Context/Value Association using Genericode 1.0

Public Review Draft 01

22 July 2009

Specification URIs:

This Version:

http://docs.oasis-open.org/codelist/prd01-ContextValueAssociation-1.0/doc/context-value-association.html (Authoritative)
http://docs.oasis-open.org/codelist/prd01-ContextValueAssociation-1.0/doc/context-value-association.pdf
http://docs.oasis-open.org/codelist/prd01-ContextValueAssociation-1.0/doc/context-value-association.xml

Previous Version:

N/A

Latest Version:

http://docs.oasis-open.org/codelist/ContextValueAssociation/doc/context-value-association.html (Authoritative)
http://docs.oasis-open.org/codelist/ContextValueAssociation/doc/context-value-association.pdf
http://docs.oasis-open.org/codelist/ContextValueAssociation/doc/context-value-association.xml

Technical Committee:

OASIS Code List Representation Technical Committee

Chair:

G. Ken Holman, Associate member <gkholman@CraneSoftwrights.com>

Editor:

G. Ken Holman, Associate member <gkholman@CraneSoftwrights.com>

Related Works:

This re-titled specification replaces the draft "Schematron-based Value Validation Using Genericode" version 0.1 draft 1 http://www.oasis-open.org/committees/document.php?document_id=24810, which itself replaced the draft "UBL Methodology for Code List and Value Validation" version 0.8 draft 6 http://www.oasis-open.org/committees/document.php?document_id=24518, which itself replaced the "UBL Code List Value Validation Methodology" document that is directly referenced by name in the UBL 2.0 specification. This specification was originally a work product of the UBL Technical Committee before being transferred *in toto* to the Code List Representation Technical Committee.

This specification utilizes genericode for the representation of codes and identifiers. The latest approved version of genericode is found at http://docs.oasis-open.org/codelist/approved/genericode/doc/oasis-code-list-representation-genericode.pdf.

Declared XML Namespaces:

http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/

Abstract:

This Public Review Draft 01 normatively describes the file format used in a "context/value association" file (termed in short as "a CVA file"). This file format is an XML vocabulary using address expressions to specify hierarchical document contexts and their associated constraints. A document context specifies one or more locations found in an XML document or other similarly structured hierarchy of information. A constraint is expressed as either an explicit expression evaluation or as a value inclusion in one or more controlled vocabularies of values. This file format specification assumes a controlled vocabulary of values is expressed in an external resource described by the OASIS genericode standard.

Context/value association is useful in many aspects of working with an XML document using controlled vocabularies and other constraints. Two examples are (1) for the direction of user data entry in the creation of an XML document, ensuring that only valid values are proffered in a user interface selection such as a drop-down menu; and (2) for the validation of the correct use of valid values found in an XML document.

Status:

This document was last revised or approved by the Code List Representation TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/codelist/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/codelist/ipr.php).

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/codelist/errata/context-value-association/.

Notices:

Table of Contents

Appendixes

---

1. Introduction

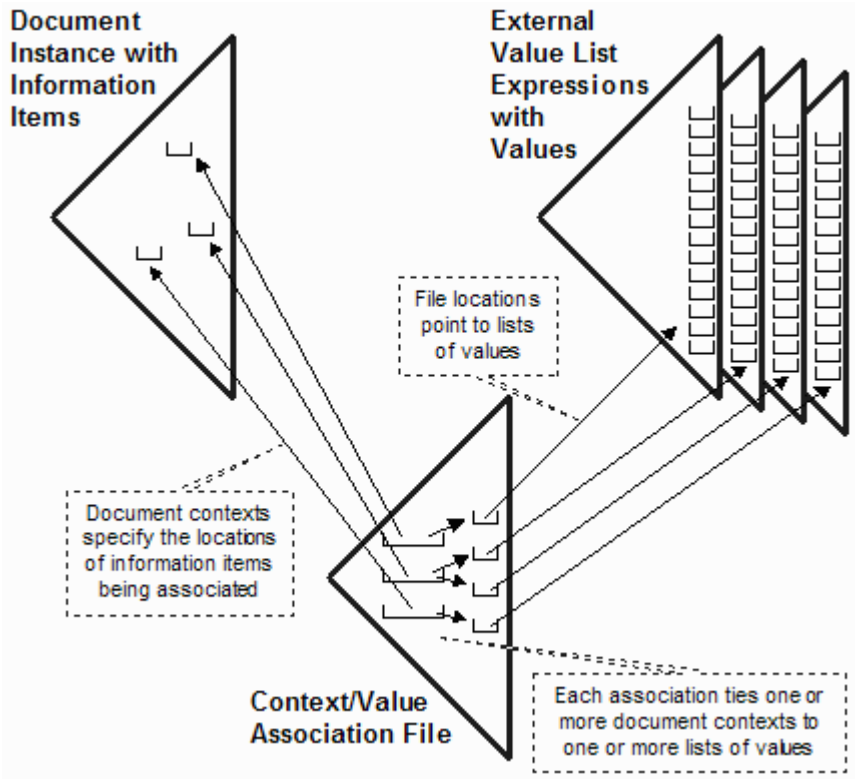
This Public Review Draft 01 specifies an XML vocabulary used to express the relationship between information items found in a structured hierarchy (such as the XML instance of a business document) and two kinds of constraints imposed on those items. One kind constraint is that the value is a member of a set of controlled vocabularies of enumerated values (such as code lists or identifiers). Another kind of constraint is an arbitrary evaluation of a boolean query expression (such as a non-enumerable code list value check (say, the checksum calculation of an ISBN number), a business rule or a superimposed lexical constraint such as a maximum string value length).

The W3C XML Path Language [XPath 1.0][XPath 2.0] is an example of an addressing and query language used to specify document contexts and to express evaluations using a built-in function library.

The OASIS genericode standard [genericode] for code list representation is used to specify sets of enumerated values, be they codes or identifiers or any other set of values. Genericode provides for list-level metadata to be expressed for the list as a whole, as well as value-level metadata for each member of the list.

An OASIS context/value association (CVA) file specifies the relationship of information items in document contexts to both external enumerations of values allowed for each item and to arbitrary evaluations processed for each item. The diagram Figure 1, “Context/value association” shows the associations in the CVA file, each with the document context pointing into the document instance and one or more file locations pointing to the enumerations. The specification of related instance-level metadata for information items identifies which external code list or value test correlates with the information item's value. The CVA vocabulary allows one to masquerade the list-level metadata found in the external expression with specified overriding metadata.

Figure 1. Context/value association



There are no mandated technologies with which one is required to implement the functionality of a context/value association file. Runtime artefacts can use any technology desired.

1.1. Terminology

1.1.1. Key words

The key words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional* are to be interpreted as described in [RFC2119]. Note that for reasons of style, these words are not capitalized in this document.

1.1.2. Definitions

co-occurrence constraint

> A *co-occurrence constraint* is a constraint on an information item that is dependent upon the value of another information item.

document context

> Every element and attribute information item of an XML document is in a *document context* described by its hierarchical ancestry of elements. A fully-qualified document context specifies the information item's precise location in the document.

pattern

> A *pattern* is an XPath location path address formally defined in XSLT, used to specify one or more document contexts. Pattern addresses are a subset of all possible XPath addresses. In XPath 1.0 [XPath 1.0] the location path address is formally specified in XSLT 1.0 [XSLT 1.0 Pattern] production [1]. In XPath 2.0 [XPath 2.0] the location path address is formally specified in XSLT 2.0 [XSLT 2.0 Pattern] production [1].

1.1.3. Key concepts

code list representation

The representation of a code list is the XML vocabulary with which the values of a set are enumerated and described. For example, the international set of currency codes is maintained by the UN/ECE who specifies the code values and their meanings. This abstract set of codes may have many *code list representations*, including database tables, comma-separated-values records, XML instances, etc. A context/value association file points to genericode [genericode] XML representations of enumerations.

lexical validation

The validation of the use of a lexical grammar addresses the token syntax or the rules for patterns of characters within a character sequence. A parallel to the structural grammar of elements and attributes, the lexical grammar is the structure of the members of a flat character sequence described as patterns. *Lexical validation* confirms that a character sequence matches its declared grammatical pattern constraints (not value constraints) such that every information item's character sequence is correctly formed.

structural validation

The validation of the use of an XML vocabulary grammar addresses the labeling, ordering, nesting and cardinality of elements and attributes constrained by a document schema. *Structural validation* confirms that an XML document matches its declared structural constrains such that every information item is in its correct document location.

value validation

Distinct from structural and lexical validation checking, respectively, the element and character-level constitution of the representation of an information item, *value validation* is the process where constrained information items are checked for having appropriate values. Value validation is only meaningful after both structural and lexical validation have confirmed the value meets its document location and character representation constraints.

1.2. Normative References

[genericode] OASIS Committee Specification Genericode Version 1.0 http://docs.oasis-open.org/codelist/cs-genericode-1.0/doc/oasis-code-list-representation-genericode.pdf, December 2007

[RFC2119] S. Bradner Key words for use in RFCs to Indicate Requirement Levels Internet Engineering Task Force, March 1997

[XPath 1.0] James Clark, Steve DeRose XML Path Language (XPath) Version 1.0 1999-11-16

[XPath 2.0] Anders Berglund, et al XML Path Language (XPath) Version 2.0 2007-01-23

[XSLT 1.0 Pattern] James Clark XSL Transformations (XSLT) Version 1.0 section 5.2 "Patterns" 1999-11-16

[XSLT 2.0 Pattern] Michael Kay XSL Transformations (XSLT) Version 2.0 section 5.5.2 "Syntax of Patterns" 2007-01-23

1.3. Non-Normative References

[currency] UN/ECE Working Party on Facilitation of International Trade Procedures Alphabetical Code for the Representation of Currencies Recommendation 9 (Second Edition) January 1996, ECE/TRADE/203 [Edition 96.1]

[DOM] World Wide Web Consortium Document Object Model

[Schematron] Rick Jelliffe ISO/IEC 19757-3 Schematron, Document Schema Definition Languages (DSDL) Part 3, ISO/IEC JTC 1/SC 34/WG 1

2. Example applications of context/value association files (Non-Normative)

XML documents are hierarchical arrangements of elements and attributes where some of these information items are constrained to one of a set of enumerated values and/or to a number of arbitrary evaluations. The document may also provide for such an information item some instance-level metadata in related elements and attributes. This metadata correlates the information item to the list-level metadata of the list of values that governs the value of the item.

Instance-level metadata can be defined differently for different XML vocabularies. Elements and/or attributes can be put aside to contain qualifying values that identify in the instance the list-level metadata associated with a given code or tested value. Typically, however, attributes are used for instance-level metadata for information items, attached to an element whose value is from a list, or a sibling of an attribute whose value is from a list. A simple example of such metadata is a version identifier. Consider that an element in an XML document has a value from one of two different value lists based on version 1 of the value list or version 2 of the value list, where that version number is expressed in the list's list-level metadata. The version instance-level metadata could be specified in an attribute, say `ver=` for an example. The user can then qualify the element's value by indicating `ver="2"` to indicate the qualifying values are from the second version of the value list corresponding to the list's genericode list-level metadata. A value found in one version of the list may not be available in another version of the list, thus the user's specification in instance-level metadata of the version of the list in play for the value is significant. Absent instance-level metadata, all of the lists' values are in play.

Information items may share a broad definition across the entire document, or possibly be more narrowly defined in sub-document contexts. An example of a broadly defined information item is a currency attribute with a document context that spans across an invoice: all references to currency across the document need to be associated with a single set of currency coded values. Examples of two narrowly defined information items with unique document contexts are a product identifier element named "`ID`" as a child of an element named "`Product`", and a document identifier similarly named "`ID`" but as a child of the document element: the former might be constrained by a list of identifiers derived from an inventory status query, while the latter might not be constrained at all. A broad specification of the element's value is inappropriate when other elements of the same name are used for different purposes in different document contexts.

Figure 2, “The use of context/value association files in document creation” illustrates how a context/value association file can help in a data entry scenario. It is common in data entry scenarios to limit values for information items to only those allowed by using a user interface control such as a drop-down list. By not making the data entry free-form, the value after creation is guaranteed to be one from the list. Hardwiring the list of allowed values in the application is inflexible if the enumeration changes based on the use of the document, say, for particular trading partners.

Externalizing the list in a genericode file expresses the enumeration as a resource available to the application for data entry. However, the choice of genericode file may vary on a per-information item basis due to the item's document context, or perhaps vary again for particular trading partners. Expressing the appropriate mappings in a colloquial fashion inhibits interoperability and the sharing of resources and program code.

A context/value association file specifies the relationship from information items found in different

document contexts to one or more external genericode files for each item. With these relationships a directed authoring environment can precisely direct the editing of individual information items. Different context/value association files can then be used to create instances for different purposes that have different constraints on the enumerations used therein.

The data flow illustrates the CVA file pointing to the genericode files. A one-time preparation step distills the information from the genericode files in the contexts described by the CVA file and prepares a run-time artefact incorporating all the aggregated information. At run-time this artefact informs the directed authoring facility that is governing the data entry, without needing to access the CVA or genericode files. The output of the directed authoring is the XML instance having been populated with appropriate values for the given document contexts as regulated by the CVA associations.

Figure 2. The use of context/value association files in document creation

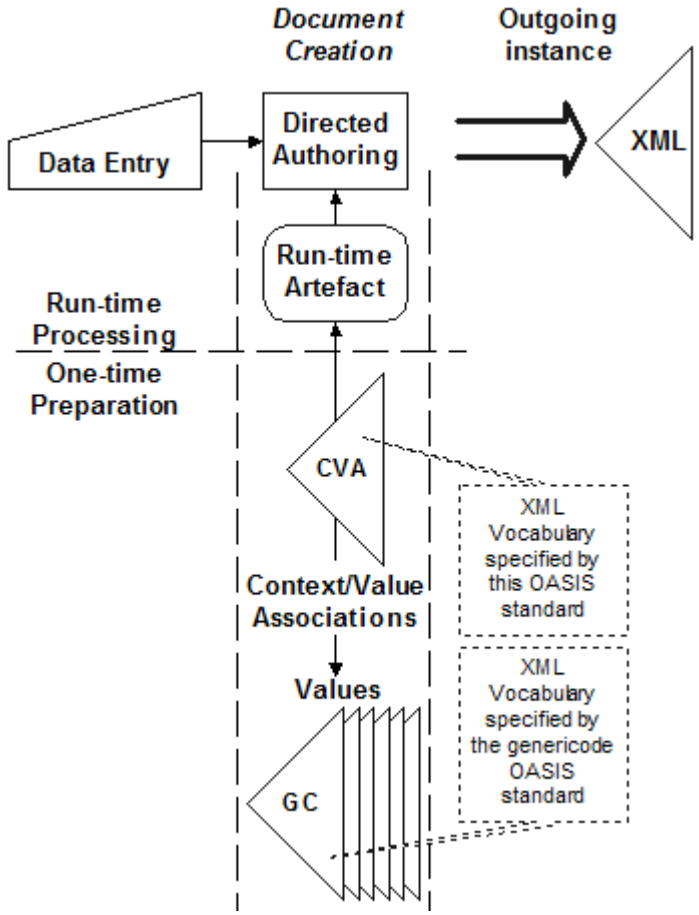


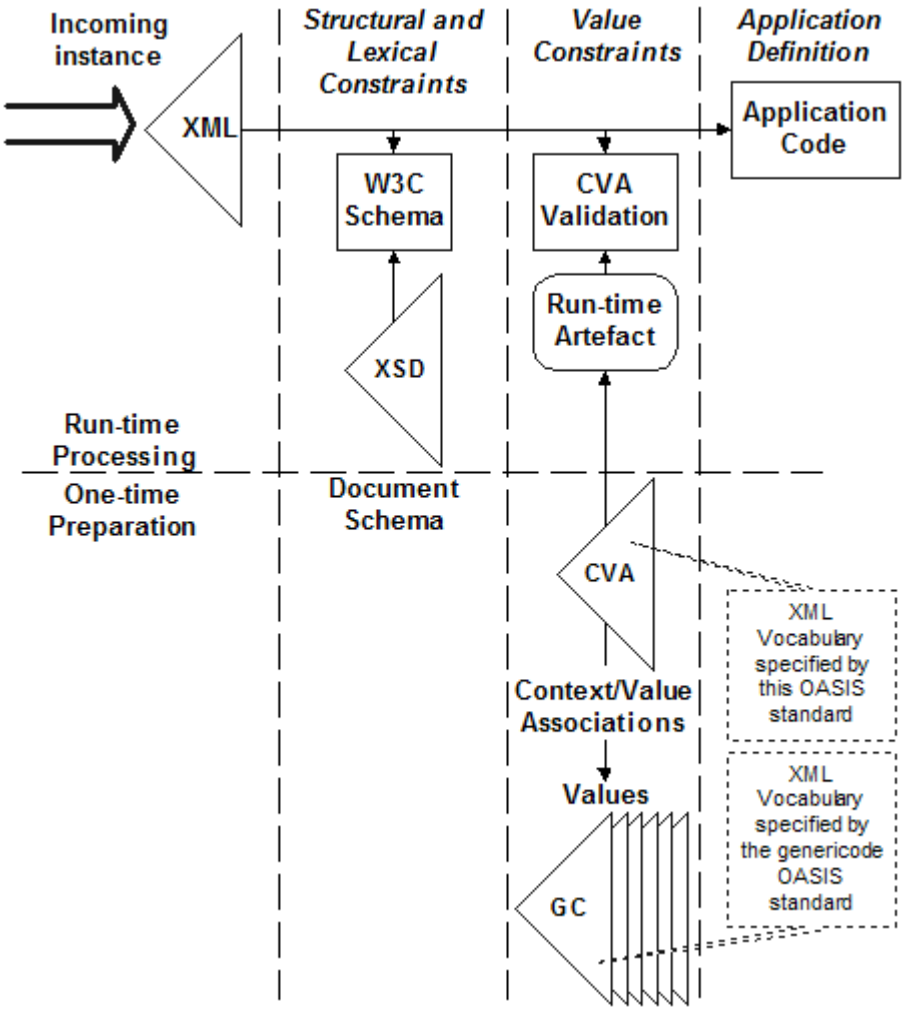
Figure 3, “The use of context/value association files in document validation” illustrates how a context/value association file can help in a validation scenario. Validation is important to confirm that the values used in an incoming XML instance are the ones allowed for each information item in each document context.

The historical use of W3C Schema enumerations for code lists and identifiers includes the value constraints along with the structural and lexical constraints. Conflating value validation with structural validation requires the structural constraint expression (i.e. the schema) to be changed to accommodate specific value validation requirements between partners in a document interchange. But often, schemas are distributed as read-only artefacts that should not be modified to accommodate trading partner nuances. By using associations a community of users can standardize on the expression of the structural constraints governing document interchange using XSD. Meanwhile, individuals in the community can tailor the value constraints on a per trading partner or scenario profile basis using context/value association files.

In document validation the value validation is irrelevant if the information items being tested are not in the expected document contexts. This suggests a first-pass structural validation to confirm the integrity of the information item locations. Only when structural validation is successful can value validation reflect the true fitness of the values used in the expected document contexts.

The data flow illustrates the CVA file pointing to the genericode files. A one-time preparation step distills the information from the genericode files in the contexts described by the CVA file and prepares a run-time artefact. The data flow shows the incoming XML instance first being checked using W3C Schema for structural integrity. When successful the information items are all correctly found and correctly formed. The incoming instance can then be checked for value validation using the run-time artefact created by the CVA associations (not the CVA or genericode files). When CVA validation is successful the application can then act on the incoming instance.

Figure 3. The use of context/value association files in document validation



Publicly-available enumerations of coded values are published and maintained by registration authorities regarded as the custodians of the code list of defined values and their associated definitions. A schema may constrain a document value to be one of the entire set of published code list values so as to ensure the document value used represents the published definition. A common example of a publicly-maintained code list is that which enumerates currency value indications [currency] and some projects have created genericode expressions of currency code lists containing all known values.

An important feature of CVA files is the ability to masquerade a genericode file in use as being another genericode file that is not being used. The typical uses of currency values illustrates this

well. Should two trading partners agree to use only a limited number of currency values, say Canadian dollars and US dollars, a custom genericode file can be created taking only those two values from the publicly available genericode file for all currency codes. The smaller genericode file with only two currency codes necessarily uses custom list-level metadata that is different than the published list-level metadata of the complete genericode file for all currency codes.

When using these two currency codes in an instance, however, the instance-level metadata needs to reflect the published list-level metadata of the complete genericode file, as that would be the international identification of the semantics behind the values being used for the currency information items. A CVA file pointing simply to the reduced trading-partner list and its list-level metadata would prohibit the association of currency values with instance-level metadata of the complete list. Therefore, a CVA file is allowed to masquerade the genericode list-level metadata being pointed at by trading partners with any genericode list-level metadata the community of all trading partners agrees to use.

Thus, a community can standardize on the use of an international list of currency coded values, while individuals can utilize their own subsets of those values without conflict.

3. Specifying value constraints in document contexts

This Public Review Draft 01 describes the document model for a context/value association file in which associations are made between document contexts of information items and either query language expression evaluations or external resources containing the values allowed for those items.

All elements in this document model are in the namespace identified at the beginning of this specification. The formal expression of this document model is found through links in the namespace documentation found at the URL of the namespace URI.

Using this document model, two partners in document interchange can exchange an instance of agreed-upon context/value associations, accompanied by instances of agreed-upon genericode expressions in support of the enumeration associations.

In summary, the association file describes the query language evaluations and points to the genericode files as system resources using URI strings, and names these pointers using XML identifiers unique to the association instance. The document context of each information item to be governed by an association is then associated with as many pointer identifiers as required to enumerate all of the possible values from all of the possible enumerations.

Note: Vocabulary namespace convention (non-normative)

As is true for genericode, only the document element of a CVA instance is in the CVA namespace. All of the other elements are in no namespace.

Note: A pro-forma example (non-normative)

A pro-forma context/value association instance reads as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<cva:ValueListConstraints xmlns:cva=
"http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/"

  ... namespace declarations as required for context addresses ...
  queryBinding="xslt"
  id="urn:x-optional-unique-identifier-for-external-referencing"
  name="required-unique-name-token-for-internal-referencing"
```

```
version="Revision: 27b - 2006-11-23 15:00z">
<Annotation>
  <Description>
    This is included for illustrative purposes.
  </Description>
  <AppInfo>
    This is included for illustrative purposes.
  </AppInfo>
</Annotation>

<Title>
  This is the main context/value association file for project X.
</Title>

<Include uri="other-assoc-file-1.cva">
  <Annotation>
    <Description>
      Incorporating information from another file.
    </Description>
  </Annotation>
</Include>
<Include uri="other-assoc-file-2.cva">
  <Annotation>
    <Description>
      Incorporating information from yet another file.
    </Description>
  </Annotation>
</Include>

<ValueTests>
  <Annotation>
    <Description>
      All the tests available to be used.
    </Description>
  </Annotation>
  <ValueTest xml:id="a1" test="this > that">
    <Annotation>
      <Description>
        An example expression comparing child values.
      </Description>
    </Annotation>
  </ValueTest>
  <ValueTest xml:id="a2" test="string-length(.)&lt;=35">
    <Annotation>
      <Description>
        An example expression checking the length of data.
      </Description>
    </Annotation>
  </ValueTest>
</ValueTests>

<ValueLists>
  <Annotation>
    <Description>
      All the lists available to be used.
    </Description>
  </Annotation>
  <ValueList xml:id="a3" uri="enumeration1.gc">
    <Annotation>
      <Description>
        A set of external values with no masquerading meta data.
      </Description>
    </Annotation>
  </ValueList>
  <ValueList xml:id="a4" uri="enumeration2.gc">
    <Annotation>
      <Description>
        A set of external values with masquerading version meta data.
      </Description>
    </Annotation>
    <Identification>
      <Version>1.4</Version>
    </Identification>
  </ValueList>
  <ValueList xml:id="a5" uri="enumeration3.gc">
```

```
  <Annotation>
    <Description>
      Another set of external values with no masquerading meta data.
    </Description>
  </Annotation>
  </ValueList>
</ValueLists>

<InstanceMetadataSets>
  <Annotation>
    <Description>
      All the kinds of instance-level meta data available.
    </Description>
  </Annotation>
  <InstanceMetadataSet xml:id="i1">
    <InstanceMetadata address="../@version"
                      identification="Version">
      <Annotation>
        <Description>
          The version in this example is an attribute for an attribute.
        </Description>
      </Annotation>
    </InstanceMetadata>
  </InstanceMetadataSet>
  <InstanceMetadataSet xml:id="i2">
    <InstanceMetadata address="@version"
                      identification="Version">
      <Annotation>
        <Description>
          The version in this example is an attribute for an element.
        </Description>
      </Annotation>
    </InstanceMetadata>
  </InstanceMetadataSet>
</InstanceMetadataSets>

<Contexts>
  <Annotation>
    <Description>
      All the contexts using lists.
    </Description>
  </Annotation>
  <Context address="@item-a" values="a4" metadata="i1">
    <Annotation>
      <Description>
        Associating a set of values with all item-a= attributes.
      </Description>
    </Annotation>
  </Context>
  <Context address="context-b1//item-b" values="a3 a5"
           metadata="i2">
    <Annotation>
      <Description>
        Associating two sets of values with all item-b descendants
        of the context-b1 element.
      </Description>
    </Annotation>
  </Context>
  <Context address="context-b2/item-b"
           values="a5" mark="characteristic-1" metadata="i2">
    <Annotation>
      <Description>
        Associating a set of values with item-b children of the
        context-b2 element.
      </Description>
    </Annotation>
  </Context>
  <Context address="item-c" values="a1 a2">
    <Annotation>
      <Description>
        Associating a pair of expressions with all item-c elements.
      </Description>
    </Annotation>
  </Context>
</Contexts>
```

```
</cva:ValueListConstraints>
```

The document element <ValueListConstraints> specified by this vocabulary is in the "http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/" namespace while all other elements specified by this vocabulary are in no namespace. Elements of other namespaces are allowed only in reserved locations where specified in this document and schema.

The required name= attribute of the document element specifies a name token for internal referencing by downstream processes that take advantage of run-time or alternative expressions of the information found in the context/association file.

The optional queryBinding= attribute of the document element specifies a name token representing the syntax of item and context addresses. Absent this attribute, the default value can be whatever the processing system assumes, thus leaving it to the processing system to interpret the syntax as it wishes.

Note: Query language names (non-normative)

The following query language names are reserved without further definition. Implementations which use different query language bindings are encouraged to use one of these names if appropriate : stx, xslt, exslt, xslt2, xpath, xpath2, xquery.

The optional id= attribute of the document element specifies a public identifier (typically, but not required to be, a URI) for externally referencing the context/value association file, such as in business process specifications and formal trading partner agreements.

The optional version= attribute of the document element is for reporting purposes suitable for identifying the version of the context/value association file. Its value can be copied to downstream synthesized files distinguishing the particular version of the information from previous versions of the same file.

The optional xml:base= attribute of the document element sets the base URI for any relative URI references in descendent URI-typed attributes.

The <Annotation> element is available everywhere where either application-specific or documentary constructs are contextually added to the content. This element is always added as the first child element of the construct being annotated. When at the top level, that is the first child of the document element, this annotates the entire context/value association file as a whole. There are two optional child elements for <Annotation>: <Description> is used to express information targeted for a human reader, and <AppInfo> is used to express information targeted for an application of some kind to process. In both cases these elements can contain text and/or elements using namespaces other than the context/value association namespace.

Most context/value association constructs allow for such annotations, and in the descriptions of elements that follow only those elements not supporting the <Annotation> first child element are so noted.

The optional <Title> element is for reporting purposes and is typically a single line of text suitable for the titling of any report generated from the documentary content. This element does not allow for an <Annotation> child element. Its string value (stripping markup) can be copied to text-only files to indicate which context/value association file has contributed a portion of its information when there are a number of portions included. This element can have any content for documentation purposes, provided any elements in that content are in a foreign namespace and not in the context/value association file namespace.

The optional and repeatable <Include> element is a directive to incorporate the associations

found in other context/value association files into the one generated result. The uri= attribute points to the file being included. The optional xml:base= attribute overrides any ancestrally-specified base URI for a relative URI reference in the uri= attribute. Where two contexts from the suite of association files match the same information item in XML documents, the priority for the single match that is acted upon is highest for the contexts in the including association file, then next highest for the last association file included by an <Include> directive, then the next-to-last association file included by an <Include> directive, and so on with the lowest priority being the first association file included by an <Include> directive. Any included association files having such directives will treat those in priority before other directives of the including file.

Each optional <ValueTest> element child of the optional <ValueTests> element uses the required xml:id= attribute to declare an identifier suitable for internally referencing an internal expression evaluation. This identifier must be unique within the context/value association file in which it is declared, but is not required to be unique across included context/value association files as well. The required test= attribute contains an evaluation expression written using the query binding language in play.

Each optional <ValueList> element child of the optional <ValueLists> element uses the required xml:id= attribute to declare an identifier suitable for internally referencing the external expression of values. This identifier must be unique within the context/value association file in which it is declared, but is not required to be unique across included context/value association files as well. The required uri= attribute points to the associated genericode-encoded file. A relative URI value is resolved relative to the base URI of this element. The optional xml:base= attribute overrides any ancestrally-specified base URI for a relative URI reference in the uri= attribute. The key= attribute is optional when the referenced list has only a single key. The attribute is required when the referenced list has multiple keys, in order to indicate which single key is to be used for value lookup for the singleton coded value. This element is allowed to have a masqueradeUri= attribute and/or a single <Identification> element whose element children represent masquerading metadata for the external genericode file.

An <Identification> element is allowed to have the same element children as a genericode file's list-level metadata element children of the <Identification> element. These being any or all of <ShortName>, <LongName>, <Version>, <CanonicalUri>, <CanonicalVersionUri>, <LocationUri>, <AlternateFormatLocationUri> and <Agency>, where <Agency> is allowed to have the children <ShortName>, <LongName> and <Identifier>. The semantics of these elements are defined by genericode and are irrelevant in meaning to the association. While the <Identification> element is allowed to have an <Annotation> first child, the other children of <Identification> are not allowed to have an <Annotation> child at all.

An application supporting context/value association, when analyzing instance-level metadata, must treat masquerading list-level metadata in the following order. First, any masquerading list-level element content in the <Identification> declaration has highest precedence. Next, any masquerading list-level metadata in the genericode file referenced by masqueradeUri= has next highest precedence. Finally, over any list-level metadata found in the genericode file of values has the lowest precedence. Empty masquerading metadata replaces lower precedence metadata with no definition. See Section 3.1, “Instance-level metadata” for more details.

Each optional <InstanceMetadataSet> element child of the optional <InstanceMetadataSets> element contains a number of <InstanceMetadata> elements each describing two addresses in the query binding syntax. The address= attribute is an address pointing to a metadata information item assuming the current node is a context information item. The identification= attribute is an address pointing to an identification information item descending from an <Identification> element found either in an external value list's list-level metadata, or a masquerading <Identification> element found as a child of <ValueList>. See Section 3.1, “Instance-level metadata” for more details.

Each optional <Context> element child of the optional <Contexts> element points to all of the

value constraints (either value expressions or lists of values) for a given information item using the `values=` attribute. The attribute string is a white-space-separated list of one or more `xml:id=` attributes used as `<ValueTest>` and `<ValueList>` identifiers in the same context/value association file.

For each specified `<Context>` all of the cited `<ValueTest>` tests must be true and the value must be present in any of the cited `<ValueList>` lists that are applicable given the presence of instance-level metadata and not empty of values, otherwise the constraints on the context are considered violated. The applicability of instance-level metadata is measured by the values at the respective addresses of the two attributes of each `<InstanceMetadata>` element that is a child of the element pointed to by the `metadata=` attribute.

Note: Validating a CVA file (non-normative)

Validating a context/value association file ensures each attribute value pointing to an existing `xml:id=` attribute is pointing to an appropriate `<ValueTest>`, `<ValueList>` or `<InstanceMetadataSet>` element. This cannot be done solely with the provided schema expression.

Note: Empty document contexts list (non-normative)

It is acceptable to have an absent or empty `<Contexts>` element in which case there are no context/value associations specified in the file. This makes the context/value association file a useful placeholder for future constraints, or perhaps an artefact used to explicitly express that no constraints are applicable in a given situation.

The `<Context>` element must declare the address of the information items being associated with tests or lists by using the `address=` attribute expressed in the bound query language syntax.

Note: Examples of XPath addresses (non-normative)

Appendix E, *Example context/value association and genericode file scenario (Non-Normative)* illustrates the following three uses of the `<Context>` element with different XPath addresses (the documentation has been elided):

```
<Context address="@currencyID" values="currency" mark="money"
         metadata="cctsV2.01-amount">
...
<Context address="cac:BuyerCustomerParty//cbc:CountrySubentityCode"
         values="provinces states"
         metadata="cctsV2.01-code">
...
<Context address="cac:TaxCategory/cbc:ID"
         values="tax-ids" mark="money"
         metadata="cctsV2.01-identifier">
...
```

The first declaration uses a context-free address for all uses of the `currencyID=` attribute across the entire document instance. This is the most typical use of the `<Context>` element as most requirements constrain information items in all places where the item is used.

The second declaration uses the descendant scope to specify the constraints for all `<cbc:CountrySubentityCode>` elements that are descendants (including immediate children) of the `<cac:BuyerCustomerParty>` element. This is the next most typical use of the `<Context>` element as there may be some requirements for constraining values in only parts of a document.

The third declaration uses the child scope to specify a precise location address constraining the `cbc:ID` element only when it is an immediate child of the `cac:TaxCategory` element. Consider the following snippet of an instance where inappropriately using the context-free address `cbc:ID` would unnecessarily (and likely in error) associate `cac:TaxScheme/cbc:ID` with the same constraints as `cac:TaxCategory/cbc:ID`:

```
<cac:TaxSubTotal>
  <cbc:TaxableAmount currencyID="USD">100.00</cbc:TaxableAmount>
  <cbc:TaxAmount currencyID="USD">15.00</cbc:TaxAmount>
  <cac:TaxCategory>
    <cbc:ID>Z</cbc:ID>
    <cbc:Percent>15</cbc:Percent>
    <cac:TaxScheme>
      <cbc:ID>Provincial Tax</cbc:ID>
    </cac:TaxScheme>
  </cac:TaxCategory>
</cac:TaxSubTotal>
```

Two or more contexts' address expressions may all be valid for a single node in the source document, but only one will get engaged in the validation process. The order of `<Context>` elements for the context/value associations is interpreted such that the more important document contexts are found first and the less important document contexts follow. Thus, a given information item from the document can be associated with only a single document context declaration (the most important, that is the first one matched) in this context/value association file.

Each `<Context>` element may have any number of optional `<Message>` children that contain any content in a foreign namespace. Each message may have a `useUri=` attribute to characterize where an implementation would use the recognized message in the processing of value violations. At this time the specification has not reserved any such URI strings representing particular violation processing semantics but does reserve any URI string in an OASIS domain for future standardization. An application that does not recognize any of the supplied message characterizations shall use the first supplied message for reporting all violations. When a `<Message>` element is present with element or non-white-space text content, that content is meant to be used in place of any automatically-generated message an application might use when reporting a violation of the context/value constraint. An otherwise absent or empty `<Message>` element will signal to the application the permitted use of any automatically-generated message text.

Note: Supporting contexts in documents of different types (non-normative)

As an alternative to having separate expressions of context/value associations for instances of different document types, it is possible using an address that specifies a context relative to the instance's document element. This combines the associations in a single file for a subset of a multi-document context. An example of a context/value association file testing information items in two instances with different document types is as follows, where `@item-a` has the same constraints in instances of both document types, but `@item-b` has different constraints in instances of each document type:

```
<?xml version="1.0" encoding="UTF-8"?>
<cva:ValueListConstraints xmlns:cva=
"http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/"
  xmlns:in="urn:oasis:names:specification:ubl:schema:xsd:Invoice-1.0"
  xmlns:po="urn:oasis:names:specification:ubl:schema:xsd:Order-1.0"
  name="code-list-rules">
  <ValueLists>
    <ValueList xml:id="a1" uri="enumeration1.gc"/>
    <ValueList xml:id="a2" uri="enumeration2.gc"/>
    <ValueList xml:id="a3" uri="enumeration3.gc"/>
```

```
  </ValueLists>
  <Contexts>
    <Context address="@item-a" values="a2"/>
    <Context address="/po:Order//context-b1//@item-b"
             values="a1 a3"/>
    <Context address="/in:Invoice//context-b1/@item-b"
             values="a3" mark="characteristic-1"/>
  </Contexts>
</cva:ValueListConstraints>
```

Each `<Context>` element may have an optional `mark=` attribute that contains a single name token used to characterize the context. Downstream processes may choose to distinguish different constraint violations by their marked characterization.

3.1. Instance-level metadata

An information item in an XML document being created or tested may or may not have associated instance-level metadata that represents the list-level metadata of the enumeration from which an item's value is obtained. Instance-level metadata disambiguates the specification of a value that might be present in more than one enumeration. Without disambiguation, an application is unable to either specify or interpret the precise semantics represented by the value.

Note: Attribute confusion (non-normative)

Instance-level metadata is not to be confused with an attribute information item that contains the constrained value. For example the UN/CEFACT amount data type has a numeric value in an element and the constrained coded value is that element's currency identifier attribute. The instance-level metadata is the currency list's version attribute, not the currency attribute.

Note: Document design considerations (non-normative)

An information item needing to unambiguously identify the list-level metadata associated with its value must be able to specify corresponding instance-level metadata. Document designers prescribing the specification of information items should make provision for also specifying instance-level metadata associated with the information item's value.

Note: Strategic omission of instance-level metadata (non-normative)

There are strategies where specifying information items without instance-level metadata can provide long-term benefit. For values unexpected to change over the life cycle of revisions of an enumerated list, avoiding specifying instance-level metadata will allow the value specification to have validity across all revisions. Of course should the semantic of the value change at some point, legacy uses of the value become ambiguous.

A genericode file contains list-level metadata attributing properties to the enumeration as a whole. This is different than the value-level metadata that may be specified for each value in the enumeration. Instance-level metadata corresponds only to the list-level metadata, and not to any value-level metadata, found in the genericode file. All differing genericode files should have unique list-level metadata to unambiguously identify the enumeration.

A context/value association file provides the optional masquerade facility whereby the `<ValueList>` declaration of an external genericode file optionally includes either a `masqueradeUri=` attribute or a `<Identification>` child element or both. The descendants of this `<Identification>`

element correspond to the like-named descendants of the `<Identification>` element in the genericode file of values and the genericode file of masquerading list-level metadata. An application interpreting the context/value association of a document context with an external value must give precedence to any masquerading metadata over any external list-level metadata when considering any instance-level metadata associated with the value. The masquerading metadata in the CVA file has highest precedence. The masquerading metadata in the masquerading genericode file has lower precedence. The list-level metadata in the genericode file referenced with `uri=` has the lowest precedence.

Note: Strategic masquerading of list-level metadata (non-normative)

A strategy involving masquerading list-level metadata in support of instance-level metadata supports a public enumeration that does not yet have a value that is anticipated to be included at a future date. Using one's own enumeration containing the future value, and masquerading one's own necessarily-different metadata with the list-level metadata of the future enumeration, allows the association of the value with document contexts as if the value already is part of the future version of the enumeration. This approach supplants any need to prematurely modify the larger enumeration to include the future value.

The `<InstanceMetadata>` element children of the `<InstanceMetadataSet>` referenced by a context's `metadata=` attribute associate a document's instance-level metadata item (relative to the context item being tested) with a value list's corresponding list-level metadata item (relative to the `<Identification>` element found either as masquerading list-level metadata or as expressed inside the referenced genericode file). For XML document vocabularies without instance-level metadata, the `metadata=` attribute is omitted.

4. Context/value association XML vocabulary expression

The `http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/` namespace URI string represents the committee draft version of the XML vocabulary for context/value association files. When the work becomes a committee specification the namespace URI string will likely be `http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/`.

A set of associations from specified document contexts to referenced value lists is expressed in an XML instance using the context/value association XML vocabulary. The W3C Schema XSD file `ContextValueAssociation.xsd` expresses the document constraints of this XML vocabulary and references the `xml.xsd` expression of reserved XML attributes.

All normative W3C Schema XSD document fragments for this XML vocabulary can be obtained by dereferencing this XML namespace URI.

5. Conformance

Two measures of conformance are described by this specification: (1) the conformance of a standalone context/value association instance and its contents; and (2) the conformance of an application acting on a context/value association file to interpret the values permitted for document contexts.

Note: Conformance clauses in schema (non-normative)

These clauses of the specification are mechanically extracted from annotations

found inside the schema expression.

5.1. Document-level conformance

An XML instance conforms to the OASIS Context/value association file document model if it does not violate any document-level constraints described herein.

D1 [document:: vocabulary constraints]

A context/value association file must not violate the W3C Schema XSD constraints expressed in the `ContextValueAssociation.xsd` associated with this version of the XML vocabulary.

Applies to: <ValueListConstraints>

D2 [document:: include file URI resolution]

The document element of the document pointed to by this attribute must be `<ValueListConstraints>` in the same namespace as the pointing document.

Applies to: <Include uri="">

D3 [document:: key specification requirement]

The `key=` attribute is required when the code list referenced by `uri=` has more than one key.

Applies to: <ValueList key="">

D4 [document:: key specification value]

The `key=` attribute, when specified, must be the value of a key identifier in the genericode file referenced by `uri=`.

Applies to: <ValueList key="">

D5 [document:: instance-level metadata item addressing]

This query binding syntax address points to an instance-level metadata item from the perspective of the addressed context item.

Applies to: <InstanceMetadata address="">

D6 [document:: identification component addressing]

This query binding syntax address points to a descendant of `<Identification>` from the perspective of the `<Identification>` element.

Applies to: <InstanceMetadata identification="">

D7 [document:: instance metadata set reference constraint]

The token in the `metadata=` attribute points to an `<InstanceMetadataSet>` element.

Applies to: <Context metadata="">

D8 [document:: value reference constraint]

Each token in the `values=` attribute points to either a `<ValueTest>` element or a `<ValueList>` element.

Applies to: <Context values="">

D9 [document:: code list URI resolution]

The document element of the document pointed to by this attribute must be `<CodeList>` in an OASIS genericode namespace.

Applies to: <ValueList uri="">,<ValueList masqueradeUri="">

D10 [document:: absolute URI specification]

All absolute URI attributes shall not have a relative URI value.

Applies to: <Message useUri="">,<CanonicalUri>,<CanonicalVersionUri>

D11 [document:: location URI irrelevance]

Any measure of equivalence for a location URI is determined between users of this specification as this specification does not confer any identification semantics for these elements.

Applies to: <LocationUri>,<AlternateFormatLocationUri>

D12 [document:: addressing]

A query address shall use an expression of the query binding syntax to address or query elements or attributes. The query binding syntax is implied by the `queryBinding=` attribute on `<ValueListConstraints>`. Absent this attribute, there is no representation that the expression is in any particular syntax.

Applies to: <InstanceMetadata address="">,<InstanceMetadata identification="">,<Context address="">

D13 [document:: expression evaluation]

A query boolean expression shall use an expression of the query binding syntax that returns a boolean value of true or false. The query binding syntax is implied by the `queryBinding=` attribute on `<ValueListConstraints>`. Absent this attribute, there is no representation that the expression is in any particular syntax.

Applies to: <ValueTest test="">

D14 [document:: XML name lexical constraint]

A name token must lexically validate as would an un-prefixed element or attribute name.

Applies to: <ValueListConstraints name="">,<ValueListConstraints queryBinding="">, <ValueList key="">,<Context mark="">

5.2. Application-level conformance

An application conforms to the OASIS Context/value association file processing rules if it does not violate any application-level constraints described herein when processing a context/value association file.

A1 [application:: document-level constraints]

An application must report violations of document-level constraints.

Applies to: <ValueListConstraints>

A2 [application:: query binding syntax interpretation]

This identifies the governing query syntax used for addresses and expressions in the document. Absent this attribute an application is free to interpret the syntax in any manner.

Applies to: <ValueListConstraints queryBinding="">

A3 [application:: include file context priority]

The document contexts described in an including CVA file have higher detection priority than those found in an included CVA file. The document contexts described in an included CVA file have higher detection priority than those found in a previously-included CVA file. Expressed another way, the document contexts in the first included CVA file have the lowest detection priority.

Applies to: <Include uri="">

A4 [application:: identification element masquerade precedence]

Any element found in a CVA file's `<Identification>` element has precedence over any like-named element that may or may not be present in the maquerading genericode file's or referenced genericode file's `<Identification>` element.

Applies to: <Identification>

A5 [application:: masquerading file masquerade precedence]

Any element found in a masquerading genericode file's `<Identification>` element has precedence over any like-named element that may or may not be present in the referenced genericode file's `<Identification>` element, but lower precedence than any specified like-name element in the CVA file's `<Identification>` element.

Applies to: <ValueList masqueradeUri="">

A6 [application:: context specification precedence]

The contexts of a given CVA file are processed in descending priority based on order declared, such that an earlier described context has higher detection priority than those described after it (that is, the following sibling `<Context>` elements). Expressed another way, the document contexts within a given CVA file are processed in descending priority order of being declared.

Applies to: <Context>

A7 [application:: context constraint evaluation]

For a context's constraints to be considered fully satisfied, all of the referenced `<ValueTest>` expressions must evaluate to logical TRUE. At the same time the value must be present in any of the referenced `<ValueList>` value lists that have values and that qualify to be used in the presence of the given instance-level metadata described by any referenced `<InstanceMetadataSet>` sets. Referencing a `<ValueList>` that points to a value list without any values imposes no constraint on the context.

Applies to: <Context>

A8 [application:: message use identification]

An implementation that does not recognize any of the message use attributes shall use the first message element for value constraint violation reporting. An implementation that does

recognize any of the message use attributes can choose any recognized message element for value constraint violation reporting. When there are no messages of any kind, or the selected message is empty, the application can use any wording in value constraint violation reporting.

Applies to: <Message>

A9 [application:: message use identification]

A message use URI standardized in the CVA specification (if any) shall, if recognized by an implementation, only represent the semantics described by the specification and not any custom semantics chosen by an application.

Applies to: <Message useUri="">

A10 [application:: relative URI resolution]

All relative URI values in `uri=` attributes are resolved to the base URI of the `uri=` attribute, which is influenced by any ancestral `xml:base=` attributes that are present.

Applies to: <Include uri="">

A. Context/value association schemas

The context/value association file XML vocabulary schemas are obtained through the namespace URI:

`http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/`

B. Document contexts in XML instances (Non-Normative)

The different types of information items that are constrained by tests or described by code lists and value lists are typically declared in few places in document models but, because of document context, the actual instances of these information items are found in possibly very many places in actual documents. Each use of an information item is in a different document context. Document grammars that validate information items based solely on global declarations cannot distinguish the uses of the items in different document contexts and any desired differences in value association required by trading partners exchanging XML instances.

Document contexts are expressed structurally as hierarchical tree locations. Without confidence that the document contexts of the information items of an XML instance are sound, no amount of contextual association of item values is going to be reliable. It is, therefore, a suggested precondition in advance of using context/value association files to validate the XML instances against, when available, a schema expression of structural constraints for information item location and lexical structure. This is a critically important precondition because the schema constraints will confirm the document contexts of information items are correctly positioned in the XML instance document hierarchy, and are correctly formed in regard to their lexical structure. Only when the information items are known to be in correct contexts with their correct form will the value association of the document contexts reflect bona fide values.

1. Using XPath patterns to specify document context

A common query binding for addresses in CVA files is the XML Path Language called XPath[XPath 1.0][XPath 2.0].

The XSLT pattern concept [XSLT 1.0 Pattern][XSLT 2.0 Pattern] specifies the subset of XPath addresses specifying document contexts.

Patterns are used to address locations in an XML document according to a data model of processed syntax. This XPath data model differs from other data models such as the Document Object Model [DOM] in that the DOM models more aspects of raw syntax used in the document. Given that syntax is irrelevant (in that it is arbitrary to the creator of XML which syntactic choices are made when marking up documents) the XPath data model is sufficient to talk about the elements and attributes found in documents.

Elements are referred to in an XPath expression by their namespace-qualified names, while the "`@`" character (an abbreviation for the XPath `attribute::` axis) prefixes attributes referred to by their namespace-qualified names.

Note: Names without prefixes

The default use of XPath considers names without prefixes to always be in no namespace, and does not use the default namespace to qualify names without prefixes. For this reason, all namespace-qualified information items in an XML vocabulary being associated with values must be prefixed when being addressed in XPath, even if the instances of the document vocabulary utilize the default namespace.

The syntax of an XPath expression separates multiple location steps of a single location path using an oblique "`/`" character. Each step to the right names the child element or attached attribute of the immediately preceding step to the left which is always an element. Child elements are one level deeper in the hierarchical nesting than their parents. Elements are also parents of their attached attributes.

A fully-qualified absolute XPath location path begins with the oblique indicating the path starts from the root node (the parent of the document element) of the XPath data model document tree. A relative XPath location path starts with the name of an information item without the oblique at the beginning.

Examples of four absolute XPath location paths are as follows (note that arbitrary white-space is allowed between steps of an XPath address):

```
/po:Order/cac:TaxTotal/cbc:TaxAmount/@currencyID
/po:Order/cbc:DocumentCurrencyCode
/po:Order/cac:BuyerCustomerParty/cac:Party/cac:PostalAddress/
                                          cbc:CountrySubentityCode
/po:Order/cac:SellerSupplierParty/cac:Party/cac:PostalAddress/
                                           cbc:CountrySubentityCode
```

An example of a relative XPath location that matches all currency values in attributes in the entire instance of a document model is as follows, as the information item does not include any ancestral distinction to the left:

```
@currencyID
```

An example of a relative XPath location that matches all country sub-entity values in elements in the entire instance of a document model is as follows, as the information item does not include any ancestral distinction to the left:

```
cbc:CountrySubentityCode
```

The minimum XPath addresses needed to precisely distinguish, for example, the country sub-entity code of the party address of each of the buyer and seller are as follows, as the information item includes explicit ancestry to the left:

```
cac:BuyerCustomerParty/cac:Party/cac:PostalAddress/cbc:CountrySubentityCode
cac:SellerSupplierParty/cac:Party/cac:PostalAddress/cbc:CountrySubentityCode
```

Two examples of the "//" operator in XPath illustrate the matching within an entire sub-tree of the document hierarchy; the XPath addresses needed to distinguish all (not just in the party address) country sub-entity codes descendent to the buyer and the seller would be as follows indicating only the required (and possibly distant) ancestor:

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

2. Example uses of document contexts

When deciding on code list and value association, partners in information interchange must agree in which contexts particular sets of values need to be constrained.

Some business rules may require the same context to be specified across all document types, such as "All currency values must be Canadian or US dollars."

Other business rules may require indistinct document contexts to be specified, such as "all country sub-entity codes used in the order and in the invoice shall be valid states according to the United States postal service."

Note: Incomplete example

In fact constraining only the country sub-entity code could be quite misleading in a business environment, but it is used here for illustrative purposes. The code "WA" validly representing both Western Australia, Australia and Washington, United States of America, would be an example of the ambiguity problem.

Yet other business rules might require more distinct document contexts to be specified, such as "The country sub-entity codes for the seller can only be states of the United States, while country sub-entity codes for the buyer can be both provinces of Canada and states of the United States."

Partners engaged in information interchange must, therefore, take the step to agree on which XPath addresses will specify the contexts at which particular values are constrained. Examining the list of contexts in which constrained-value information items are found, the partners can identify as much specificity as is required to match those contexts in which the values are constrained.

C. Using association to restrict a schema-enumerated code list (Non-Normative)

When associating the values from a list that is described in a schema with an enumeration, the genericode file of values can at most have only those values found in the schema. Any superfluous values would never be properly associated because the schema validation precludes their use.

Partners involved in information interchange can pare down this complete list and prune unwanted values in a new genericode file leaving in only the values agreed to be used in XML instances. Creating the new list, however, obligates the user to utilize different list-level metadata in the new list than that found in the original list, as the new list does not in fact reflect the complete list as described by the original metadata.

An example of this is a genericode file based on the UN/CEFACT currency values that includes over 160 entries. A copy of this file is edited where the entire list of coded values has been pruned to only the Canadian dollar and the US dollar, and the metadata necessarily modified to

reflect the qualified list and not the complete list. An example file is as follows:

```
<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>
```

Note that instance-level metadata has values corresponding to the cited list's list-level metadata, thus defining the semantics of the values being used. For UN/CEFACT CCTS V2.01 unqualified data types for core component types, instance-level metadata corresponds to the attributes expressing the corresponding supplementary components. A complete expression of the correspondences of these supplementary component metadata attributes to genericode list-level metadata elements is as follows:

```
    <InstanceMetadataSet xml:id="cctsV2.01-amount">
      <InstanceMetadata address="../@currencyCodeListVersionID"
                        identification="Version"/>
    </InstanceMetadataSet>
    <InstanceMetadataSet xml:id="cctsV2.01-measure">
      <InstanceMetadata address="../@unitCodeListVersionID"
                        identification="Version"/>
    </InstanceMetadataSet>
    <InstanceMetadataSet xml:id="cctsV2.01-quantity">
      <InstanceMetadata address="../@unitCodeListID"
                        identification="Version"/>
      <InstanceMetadata address="../@unitCodeListAgencyName"
                        identification="Agency/LongName"/>
      <InstanceMetadata address="../@unitCodeListAgencyID"
                        identification="Agency/Identifier"/>
    </InstanceMetadataSet>
    <InstanceMetadataSet xml:id="cctsV2.01-code">
      <InstanceMetadata address="@listName"
                        identification="LongName[not(@Identifier='listID')]"/>
```

```
      <InstanceMetadata address="@listID"
                        identification="LongName[@Identifier='listID']"/>
      <InstanceMetadata address="@listVersionID"
                        identification="Version"/>
      <InstanceMetadata address="@listSchemeURI"
                        identification="CanonicalUri"/>
      <InstanceMetadata address="@listURI"
                        identification="LocationUri"/>
      <InstanceMetadata address="@listAgencyName"
                        identification="Agency/LongName"/>
      <InstanceMetadata address="@listAgencyID"
                        identification="Agency/Identifier"/>
    </InstanceMetadataSet>
    <InstanceMetadataSet xml:id="cctsV2.01-identifier">
      <InstanceMetadata address="@schemeName"
                        identification="LongName"/>
      <InstanceMetadata address="@schemeVersionID"
                        identification="Version"/>
      <InstanceMetadata address="@schemeURI"
                        identification="CanonicalUri"/>
      <InstanceMetadata address="@schemeDataURI"
                        identification="LocationUri"/>
      <InstanceMetadata address="@schemeAgencyName"
                        identification="Agency/LongName"/>
      <InstanceMetadata address="@schemeAgencyID"
                        identification="Agency/Identifier"/>
    </InstanceMetadataSet>
```

Consider the following instance fragment that specifies version metadata for the currency attribute:

```
<cac:TaxTotal>
  <cbc:TaxAmount currencyCodeListVersionID="2001"
                 currencyID="USD">15.00</cbc:TaxAmount>
  <cbc:TaxEvidenceIndicator>false</cbc:TaxEvidenceIndicator>
  <cac:TaxSubTotal>
```

This instance would validate with the original genericode file but, as is, would not validate with the restricted code list above because the version information in the restricted code list list-level metadata, amongst other metadata, doesn't match the original list.

The context/value association file in Appendix E, *Example context/value association and genericode file scenario (Non-Normative)* accommodates this with the following declaration of the use of the restricted code list, while explicitly masquerading the restricted list of values as being that from the bona fide complete list of values, while at the same time overriding one of those bona fide list values (such overriding of bona fide list values is not a suggested practice, but it is included here for illustrative purposes):

```
<ValueList xml:id="currency" uri="CAUS_CurrencyCode.gc"
           masqueradeUri="UBL_CurrencyCode-2.0.gc">
  <Annotation>
    <Description>
      <x:p>Restricted to only Canadian and US dollars.</x:p>
    </Description>
  </Annotation>
  <Identification>
    <LongName>ISO Currency List</LongName>
  </Identification>
</ValueList>
```

By masquerading the metadata, the code list genericode file properly includes the unique metadata of the derived set of codes, while the association properly matches against the original list's metadata values. There is, of course, a risk that one could accidentally or maliciously improperly purport a derived list to be from a different list and cause instances to pass value validation without an error. Note that the `masqueradeUri=` and `<Identification>` items are both optional, with a precedence described in detail in Section 3, “Specifying value constraints in document contexts”. For example, one could choose to use only `<Identification>` should a genericode file for the original list not be available. However, where genericode files are available

for both the original list and the derived list, only the attributes need be used.

D. Using association to extend a code list (Non-Normative)

The context/value association file in Appendix E, *Example context/value association and genericode file scenario (Non-Normative)* illustrates the extension of the codes permitted for an information item, in this case for payment means in a document.

Two code lists for payment means are declared: one with the identifier "`payments`", pointing to an original list of agreed-upon payment means for a community, and the other with the identifier "`additional_payments`", pointing to a set of values meant to augment the list of available values.

The following declaration associates the members of both lists with the document-wide context of `<cbc:PaymentMeansCode>` elements:

```
    <Context address="cbc:PaymentMeansCode" mark="money"
             values="payments additional_payments"
             metadata="cctsV2.01-code">
      <Annotation>
        <Description>
      <x:p>The payments can be by either standard or supplemental means.</x:p>
        </Description>
      </Annotation>
    </Context>
```

E. Example context/value association and genericode file scenario (Non-Normative)

In this scenario two trading partners are going to interchange documents between buyer and seller parties. At a structural level, they are using publicly-available W3C Schema expressions for the structural integrity of their XML instances.

At a business level, the trading partners have agreed that all currencies used in an instance can be only Canadian or US dollars. The `CAUS_CurrencyCode.gc` file documented above expresses this limited number of coded values.

The partners have also agreed that the buyer's country sub-entity codes may be either a US state or a Canadian province, but that the seller's country sub-entity codes may only be a US state. The two genericode files `US_CountrySubentityCode.gc` and `CA_CountrySubentityCode.gc` each represent respectively state and province name abbreviations.

An excerpt from the `CA_CountrySubentityCode.gc` file created by hand to represent the list of Canadian provinces and territories after April 1, 1999 reads as:

```
<gc:CodeList
xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/">
   <Identification>
      <ShortName>provinces</ShortName>
      <LongName>Canadian Provinces</LongName>
      <Version>2</Version>
      ...
   </Identification>
   ...
   <SimpleCodeList>
      <Row>
         <Value ColumnRef="code">
            <SimpleValue>AB</SimpleValue>
         </Value>
         <Value ColumnRef="name">
```

```
            <SimpleValue>Alberta</SimpleValue>
         </Value>
      </Row>
      <Row>
         <Value ColumnRef="code">
            <SimpleValue>BC</SimpleValue>
         </Value>
         <Value ColumnRef="name">
            <SimpleValue>British Columbia</SimpleValue>
         </Value>
      </Row>
      ...
```

An excerpt from the `US_CountrySubentityCode.gc` file created by hand to represent US states reads as:

```
<gc:CodeList
xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/">
   <Identification>
      <ShortName>states</ShortName>
      <LongName>US States</LongName>
      <Version>1</Version>
      ...
   </Identification>
   ...
   <SimpleCodeList>
      <Row>
         <Value ColumnRef="code">
            <SimpleValue>AL</SimpleValue>
         </Value>
         <Value ColumnRef="name">
            <SimpleValue>ALABAMA</SimpleValue>
         </Value>
      </Row>
      <Row>
         <Value ColumnRef="code">
            <SimpleValue>AK</SimpleValue>
         </Value>
         <Value ColumnRef="name">
            <SimpleValue>ALASKA</SimpleValue>
         </Value>
      </Row>
      ...
```

Using the `TaxIdentifier.gc` set of identifiers, the instances can make reference to appropriate taxes.

Finally, the trading partners have agreed to use both the complete set of payment means agreed on by the community, extended by an additional payment means expressed in `Additional_PaymentMeansCode.gc`.

These genericode files and context/value association file together form a formal and unambiguous expression of the contextual coded value constraints that go beyond the constraints found in schema expressions. The package of these files can, therefore, be included in a contractual agreement between the trading partners. Each trading partner can then use the formal expression of context/value associations in data entry, validation or other processes responsible for writing or confirming specified values in instances.

1. Example context/value association file

The following complete example of a context/value association file constrained by this specification and is similar to that used in the example scenario. The example uses ISO/IEC 19757-3 Schematron [Schematron], an assertion-based schema language used to formally express the constraints and co-occurrence constraints on information items in XML documents, in the message content:

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

```
<?xml-stylesheet type="text/xsl" href="Crane-cva2html.xsl"?>
<cva:ValueListConstraints
  xmlns:cva=
         "http://docs.oasis-open.org/codelist/ns/ContextValueAssociation/1.0/"
  xmlns:cbc="urn:oasis:names:draft:ubl:schema:xsd:CommonBasicComponents-2"
  xmlns:cac="urn:oasis:names:draft:ubl:schema:xsd:CommonAggregateComponents-2"
  xmlns:x="http://www.w3.org/TR/REC-html40"
  xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  id="urn:x-illustration"
  name="code-list-rules"
  version=
"$ Id: order-constraints-doc.cva,v 1.18 2009/07/12 01:09:31 gkholman Exp $">
  <Annotation>
    <Description>
      <x:p>
        This is an illustrative example of all of the features of specifying
        the context/value constraints that one can express for XML documents.
      </x:p>
      <x:p>
        The validation requirements for this contrived scenario are as follows:
        <x:ul>
          <x:li>the UN/CEFACT currency code list is restricted to be
          only Canadian and US dollars:</x:li>
          <x:li>the seller must be in the US</x:li>
          <x:li>the buyer may be in either Canada or the US</x:li>
          <x:li>the definition for Payment Means is extended to include
          both UBL definitions and additional definitions</x:li>
        </x:ul>
      </x:p>
    </Description>
  </Annotation>

  <Title>
    Illustration of code list constraints -
    <x:samp>order-constraints.cva</x:samp>
  </Title>

  <!--list all test expressions-->
  <ValueTests>
    <ValueTest xml:id="length-35" test="string-length(.)&lt;=35">
      <Annotation>
        <Description>
          <x:p>Certain fields are restricted to 35 characters in length.</x:p>
        </Description>
      </Annotation>
    </ValueTest>
  </ValueTests>

  <!--list all of the genericode expressions of agreed-upon code list
      value enumerations-->
  <ValueLists>
    <ValueList xml:id="currency" uri="CAUS_CurrencyCode.gc"
               masqueradeUri="UBL_CurrencyCode-2.0.gc">
      <Annotation>
        <Description>
          <x:p>Restricted to only Canadian and US dollars.</x:p>
        </Description>
      </Annotation>
      <Identification>
        <LongName>ISO Currency List</LongName>
      </Identification>
    </ValueList>
    <ValueList xml:id="states" uri="US_CountrySubentityCode.gc">
      <Annotation>
        <Description>
          <x:p>List of US states.</x:p>
        </Description>
      </Annotation>
    </ValueList>
    <ValueList xml:id="provinces" uri="CA_CountrySubentityCode.gc">
      <Annotation>
        <Description>
          <x:p>List of Canadian provinces</x:p>
        </Description>
      </Annotation>
```

```
  </ValueList>
  <ValueList xml:id="tax-ids" uri="TaxIdentifier.gc" key="codeKey">
    <Annotation>
      <Description>
        <x:p>List of tax type identifiers</x:p>
      </Description>
    </Annotation>
  </ValueList>
  <ValueList xml:id="payments" uri="UBL_PaymentMeansCode-2.0.gc">
    <Annotation>
      <Description>
        <x:p>
          Copied from the UBL 2.0 suite:
          <x:a href="http://docs.oasis-open.org/ubl/cs-UBL-2.0/">
            <x:samp>http://docs.oasis-open.org/ubl/cs-UBL-2.0/</x:samp>
          </x:a>
        </x:p>
      </Description>
    </Annotation>
  </ValueList>
  <ValueList xml:id="additional_payments"
             uri="Additional_PaymentMeansCode.gc">
    <Annotation>
      <Description>
        <x:p>An extra set of possible payment means.</x:p>
      </Description>
    </Annotation>
  </ValueList>
</ValueLists>
<!--list all of the instance-level metadata components associated with
    genericode <Identification> components-->
<InstanceMetadataSets>
  <Annotation>
    <Description>
      <x:p>UN/CEFACT CCTS V2.01 supplementary components to genericode</x:p>
    </Description>
    </Annotation>
  <InstanceMetadataSet xml:id="cctsV2.01-amount">
    <Annotation>
      <Description>
        <x:p>CCTS 2.01 AmountType instance metadata</x:p>
      </Description>
    </Annotation>
    <InstanceMetadata address="../@currencyCodeListVersionID"
                      identification="Version"/>
  </InstanceMetadataSet>
  <InstanceMetadataSet xml:id="cctsV2.01-measure">
    <Annotation>
      <Description>
        <x:p>CCTS 2.01 MeasureType instance metadata</x:p>
      </Description>
    </Annotation>
    <InstanceMetadata address="../@unitCodeListVersionID"
                      identification="Version"/>
  </InstanceMetadataSet>
  <InstanceMetadataSet xml:id="cctsV2.01-quantity">
    <Annotation>
      <Description>
        <x:p>CCTS 2.01 QuantityType instance metadata</x:p>
      </Description>
    </Annotation>
    <InstanceMetadata address="../@unitCodeListID"
                      identification="Version"/>
    <InstanceMetadata address="../@unitCodeListAgencyName"
                      identification="Agency/LongName"/>
    <InstanceMetadata address="../@unitCodeListAgencyID"
                      identification="Agency/Identifier"/>
  </InstanceMetadataSet>
  <InstanceMetadataSet xml:id="cctsV2.01-code">
    <Annotation>
      <Description>
        <x:p>CCTS 2.01 CodeType instance metadata</x:p>
      </Description>
    </Annotation>
    <InstanceMetadata address="@listName"
```

```
                        identification="LongName[not(@Identifier='listID')]"/>
      <InstanceMetadata address="@listID"
                        identification="LongName[@Identifier='listID']"/>
      <InstanceMetadata address="@listVersionID"
                        identification="Version"/>
      <InstanceMetadata address="@listSchemeURI"
                        identification="CanonicalUri"/>
      <InstanceMetadata address="@listURI"
                        identification="LocationUri"/>
      <InstanceMetadata address="@listAgencyName"
                        identification="Agency/LongName"/>
      <InstanceMetadata address="@listAgencyID"
                        identification="Agency/Identifier"/>
    </InstanceMetadataSet>
    <InstanceMetadataSet xml:id="cctsV2.01-identifier">
      <Annotation>
        <Description>
          <x:p>CCTS 2.01 IdentifierType instance metadata</x:p>
        </Description>
      </Annotation>
      <InstanceMetadata address="@schemeName"
                        identification="LongName"/>
      <InstanceMetadata address="@schemeVersionID"
                        identification="Version"/>
      <InstanceMetadata address="@schemeURI"
                        identification="CanonicalUri"/>
      <InstanceMetadata address="@schemeDataURI"
                        identification="LocationUri"/>
      <InstanceMetadata address="@schemeAgencyName"
                        identification="Agency/LongName"/>
      <InstanceMetadata address="@schemeAgencyID"
                        identification="Agency/Identifier"/>
    </InstanceMetadataSet>
  </InstanceMetadataSets>

  <!--list all of the contexts in which the value enumerations are used;
      where two or more contexts might match a given node in the input,
      list them here in order of most-important to least important match-->
  <Contexts>
    <Context address="@currencyID" values="currency" mark="money"
             metadata="cctsV2.01-amount">
      <Annotation>
        <Description>
      <x:p>All currencies are restricted to only Canadian and US dollars.</x:p>
        </Description>
      </Annotation>
    </Context>
    <Context address="cac:BuyerCustomerParty//cbc:CountrySubentityCode"
             values="provinces states"
             metadata="cctsV2.01-code">
      <Annotation>
        <Description>
          <x:p>The buyer can be in either Canada or the US.</x:p>
        </Description>
      </Annotation>
      <Message>Invalid province or state '<sch:value-of select="."/>' for
buyer "<sch:value-of select="ancestor::cac:BuyerCustomerParty/cac:Party/
cac:PartyName/cbc:Name"/>"</Message>
    </Context>
    <Context address="cac:SellerSupplierParty//cbc:CountrySubentityCode"
             values="states"
             metadata="cctsV2.01-code">
      <Annotation>
        <Description>
          <x:p>The seller can only be in the US.</x:p>
        </Description>
      </Annotation>
      <Message>Invalid state '<sch:value-of select="."/>' for seller
"<sch:value-of select="ancestor::cac:SellerSupplierParty/cac:Party/
cac:PartyName/cbc:Name"/>"</Message>
    </Context>
    <Context address="cac:TaxCategory/cbc:ID"
             values="tax-ids" mark="money"
             metadata="cctsV2.01-identifier">
      <Annotation>
```

```
        <Description>
          <x:p>Limit the recognized tax identifiers</x:p>
        </Description>
      </Annotation>
    </Context>
    <Context address="cbc:PaymentMeansCode" mark="money"
             values="payments additional_payments"
             metadata="cctsV2.01-code">
      <Annotation>
        <Description>
      <x:p>The payments can be by either standard or supplemental means.</x:p>
        </Description>
      </Annotation>
    </Context>
    <Context address="cbc:Name" values="length-35">
      <Annotation>
        <Description>
          <x:p>Names are not allowed to be too long.</x:p>
        </Description>
      </Annotation>
    </Context>
    <Context address="cbc:StreetName" values="length-35">
      <Annotation>
        <Description>
          <x:p>Addresses are not allowed to be too long.</x:p>
        </Description>
      </Annotation>
    </Context>
  </Contexts>
</cva:ValueListConstraints>
```

F. Acknowledgments (Non-Normative)

The following individuals have participated in the creation of this specification and are gratefully acknowledged.

Participants:

- Jon Bosak, Sun Microsystems
- Anthony Coates, Miley Watts LLP
- Jim Harris, National Center for State Courts
- G. Ken Holman, Associate Member
- Paul Spencer, Associate Member