

3 section three

Using Genericode in a Code List and Value Validation Methodology for XML Documents



Traditional approaches to expressing the valid values of code lists (a.k.a. controlled vocabularies) in XML instances have embedded such values directly into schema expressions. When publishing XML document models to a diverse community of users with differing requirements, this approach does not deliver the flexibility to meet everyone's needs for specialisation. Such a community building new XML vocabularies must rely on the integrity of fixed document structures. An approach to expressing codes for controlled vocabularies using genericode, and a methodology for validating XML instances against those expressions, provides the flexibility needed to make an XML vocabulary adaptable to all users.

G. Ken Holman
Crane Softwrights Ltd
gkholman@CraneSoftwrights.com

1 Introduction

Controlled vocabularies have been an important part of business documents ever since there have been business documents. Two examples of controlled vocabularies are code lists and identifiers. Code lists are agreed-upon abbreviations used in place of concepts and identifiers are abbreviations used in place of things.

Examples of code lists would be the internationally-standardised list of currency codes (e.g. "USD", "CAD", "GBP", etc.), and the internationally-standardised list of country codes (e.g. "US", "CA", "GB", etc.). Rather than spelling out such concepts in any one language, or many different languages following many different conventions, all users of code lists can use the abbreviations in the business documents without confusion. Two non-standardised lists of codes could be the codes of "North" and "South" for lines of latitude, and "Multiply" and "Divide" as operators used in currency exchange rate calculations. Such lists need not be formally standardised, but by having a convention of codes agreed upon by parties in an exchange, the meaning of the information being expressed can be conveyed without ambiguity or confusion. For interoperability, such codes are under the purview of the community as a whole, and not left to individual trading partners to come up with new values unknown to the community. Nevertheless, trading partners may wish to restrict their transactions to subsets of the codes for business reasons, thus maintaining compatibility with the community without violating one's own rules.

Examples of identifier lists would be the account codes or product codes used by trading partners. Neither of these are suitable for international standardisation, yet the "owner" of a set of accounts or products needs to be able to use an unambiguous and abbreviated reference to each account and product for lookup purposes. Sellers offer

their products identified by unique codes. Buyers specify which products are being purchased by making reference to the products' associated identifier. Managing the accounting records requires uniquely and unambiguously identifying accounts. Such identifiers are under the purview of the organisations using the community standards, not the community as a whole.

Communities of users who are exchanging business documents need to be able to publish their use of abbreviations for different codes and identifiers so that others in the community can utilise the correct value when they wish to convey the agreed-upon concept or item represented by the value. Now, as in the past, this has been accomplished in printed documentation by simple lookup charts or web sites with documented lists of codes and their associated values, referenced by the unique value of the code. Database applications can have tables of values loaded with the values to be used in validating information items stored in columns. Formal standards have not been published where this information can be conveyed unambiguously between users on different platforms in a standardised format.

With the advent of structured document markup, it has been necessary to include coded values and identifiers in XML document instances. Traditional approaches of using XML 1.0 Document Type Definition [12], W3C Schema XSD [13] and ISO/IEC 19757-2 RELAX-NG [8] to express the sets of allowed values have conflated document structure (the nesting of attributes and elements within other elements) with document content (the information found within attributes and element text). This conflation exists by embedding the constraints of coded values (aspects of content) inboard with a set of hierarchical constraints (aspects of structure). Such an approach is suitable for closed and inflexible environments but not for environments that need to be open and adaptable. This is because document constraints related to document structure are inherently different than document constraints related to document content.

Different individuals within communities of users often have diverse requirements for the content of XML documents whose structure is standardised by their community. This is quite the norm and the main reason for XML markup: a user can adopt a standardised structure within which their own information is placed in order to be conveyed to another person or programme. However, some users will want to constrain some of the values in their XML documents, perhaps for business reasons. These business reasons might be because of their own business practices, or the business reasons may vary with the different trading partners with whom they interact. Flexibility in expressing those value constraints allows them to accommodate different situations. Flexibility is not required in the document structures, just in this aspect of document content. Having content constraints conflated with structural constraints is very inflexible when one wishes to modify the content constraints while not modifying the structural constraint expressions.

So communities of users should be focused on standardising document structures for interchange, leaving document content under the purview of the individuals within the community. Guidelines, or lists of codes and identifiers, can be published by the community to be taken advantage of by individuals, as the individuals desire but without impacting on the structures governing the interchange. The validity of the document structure can be checked against constraints published by the committee, whereas the validity of the document content can be checked by an individual against the constraints governing their own policies or applications or their interfaces with trading partners.

The Organization for the Advancement of Structured Information Standards (OASIS) [5] has standardised genericcode [4] as the expression of the values in a controlled vocabulary. The genericcode standard is a work product of the OASIS Code List Representation Technical Committee [2]. This standard describes an XML vocabulary suitable for describing a set of codes for any purpose where machine access to the codes is necessary, such as defining user interfaces, loading database tables, validating XML documents, and other necessities.

However, by appropriately taking content validation out of the validation of XML document structures, there remains a void that needs to be satisfied in the information flow. The values utilised from the code lists need to be validated according to the rules set out by the community and the business rules specific to trading partners. There are different approaches to using genericcode files to satisfy such XML instance value validation requirements, one of which is based on ISO/IEC 19757-3 Schematron [9]. This approach, called the Schematron-based Value Validation Using Genericcode [10], has been successfully deployed by a number of OASIS technical committees and some user communities around the world very soon after being published. Other committees and communities are considering its deployment in light of the features it provides to a group of users with differing value validation requirements.

Finally, the traditional approaches need not be abandoned in communities where it is not possible to change the status quo of exclusively using XSD Schema technology for validation. As an XML instance itself, a genericcode file can be translated into an XSD schema fragment suitable for incorporation into a comprehensive structure and validation set of document constraints.

Nevertheless, those who are not so constrained to use only XSD Schema for document constraints will find the genericcode outboard expressions of code lists, and the Schematron-based Value Validation Using Genericcode methodology for code list validation, of interest as ways to flexibly meet the diverse requirements for code list management within a single community of users.

2 Traditional Approaches

An application preparing to act on an XML instance must first check that the instance satisfies the structural, lexical and value constraints governing the interchange. The structural constraints regulate the presence of attributes and elements within the elements of an XML instance, thus ensuring all of the information is correctly found. The lexical constraints regulate the character-level structure of each information item, thus ensuring all of the information found in the instance is correctly formed. The value constraints regulate the string-level composed value of each information item, thus ensuring all of the information specified in the instance is going to be correctly understood by the application. Any violations of these constraints warrant an application to reject the XML instance before doing any processing.

For many years designers of XML vocabularies have been addressing the needs for itemising the set of valid codes or identifiers for a given information item by describing in a schema expression an enumeration of values. There are mechanisms for doing this in the XML DTD, W3C Schema XSD and ISO/IEC 19757-2 RELAX-NG by specifying the value space for an information item to be one of a set of explicitly enumerated values.

An example excerpt of an XSD schema declaring currency codes is as follows and would be very familiar to writers of XSD files. Here the lexical constraint of being a token (a string without spaces) is constrained to only allow values of "AED" or "AFN" or "ALL" or one of many other enumerated allowed values.

```

<xsd:restriction base="xsd:token">
  <xsd:enumeration value="AED">
    <xsd:annotation>
      <xsd:documentation>
        <ccts:CodeName>Dirham</ccts:CodeName>
      </xsd:documentation>
    </xsd:annotation>
  </xsd:enumeration>
<xsd:enumeration value="AFN">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:CodeName>Afghani</ccts:CodeName>
    </xsd:documentation>
  </xsd:annotation>
</xsd:enumeration>
<xsd:enumeration value="ALL">
  <xsd:annotation>
    <xsd:documentation>
      <ccts:CodeName>Lek</ccts:CodeName>
    </xsd:documentation>
  </xsd:annotation>
  ...

```

When conflating structural, lexical and value constraints in a single constraint expression, there need be only a single validation step in confirming that none of the constraints have been violated. This is illustrated in Figure 1 where the set of constraints is established in advance of the run-time use of those constraints in validation.

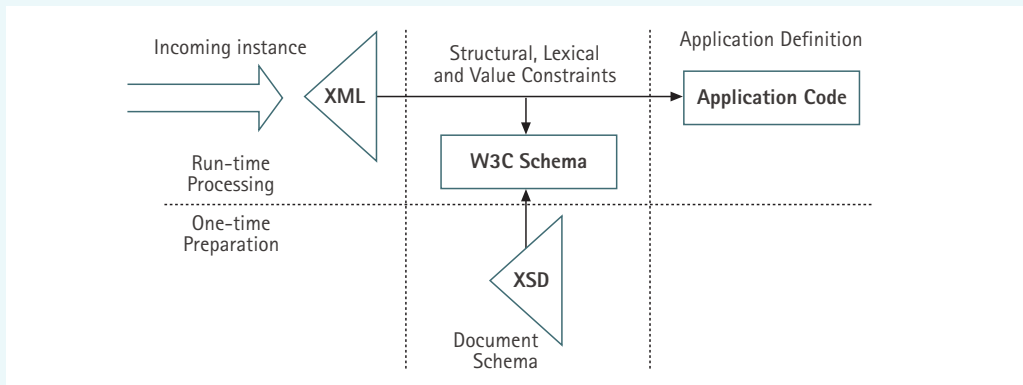


Figure 1: Traditional one-step validation approach

3 Communities of Diverse Users

As often happens in a diverse community of individuals, one size does not fit all.

For interoperability between different individuals in the community, it is necessary to mandate the structural constraints and the lexical constraints of interchanged documents so that applications can properly find the

information they need from the XML document and be assured that the lexical makeup of the information is correctly formed.

Value constraints can be, however, very different between individuals as well as different for any one individual when dealing with different trading partners.

For interoperability, value constraints need to be agreed upon between trading partners, and the community has a responsibility to provide a basic set of values for codes that represent common semantics. Fulfilling this responsibility ensures two trading partners can adopt community-specified values for commonly-understood concepts.

But the responsibility cannot be taken so far as to restrict individuals from adding concepts to accommodate specialised requirements they have for conducting business. Consider that an individual's workflow practices introduce a particular "document state" that reflects a status of processing not typified by the community's practices. For completeness, the individual will need to indicate to a trading partner the wider set of document states, extending the community set with the additional values the partner is likely to encounter.

Likewise, the responsibility cannot force individuals to accept a wide range of community-specified values when the individual's business practices restrict them to requiring a smaller subset of values. Consider that there are 167 possible currencies described by an international standard of currency abbreviations, but a trading partner wishes to engage solely in US dollars or Canadian dollars. For precision, the individual will need to indicate to a trading partner the narrower set of currency values, restricting the community set to only the values the partner is allowed to use in interchange.

These diverse requirements impact on the validation process. When traditional approaches of conflating structural and lexical constraints with value constraints are used, an individual's specialisations may require modifying the community-published artefacts. This may break the integrity of the validation process by using modified expressions that may not have been modified accurately.

Moreover if the community-published specifications utilise global declarations of the types of information items, the value constraints therein are applied to all information items of the given type across the entire document. An individual's specialisation may require multiple different modifications of value constraints for the same information item found in different document locations, such as different currencies allowed for selling items rather than buying items.

4 Standardising Document Structures for Interchange

The integrity of the document structure and the lexical space for information items is of utmost importance when supporting document interchange in a community of users.

It is entirely appropriate for a community to standardise the structural and lexical constraints of documents using a formal expression such as an XSD or DTD document model. Such a document structure may contain a wild-card extension point under which trading partners add custom requirements. Trading partners may restrict the use of certain optional constructs known not to be needed. The Universal Business Language (UBL) 2.0 [11] specification is an XML vocabulary described by XSD for business documents such as a purchase order, invoice, waybill and 28 other documents used in transactions, and supports such customisation.

Thus, trading partners within the UBL community can safely interchange documents according to the standardised published structures knowing that the information therein can be correctly found and that it is correctly formed. Having divorced the value validation from the structural and lexical validation, trading partners now have the opportunity to layer their specialised constraints as a separate process.

This does not mean value validation is unimportant or must be abandoned in the information flow.

5 Standardising the Expression of Controlled Vocabularies

The value set of information items is a resource that can be used for more than just document validation.

Consider the definition of user interfaces in the scenarios described here where trading partners need to extend the set of document statuses or restrict the set of currencies, particularly when the restrictions are different in different locations of the document. When dealing with enumerations, a drop-down list is a common control used in human interface design. The user clicks on the down arrow to bring up the predefined list and can only choose from that list. Having a machine-readable representation in XML of the available codes equips an application developer to populate the controls appropriately.

Moreover there are database applications that will validate the use of values in columns by joining the tables with other columns of allowed values. If trading partners need to exchange the lists of codes to be loaded into their respective databases, having a machine-readable representation in XML is platform independent.

The OASIS Code List Representation Technical Committee has standardised an XML vocabulary named "genericode" specifically to express a set of coded values in a form suitable for interchange. This vocabulary provides for the specification of list-level meta data (describing the list as a whole), and item-level meta data (describing individual members of the list).

Genericode is not intended as a run-time format, though an implementation may choose to use it in this fashion. Typically, the contents of a genericode file are massaged into a form suitable for run-time deployment. In this way trading partners can deploy dissimilar run-time systems with information derived from the interchanged XML document of code and identifier values.

Genericode describes a generic keyed-table structure for information representation, unlike other standardised table structures such as the CALS/OASIS Open Table Model [1] used for information presentation.

An example genericode file is as follows, where an organisation has created a small list restricting currency codes to only the US or Canadian dollar being used:

```
<gc:CodeList
  xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/">
  <Identification>
    <ShortName>CAUSCurrencyCode</ShortName>
    <LongName>Canadian and US Currency Codes</LongName>
    <Version>1</Version>
    <CanonicalUri>urn:x-company:CAUS-currency</CanonicalUri>
    <CanonicalVersionUri>urn:x-company:CAUS-currency:
```

```

1</CanonicalVersionUri>
  </Identification>
  <ColumnSet>
    <Column Id="code" Use="required">
      <ShortName>Code</ShortName>
      <Data Type="xsd:normalizedString"/>
    </Column>
    <Column Id="name" Use="optional">
      <ShortName>Name</ShortName>
      <Data Type="xsd:string"/>
    </Column>
    <Key Id="codeKey">
      <ShortName>CodeKey</ShortName>
      <ColumnRef Ref="code"/>
    </Key>
  </ColumnSet>
  <SimpleCodeList>
    <Row>
      <Value ColumnRef="code">
        <SimpleValue>CAD</SimpleValue>
      </Value>
      <Value ColumnRef="name">
        <SimpleValue>Canadian Dollar</SimpleValue>
      </Value>
    </Row>
    <Row>
      <Value ColumnRef="code">
        <SimpleValue>USD</SimpleValue>
      </Value>
      <Value ColumnRef="name">
        <SimpleValue>US Dollar</SimpleValue>
      </Value>
    </Row>
  </SimpleCodeList>
</gc:CodeList>

```

Freely-available resources [3] can be downloaded for the presentation of genericode files as HTML documents. This allows non-technical readers to assess the contents of a code or identifier list without having to wrestle with reading the angle brackets of XML.

6 A Methodology for Validating the Use of External Controlled Vocabularies

Document validation of the use of a controlled vocabulary of values remains a most important aspect of information processing.

For those systems where the use of XSD is mandatory, a genericode expression can be translated into XSD enumerations. Unfortunately, though, this does not address some of the diversity requirements when the XSD enumerations are used as global data types for information items found in different document locations.

The anticipated audience for UBL 2.0 is incredibly diverse and all users will have some custom requirements for value validation for many of the 91 different information items that are code lists. Many other information items

are identifiers. The UBL committee provides prototypical values for only a dozen of these code lists, anticipating trading partners to layer business value constraints on top of the structural and lexical constraints in the XSD.

To meet this need, the UBL technical committee adopted the use of genericcode to express code lists and identifier lists, and formalised a two-step validation process, illustrated in Figure 2. The first step confirms using XSD constraints that all of the information items in the UBL document are properly structured and the values of codes are appropriately structured at the lexical level. The second step utilises ISO 19757-3 Schematron at run time to assert the allowable code and identifier values are in the information items.

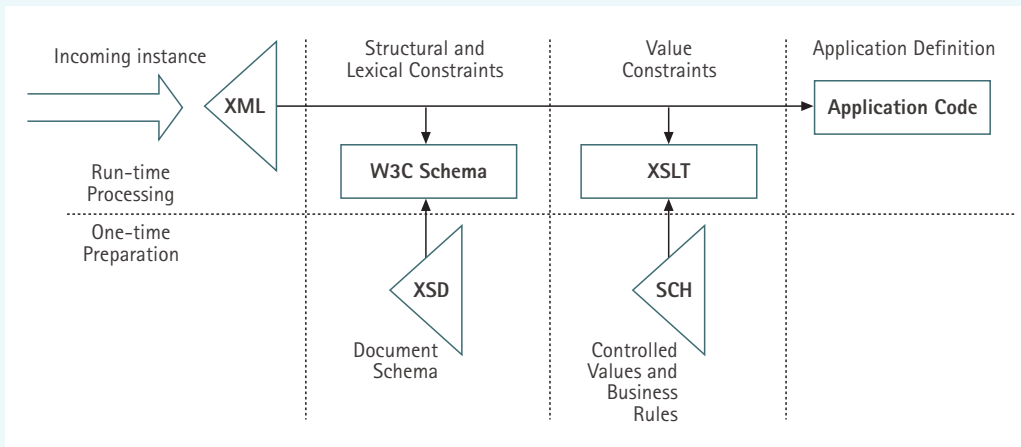


Figure 2: Two-step validation approach

Note how the second step has no meaning unless the first step has succeeded without constraint violations. The values need not be checked if the information is in the wrong place or is structured improperly.

The creation of the second pass Schematron schema is done once to create the run time artefact used for every instance that passes structural and lexical validation. The inputs to the creation process are illustrated in Figure 3. The context/value association file describes the locations of code list and identifier list values in an XML document by pointing to the appropriate genericcode files. This information is processed into a Schematron pattern suitable for inclusion in a complete Schematron schema. This complete Schematron schema can contain business rule assertions, also written as patterns, to describe a complete set of nuanced value constraints between trading partners. The Schematron run-time artefact is then prepared statically, changing only when any of the inputs are changing.

Note that the deployment of Schematron utilises an XSLT 1.0 [14] implementation of run-time validation. Other implementations of Schematron are available including one in the Python programming language [7].

As this methodology has no dependencies on UBL (or any document vocabulary), the Universal Business Language Technical Committee formally transferred the specification and maintenance of this two-step validation approach to the Code List Representation Technical Committee.

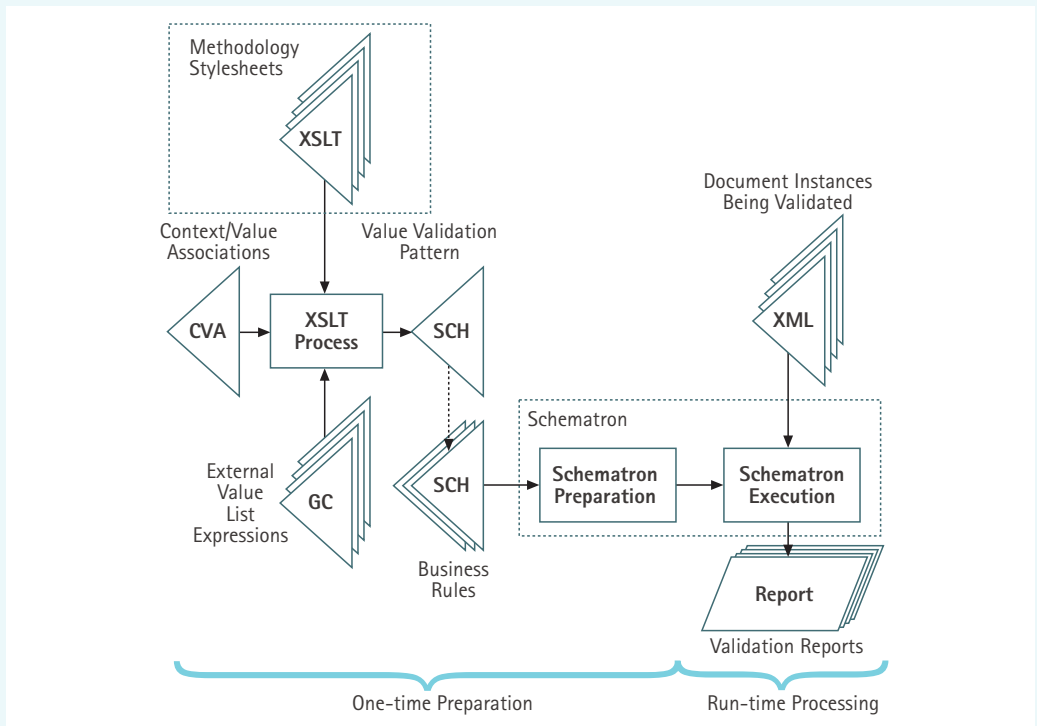


Figure 3: Preparing the value validation step

7 Conclusion

Traditional approaches have conflated code list and identifier value constraints with document structure constraints in XSD document models for XML instances. These approaches do not meet well the needs of a community with diverse user requirements for value constraints within a strict set of community requirements for structure constraints.

The OASIS genericcode specification [4] provides a flexible method of expressing lists of codes and identifiers. This expression is suitable for the interchange of lists of content values between trading partners. Each trading partner can then apply the list of codes in their respective run-time environments with their respective run-time technologies independent of those of the other partner.

When the run-time environment incorporates validating the use of controlled vocabularies of values in XML document information items, the Schematron-based value validation for genericcode [10] is an appropriate and powerful approach to flexibly meet business requirements using formal artefacts that can be interchanged between trading partners.

A case study [6] detailing the steps required to analyse and adopt the use of this two-pass Schematron-based validation methodology was published providing guidelines for another OASIS committee's consideration. This case study can be used as a model for weighing the methodology's applicability for any XML document vocabulary.

These new approaches addressing information interchange requirements compel their consideration by communities looking to standardise XML document structure and value constraints. Flexibility, combined with unambiguous formalism, can build a community-based environment where business-driven specialisation can be accomplished successfully within community-driven standardisation.

8 References

- [1] Harvey Bingham CALS Table Model Document Type Definition - <http://www.oasis-open.org/specs/a502.htm>
- [2] G. Ken Holman, Chair Code List Representation Technical Committee
- <http://www.oasis-open.org/committees/codelist>
- [3] Crane Softwrights Ltd. Free developer resources - <http://www.CraneSoftwrights.com/links/res-ublo.htm>
- [4] Tony Coates genericcode - <http://www.genericcode.org/>, UBL repository <http://docs.oasis-open.org/codelist/genericcode>
- [5] Organization for the Advancement of Structured Information Standards (OASIS) - <http://www.oasis-open.org/>
- [6] G. Ken Holman OASIS code list adaptation case study (CIQ)
- http://www.oasis-open.org/committees/document.php?document_id=24813
- [7] Python - <http://www.python.org>
- [8] James Clark, Makoto Murata ISO/IEC 19757-2 RELAX-NG (Regular Language for XML)
- <http://www.relax-ng.org/>, ISO/IEC JTC 1/SC 34/WG 1 <http://www.jtc1sc34.org/>
- [9] Rick Jelliffe ISO/IEC 19757-3 Schematron <http://www.schematron.com/>, ISO/IEC JTC 1/SC 34/WG 1
- <http://www.jtc1sc34.org/>
- [10] G. Ken Holman Schematron-based Value Validation Using Genericcode
- http://www.oasis-open.org/committees/document.php?document_id=24810
- [11] Jon Bosak, Tim McGrath, G. Ken Holman Universal Business Language (UBL) Version 2.0 - <http://docs.oasis-open.org/ubl/os-ubl-2.0/>, OASIS UBL Technical Committee <http://www.oasis-open.org/committees/ubl/> 2006
- [12] Tim Bray, et al. XML 1.0 - Extensible Markup Language (Fourth Edition)
- <http://www.w3.org/TR/2006/REC-xml-20060816> 2006-08-16
- [13] Henry S. Thomson, et al. XML Schema Part 1: Structures Second Edition
- <http://www.w3.org/TR/2004/PER-xmlschema-1-20040318> 2004-03-18
- [14] James Clark XSL Transformations (XSLT) Version 1.0
- <http://www.w3.org/TR/1999/REC-xslt-19991116> 1999-11-16