

**From:** Jim Harris  
**To:** OASIS Electronic Court Filing Technical Committee (ECF TC)  
**Subject:** Code Lists in Court Policy  
**Date:** December 21, 2007

---

This memo summarizes my analysis of the issues with specifying context restrictions relating to code lists in machine readable court policy.

## ***Requirement***

The code list requirement communicated by Shane Durham was the need for context restrictions relating to codes. For example, there may be a set of party type codes that are only valid when filing a specific case type. Another example may be filing type codes that are valid only with new-case filings versus those valid with existing-case filings. More detail is provided in the memo from Shane dated Nov 10, 2006 (<http://www.oasis-open.org/apps/org/workgroup/legalxml-courtfilling/email/archives/200611/bin00000.bin>).

## ***Shane's Proposal***

Shane has proposed a relatively simple method of expressing code-set hierarchies and schema enhancements to allow each code to express zero or more implementation-specific context restrictions. His approach is detailed in the aforementioned Nov 10, 2006 memo.

In summary, the proposal involves allowing nested code lists and allowing any number of context restrictions to be associated with a given code value. Each context restriction would consist of a name/value pair that would be completely implementation specific. Note, the nested code list would itself be a context restriction that specifies which code values are valid for a code list nested within a particular value from another code list (in other words a more convenient way of specifying this particular type of context restriction).

## ***Observations***

While the proposed changes would address the specific requirement communicated, there is some concern that it may be somewhat limiting and may not provide flexibility that may be needed to enable specifying more complex business rules relating to code lists. Indeed, the whole issue of how to specify business rules (whether relating to code lists or not) is an area of concern that isn't addressed by the current ECF specification beyond rules inherent within the architecture provided by the MDE's and operational messages. It is left to each implementation to deal with context restrictions and more complex business rules through extensions, constraint schemas, or hard-coded functionality within affected MDE's. Shane's approach addresses this to a limited degree for context restrictions relating to code lists.

## ***CLR TC Specifications (genericode & CVA)***

At the August 2007 face-to-face meeting of the ECF TC, Ken Holman of the Code List Representation TC presented details relating to the genericode specification and a proposed

Schematron based methodology for context/value associations. Ken's presentation was not well received by the ECF TC as it appeared to be a more complicated approach to dealing with code lists than what the TC perceived was needed. I have since concluded that the distinction between the genericcode specification and the validation methodology was not clear in Ken's presentation. Hence, confusion and reluctance on the part of members of the TC to consider genericcode for representing code lists in ECF.

Genericcode is a specification for expressing simple code lists and is not tied to any particular method for validation. Indeed, genericcode itself doesn't specifically provide for nested code lists (which itself is just a specific form of context validation). Here is a simple genericcode example specifying A, B, and C values for party types (CasePersonRoleTypeCode):

```
<Identification>
  <ShortName>CasePersonRoleTypeCode</ShortName>
</Identification>
<ColumnSet>
  <Column Id="code" Use="required">
    <ShortName>Code</ShortName>
    <Data Type="normalizedString"/>
  </Column>
  <Column Id="name" Use="optional">
    <ShortName>Name</ShortName>
    <Data Type="string"/>
  </Column>
  <Key Id="codeKey">
    <ShortName>CodeKey</ShortName>
    <ColumnRef Ref="code"/>
  </Key>
</ColumnSet>
<SimpleCodeList>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>A</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>Party Type A</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>B</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>Party Type B</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>C</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>Party Type C</SimpleValue>
    </Value>
  </Row>
</SimpleCodeList>
```

Note, any number of columns may be specified to dynamically identify the codes, their names, and any other metadata required for a given implementation. Column references within the actual code list are optional and inferred based on position when not explicitly referenced. Also note that columns may be specified as optional. There are also provisions for multiple keys as well as XHTML for human-readable annotations or images. As you can see, genericcode

provides a relatively simple yet flexible approach to expressing code lists. The current version of the genericode specification may be downloaded from <http://www.oasis-open.org/committees/download.php/26153/oasis-code-list-representation-genericode.pdf>.

Context/value association (i.e. validation) is a separate specification that is currently under development by the CLR TC. The initial work on this actually started with the UBL TC and included an implementation methodology using Schematron. Since Ken's presentation in August, the CLR TC has decided to remove the implementation methodology and is working on a context/value association specification (CVA) independent of implementation methodologies. The CVA specification will provide a standardized file structure expressing the association between elements in contexts (using XPath patterns), sets of genericode files associated with those contexts (using URI locations), and optionally sets of masquerading metadata values (used to override genericode list-level metadata with anticipated instance-level meta data values). The current working draft of the CVA specification may be downloaded from <http://www.oasis-open.org/committees/download.php/26175/context-value-association-0.2-D1-20071121-0250z.zip>.

## **Options**

Given the above explanations, there are several options the TC may consider to address the issue of code lists and context restrictions. First of all, we need to look at this as two separate and distinct issues: 1] Consider use of genericode to express code lists, and 2] Decide how to address the need for context restrictions. With these 2 questions in mind, here are some options the TC may want to consider:

Option 1: Do nothing. Leave code list specifications as they are now and explore other options in a future ECF release. This will require implementers to deal with the inability to express context restrictions (using extensions, constraint schemas, etc).

Option 2: Incorporate Shane's proposed changes into ECF 4. This would certainly address the requirement communicated by Shane and would provide for what is essentially free-form implementation-specific context restrictions to be associated with specific code values. This may not, however, provide long-term flexibility to support anticipated need for expressing more complex business rules. Genericode and CVA could still be considered for future versions of ECF, but backward compatibility may be problematic.

Option 3: Adopt genericode for specifying code lists. This would establish a standardized approach embedded in other standards (e.g., UBL) for expressing code lists and would position us for incorporating the CVA approach (or other validation methodologies that support genericode files) for context restrictions in the future. Though genericode does not specifically support nested code lists or context restrictions, features of genericode do allow specifying additional columns associated for each code list that may be used to effect the hierarchies and context restrictions identified by Shane. This is discussed further in the "Recommendations" section below.

There are various hybrids of options 2 and 3 (using genericode while also incorporating changes suggested by Shane) the TC may consider. However, this would likely result in inconsistencies and may not be necessary if needs identified by Shane can be accomplished using features of genericode.

## Recommendation

Jim Cabral and I are recommending Option 3 (adopt genericode for all code lists). We came to this conclusion after making the distinction between the relatively clear-cut genericode specification and the more complex validation methodologies (which we are not ready for). After reviewing the genericode specification, we feel the approach is flexible yet straightforward and better positions us for eventual support of validation methodologies and other schemes for expressing and dealing with business rules.

We also feel the requirements communicated by Shane can be addressed using features of genericode that allow for additional columns and keys for each code set. Genericode uses a tabular structure for code list information. Each row represents a distinct entry in the code list and each column represents a metadata value associated with a distinct entry in the code list. Additional columns may be added as needed to express implementation-specific context restrictions.

The three tables below illustrate the case types, party types (roles), and filing types from Shane's example (see aforementioned memo). The "Code" and "Description" columns would be specified for all code lists in the ECF spec. However, in this example, we've added "Case Type" and "Restrictions" columns to both the party type code list and the filing type code list. Note, this may result in the code list containing multiple instances of the same code value. If that is intended to be allowed, the code and case type columns may be combined to define a compound key (rather than just the code being the key). The optional "Restrictions" columns are added as freeform text elements used for implementation-specific strings. Genericode files for each of the three tables are shown following the tabular tables.

Example Case Types Code List:

Code	Description	Restrictions (optional)
Civil - Probate	Probate	
Civil - General	General Civil	

Example Party Type Code List:

Code	Description	Case Type	Restrictions (optional)
A	Party Type A	Civil - Probate	PeerPartyTypesWhenNewCaseFiling=B
B	Party Type B	Civil - Probate	PeerPartyTypesWhenNewCaseFiling=A
C	Party Type C	Civil - Probate	
X	Party Type X	Civil - General	
Y	Party Type Y	Civil - General	
Z	Party Type Z	Civil - General	

### Example Filing Type Code List:

Code	Description	Case Type	Restrictions (optional)
1	Filing Type 1	Civil - Probate	CanSubmitWithNewCaseFilings=True CanSubmitWithExistingCaseFilings=False
2	Filing Type 2	Civil - Probate	CanSubmitWithNewCaseFilings=False CanSubmitWithExistingCaseFilings=True
3	Filing Type 3	Civil - Probate	
4	Filing Type 4	Civil - Probate	ForCourtUseOnly=True CanSubmitWithNewCaseFilings=False CanSubmitWithExistingCaseFilings=False
3	Filing Type 3	Civil - General	
4	Filing Type 4	Civil - General	ForCourtUseOnly=True CanSubmitWithNewCaseFilings=False CanSubmitWithExistingCaseFilings=False
5	Filing Type 5	Civil - General	

### Genericcode file for example Case Type Code List:

```

<Identification>
  <ShortName>CaseTypeCodes</ShortName>
</Identification>
<ColumnSet>
  <Column Id="code" Use="required">
    <ShortName>Code</ShortName>
    <Data Type="normalizedString"/>
  </Column>
  <Column Id="desc" Use="optional">
    <ShortName>Description</ShortName>
    <Data Type="string"/>
  </Column>
  <Column Id="restrictions" Use="optional">
    <ShortName>Restrictions</ShortName>
    <Data Type="string"/>
  </Column>
  <Key Id="codeKey">
    <ShortName>CodeKey</ShortName>
    <ColumnRef Ref="code"/>
  </Key>
</ColumnSet>
<SimpleCodeList>
  <Row>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
    <Value><SimpleValue>Probate</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
    <Value><SimpleValue>General Civil</SimpleValue></Value>
  </Row>
</SimpleCodeList>

```

## Genericcode file for example Party Type Code List:

```

<Identification>
  <ShortName>PartyTypeCodes</ShortName>
</Identification>
<ColumnSet>
  <Column Id="code" Use="required">
    <ShortName>Code</ShortName>
    <Data Type="normalizedString"/>
  </Column>
  <Column Id="desc" Use="optional">
    <ShortName>Description</ShortName>
    <Data Type="string"/>
  </Column>
  <Column Id="casetype" Use="required">
    <ShortName>CaseType</ShortName>
    <Data Type="string"/>
  </Column>
  <Column Id="restrictions" Use="optional">
    <ShortName>Restrictions</ShortName>
    <Data Type="string"/>
  </Column>
  <Key Id="codeKey">
    <Annotation>
      <Description>This list assumes that a given party type code may be
used with one or more case types, hence this compound key.</Description>
    </Annotation>
    <ShortName>CodeKey</ShortName>
    <ColumnRef Ref="code"/>
    <ColumnRef Ref="casetype"/>
  </Key>
</ColumnSet>
<SimpleCodeList>
  <Row>
    <Value><SimpleValue>A</SimpleValue></Value>
    <Value><SimpleValue>Party Type A</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
    <Value><SimpleValue>PeerPartyTypesWhenNewCaseFiling=B</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>B</SimpleValue></Value>
    <Value><SimpleValue>Party Type B</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
    <Value><SimpleValue>PeerPartyTypesWhenNewCaseFiling=A</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>C</SimpleValue></Value>
    <Value><SimpleValue>Party Type C</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>X</SimpleValue></Value>
    <Value><SimpleValue>Party Type X</SimpleValue></Value>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>Y</SimpleValue></Value>
    <Value><SimpleValue>Party Type Y</SimpleValue></Value>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>Z</SimpleValue></Value>
    <Value><SimpleValue>Party Type Z</SimpleValue></Value>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
  </Row>
</SimpleCodeList>

```

## Genericcode file for example Filing Type Code List:

```

<Identification>
  <ShortName>FilingTypeCodes</ShortName>
</Identification>
<ColumnSet>
  <Column Id="code" Use="required">
    <ShortName>Code</ShortName>
    <Data Type="normalizedString"/>
  </Column>
  <Column Id="desc" Use="optional">
    <ShortName>Description</ShortName>
    <Data Type="string"/>
  </Column>
  <Column Id="casetype" Use="required">
    <ShortName>CaseType</ShortName>
    <Data Type="string"/>
  </Column>
  <Column Id="restrictions" Use="optional">
    <ShortName>Restrictions</ShortName>
    <Data Type="string"/>
  </Column>
  <Key Id="codeKey">
    <Annotation>
      <Description>This list assumes that a given filing type code may be used
with one or more case types, hence this compound key.</Description>
    </Annotation>
    <ShortName>CodeKey</ShortName>
    <ColumnRef Ref="code"/>
    <ColumnRef Ref="casetype"/>
  </Key>
</ColumnSet>
<SimpleCodeList>
  <Row>
    <Value><SimpleValue>1</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 1</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
    <Value><SimpleValue>CanSubmitWithNewCaseFilings=True
CanSubmitWithExistingCaseFilings=False</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>2</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 2</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
    <Value><SimpleValue>CanSubmitWithNewCaseFilings=False
CanSubmitWithExistingCaseFilings=True</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>3</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 3</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>4</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 4</SimpleValue></Value>
    <Value><SimpleValue>Civil - Probate</SimpleValue></Value>
    <Value><SimpleValue>ForCourtUseOnly=True CanSubmitWithNewCaseFilings=False
CanSubmitWithExistingCaseFilings=False</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>3</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 3</SimpleValue></Value>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>4</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 4</SimpleValue></Value>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
    <Value><SimpleValue>ForCourtUseOnly=True CanSubmitWithNewCaseFilings=False
CanSubmitWithExistingCaseFilings=False</SimpleValue></Value>
  </Row>
  <Row>
    <Value><SimpleValue>5</SimpleValue></Value>
    <Value><SimpleValue>Filing Type 5</SimpleValue></Value>
    <Value><SimpleValue>Civil - General</SimpleValue></Value>
  </Row>
</SimpleCodeList>

```