HL7 Vocabulary Maintenance Language

Version 0.5

1 Introduction

This document describes a formal, XML based language that can be used to create and maintain the HL7 Version 3 reference vocabulary. The immediate purpose of this language is to provide vocabulary facilitators a consistent and rigorous mechanism that can be used to specify HL7 vocabulary additions and updates. This mechanism will be supplemented with graphical maintenance tools that may partially or completely replace this language as the primary form of input. As this occurs, it is anticipated that the language specified here will evolve a form that will be used to record and transmit vocabulary changes between systems and tools.

This specification takes a different approach to vocabulary maintenance than that of its Excel-based predecessor. One of the major differences is that this language is *procedural* vs. table-driven. A vocabulary maintenance description consists of a sequential list of parameterized function calls such as `RegisterCodeSystem`, `AddCodes`, `CreateValueSet`, etc.

A second major difference between this approach and its predecessor is the underlying model. The Excel-based maintenance system blurred the distinction between value sets, concept codes and vocabulary domains. The approach taken in this document is to separate the maintenance task into three separate parts:

1) Code Systems – A code system contains a set of unique *concept codes*. Each *concept code* serves as a token to represent a useful category or class as viewed from a particular perspective. The definition and organization of the tokens within a code system represents assertions about the organization of the corresponding categories and classes within a real world. A code system may also carry information about the various ways that the categories or classes are identified in different situations and languages, as well as additional defining and identifying information that serves to clarify the intended meaning of the tokens

2) Value Sets – A value set represents a list of concept codes. Value sets are used to specify a set of possible values for one or more RIM-derived coded attributes.

3) Vocabulary Domains – A vocabulary domain represents an abstract conceptual space that can be associated with RIM-derived coded attributes. A vocabulary domain can be represented by one or more value sets, where each associated value set applies in a given context.

Each of the above parts is maintained separately, with the revisions to the code system(s) occurring first followed by changes to the value sets followed by any revisions to vocabulary domain/value set associations that might be necessary.

2 Vocabulary Revisions

A vocabulary revision represents a related collection of updates to one or more code systems, value sets and/or vocabulary domains. Each vocabulary revision comes from a single HL7 committee and is represented by one primary contact person. The vocabulary revision element has to be the outermost (first) element in any vocabulary submission.

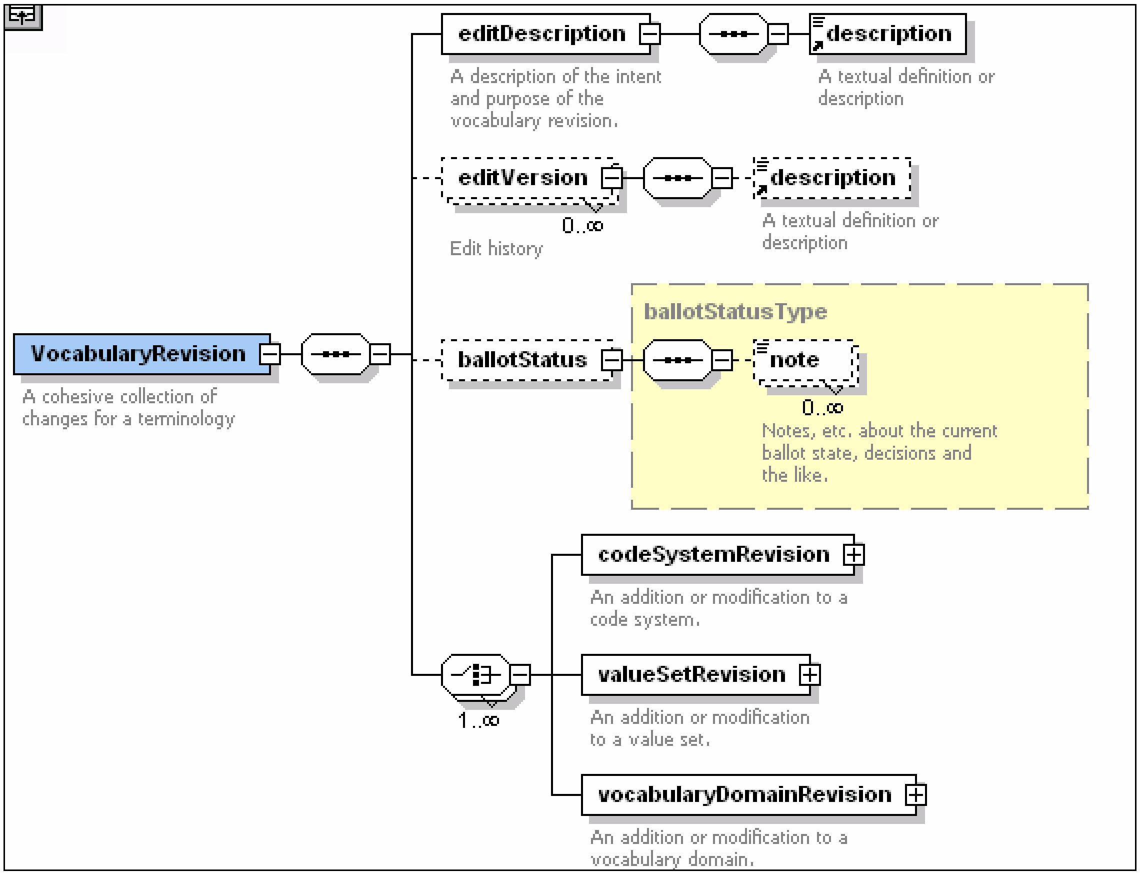
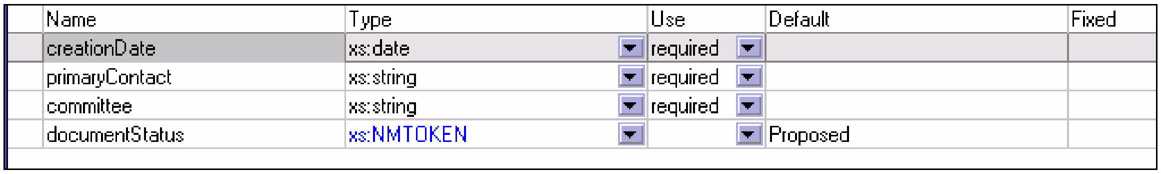

editDescription
A description of the intent and purpose of the vocabulary revision.
description
A textual definition or description
editVersion
0..∞
Edit history
description
A textual definition or description
ballotStatusType
VocabularyRevision
A cohesive collection of changes for a terminology
ballotStatus
note
0..∞
Notes, etc. about the current ballot state, decisions and the like.
codeSystemRevision
An addition or modification to a code system.
1..∞
valueSetRevision
An addition or modification to a value set.
vocabularyDomainRevision
An addition or modification to a vocabulary domain.

Figure 1 - Vocabulary Revision

A **vocabularyRevision** always begins with an **editDescription** element that describes the intent and purpose of the revision. The actual description should be recorded in the **description** element. The attributes of the **editDescription** are listed below:



Name	Type	Use	Default	Fixed
creationDate	xs:date	required		
primaryContact	xs:string	required		
committee	xs:string	required		
documentStatus	xs:NMTOKEN		Proposed	

Figure 2 - editDescription attributes

creationDate - when the document was first created (format: yyyy-mm-dd).
primaryContact - the name of the person responsible for the document
committee - the committee identifier primarily responsible for the changes
documentStatus - see table below:

Status	Meaning
Proposed	The document is in the initial stages
Submitted	The document has been submitted for review by the appropriate committees.
Reviewed	The document has been reviewed by the appropriate committees and is awaiting harmonization.
Harmonized	The document has been reviewed and voted on by the RIM Harmonization Committee
Final	The document has been recorded in the official RIM database and can undergo no further changes.
Rejected	The document was not accepted at some stage in the process.

Table 1 - documentStatus values

The tools used to update the actual vocabulary tables won't process *Rejected* status documents. In addition, warnings will be generated if Harmonized or Final documents contain *Proposed* **ballotStatus** action entries (described below).

Final and *Harmonized* submissions must also supply OID's for all new code system registrations.

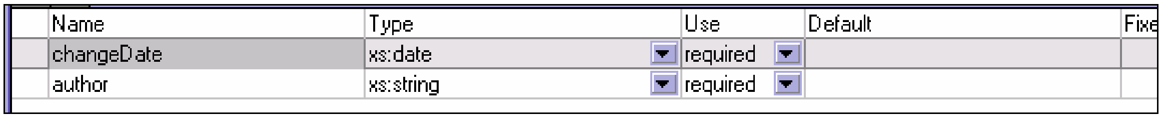
```
<?xml version="1.0" encoding="UTF-8"?>
<VocabularyRevision xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xsi:noNamespaceSchemaLocation="VocabularyRevision.xsd">
   <editDescription creationDate="2003-04-28"
                      primaryContact="Harold Solbrig"
                      committee="C21"
                      documentStatus="Proposed">
      <description>Specification of the orderable beer codes for order message</description>
   </editDescription>
```

Figure 3 - Sample edit description

The sample above describes a new edit created on April 28, 2003 by Harold Solbrig for the Vocabulary (C21) committee. The first three lines of a vocabulary submission should always appear as they do above.

2.1 Edit Versions

A vocabulary revision may also include one or more edit version entries that track the history of the submission. Edit versions are optional and must immediately follow the **editDescription** entry.



Name	Type	Use	Default	Fixe
changeDate	xs:date	required		
author	xs:string	required		

Figure 4 - editVersion attributes

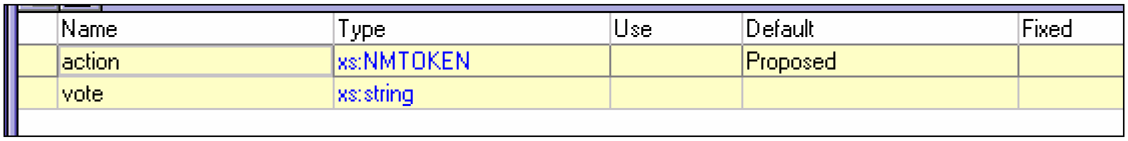
changeDate	- date that the submission was changed
author	- name of the person making the change

```
<editVersion changeDate="2003-05-12" author="Harold Solbrig">
    <description>Enhanced examples</description>
</editVersion>
<editVersion changeDate="2003-05-19" author="Harold Solbrig">
    <description>Adjustments for the revised template</description>
</editVersion>
```

Figure 5 - Sample editVersion(s)

2.2 Ballot Status

ballotStatus reflects the current status of the document in terms of the RIM Harmonization process. A ballotStatus can occur at multiple points throughout a vocabulary submission document, with the “innermost” status taking precedence. A ballot status has an optional note element that can be used to clarify the current state. ballotStatus has the following attributes:



Name	Type	Use	Default	Fixed
action	xs:NMTOKEN		Proposed	
vote	xs:string			

Figure 6 - ballotStatus attributes

The first component of a **registerCodeSystem** entry is the ballot status of the action. The **ballotStatusEntry** is optional. If it is omitted, it defaults to the innermost surrounding **ballotStatus**. If, for example, the **ballotStatus** for **addCodesForCodeSystem** is omitted, the surrounding **registerCodeSystem** or **selectCodeSystem ballotStatus** would apply. If there is no surrounding **ballotStatus**, the status defaults to “Proposed”.

action - the current status on this particular part of the submission as described by the table below.

vote - the actual vote (format: *yy-nn-aa*) where *yy* is the number of yes votes, *nn* the number of no votes and *aa* the number of abstentions. The **vote** attribute applies to Passed, PassedWithChanges and Withdrawn actions

Action	Meaning
Proposed	The proposed revision has been proposed (default).
Passed	The revision has passed RIM harmonization
PassedWithChanges	The revision has passed RIM harmonization subject to the changes outlined in the note.
Tabled	The revision has been set aside and will be reconsidered at a future time.
Withdrawn	The revision has been withdrawn, voted down or otherwise has not been accepted. It cannot be reconsidered in this context.
NonVotingItem	The revision consists of technical changes that do not need to be voted on.

Table 2 - ballotStatus action values

Tabled and *Withdrawn* entries will not be processed by the tools used to update the RIM vocabulary database. In addition, *Proposed* items will be considered in error when the **vocabularyRevision.documentStatus** is *Final*.

2.3 Revisions

A list of revision entries follows the edit description and edit versions (if any) . Revision entries are applied in the order that they occur. A revision entry can create or modify a code system, a value set or a vocabulary domain. The following tags identify revision entities:

- **codeSystemRevision** - Register (create) a new code system or modify the contents of an existing code system.
- **valueSetRevision** - Create a new value set or modify the contents of an existing value set.
- **vocabularyDomainRevision –** Create or modify a vocabulary domain.

The contents of these revision records are described in more detail in the following sections.

Revision records will be applied to the vocabulary database in sequential order. A typical sequence of revision records might consist of:

1) Register a new code system and define its contents
2) Create the value set (or sets) drawn from the code system
3) Create the vocabulary domain and connect it to the value sets.

3 Code System Revisions

A code system revision can either create (register) a new code system or update an existing system.

3.1 Registering a New Code System

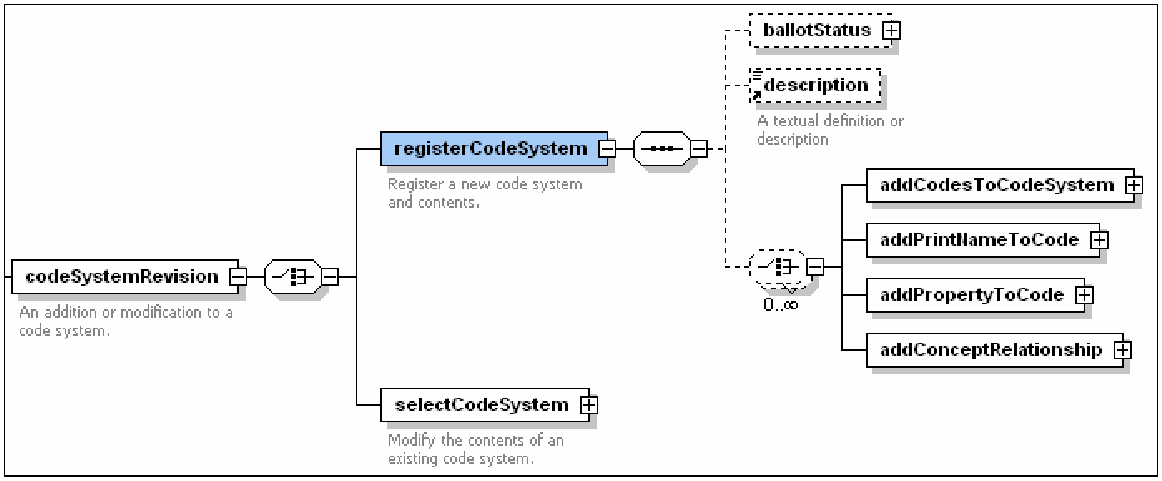
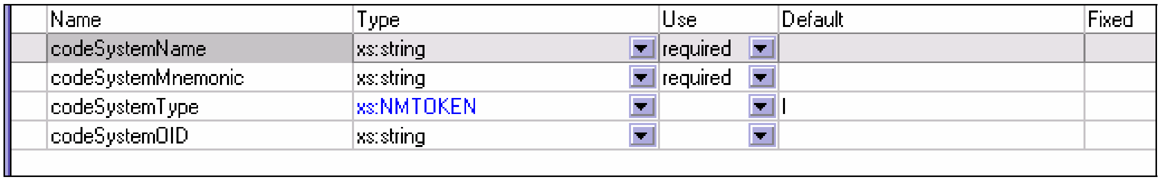

ballotStatus
description
A textual definition or description
registerCodeSystem
Register a new code system and contents.
addCodesToCodeSystem
addPrintNameToCode
addPropertyToCode
addConceptRelationship
0..∞
codeSystemRevision
An addition or modification to a code system.
selectCodeSystem
Modify the contents of an existing code system.

Figure 7 - registerCodeSystem

The **registerCodeSystem** tag is used to register a new external, internal or external/internally coded code system with the HL7 RIM vocabulary. The attributes of the **registerCodeSystem** entry are described below



Name	Type	Use	Default	Fixed
codeSystemName	xs:string	required		
codeSystemMnemonic	xs:string	required		
codeSystemType	xs:NMTOKEN		I	
codeSystemOID	xs:string			

Figure 8 - registerCodeSystem attributes

codeSystemName The full name used to describe the code system.

codeSystemMnemonic A short mnemonic used to represent the code system within the HL7 vocabulary. The mnemonic must be unique.

codeSystemType - (see table below)

codeSystemOID - the official ISO Object Identifier (OID) of the code system. The OID can be omitted when **vocabularyRevision.documentStatus** is anything other than *Final* . The **codeSystemOID** must be explicitly stated before the code system can be registered in the official RIM database.

If the code system OID is omitted, it will be assigned an available number from the HL7 example root branch (2.16.840.1.113883.19). A number will be assigned from the 2.16.840.1.113883.19.5 branch for internal code systems and the 2.16.840.1.113883.19.6 branch for External systems.

codeSystemType	Meaning
I	(Internal) The code system is created and maintained by HL7 (default)
E	(External) The code system is created and maintained by an outside party and is not carried in the HL7 tables
EI	(External/Internally maintained) The code system is created and maintained by an outside party, but a copy is carried within the HL7 tables for convenience.

Table 3 - codeSystemType values

3.1.1 Code System Description

The **description** element immediately follows the **ballotStatus** if it exists. Otherwise it is the first element in the code system registration block. It is optional and describes the intent or purpose of the code system revision.

```
   <codeSystemRevision>

      <registerCodeSystem
         codeSystemName="Beer and Flavor Classification" codeSystemMnemonic="BEERS">

         <ballotStatus action="Passed" vote="17-4-1"/>

         <description>A beer and flavor classification derived from the Castello web site:
http://www.birracastello.com/html/masgustoeng.html</description>
```

Figure 9 - Sample codeSystemRevision header

The example above registers a new *Beer and Flavor Classification* code system under the mnemonic *BEERS*. The entry has passed RIM harmonization with 17 for, 4 against and one abstention. The code system is *Internal* by default. Because an OID isn't supplied, an OID will automatically be generated from the 2.16.840.1.113883.19.5 branch.

```xml
    <codeSystemRevision>
        <registerCodeSystem codeSystemName="Logical Observation Identifiers Names and
Codes" codeSystemMnemonic="LOINC" codeSystemOID="2.16.840.1.113883.6.1"
codeSystemType="E">
            <description>The purpose of the LOINC database is to facilitate the exchange and
pooling of results, such as blood hemoglobin, serum potassium, or vital signs, for clinical care,
outcomes management, and research. Currently, most laboratories and other diagnostic services
use HL7 to send their results electronically from their reporting systems to their care systems.
However, most laboratories and other diagnostic care services identify tests in these messages
by means of their internal and idiosyncratic code values. Thus, the care system cannot fully
"understand" and properly file the results they receive unless they either adopt the producer's
laboratory codes (which is impossible if they receive results from multiple sources), or invest in
the work to map each result producer's code system to their internal code system. LOINC codes
are universal identifiers for laboratory and other clinical observations that solve this problem.
The laboratory portion of the LOINC database contains the usual categories of chemistry,
hematology, serology, microbiology (including parasitology and virology), and toxicology; as well
as categories for drugs and the cell counts you would find reported on a complete blood count or
a cerebrospinal fluid cell count. Antibiotic susceptibilities are a separate category. The clinical
portion of the LOINC database includes entries for vital signs, hemodynamics, intake/output,
EKG, obstetric ultrasound, cardiac echo, urologic imaging, gastroendoscopic procedures,
pulmonary ventilator management, selected survey instruments, and other clinical observations.
The Regenstrief Institute (www.regenstrief.org) maintains the LOINC database and its supporting
documentation.
            </description>
        </registerCodeSystem>
    </codeSystemRevision>
```

Figure 10 - Sample external code system registration

The above example registers the LOINC code system as an external entry under the mnemonic “LOINC” with the OID “2.16.840.1.113883.6.1”

3.1.2 Adding Concept Codes to a Code System

The **registerCodeSystem** operation allows concept codes to be added to the newly registered system.

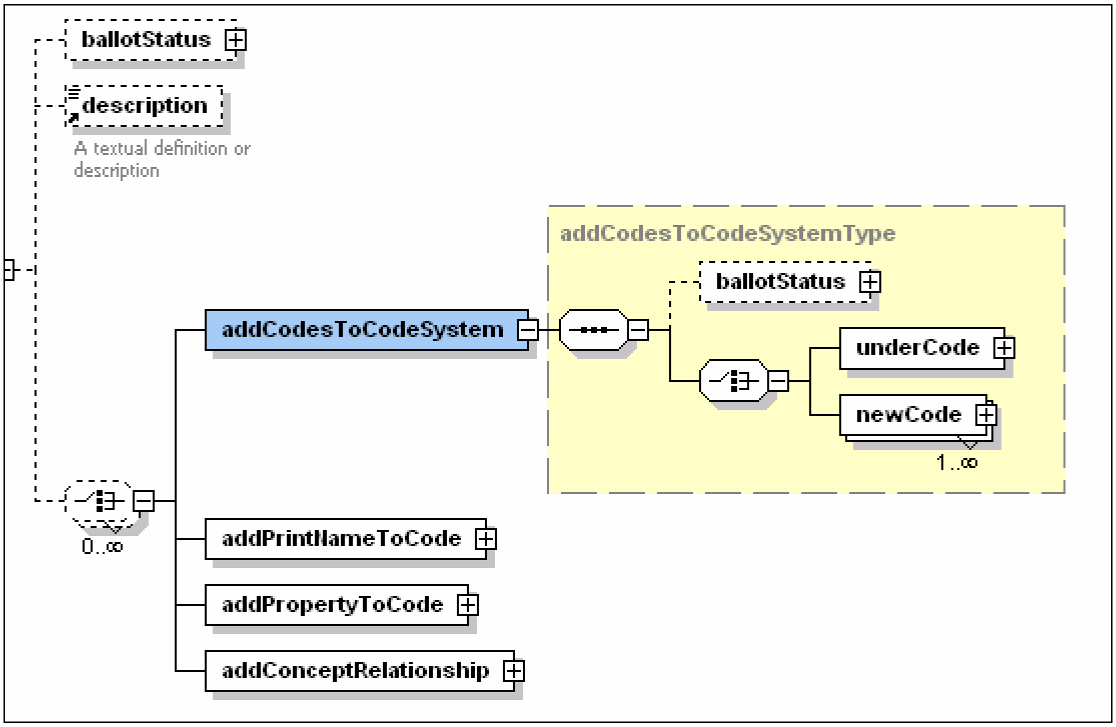
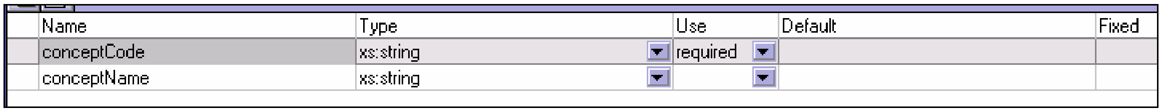


Figure 11 – addCodesToCodeSystem

The first component of **addCodesToCodeSystem** a **ballotStatus** which was described in section 2.2. Codes may either be added underneath an already existing concept code or as a completely new hierarchical node. In this context “under” is used to imply subsumption. If, for example, *Pale Ale* is a kind of *Ale*, the code for *Pale Ale* would occur “under” the code for *Ale*.

It is anticipated that the **underCode** option will be used primarily when rearranging existing code systems.

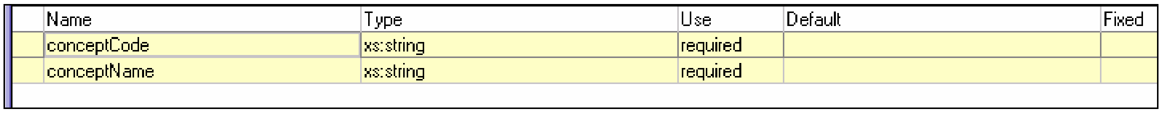


Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			

Figure 12 - underCode attributes

conceptCode	The concept code of an existing parent node
conceptName	A valid designation for the parent node. **conceptName** is optional, and will be validated if it is supplied.

The **newCode** node identifies a new concept code to be added to the code system.



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string	required		

Figure 13 - newCode attributes

conceptCode	The new concept code to be added to the code system. The code must not already exist in the system.
conceptName	A valid designation for the concept code. Optional, but validated if supplied.

Each **newCode** entry can also a description of the intent and purpose of the code. The description element is optional. Additional **newCode** entries may be nested underneath other entries as needed. The figure below shows a sample set of **newCode** entries.

```
<addCodesToCodeSystem>
    <ballotStatus action="Proposed"/>
    <newCode conceptCode="1001" conceptName="ALE">
        <description>The name which was given to a brew drink acquire from the ancient Celts
Anglo-Saxon origin but today synonym to high brew beer.</description>
        <newCode conceptCode="1002" conceptName="PALE ALE">
            <description>Light amber colouring, with, abundant hop and fairly dry. Slightly bitter.
Below 5ø alcoholic contents. Served at cellar temperature.</description>
        </newCode>
        <newCode conceptCode="1003" conceptName="BITTER ALE">
            <description>Very dry beer and decisively bitter. Distinct malt taste and aroma. Dark
coppery colour. Low carbon dioxide contents. 3-4 alcoholic contents. Traditionally
drafted.</description>
        </newCode>
        <newCode conceptCode="1004" conceptName="LIGHT ALE">
                <description>Lower alcoholic contents than the bitter. Traditionally dunk out of a
bottle.</description>
        </newCode>
    </newCode>
    <newCode conceptCode="1005" conceptName="PORTER">
        <description>Originated in London. Both it's dark brown colour and it's characteristic bitter
taste is due to the roasted barley and not to the malt. The bitter taste increased with the high hop
quantities which makes it also sour.</description>
    </newCode>
    <newCode conceptCode="1006" conceptName="STOUT">
        <description>It is thicker and richer than the Porter, also made with roasted barley and not
malt. National Irish beer and very sour, dark with a rich creamy froth.</description>
    </newCode>
    <newCode conceptCode="1007" conceptName="BITTER STOUT">
        <description>Very dry and distinctively bitter. Only sold in Ireland</description>
    </newCode>
</addCodesToCodeSystem>
```

Figure 14 - Sample addCodesToCodeSystem entry

The above example adds four new concept codes as “root nodes” – 1001 (ALE), 1005 (PORTER), 1006 (STOUT) and 1007 (BITTER STOUT). It also adds concept codes 1002 (PALE ALE), 1003 (BITTER ALE) and 1004 (LIGHT ALE) as children of the ALE node.

```
<codeSystemRevision>
          <selectCodeSystem codeSystemMnemonic="BEERS">
               <addCodesToCodeSystem>
                    <underCode conceptCode="1006" conceptName="STOUT">
                         <newCode conceptCode="1008" conceptName="SWEET STOUT">
                              <description>Dark brown colour with a sweeter taste, malted, slightly spicy.
     </description>
                         </newCode>
                         <newCode conceptCode="1009" conceptName="MILK STOUT">
                              <description>Sweet stout version with added milk sugar to soften the sour
taste.  </description>
                         </newCode>
                    </underCode>
               </addCodesToCodeSystem>
          </selectCodeSystem>
     </codeSystemRevision>
```

Figure 15 - Sample addCodesToCodeSystem on existing system

The example above shows how concept codes can be added underneath an existing concept code in an existing code system. In the example we add two new concept codes – 1008 and 1009 as children of the existing concept code 1006 (STOUT).

3.1.3 Adding Print Names to a Concept Code

Both the **registerCodeSystem** and the **selectCodeSystem** branches allow additional print names to be added to existing concept codes.

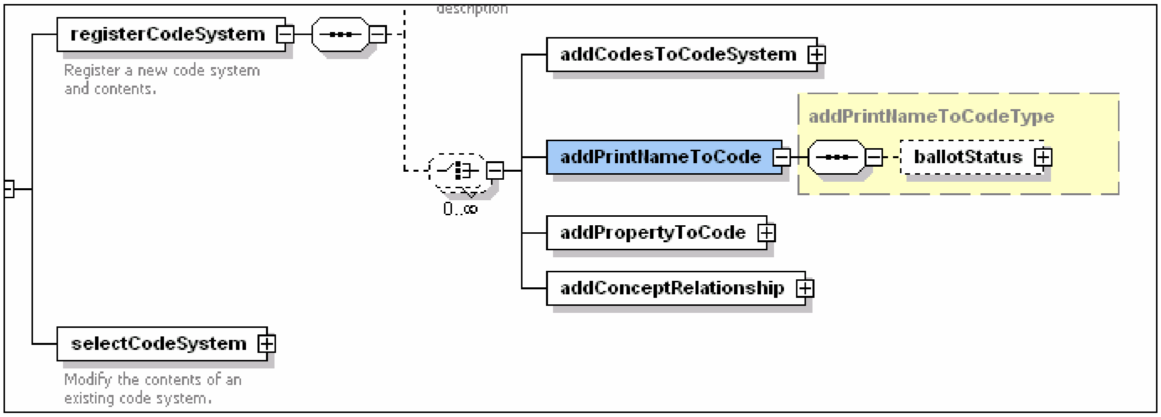
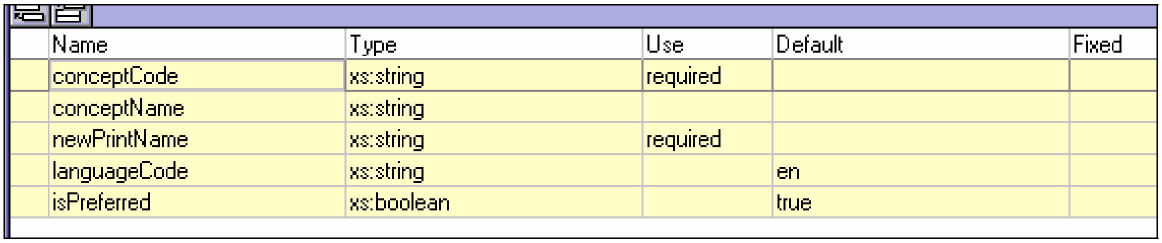


Figure 16 – addPrintNameToCode

Each **addPrintNameToCode** entry can have its own ballot status (see: section 2.2). If omitted, the **ballotStatus** of the containing element will be used.



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			
newPrintName	xs:string	required		
languageCode	xs:string		en	
isPreferred	xs:boolean		true	

Figure 17 - addPrintNameToCode attributes

conceptCode	The concept code to add the new print name to
conceptName	A valid designation for concept code. Optional, but will be validated if it is supplied
newPrintName	The print name to be added to the concept code
languageCode	The language code for the print name. Default is “en” for English
isPreferred	Flag indicating whether the print name is preferred for the language. Default is “true”.

If the **isPreferred** flag is “true”, the default value, all other print names for the same concept code and language will have their **isPreferred** flags set to “false”.

```
<codeSystemRevision>
    <selectCodeSystem codeSystemMnemonic="BEERS">
        <addPrintNameToCode conceptCode="1004" conceptName="LIGHT ALE"
                              newPrintName="Pils" languageCode="de"/>
        <addPrintNameToCode conceptCode="1004" newPrintName="Lager"
                              languageCode="de" isPreferred="false"/>
    </selectCodeSystem>
</codeSystemRevision>
```

Figure 18 - Sample addPrintNameToCode

The example above adds two new German print names to concept code 1004 – LIGHT ALE. The first name, Pils, is the preferred German name.

3.1.4 Adding Properties to a Concept Code

Properties such as “openIssue”, “appliesTo”, “howApplies”, etc. can be added to concept codes during either the code system registration or code system update phases.

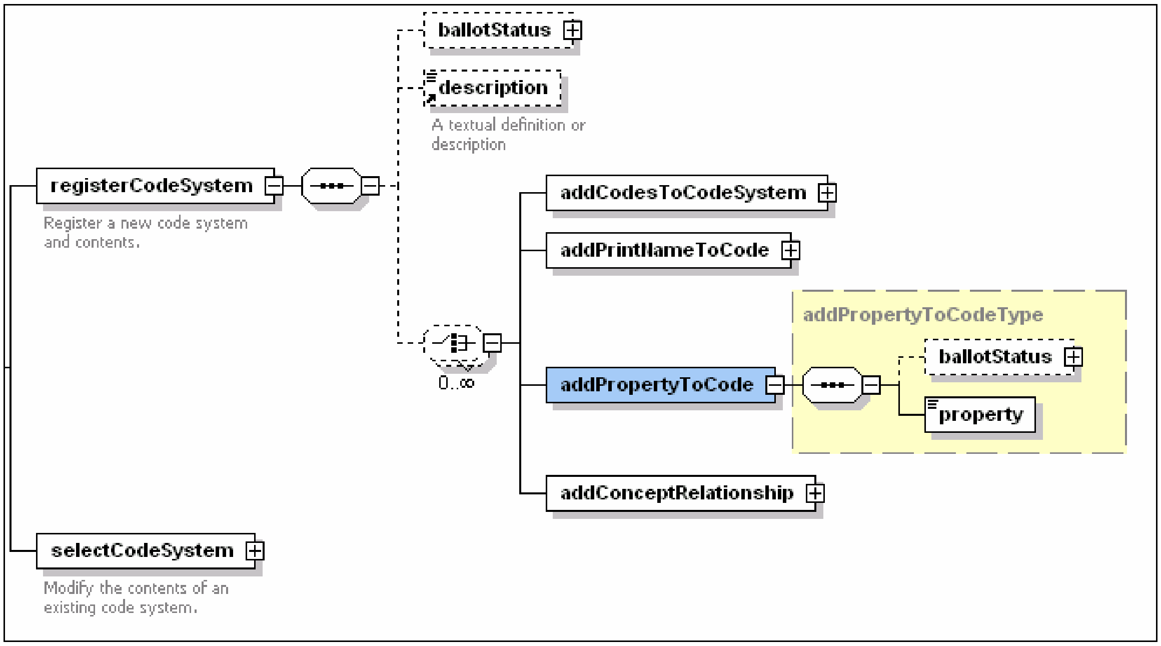
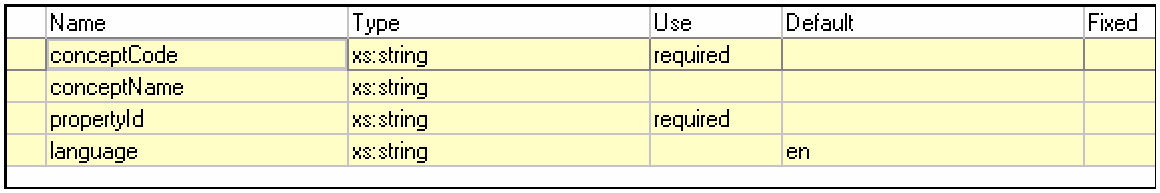


Figure 19 - addPropertyToCode

addPropertyToCode can have its own **ballotStatus** (see section 2.2) or can inherit the **ballotStatus** from the containing entry. The concept code, property id and language code are defined as attributes (see below). The property value itself is entered in the **property** element.



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			
propertyId	xs:string	required		
language	xs:string		en	

Figure 20 - addPropertyToCode attributes

conceptCode	The concept code to add the property to
conceptName	The preferred name of the code. Optional, but validated if supplied.
propertyId	The unique identifier of the property to be added
language	The language of the property. Default is “en” if not supplied.

```
    <addPropertyToCode conceptCode="1007"
                       conceptName="BITTER STOUT"
                       propertyId="appliesTo">
        <ballotStatus action="Withdrawn" vote="3-12-1"/>
        <property>Colds and whatever else ails you</property>
    </addPropertyToCode>
```

Figure 21 - addPropertyToCode sample

The above example adds an *appliesTo* property to concept code “1007”. The sample also shows an example of a negative vote – probably because the *appliesTo* property was being badly misused.

3.1.5 Defining Relationships between Concept Codes

Additional relationships other than subsumption may be asserted to exist between concept codes. As an example, one may want to assert that one concept code occurs “before” a second in a basic ordering. Relationships can be added to concept codes during either the code system registration or code system update phases.

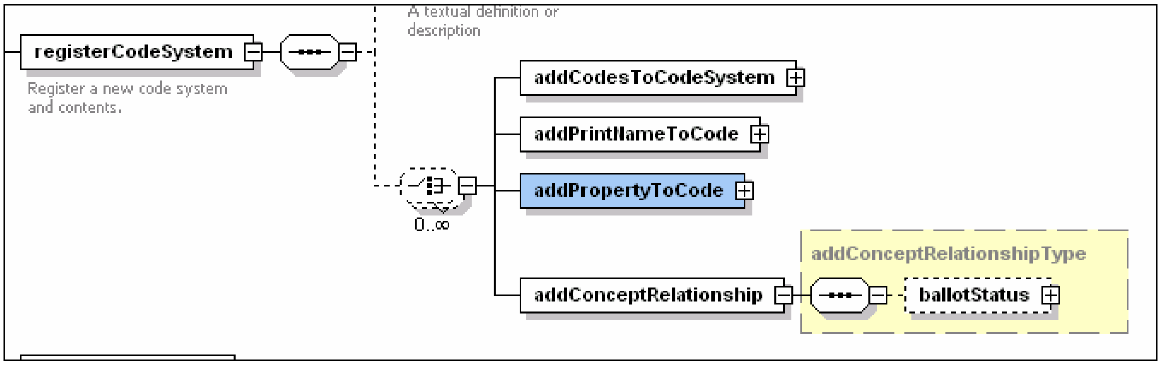
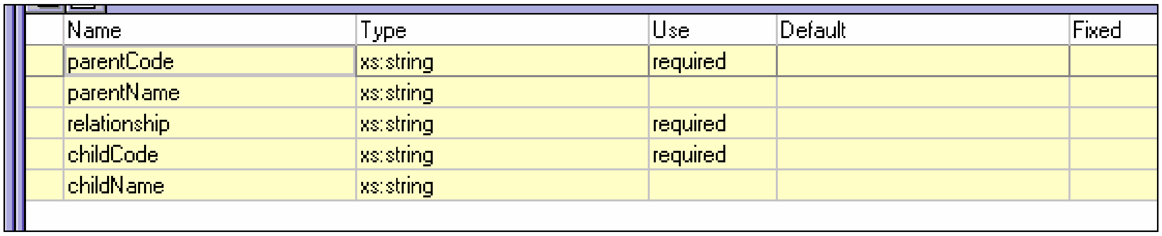


Figure 22 – addConceptRelationship

As with the other operations, addConceptRelationship can have own **ballotStatus see section 2.2** or can inherit the **ballotStatus** from the containing entry. The addConceptRelationship attributes define the relationship to be added:



Name	Type	Use	Default	Fixed
parentCode	xs:string	required		
parentName	xs:string			
relationship	xs:string	required		
childCode	xs:string	required		
childName	xs:string			

Figure 23 - addConceptRelationship attributes

parentCode	The concept code that occurs on the “left hand” or parent side of the relationship
parentName	A valid designation for the parent code. Optional but validated if present
relationship	A valid relationship code (e.g. smallerThan)
childCode	The concept code that occurs on the “right hand” or child side of the relationship
childName	A valid designation for the child code. Optional but validated if present.

```
<addConceptRelationship parentCode="1002" parentName="LIGHT ALE"
relationship="smallerThan"
                                        childCode="1003" childName="BITTER ALE"/>
```

Figure 24 - Sample addConceptRelationship entry

The above example asserts that concept code “1002” is “less than” concept code “1003” in an ordinal sense.

3.2 Modifying an Existing Code System

It is also possible to modify the contents of a previously registered code system. As with new code systems, it is possible to add new concept codes to the code system, new print names or properties to concept codes, and new relationships between concept codes. In addition it is also possible to modify the properties of the code system itself, modify or remove print names, descriptions, properties and relationships. It is also possible to retire concept codes from code systems and adjust subsumption relationships.

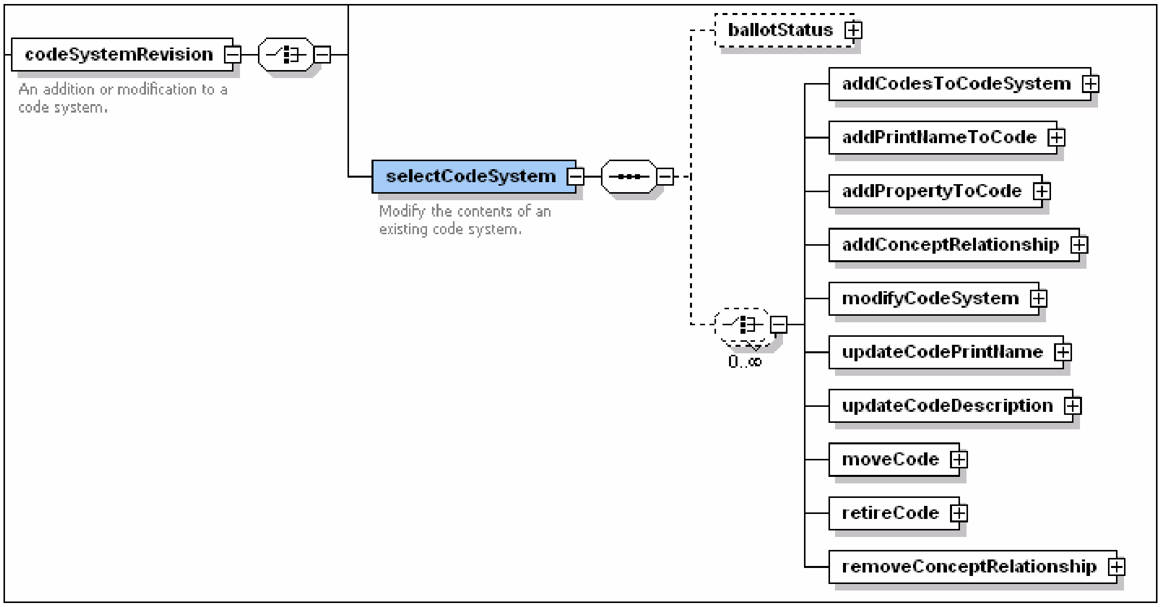
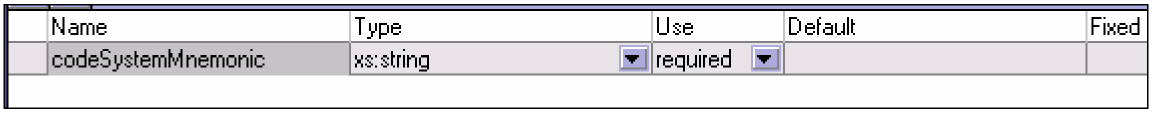


Figure 25 – selectCodeSystem

selectCodeSystem has an optional **ballotStatus** element that can be used to record the vote on the entire modification (see section 2.2). It has a single attribute, **codeSystemMnemonic**, that selects the code system to be modified.



Name	Type	Use	Default	Fixed
codeSystemMnemonic	xs:string	required		

Figure 26 - selectCodeSystem attributes

codeSystemMnemonic The HL7 mnemonic code for the code system to be selected

The first four selectCodeSystem operations

```
<codeSystemRevision>
    <selectCodeSystem codeSystemMnemonic="BEERS">
        <ballotStatus action="PassedWithChanges" vote="12-1-0">
            <note>Author will sample all changes first</note>
        </ballotStatus>
```

Figure 27 - Sample selectCodeSystem element

The above example selects the “BEERS” code system for modification. Unless otherwise noted, all of the changes within the selection were passed by a 12-1-0 vote (the author being the single negative vote) with the provision that author will sample all of the changes before they are officially recorded.

3.2.1 Modifying the Code System Mnemonic, Name or Description

The modifyCodeSystem element is used to update the properties of the code system itself.

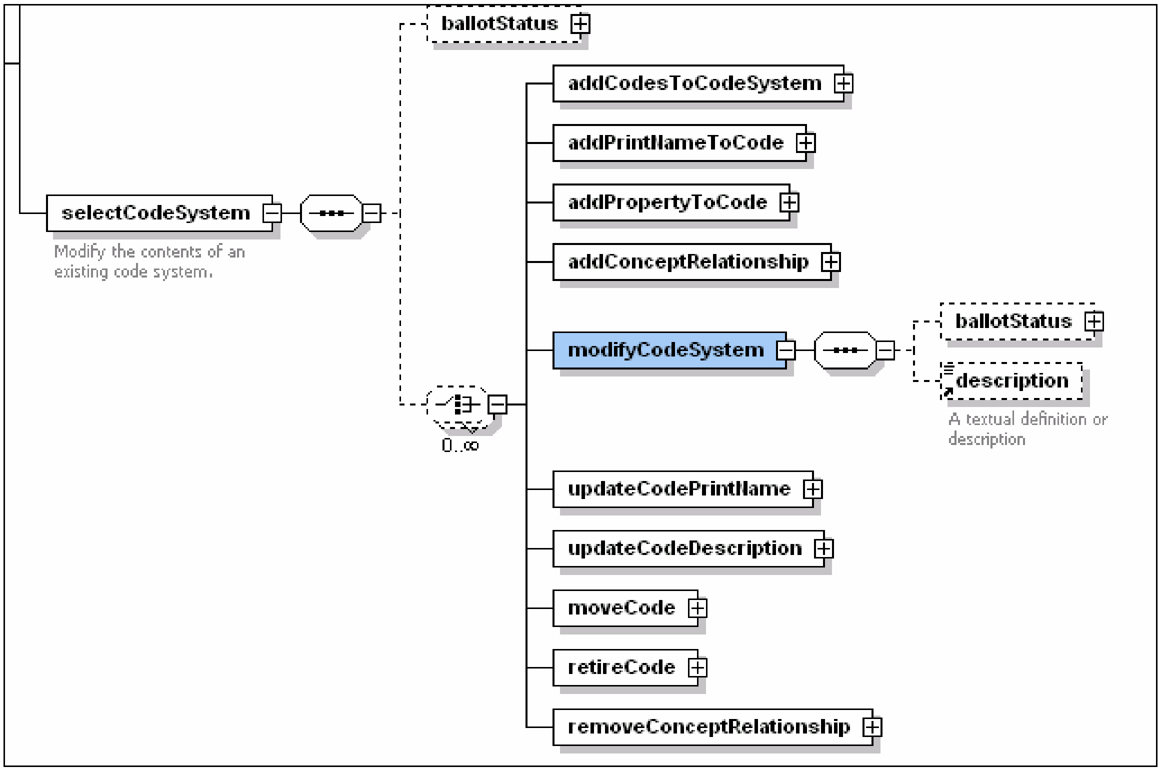
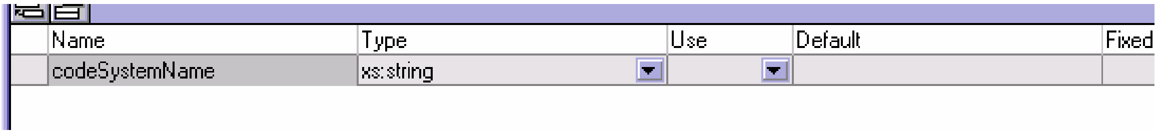


Figure 28 - modifyCodeSystem

modifyCodeSystem has two optional elements. The first, **ballotStatus**, defines the status of the modification if it is different than the status as it occurs on the **selectCodeSystem** node. The second optional element, **description**, is used to add, update or remove the existing code system description. If the **description** element is not supplied, the existing code system description is unchanged. The name of a code system can be changed as well, using the **codeSystemName** attribute.



Name	Type	Use	Default	Fixed
codeSystemName	xs:string			

Figure 29 - modifyCodeSystem attributes

odeSystemName If present, the code system name will be updated to this value

```
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <ballotStatus action="PassedWithChanges" vote="12-1-0">
                <note>Author will sample all changes first</note>
            </ballotStatus>
            <modifyCodeSystem codeSystemName="Castello Beer and Flavor Classification"/>
        </selectCodeSystem>
```

Figure 30 - Sample modifyCodeSystem entry

The above example changes the BEERS code system name from what it was previously (Beer and Flavor Classification) to “Castello Beer and Flavor Classification.

3.2.2 Updating the Print Name of a Concept Code

Print names for concept codes can be removed or updated, and the setting for which print name is preferred for a given language can be changes as well.

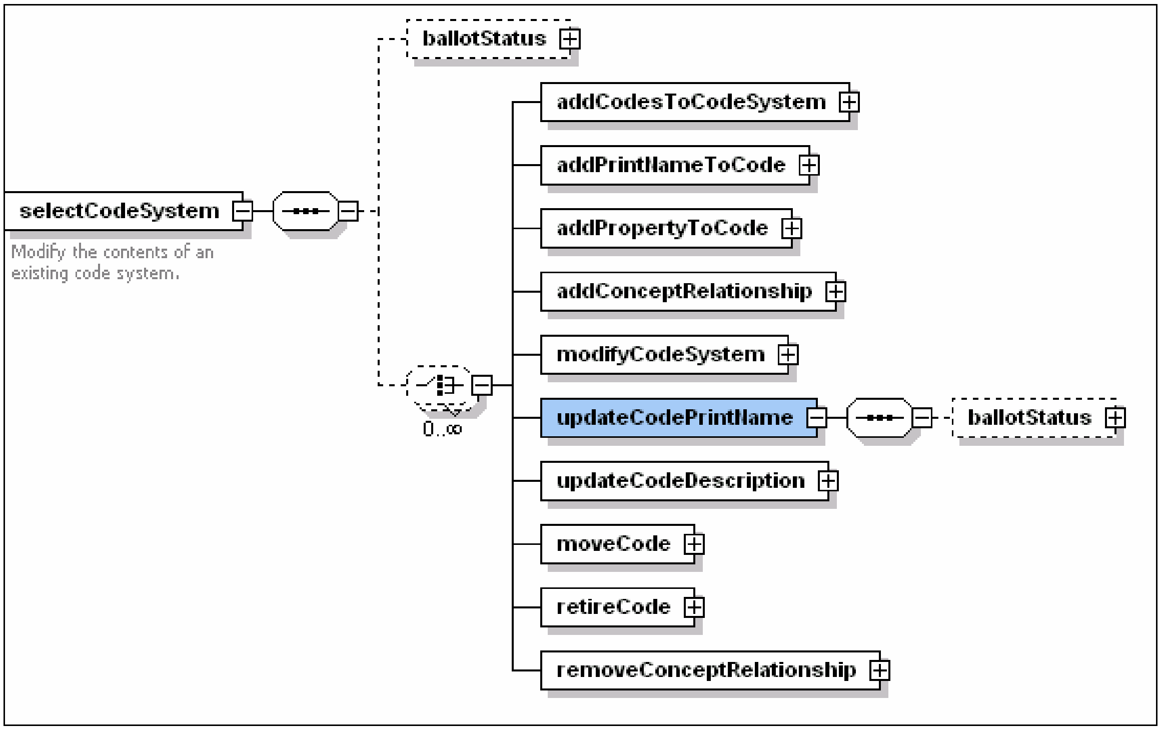
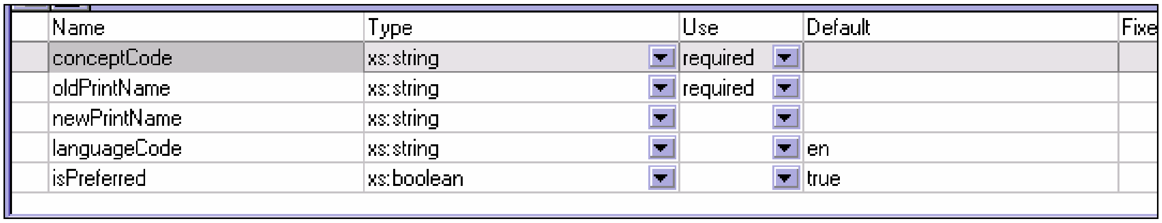


Figure 31 – updateCodePrintName

updateCodePrintName includes an optional **ballotStatus** that reflects the status of the change if it is different from that of the entire revision.



Name	Type	Use	Default	Fixe
conceptCode	xs:string	required		
oldPrintName	xs:string	required		
newPrintName	xs:string			
languageCode	xs:string		en	
isPreferred	xs:boolean		true	

Figure 32 - updateCodePrintName attributes

conceptCode	The concept code to be changed
oldPrintName	The original print name
newPrintName	The new print name. If omitted, the name isn't changed. If present, but empty, the print name is removed.
languageCode	The language code for both the old and new print names. Default: en. Note: the language of a print name cannot be changed. If this is necessary, the print name for one language must first be removed and then re-added under the second language.
isPreferred	true means that this print name is the preferred one for the supplied language. Setting this to "true" sets all other **isPreferred** flags to "false" to for the given concept code/language.

```
<selectCodeSystem codeSystemMnemonic="BEERS">
    <updateCodePrintName conceptCode="1004"
                   oldPrintName="LIGHT ALE" newPrintName="Light Ale"/>
    <updateCodePrintName conceptCode="1004"
                   oldPrintName="Pils" newPrintName="Kölsch" languageCode="de"/>
    <updateCodePrintName conceptCode="1004"
                   oldPrintName="Lager"
                   languageCode="de" isPreferred="true"/>
    <updateCodePrintName conceptCode="1004"
                   oldPrintName="Kölsch" newPrintName="" languageCode="de"/>
</selectCodeSystem>
```

Figure 33 - Sample updateCodePrintName entries

The first entry above changes the print name of concept code "1004" from "LIGHT ALE" to "Light Ale". The second entry changes the German print name for concept code "1004" from "Pils" to "Kölsch". Note that this operation would fail if there wasn't already a German print name "Pils". The third entry changes "Lager" to be the preferred German print name if it isn't already. If it is, this operation would still succeed. The fourth entry removes the "Kölsch" print name completely – a somewhat odd request since we went to all the trouble of changing it a couple lines earlier.

3.2.3 Changing the Description of a Concept Code

The description of a concept code may be added, modified or removed using the **updateCodeDescription** element.

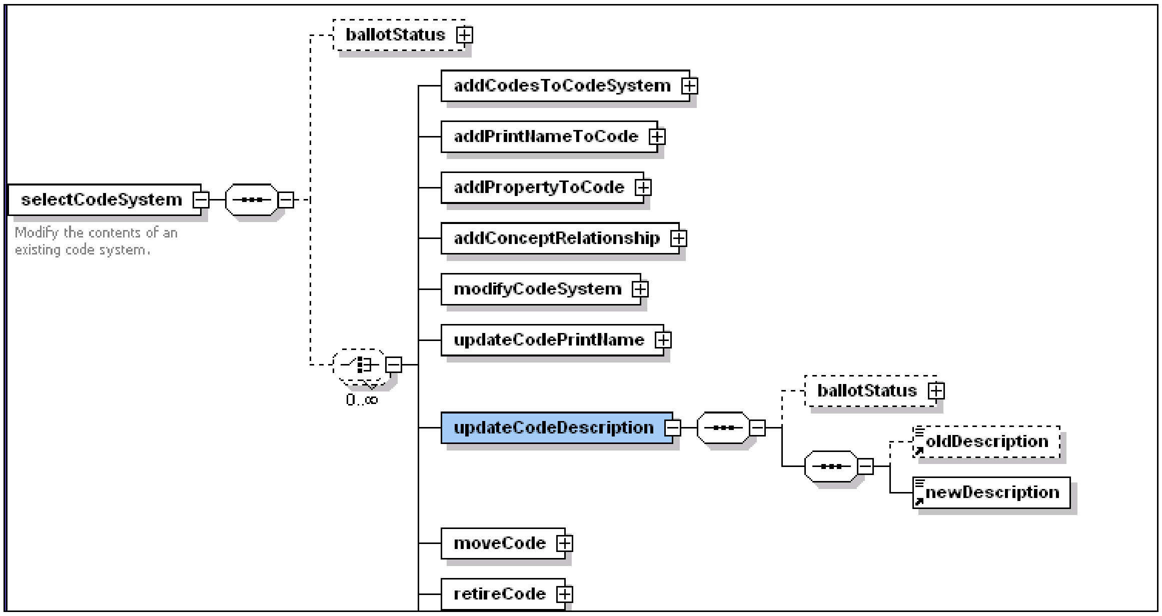
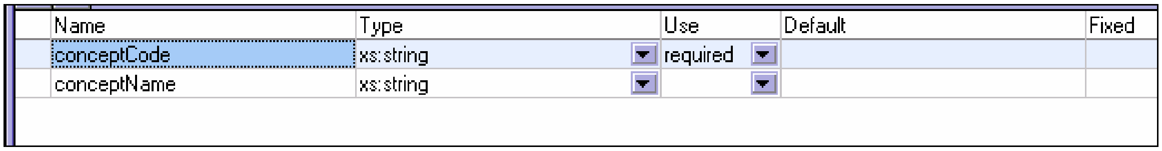


Figure 34 - updateCodeDescription

As with all updates, **updateCodeDescription** can have its own **ballotStatus** if needed to reflect special cases. It has an optional **oldDescription** element that can contain the description that is being changed and a **newDescription** element that contains the revised description or can be empty if the description is to be completely removed. If present, the **oldDescription** will be validated against the database before the change is performed.



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			

Figure 35 - updateCodeDescription attributes

conceptCode	The concept code that the change is being applied to
conceptName	A valid designation for the concept code. Optional, but will be validated if supplied.

```
<codeSystemRevision>
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <updateCodeDescription conceptCode="1004" conceptName="Light Ale">
                <oldDescription>Lower alcoholic contents than the bitter. Traditionally dunk out of a
bottle.</oldDescription>
                <newDescription>Lower alcoholic contents than bitter.</newDescription>
            </updateCodeDescription>
        </selectCodeSystem>
    </codeSystemRevision>
```

Figure 36 - Sample updateCodeDescription

The example above removes the second sentence from the “Light Ale” description. Note that the preferred name is “Light Ale” rather than “LIGHT ALE” because of an update that occurred earlier in both this document and the actual XML update source.

3.2.4 Moving a Code in the Subsumption Hierarchy

Concept codes can be rearranged within the concept subsumption hierarchy. moveCode can be used to assert that one concept code is a “kind of” or “is implied by” a second concept code. It can also remove this assertion.

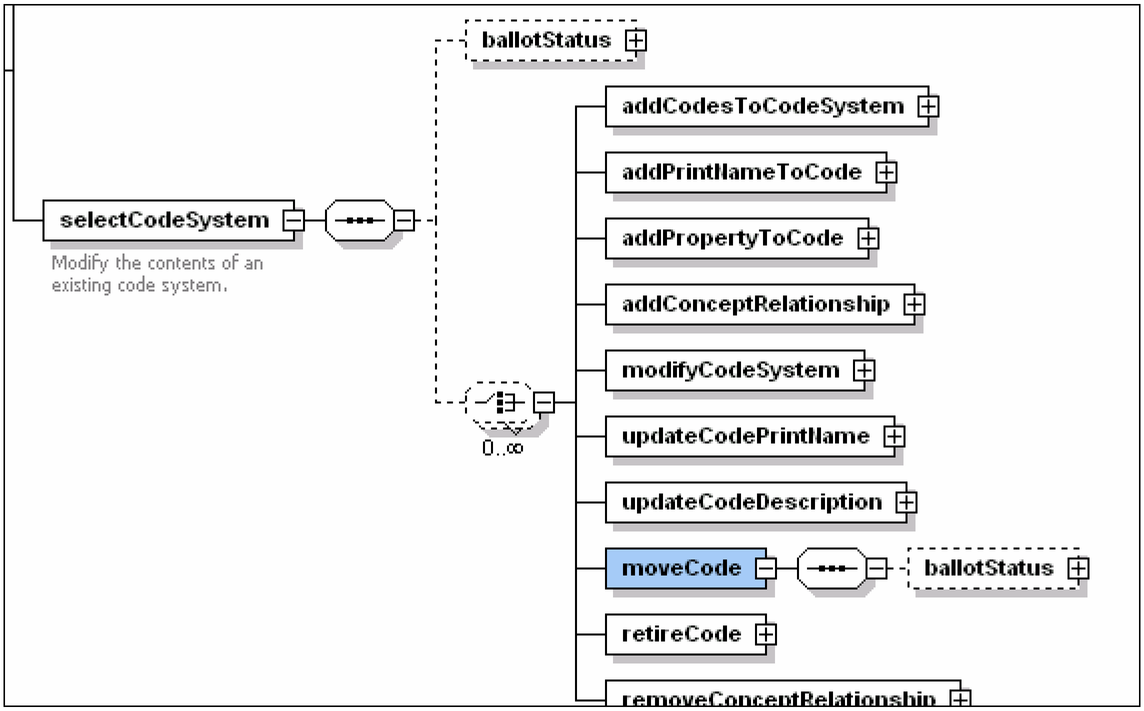
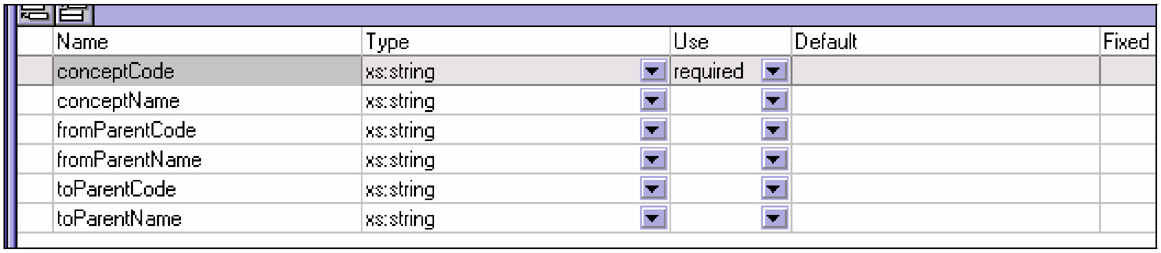


Figure 37 – moveCode

As with all update operations, moveCode has a separate, optional ballotStatus node. The moveCode attributes specify the operation to be performed:



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			
fromParentCode	xs:string			
fromParentName	xs:string			
toParentCode	xs:string			
toParentName	xs:string			

Figure 38 - moveCode attributes

conceptCode	The code of the concept to be moved.
conceptName	A valid designation for the concept to be moved. Optional, and if supplied it will be validated against the database before the operation occurs.
fromParentCode	The current direct parent of the concept code, if any. This attribute should be omitted if **conceptCode** doesn't currently occur under any other node.
fromParentName	A valid designation for the from parent code. Optional, but validated if supplied.
toParentCode	The new parent of **conceptCode**. If omitted, **conceptCode** no longer has a parent in the subsumption hierarchy.
toParentName	A valid designation for the to parent code. Optional, but validated if supplied.

```
<selectCodeSystem codeSystemMnemonic="BEERS">
        <moveCode conceptCode="1008" conceptName="SWEET STOUT"
                        fromParentCode="1006" fromParentName="STOUT"/>
        <moveCode conceptCode="1006"
                        toParentCode="1008" toParentName="SWEET STOUT"/>
</selectCodeSystem>
```

Figure 39 - Sample moveCode entries

In the above example, concept code "1008" (SWEET STOUT) is first removed as a child of "1006" (STOUT). If this operation succeeds, both STOUT and SWEET STOUT would be root nodes in the subsumption hierarchy. The second part the operation asserts that 1006 (STOUT) is a *kind of* 1008 (SWEET STOUT). Note that all other entries in the subsumption hierarchy remain unchanged (e.g. 1009 – MILK STOUT) remains a kind of STOUT throughout.

3.2.5 Retiring a Concept Code

While a concept code may never be reused within a code system, it may be “retired” from active use.

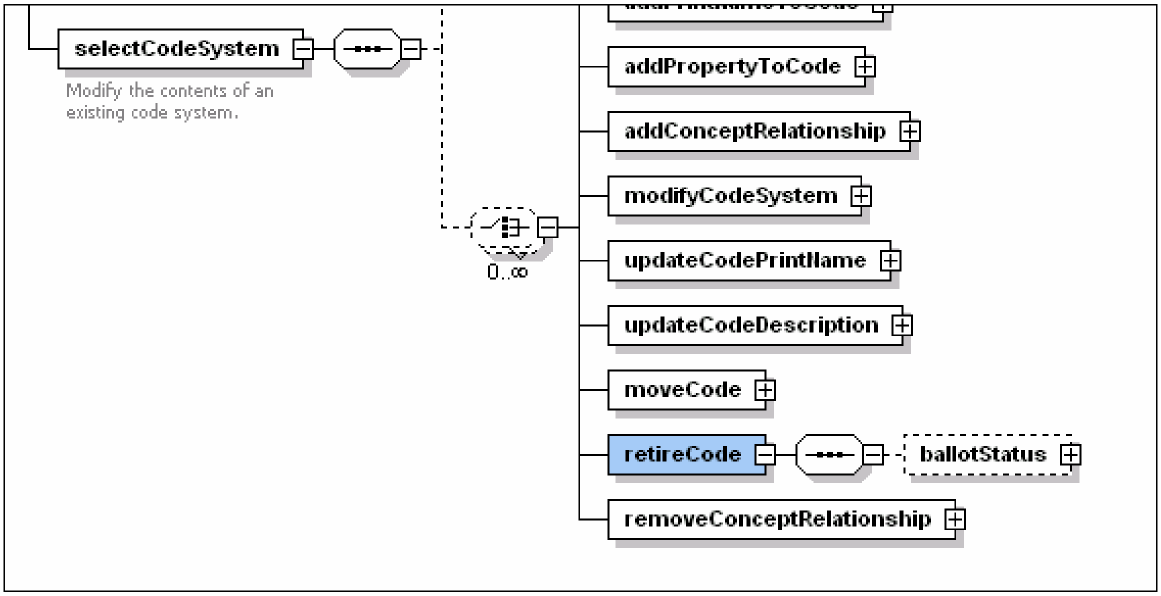
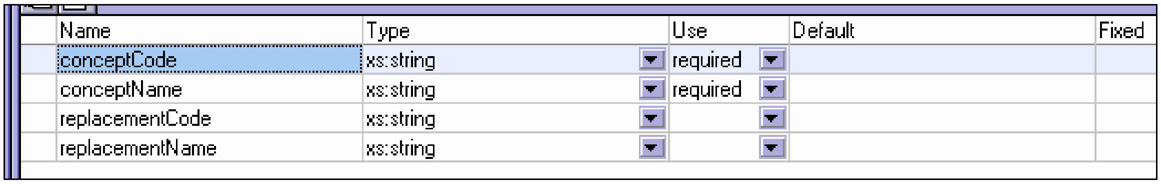


Figure 40 - retireCode

As with all operations, **retireCode** may be accompanied by an optional **ballotStatus**. The attributes of **retireCode** determine what actually happens:



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string	required		
replacementCode	xs:string			
replacementName	xs:string			

Figure 41 - retireCode attributes

conceptCode	The code of the concept to be retired
conceptName	A valid designation for the concept to be retired. Optional, but validated if supplied.
replacementCode	A new or existing concept code that will replace the retired code. If new, all of the retired code’s names, properties and relationships are copied over to the new code. If **replacementCode** already exists, the current concept code’s properties, etc. are dropped.
replacementName	The name of the replacement concept code. If the replacement code exists and this name is supplied it will be validated. If the replacement

code is new and this name is supplied, it will become the preferred name for the replacement code.

3.2.6 Removing Existing Concept Relationships

It is also possible to undo relationships that have previously been asserted using the addConceptRelationship node.

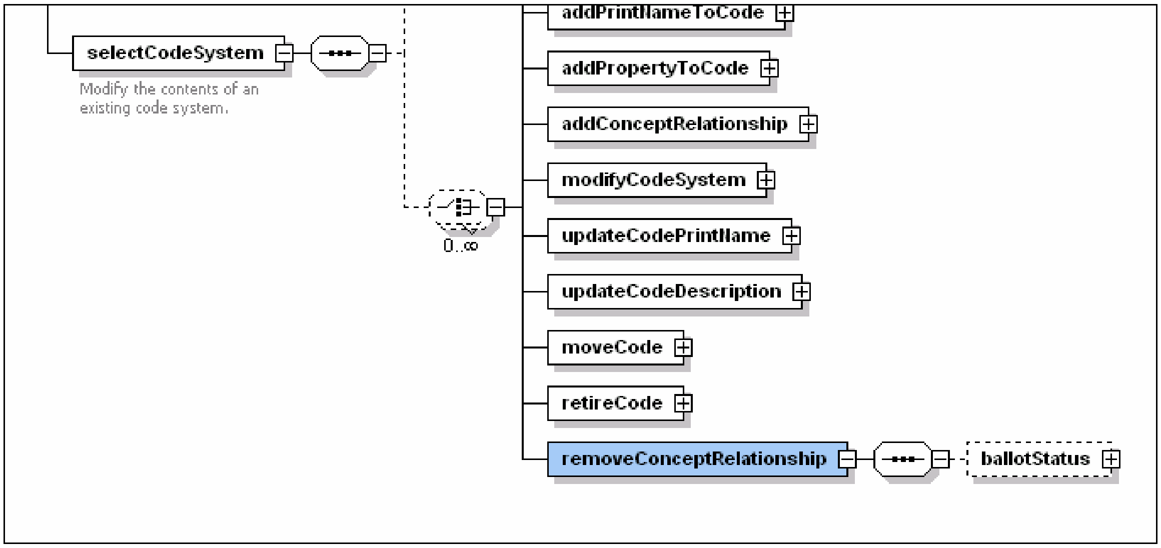
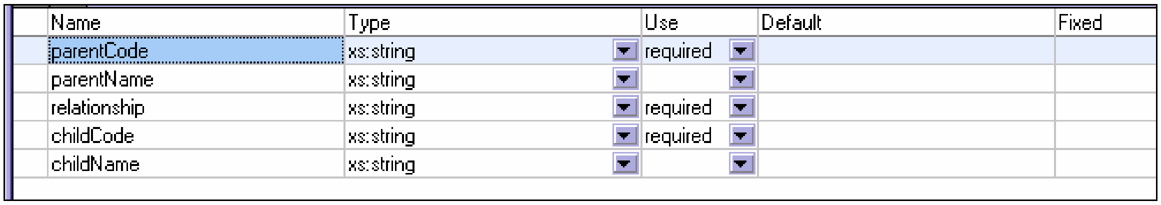


Figure 42 – removeConceptRelationship

Like all operations, **removeConceptRelationship** can have a ballot status entry that overrides the outside status. **removeConceptRelationship** has the following attributes:



Name	Type	Use	Default	Fixed
parentCode	xs:string	required		
parentName	xs:string			
relationship	xs:string	required		
childCode	xs:string	required		
childName	xs:string			

Figure 43 - removeConceptRelationship attributes

parentCode	The left hand or parent side of an existing relationship entry
parentName	A valid designation for the **parentCode**. Optional but validated if present.
relationship	The code for the existing relationship (e.g. smallerThan)
childCode	The right hand or child side of an existing relationship entry

childName	A valid designation for the **childCode**. Optional but validated if present.

4 Value Set Revisions

A value set represents a list of concept codes from a single code system. Value sets are used to identify the list of possible values for a coded attribute based on the HL7 RIM.

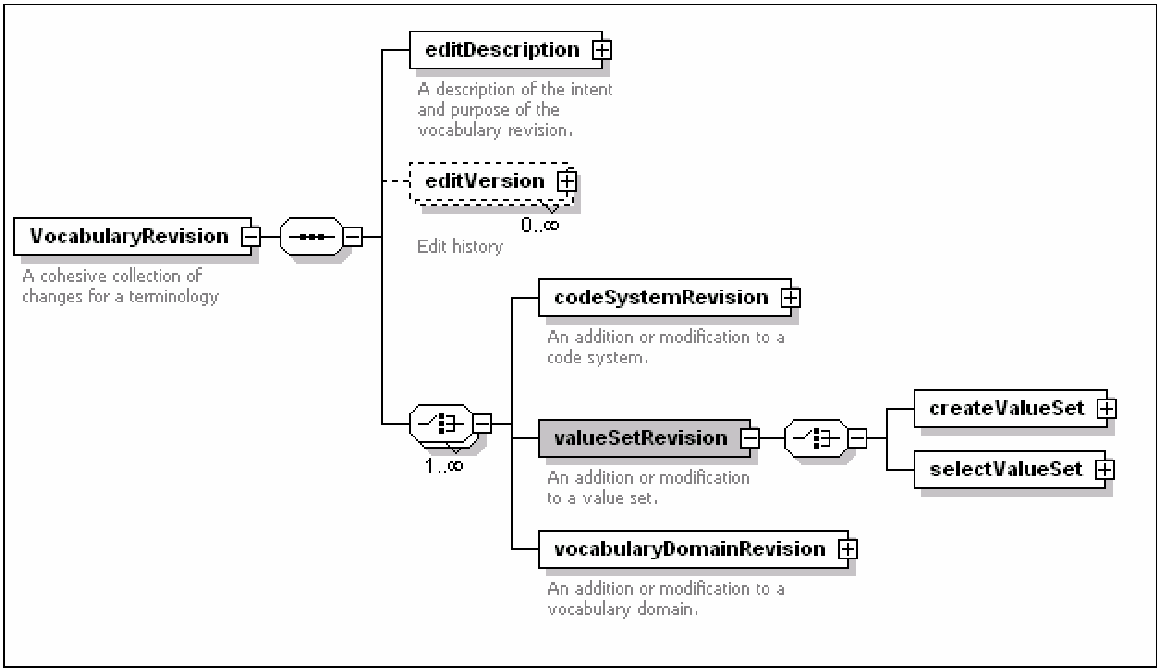
Value sets can be constructed in one of three ways:

1) A value set can represent *all* of the concept codes in a given code system
2) A value set can represent selected codes from a code system. Codes may be selected by:
    a. Identifying individual concept codes
    b. Identifying a concept code and a relationship (e.g. hasSubtype, hasPart) and stating one of:
        i. The concept code *and* all related concepts are included in the set
        ii. All related concepts *except* the named concept are included in the set
        iii. All *leaf nodes* of the specified relationship are included in the set
3) A value set can include codes from other value sets.

Note that the third form of construction actually makes it possible to mix concept codes from more than one code system in a single value set. **This form should only be used, however, when the semantic space of the two value sets are disjoint - meaning that there is no possibility that the same concept could be represented using different codes from different systems.**

When one value set is included in another set, sometimes it is useful to identify a concept code that represents the included value set as a whole (e.g. in the nullFlavor value set, one of the choices is “NoInformation (NI)”, which can be used instead of more specific flavors such as “masked (MSK)”, “not applicable (NA)”, etc.). When this is the case, the “whole” concept code can be identified by associating it with a value set as the “head code”. When this value set is included in a second value set, the include specifies whether the “head code” is a selectable value or not. Refer to the examples on the following pages for further clarification.

The vocabulary revision process can be used to create new value sets as well as to modify the contents of existing sets.



editDescription
A description of the intent and purpose of the vocabulary revision.
editVersion
0..∞
Edit history
VocabularyRevision
A cohesive collection of changes for a terminology
codeSystemRevision
An addition or modification to a code system.
1..∞
valueSetRevision
An addition or modification to a value set.
createValueSet
selectValueSet
vocabularyDomainRevision
An addition or modification to a vocabulary domain.

Figure 44 - valueSetRevision

4.1 Creating a New Value Set

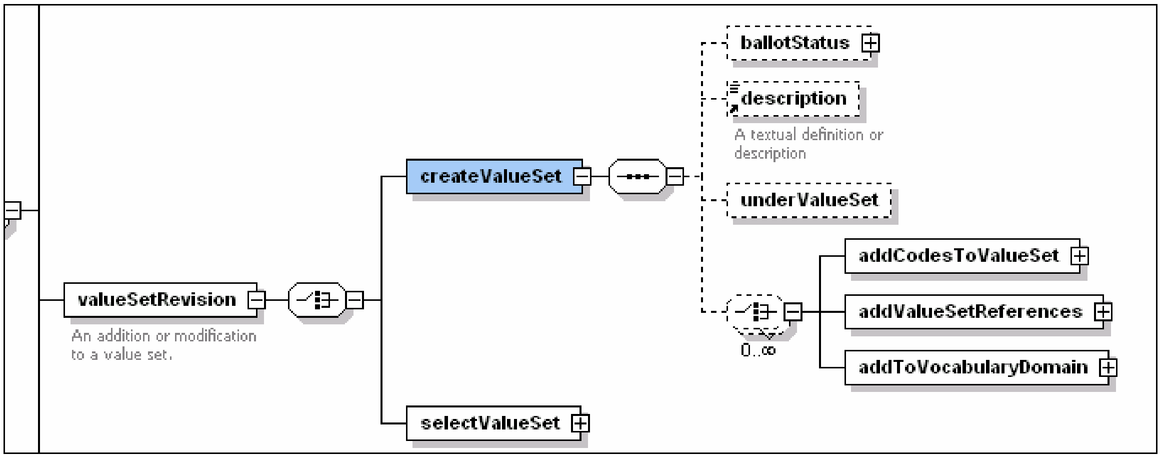
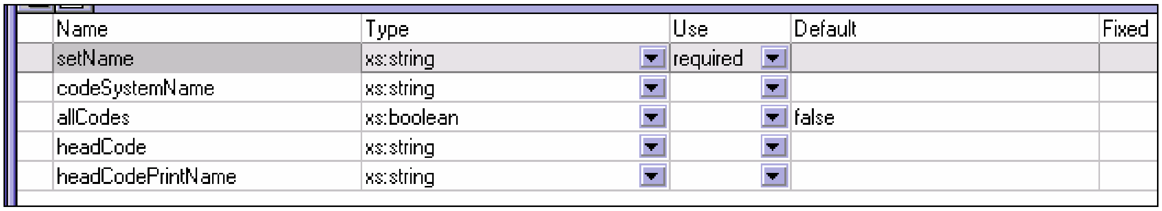


Figure 45 – createValueSet

ballotStatus is the first element defined when creating a value set. As described in section **Error! Reference source not found. Error! Reference source not found.**, it is optional except in the case when the **vocabularyRevision documentStatus** is Final, in which case all revisions have to be associated with a non- “Proposed” status.

The **description** element is also optional and, when present, provides a description of the use and purpose of the value set.



Name	Type	Use	Default	Fixed
setName	xs:string	required		
codeSystemName	xs:string			
allCodes	xs:boolean		false	
headCode	xs:string			
headCodePrintName	xs:string			

Figure 46 - createValueSet attributes

setName	The unique name of the value set
codeSystemName	The mnemonic for the code system associated with the value set (if any)
allCodes	If **codeSystemName** is supplied, this flag determines whether all of the codes in the code system are included in the value set or just selected codes.
headCode	It **codeSystemName** is supplied, this can be the concept code of the "head code" – the code that represents the entire value set.
headCodePrintName	A valid designation for the **headCode**. Optional, but validated if supplied.

Note: The current version of RoseTree requires that all value sets specify a code system name.

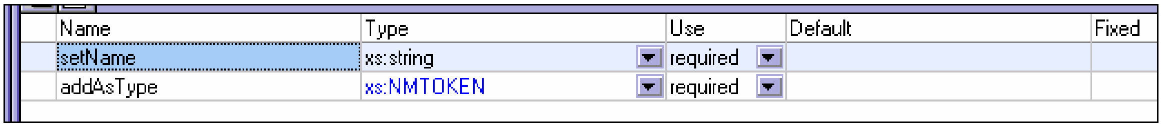
Note: The current version of RoseTree won't represent the hierarchical structure of code systems. Currently, if <u>allCodes</u> is selected and the supplied code system has a hierarchical structure, the code system will be "unraveled" into the corresponding series of value sets.

```
<valueSetRevision>
        <createValueSet  setName="OrderableBeers"
                          codeSystemName="BEERS" allCodes="true"/>
        <createValueSet setName="OrderableAles"
                          codeSystemName="BEERS" headCode="1001">
```

Figure 47 - Sample createValueSet entry

In the example above creates a value set called "OrderableBeers" and assigns all of the codes from the BEERS code system to the set. It then creates a second value set called OrderableAles that is also drawn from the BEERS code system and assigns code 1001 (ALES) as the head code for the value set.

The **underValueSet** element allows the newly created value set to be added as nested element in an already existing value set.



Name	Type	Use	Default	Fixed
setName	xs:string	required		
addAsType	xs:NMTOKEN	required		

Figure 48 - underValueSet attributes

setName	The name of an existing value set to add this new value set to
addAsType	One of *abstract* or *specializable*. *Abstract* means that the head code (if any) of the new value set is not to be considered part of **setName**. *Specializable* means that the head code (if any) is to be included in **setName**.

4.1.1 Adding Concept Codes to a Value Set

A list of concept codes may be added to a newly created value set as well as a previously existing value set.

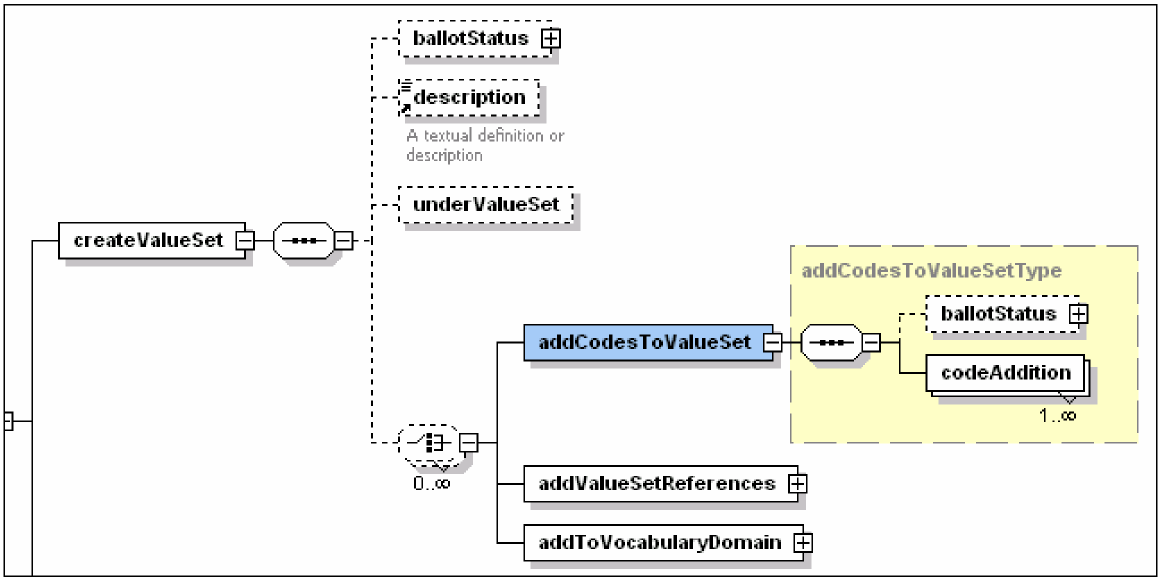
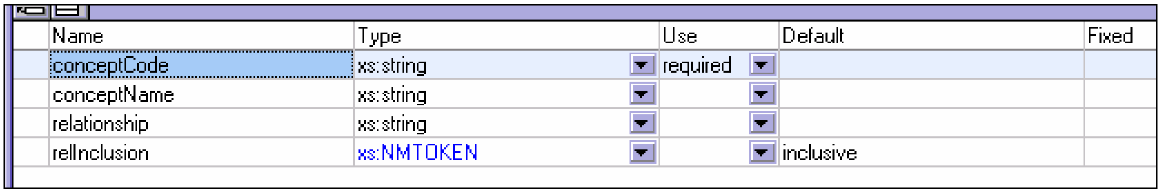


Figure 49 – addCodesToValueSet

The first element, ballot status, is optional. If supplied, it overrides any other applicable ballot status settings. Each **codeAddition** entry adds a concept code to the current value set. The order of the **codeAddition** entries is not important. A value set must have had a code system assigned to it in order to add individual concept codes.



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			
relationship	xs:string			
relInclusion	xs:NMTOKEN		inclusive	

Figure 50 - codeAddition attributes

conceptCode	The concept code to be added to the value set.
conceptName	A valid designation for the concept code. Optional, but validated if supplied
relationship	An optional relationship code. If present, it indicates that codes that are “children” of **conceptCode** are to be included in the value set as well. If omitted, only **conceptCode** itself is included. Note that “hasSubtype” is the official subsumption relationship.
relInclusion	Indicates which of the related codes are to be included. Applies only if **relationship** is supplied.

Note: The current version of RoseTree doesn’t process relationship and relInclusion settings.

relInclusion	Meaning
inclusive	**conceptCode** and all of its direct and indirect “children” are included in the value set.
exclusive	All of the direct and indirect “children” of **conceptCode** are included, but *not* conceptCode itself.
leafOnly	Only the “leaf” descendents of **conceptCode** are to be included in the value set.

Table 4 - relInclusion values

```
<valueSetRevision>
    <createValueSet setName="OrderableAles"
                         codeSystemName="BEERS" headCode="1001">
        <description>Ales that can be ordered.</description>
        <addCodesToValueSet>
            <codeAddition conceptCode="1002"/>
            <codeAddition conceptCode="1003"/>
            <codeAddition conceptCode="1004"/>
        </addCodesToValueSet>
        <addToVocabularyDomain vocabularyDomain="OrderableAles"/>
    </createValueSet>
</valueSetRevision>
```

Figure 51 - Sample addCodeToValueSet entries

The above example adds concept codes 1002, 1003 and 1004 to the OrderableAles value set.

```
<valueSetRevision>
    <createValueSet setName="SpecificAles"
                    codeSystemName="BEERS" headCode="1001">>
        <description>Ales that can be ordered.</description>
        <addCodesToValueSet>
            <codeAddition conceptCode="1001" conceptName="ALES"
                          relationship="hasSubtype" relInclusion="exclusive"/>
        </addCodesToValueSet>
        <addToVocabularyDomain vocabularyDomain="OrderableAles"/>
    </createValueSet>
</valueSetRevision>
```

Figure 52 - Sample valueSetRevision with a specified relationship

The above example adds all of the children of the “ALES” node to the “SpecificAles” value set. Note that this is subtly different than the preceding example. In the first example, codes 1001, 1002, 1003 and 1004 are the only members of the OrderableAles value set, even if new types of ales are added to the code system. The second example creates exactly the same set to start with, but would automatically acquire new members were new children added to the 1001 (ALES) node.

4.1.2 Adding References to other Value Sets

As described earlier, value sets can also reference other value sets. One way to establish this sort of reference was the **underValueSet** element of **createValueSet**. This mechanism, which was described previously, created a reference from an existing value set to a newly created set. A second way to create value set references is through the **addValueSetReference** element. **addValueSetReference** elements may occur under both the value set creation (**createValueSet**) and value set modification (**selectValueSet**) nodes.

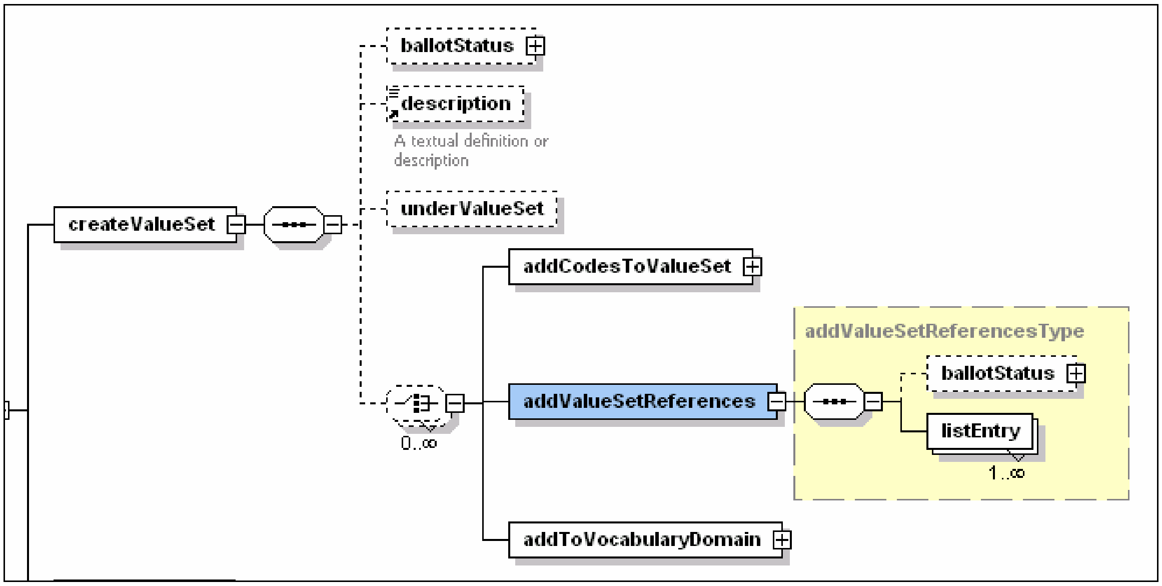
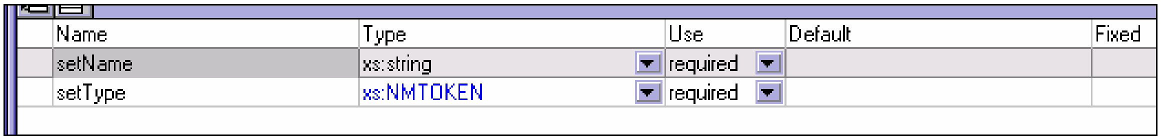


Figure 53 – addValueSetReference

Each **addValueSetReference** entry can include an optional ballot status which, if present, overrides any outer ballot status entry. **addValueSetReference** consists of one or more **listEntries**, each of which names a value set to be added to the current value set being created or modified.



Name	Type	Use	Default	Fixed
setName	xs:string	required		
setType	xs:NMTOKEN	required		

Figure 54 - listEntry attributes

setName	The name of the value set to reference.
setType	One of *abstract* or *specializable*. *Abstract* means that the head code of **setName** (if any) is not included as part of the value set being defined. *Specializable* means that the head code (if any) is to be included.

```
<valueSetRevision>
    <createValueSet setName="OrderableBeers"
                    codeSystemName="BEERS">
        <addValueSetReferences>
            <listEntry setName="SpecificAles" setType="abstract"/>
            <listEntry setName="SpecificStouts" setType="specializable"/>
        </addValueSetReferences>
    </createValueSet>
</valueSetRevision>
```

Figure 55 - Sample addValueSetReferences entry

The above example creates a value set named “OrderableBeers”, which is composed of two other value sets – “SpecificAles” and “SpecificStouts”. The head code for SpecificAles is not included in the value set, while the head code for SpecificStouts is.

4.1.3 Adding the Value Set to an Existing Vocabulary Domain

The newly created or previously selected value set can be added to one or more already existing vocabulary domains. When adding the value set it is also possible to specify in which context the value set is applicable.

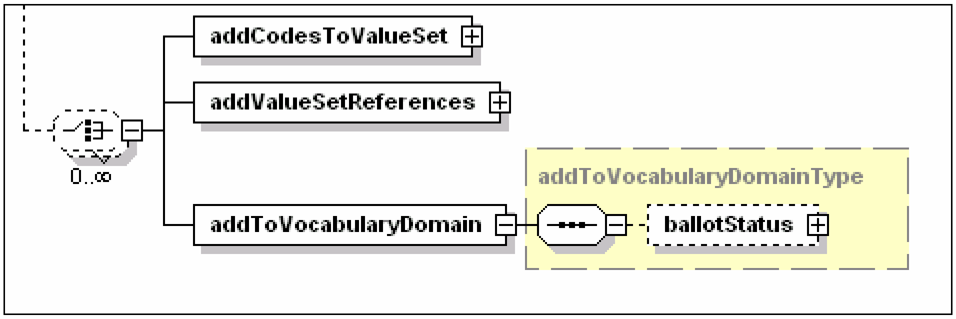
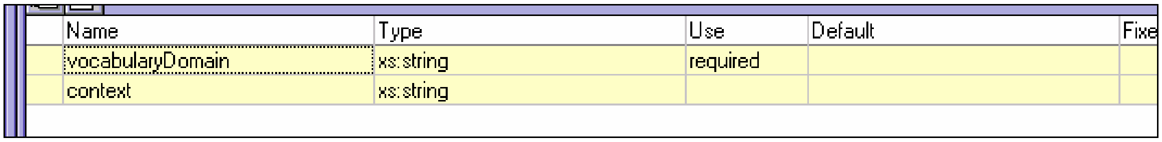


Figure 56 – addToVocabularyDomain

Like all other update elements, **addToVocabularyDomain** can specify a ballot status. If supplied, this ballot status overrides any previous applicable status.



Name	Type	Use	Default	Fixe
vocabularyDomain	xs:string	required		
context	xs:string			

Figure 57 - addToVocabularyDomain attributes

vocabularyDomain	The name of the vocabulary domain to add the value set to
context	An optional context identifier. If present, it indicates that the value set only applies in that given domain.

```
    <valueSetRevision>
        <createValueSet setName="SpecificAles" codeSystemName="BEERS"
headCode="1001">
            <description>Ales that can be ordered.</description>
            <addCodesToValueSet>
                <codeAddition conceptCode="1001" conceptName="ALES"
relationship="hasSubtype" relInclusion="exclusive"/>
            </addCodesToValueSet>
            <addToVocabularyDomain vocabularyDomain="OrderableAles"/>
        </createValueSet>
    </valueSetRevision>
```

Figure 58 - Sample addToVocabularyDomain entry

The above example adds the “SpecificAles” value set to the “OrderableAles” vocabulary domain. Note that the vocabulary domain has to exist in order for this operation to succeed.

4.2 Modifying Existing Value Sets

Previously defined value sets may be updated as well. The value set to be revised is identified by the **selectValueSet** node. All of the modifications are specified within the **selectValueSet** node.

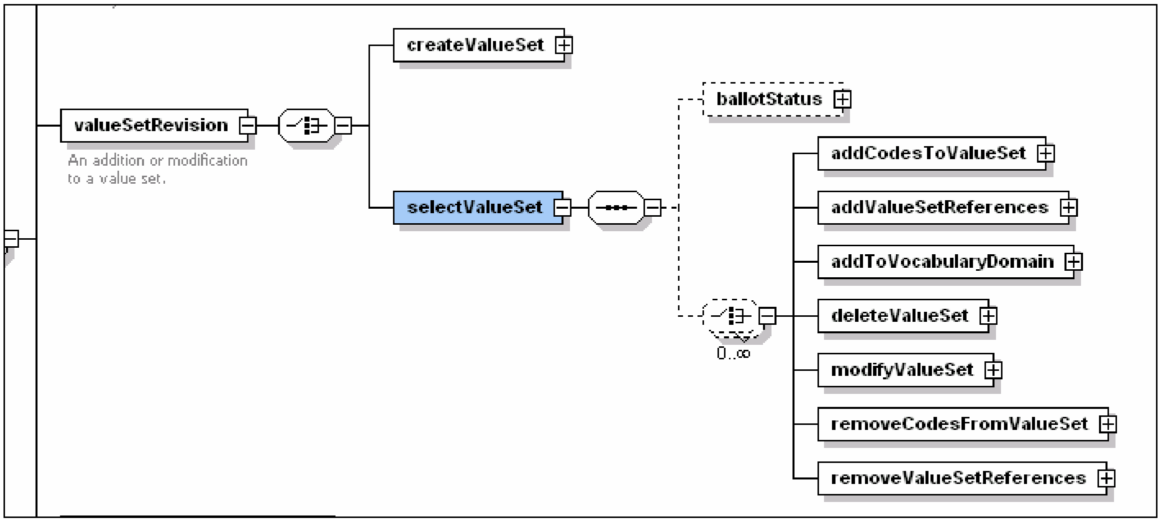
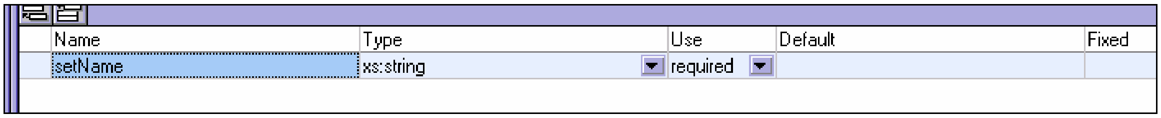


Figure 59 - selectValueSet

The first element of a **selectValueSet** node can be a ballot status (see: **Error! Reference source not found. Error! Reference source not found.**). If specified, this status applies to all operations within the **selectValueSet** element unless it is specifically overridden. **selectValueSet** has one attribute:



Name	Type	Use	Default	Fixed
setName	xs:string	required		

Figure 60 - selectValueSet attributes

setName	The name of the value set to be modified.

Concept codes or references to other value sets may be added to the selected set, and the set may be added to one or more vocabulary domains. These operations are all described under the section on creating value sets. In addition to the creation operations, the selected value set can be modified, deleted, and have references to codes or other value sets removed.

4.2.1 Deleting a Value Set

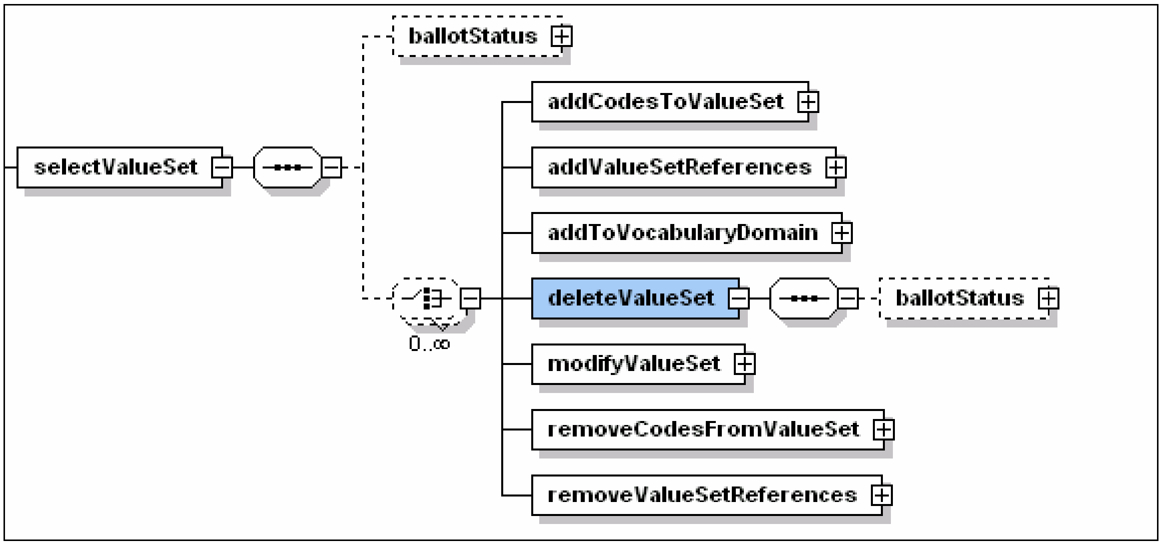


Figure 61 – deleteValueSet

deleteValueSet has no attributes. It has one optional element, the ballot status, which can be used to override an outer status.

```
<valueSetRevision>
    <selectValueSet setName="SpecificStouts">
        <deleteValueSet/>
    </selectValueSet>
</valueSetRevision>
```

Figure 62 - Sample deleteValueSet operation

4.2.2 Modifying the Definition of a Value Set

The name, code system, head code, **allCodes** setting and description of a value set can all be changed.

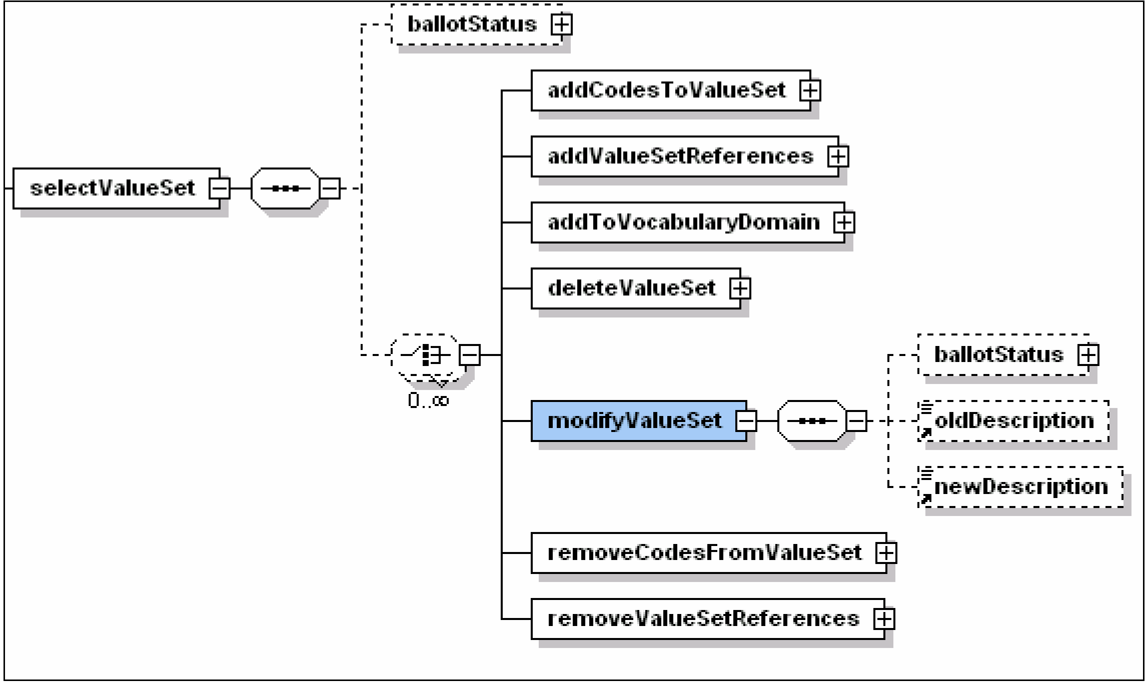
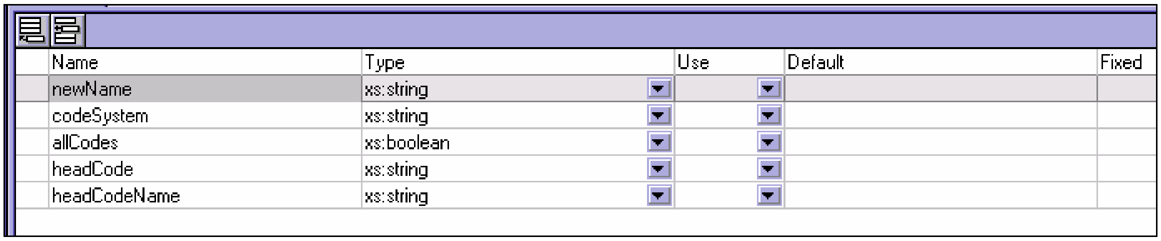


Figure 63 - modifyValueSet

modifyValueSet has three sub-elements that are all optional - the ballot status, an **oldDescription** and a **newDescription**. The ballot status needs be present only if it overrides an outer ballot status setting. If present, **oldDescription** contains the existing description for the value set if any. If present, it will be validated by comparing it with the existing database. If **newDescription** is present, the description of the value set will be updated accordingly. **modifyValueSet** also has a number of attributes, which are listed below.



Name	Type	Use	Default	Fixed
newName	xs:string			
codeSystem	xs:string			
allCodes	xs:boolean			
headCode	xs:string			
headCodeName	xs:string			

Figure 64 - modifyValueSet attributes

newName	If present, the name of the selected value set will be changed to **newName**.

codeSystem	If present, the code system associated with the value set will be updated to this value.
allCodes	If present, the **allCodes** setting will be changed to reflect this value
headCode	If present, the head code will be changed to this value (or deleted if an empty string is supplied
headCodeName	A designation for **headCode**. Optional but if supplied it will be validated against **headCode**.

```
    <valueSetRevision>
        <selectValueSet setName="OrderableBeers">
            <modifyValueSet allCodes="false" codeSystem="">
                <newDescription>Beers that can actually be ordered at an HL7
meeting</newDescription>
            </modifyValueSet>
        </selectValueSet>
    </valueSetRevision>
```

Figure 65 - Sample modifyValueSet entry

The above example:

1) Changes the **allCodes** setting on “OrderableBeers” to false
2) Removes any code system that may currently be specified
3) Updates or adds a description.

4.2.3 Removing Concept Codes from a Value Set

Concept codes that are currently associated with a value set can be removed.

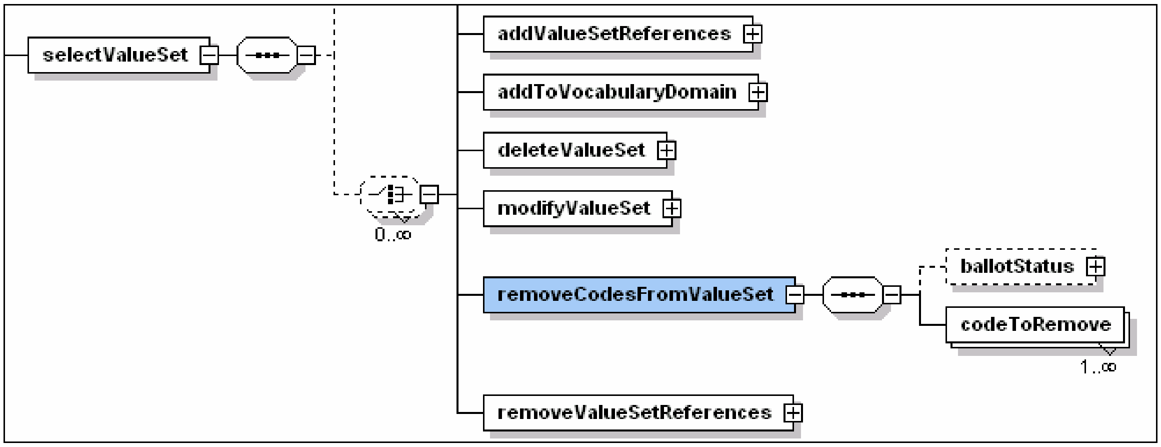
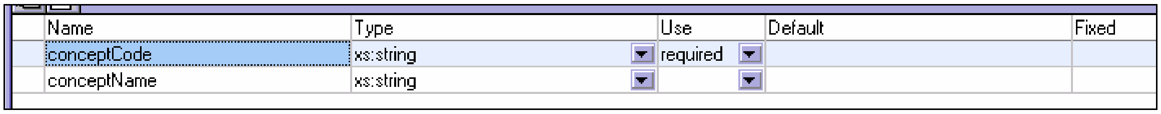


Figure 66 – removeCodesFromValueSet

removeCodesFromValueSet contains an optional ballot status followed by a list of codes to be removed. Each **codeToRemove** element has two attributes:



Name	Type	Use	Default	Fixed
conceptCode	xs:string	required		
conceptName	xs:string			

Figure 67 – codeToRemove attributes

conceptCode	The coded concept to remove from the value set. If the coded concept referenced other concepts via a relationship, those references will be removed as well.
conceptName	A valid designation for **conceptCode**. Validated if supplied.

```
<valueSetRevision>
    <selectValueSet setName="OrderableAles">
        <removeCodesFromValueSet>
            <codeToRemove conceptCode="1003" conceptName="BITTER ALE"/>
        </removeCodesFromValueSet>
    </selectValueSet>
</valueSetRevision>
```

Figure 68 - Sample of removeCodesFromValueSet

The above example removes concept code 1003 (BITTER ALE) from the “OrderableAles” value set.

4.2.4 Removing References to Other Value Sets

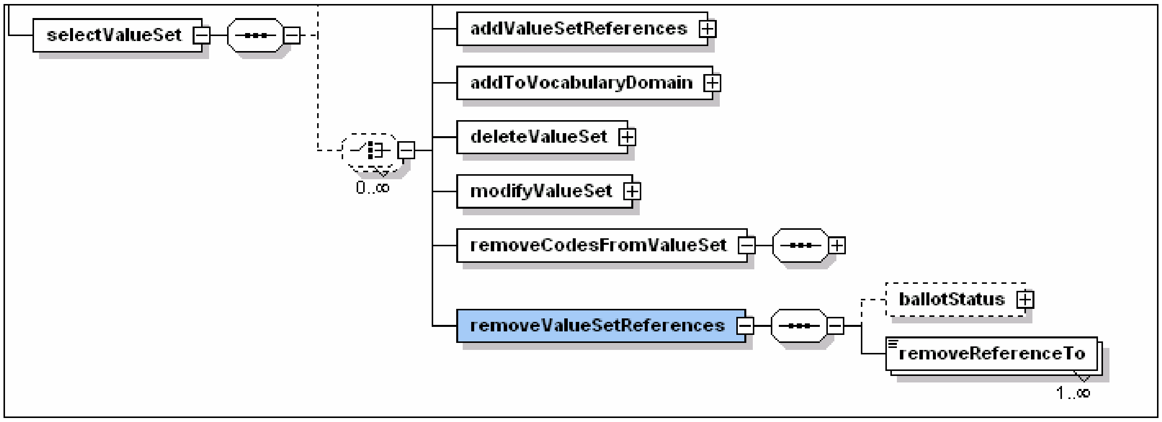
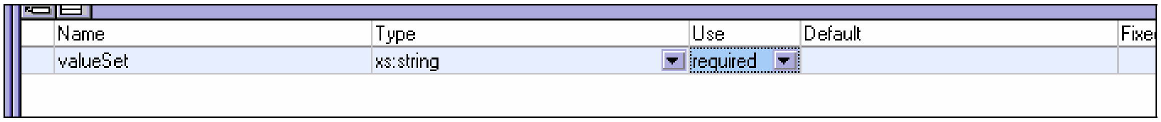


Figure 69 – removeValueSetReferences

removeValueSetReferences removes one or more references to other value sets from the selected set. As with any modification operation, it has an optional ballot status that can be used to specifically state the status for this item. It then contains a list of **removeReferenceTo** elements, each of which identifies a value set reference to be removed.



Name	Type	Use	Default	Fixe
valueSet	xs:string	required		

Figure 70 - removeReferenceTo attributes

Each **removeReferenceTo** element has one attribute **– valueSet** , which is the name of the value set to be removed.

4.3 Working with Vocabulary Domains

The vast majority of the vocabulary domains will be created during the RIM modeling process. The following operation can be used in cases where the domain doesn't already exist in the RIM model.

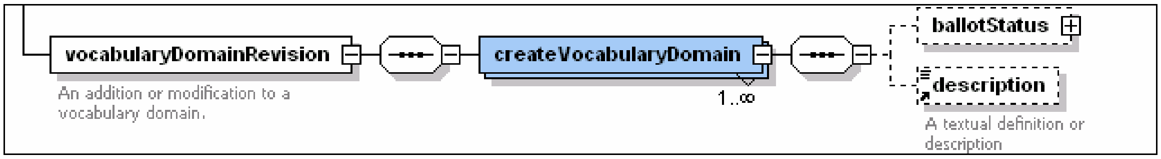
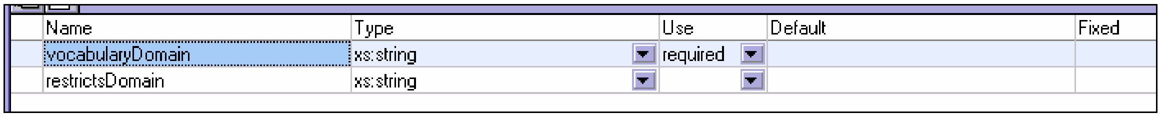

vocabularyDomainRevision
An addition or modification to a vocabulary domain.
createVocabularyDomain
1..∞
ballotStatus
description
A textual definition or description

Figure 71 – createVocabularyDomain

The creation of a new domain needs to be balloted, so createVocabularyDomain has an optional ballotStatus element (see: **Error! Reference source not found. Error! Reference source not found.** for details). The optional description is used to supply a description of the vocabulary domain. createVocabularyDomain has the following attributes



Name	Type	Use	Default	Fixed
vocabularyDomain	xs:string	required		
restrictsDomain	xs:string			

Figure 72 - createVocabularyDomain attributes

vocabularyDomain	The name of the new vocabulary domain to be created
restrictsDomain	The name of the parent vocabulary domain (optioinal).

```
<vocabularyDomainRevision>
    <createVocabularyDomain vocabularyDomain="OrderableBeers"/>
</vocabularyDomainRevision>
<vocabularyDomainRevision>
    <createVocabularyDomain vocabularyDomain="OrderableAles"
                                restrictsDomain="OrderableBeers"/>
 </vocabularyDomainRevision>
```

Figure 73 - Sample createVocabularyDomain entries

The example above creates two vocabulary domains – OrderableBeers and OrderableAles. OrderableAles is defined as a proper subset (restriction or constraint) on OrderableBeers.

Appendix A – Beer Classification Example

This example was copied verbatim from the Birra Castello web site - http://www.birracastello.com/html/masgustoeng.html

Natural amber, brown and clear beers.

ALE
Is the name which was given to a brew drink acquire from the ancient Celts
Anglo-Saxon origin but today synonym to high brew beer.

> **PALE ALE**
> Light amber colouring, with, abundant hop and fairly dry. Slightly bitter. Below 5° alcoholic contents. Served at cellar temperature.
> **BITTER ALE**
> Very dry beer and decisively bitter. Distinct malt taste and aroma. Dark coppery colour. Low carbon dioxide contents. 3-4 alcoholic contents. Traditionally drafted.
> **LIGHT ALE**
> Lower alcoholic contents than the bitter. Traditionally dunk out of a bottle.
> **MILD ALE**
> Well mature and slight hop taste. Pleasant taste with a trace of caramel and malt. Fairly light, 3/3.5 alcoholic contents. Normally with a dark amber colouring, and occasionally with a lighter coppery tonality.
> Almost always draft and served at cellar temperature.
> **BROWN ALE**
> Bottled version of the Mild Ale.
> **STRONG ALE**
> Strong robust with a full round smooth taste. Dark colouring 5/6 alcoholic contents.
> **INDIAN PALE ALE**
> Stong beer with a very strong hop taste. Served mostly by bottle. Around 6°.
> **SCOTCH ALE**
> Scotch beer, undoubtedly strong and dense. Dark brown almost black colouring, at 8/9 alcoholic contents very thick and an intense malt flavour.
> **BARLEY WINE**
> It is so called because it is a stronger ‘Strong'. Fruity wine taste. Reddish colour and very dense.

PORTER
Originated in London. Both it's dark brown colour and it's characteristic bitter taste is due to the roasted barley and not to the malt. The bitter taste increased with the high hop quantities which makes it also sour.

STOUT
It is thicker and richer than the Porter, also made with roasted barley and not malt. National Irish beer and very sour, dark with a rich creamy froth.

> **BITTER STOUT**
> Very dry and distinctively bitter. Only sold in Ireland.
> **SWEET STOUT**
> Dark brown colour with a sweeter taste ,malted, slightly spicy.
> **MILK STOUT**

Sweet stout version with added milk sugar to soften the sour taste.

IMPERIAL STOUT

Dark brown colour with a taste between dry and mildly bitter , slightly fruity.High alcoholic contents.

TRAPPIST

Highly brewed beer , produced by Trappist Cistercian monks with 95% amber malt and 5% roasted caramel malt. Plenty hop, fairly strong. It is bottled before the brewing has terminated. It is re-brewed by adding fresh yeast. It's colour is between amber and dark brown. Malt taste, spicy, wine like, fruity.

BIRRE D'ABBAZIA

Synonym to quality and tradition. Currently none are produced in abbeys by monks. There are many varieties with vast differences between each other. Often very dense and re-brewed in bottle.

SPECIALI BELGHE

Groups beers of all types. All of high brew.

BIRRE BIANCHE BELGHE

Speciality of the Lovian Region. Originally produced with 45% barley, 45% wheat and 10% oat. Currently produced by only barley and wheat. Cool taste, refreshing, fruity, mildly sour. Slightly doughy. The name comes from cloudiness which gives a milky white colouring.

BIERE DE GARDE (FRENCH)

North France speciality, renowned for its lengthy cellar maturation in barrel and wooden tubs. Presented in vine bottles with cork tap. These cellar beers are produced from a mixture of light and dark malt from Munich. A intense amber colouring, with 5/6 alcoholic contents.

KRIEK

A barley based beer from Belgium with soaked cherries.

ALT BIER

This is the German equivalent of Ale. Produced in the Düsseldorf area, and one of the few high brew German beers. Own its character to dark malt and large quantity of hop. Full and hop taste. A fruity after taste. Not too alcoholic only 4°.

KÖLSCH

A typical beer from Koln. A clear golden blond colouring, dry hop taste, slightly fizzy. In all mildly sour with decisive hop flavour however smooth. It is a beer that makes hungry and helps digest, which explains why the Germans have it as aperitif.

BIRRE TEDESCHE DI FRUMENTO

During the production process, with the barley, in variable proportions, wheat malt is added which gives the beer a distinctive aroma.

BERLINER WEISSE

Litterally means "Berliner White". In reality it only very pale at times opaque with a candid froth. Produced with a quarter malt wheat and three quarters barley malt. A cooked "lattobacilli" is added to induce a second brewing.

Low alcoholic contents , about 2.5°. Renowned sour flavour, mildly bitter.

WEIZENBIER

Produced in Bavaria with 60-70% wheat malt. Non much hop, very refreshing, light and sweet.

HEFE WEIZEN

Non filtered light coloured beer, brewed in bottle.
DUNKEL WEIZEN
Dark coloured beer. Malt and full flavour.

Appendix B – Beer Classification Example in XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Harold R. Solbrig (Mayo Clinic) -->
<VocabularyRevision xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="VocabularyRevision.xsd">
    <editDescription creationDate="2003-05-21" primaryContact="Harold Solbrig" committee="C21">
        <description>Example of creating a complete new code system of beers.</description>
    </editDescription>
    <ballotStatus action="Passed" vote="17-0-0"/>
    <codeSystemRevision>
        <registerCodeSystem codeSystemName="Castello Beer Classification" codeSystemMnemonic="BEER">
            <description>     Natural amber, brown and clear beers.</description>
            <addCodesToCodeSystem>
                <newCode conceptCode="ALE" conceptName="Ale">
                    <description>Is the name which was given to a brew drink acquire from the ancient Celts. Anglo-
Saxon origin but today synonym to high brew beer.</description>
                    <newCode conceptCode="PALE ALE" conceptName="Pale Ale">
                        <description>Light amber colouring, with, abundant hop and fairly dry. Slightly bitter. Below 5°
alcoholic contents. Served at cellar temperature.</description>
                    </newCode>
                    <newCode conceptCode="BITTER ALE" conceptName="Bitter Ale">
                        <description>Very dry beer and decisively bitter. Distinct malt taste and aroma. Dark coppery
colour. Low carbon dioxide contents. 3-4 alcoholic contents. Traditionally drafted.</description>
                    </newCode>
                    <newCode conceptCode="LIGHT ALE" conceptName="Light Ale">
                        <description>Lower alcoholic contents than the bitter. Traditionally dunk out of a
bottle.</description>
                    </newCode>
                    <newCode conceptCode="MILD ALE" conceptName="Mile Ale">
                        <description>Well mature and slight hop taste. Pleasant taste with a trace of caramel and malt.
Fairly light, 3/3.5 alcoholic contents. Normally with a dark amber colouring, and occasionally with a lighter coppery
tonality. Almost always draft and served at cellar temperature.</description>
                    </newCode>
                    <newCode conceptCode="BROWN ALE" conceptName="Brown Ale">
                        <description>Bottled version of the Mild Ale.</description>
                    </newCode>
                    <newCode conceptCode="STRONG ALE" conceptName="Strong Ale">
                        <description>Strong robust with a full round smooth taste. Dark colouring 5/6 alcoholic
contents.</description>
                    </newCode>
                    <newCode conceptCode="INDIAN PALE ALE" conceptName="Indian Pale Ale">
                        <description>Stong beer with a very strong hop taste. Served mostly by bottle. Around
6°.</description>
                    </newCode>
                    <newCode conceptCode="SCOTCH ALE" conceptName="Scotch Ale">
                        <description>Scotch beer, undoubtedly strong and dense. Dark brown almost black colouring, at
8/9 alcoholic contents very thick and an intense malt flavour.</description>
                    </newCode>
                    <newCode conceptCode="BARLEY WINE" conceptName="Barley Wine">
                        <description>It is so called because it is a stronger `Strong'. Fruity wine taste. Reddish colour
and very dense.</description>
                    </newCode>
                </newCode>
                <newCode conceptCode="PORTER" conceptName="Porter">
                    <description>Originated in London. Both it's dark brown colour and it's characteristic bitter taste is
due to the roasted barley and not to the malt. The bitter taste increased with the high hop quantities which makes it also
sour.</description>
                </newCode>
                <newCode conceptCode="STOUT" conceptName="Stout">
                    <description>It is thicker and richer than the Porter, also made with roasted barley and not malt.
National Irish beer and very sour, dark with a rich creamy froth.</description>
                    <newCode conceptCode="BITTER STOUT" conceptName="Bitter Stout">
                        <description>Very dry and distinctively bitter. Only sold in Ireland.</description>
                    </newCode>
                    <newCode conceptCode="SWEET STOUT" conceptName="Sweet Stout">
                        <description>Dark brown colour with a sweeter taste ,malted, slightly spicy.</description>
                    </newCode>
                    <newCode conceptCode="MILK STOUT" conceptName="Milk Stout">
                        <description>Sweet stout version with added milk sugar to soften the sour taste.</description>
                    </newCode>
                    <newCode conceptCode="IMPERIAL STOUT" conceptName="Imperial Stout">
                        <description>Dark brown colour with a taste between dry and mildly bitter , slightly fruity.High
alcoholic contents.</description>
                    </newCode>
                </newCode>
                <newCode conceptCode="TRAPPIST" conceptName="Trappist">
                    <description>Highly brewed beer , produced by Trappist Cistercian monks with 95% amber malt and
5% roasted caramel malt. Plenty hop, fairly strong. It is bottled before the brewing has terminated. It is re-brewed by
adding fresh yeast. It's colour is between amber and dark brown. Malt taste, spicy, wine like, fruity.</description>
                </newCode>
                <newCode conceptCode="BIRRE D'ABBAZIA" conceptName="Birre D'Abbazia">
                    <description>Synonym to quality and tradition. Currently none are produced in abbeys by monks.
There are many varieties with vast differences between each other. Often very dense and re-brewed in
bottle.</description>
                </newCode>
                <newCode conceptCode="SPECIALI BELGHE" conceptName="Speciali Belghe">
                    <description>Groups beers of all types. All of high brew.</description>
                </newCode>
                <newCode conceptCode="BIRRE BIANCHE BELGHE" conceptName="Birre Bianche Belghe">
                    <description>Speciality of the Lovian Region. Originally produced with 45% barley, 45% wheat and
10% oat. Currently produced by only barley and wheat. Cool taste, refreshing, fruity, mildly sour. Slightly doughy. The
name comes from cloudiness which gives a milky white colouring.</description>
                </newCode>
                <newCode conceptCode="BIERE DE GARDE" conceptName="Biere De Garde (French)">
                    <description>North France speciality, renowned for its lengthy cellar maturation in barrel and
wooden tubs. Presented in vine bottles with cork tap. These cellar beers are produced from a mixture of light and dark
malt from Munich. A intense amber colouring, with 5/6 alcoholic contents.</description>
                </newCode>
                <newCode conceptCode="KRIEK" conceptName="Kriek">
                    <description>A barley based beer from Belgium with soaked cherries.</description>
                </newCode>
                <newCode conceptCode="ALT BIER" conceptName="Alt Bier">
                    <description>This is the German equivalent of Ale. Produced in the Düsseldorf area, and one of the
few high brew German beers. Own its character to dark malt and large quantity of hop. Full and hop taste. A fruity after
taste. Not too alcoholic only 4°.</description>
                </newCode>
                <newCode conceptCode="KÖLSCH" conceptName="Kölsch">
                    <description>A typical beer from Koln.  A clear golden blond colouring, dry hop taste, slightly fizzy. In
all mildly sour with decisive hop flavour however smooth. It is a beer that makes hungry and helps digest, which explains
why the Germans have it as aperitif.</description>
                </newCode>
                <newCode conceptCode="BIRRE TEDESCHE DI FRUMENTO" conceptName="Birre Tedesche Di
Frumento">
                    <description>During the production process, with the barley, in variable proportions, wheat malt is
added which gives the beer a distinctive aroma.</description>
                    <newCode conceptCode="BERLINER WEISSE" conceptName="Berliner Weisse">
                        <description>Litterally means "Berliner White". In reality it only very pale at times opaque with a
candid froth. Produced with a quarter malt wheat and three quarters barley malt. A cooked "lattobacilli" is added to induce
a second brewing.
Low alcoholic contents , about 2.5°. Renowned sour flavour, mildly bitter.</description>
                    </newCode>
                    <newCode conceptCode="WEIZENBIER" conceptName="Weizenbier">
                        <description>Produced in Bavaria with 60-70% wheat malt. Non much hop, very refreshing, light
and sweet.</description>
                    </newCode>
                    <newCode conceptCode="HEFE WEIZEN" conceptName="Hefe Weizen">
                        <description>Non filtered light coloured beer, brewed in bottle.</description>
                    </newCode>
                    <newCode conceptCode="DUNKEL WEIZEN" conceptName="Dunkel Weizen">
                        <description>Dark coloured beer. Malt and full flavour.</description>
                    </newCode>
                </newCode>
            </addCodesToCodeSystem>
        </registerCodeSystem>
    </codeSystemRevision>
    <vocabularyDomainRevision>
        <createVocabularyDomain vocabularyDomain="Beer"/>
    </vocabularyDomainRevision>
    <valueSetRevision>
```

```
        <createValueSet setName="Beer" codeSystemName="BEER" allCodes="true">
            <addToVocabularyDomain vocabularyDomain="Beer"/>
        </createValueSet>
    </valueSetRevision>
</VocabularyRevision>
```

Appendix C – Complete Listing of Examples Used in Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Harold R. Solbrig (Mayo Clinic) -->
<VocabularyRevision xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="VocabularyRevision.xsd">
    <!-- A new sequences of edits -->
    <editDescription creationDate="2003-04-28" primaryContact="HRS" committee="C21" documentStatus="Proposed">
        <description>Specification of the orderable beer codes for the HL7 drink ordering message system</description>
    </editDescription>
    <editVersion changeDate="2003-05-12" author="Harold Solbrig">
        <description>Enhanced examples</description>
    </editVersion>
    <editVersion changeDate="2003-05-19" author="Harold Solbrig">
        <description>Adjustments for the revised template</description>
    </editVersion>
    <!-- Create a new beer code system and add codes -->
    <codeSystemRevision>
        <registerCodeSystem codeSystemName="Beer and Flavor Classification" codeSystemMnemonic="BEERS">
            <!-- This registration action has passed committee vote -->
            <ballotStatus action="Passed" vote="17-4-1"/>
            <description>A beer and flavor classification derived from the Castello web site:
http://www.birracastello.com/html/masgustoeng.html</description>
            <addCodesToCodeSystem>
                <ballotStatus action="Proposed"/>
                <newCode conceptCode="1001" conceptName="ALE">
                    <description>The name which was given to a brew drink acquire from the ancient Celts Anglo-Saxon
origin but today synonym to high brew beer.</description>
                    <newCode conceptCode="1002" conceptName="PALE ALE">
                        <description>Light amber colouring, with, abundant hop and fairly dry. Slightly bitter. Below 5ø
alcoholic contents. Served at cellar temperature.</description>
                    </newCode>
                    <newCode conceptCode="1003" conceptName="BITTER ALE">
                        <description>Very dry beer and decisively bitter. Distinct malt taste and aroma. Dark coppery
colour. Low carbon dioxide contents. 3-4 alcoholic contents. Traditionally drafted.
                        </description>
                    </newCode>
                    <newCode conceptCode="1004" conceptName="LIGHT ALE">
                        <description>Lower alcoholic contents than the bitter. Traditionally dunk out of a
bottle.</description>
                    </newCode>
                </newCode>
                <newCode conceptCode="1005" conceptName="PORTER">
                    <description>Originated in London. Both it's dark brown colour and it's characteristic bitter taste is
due to the roasted barley and not to the malt. The bitter taste increased with the high hop quantities which makes it also
sour.</description>
                </newCode>
                <newCode conceptCode="1006" conceptName="STOUT">
                    <description>It is thicker and richer than the Porter, also made with roasted barley and not malt.
National Irish beer and very sour, dark with a rich creamy froth.</description>
                </newCode>
                <newCode conceptCode="1007" conceptName="BITTER STOUT">
                    <description>Very dry and distinctively bitter. Only sold in Ireland</description>
                </newCode>
            </addCodesToCodeSystem>
            <addPrintNameToCode conceptCode="1006" newPrintName="Stoot" isPreferred="false"/>
            <addPropertyToCode conceptCode="1007" conceptName="BITTER STOUT" propertyId="appliesTo">
                <ballotStatus action="Withdrawn" vote="3-12-1"/>
                <property>Colds and whatever else ails you</property>
            </addPropertyToCode>
            <addConceptRelationship parentCode="1002" parentName="Pale Ale" relationship="smallerThan"
childCode="1003" childName="BITTER ALE"/>
        </registerCodeSystem>
    </codeSystemRevision>
    <!-- Example change to code system name -->
    <codeSystemRevision>
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <ballotStatus action="PassedWithChanges" vote="12-1-0">
                <note>Author will sample all changes first</note>
            </ballotStatus>
            <modifyCodeSystem codeSystemName="Castello Beer and Flavor Classification"/>
        </selectCodeSystem>
    </codeSystemRevision>
    <!-- Add codes under an existing node -->
    <codeSystemRevision>
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <addCodesToCodeSystem>
                <underCode conceptCode="1006" conceptName="STOUT">
                    <newCode conceptCode="1008" conceptName="SWEET STOUT">
                        <description>Dark brwon colour with a sweeter taste, malted, slightly spicy.  </description>
                    </newCode>
                    <newCode conceptCode="1009" conceptName="MILK STOUT">
                        <description>Sweet stout version with added milk sugar to soften the sour taste.
    </description>
                    </newCode>
                </underCode>
            </addCodesToCodeSystem>
        </selectCodeSystem>
    </codeSystemRevision>
    <!-- Switch the SWEET STOUT and STOUT positions - making STOUT a kind of SWEET STOUT -->
    <codeSystemRevision>
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <moveCode conceptCode="1008" conceptName="SWEET STOUT" fromParentCode="1006"
fromParentName="STOUT"/>
            <moveCode conceptCode="1006" toParentCode="1008" toParentName="SWEET STOUT"/>
        </selectCodeSystem>
    </codeSystemRevision>
    <!-- Examples of print name revisions & deletions -->
    <codeSystemRevision>
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <addPrintNameToCode conceptCode="1004" conceptName="LIGHT ALE" newPrintName="Pils"
languageCode="de"/>
            <addPrintNameToCode conceptCode="1004" newPrintName="Lager" languageCode="de"
isPreferred="false"/>
        </selectCodeSystem>
    </codeSystemRevision>
    <codeSystemRevision>
        <selectCodeSystem codeSystemMnemonic="BEERS">
            <updateCodePrintName conceptCode="1004" oldPrintName="LIGHT ALE" newPrintName="Light Ale"/>
            <updateCodePrintName conceptCode="1004" oldPrintName="Pils" newPrintName="Kölsch"
languageCode="de"/>
            <updateCodePrintName conceptCode="1004" oldPrintName="Lager" newPrintName="Lager"
languageCode="de" isPreferred="true"/>
            <updateCodePrintName conceptCode="1004" oldPrintName="Kölsch" newPrintName=""
languageCode="de"/>
        </selectCodeSystem>
    </codeSystemRevision>
    <!-- Create some vocabulary domains -->
    <vocabularyDomainRevision>
        <createVocabularyDomain vocabularyDomain="OrderableBeers"/>
    </vocabularyDomainRevision>
    <vocabularyDomainRevision>
        <createVocabularyDomain vocabularyDomain="OrderableAles" restrictsDomain="OrderableBeers"/>
    </vocabularyDomainRevision>
    <!-- Make a couple of value sets -->
    <valueSetRevision>
        <createValueSet setName="SpecificAles" codeSystemName="BEERS" headCode="1001">
            <description>Ales that can be ordered.</description>
            <addCodesToValueSet>
                <codeAddition conceptCode="1001" conceptName="ALE" relationship="hasSubtype"
relInclusion="exclusive"/>
            </addCodesToValueSet>
            <addToVocabularyDomain vocabularyDomain="OrderableAles"/>
        </createValueSet>
    </valueSetRevision>
    <valueSetRevision>
        <createValueSet setName="SpecificStouts" codeSystemName="BEERS" headCode="1006">
            <description>Stouts that can be ordered.</description>
            <addCodesToValueSet>
                <codeAddition conceptCode="1006" conceptName="STOUT" relationship="hasSubtype"
relInclusion="exclusive"/>
            </addCodesToValueSet>
            <addToVocabularyDomain vocabularyDomain="OrderableAles"/>
        </createValueSet>
    </valueSetRevision>
    <valueSetRevision>
        <createValueSet setName="OrderableBeers" codeSystemName="BEERS" allCodes="true">
            <addValueSetReferences>
                <listEntry setName="SpecificAles" setType="abstract"/>
                <listEntry setName="SpecificStouts" setType="specializable"/>
            </addValueSetReferences>
            <addToVocabularyDomain vocabularyDomain="OrderableBeers"/>
        </createValueSet>
    </valueSetRevision>
    <valueSetRevision>
        <createValueSet setName="OrderableAles" codeSystemName="BEERS" headCode="1001">
            <description>Ales that can be ordered.</description>
            <addCodesToValueSet>
                <codeAddition conceptCode="1002"/>
                <codeAddition conceptCode="1003"/>
                <codeAddition conceptCode="1004"/>
            </addCodesToValueSet>
            <addToVocabularyDomain vocabularyDomain="OrderableAles"/>
        </createValueSet>
    </valueSetRevision>
    <valueSetRevision>
        <selectValueSet setName="SpecificStouts">
            <deleteValueSet/>
        </selectValueSet>
    </valueSetRevision>
    <valueSetRevision>
        <selectValueSet setName="OrderableBeers">
            <modifyValueSet allCodes="false" codeSystem="">
                <oldDescription/>
                <newDescription>Beers that can actually be ordered at an HL7 meeting</newDescription>
            </modifyValueSet>
        </selectValueSet>
    </valueSetRevision>
    <valueSetRevision>
        <selectValueSet setName="OrderableAles">
            <removeCodesFromValueSet>
                <codeToRemove conceptCode="1003" conceptName="BITTER ALE"/>
            </removeCodesFromValueSet>
        </selectValueSet>
    </valueSetRevision>
</VocabularyRevision>
```

Appendix D – VocabularyRevision Schema Listing

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com) by Harold R. Solbrig (Mayo Clinic) -
->
<!--W3C Schema generated by XMLSPY v5 rel. 4 U (http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="VocabularyRevision">
        <xs:annotation>
            <xs:documentation>A cohesive collection of changes for a
terminology</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="editDescription">
                    <xs:annotation>
                        <xs:documentation>A description of the intent and purpose of the vocabulary
revision.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="description"/>
                        </xs:sequence>
                        <xs:attribute name="creationDate" type="xs:date" use="required"/>
                        <xs:attribute name="primaryContact" type="xs:string" use="required"/>
                        <xs:attribute name="committee" type="xs:string" use="required"/>
                        <xs:attribute name="documentStatus" default="Proposed">
                            <xs:annotation>
                                <xs:documentation>The current state of the
proposition</xs:documentation>
                            </xs:annotation>
                            <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                    <xs:enumeration value="Proposed"/>
                                    <xs:enumeration value="Submitted"/>
                                    <xs:enumeration value="Reviewed"/>
                                    <xs:enumeration value="Harmonized"/>
                                    <xs:enumeration value="Final"/>
                                    <xs:enumeration value="Rejected"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:attribute>
                    </xs:complexType>
                </xs:element>
                <xs:element name="editVersion" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Edit history</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="description" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute name="changeDate" type="xs:date" use="required"/>
                        <xs:attribute name="author" type="xs:string" use="required"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
                <xs:choice maxOccurs="unbounded">
                    <xs:element name="codeSystemRevision">
                        <xs:annotation>
                            <xs:documentation>An addition or modification to a code
system.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:choice>
                                <xs:element name="registerCodeSystem">
                                    <xs:annotation>
                                        <xs:documentation>Register a new code system and
contents.</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus" type="ballotStatusType"
minOccurs="0"/>
                                            <xs:element ref="description" minOccurs="0"/>
                                            <xs:choice minOccurs="0" maxOccurs="unbounded">
                                                <xs:element name="addCodesToCodeSystem"
type="addCodesToCodeSystemType"/>
                                                <xs:element name="addPrintNameToCode"
type="addPrintNameToCodeType"/>
                                                <xs:element name="addPropertyToCode"
type="addPropertyToCodeType"/>
                                                <xs:element name="addConceptRelationship"
type="addConceptRelationshipType"/>
                                            </xs:choice>
                                        </xs:sequence>
                                        <xs:attribute name="codeSystemName" type="xs:string"
use="required"/>
                                        <xs:attribute name="codeSystemMnemonic" type="xs:string"
use="required"/>
                                        <xs:attribute name="codeSystemType" default="I">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:NMTOKEN">
                                                    <xs:enumeration value="E"/>
                                                    <xs:enumeration value="I"/>
                                                    <xs:enumeration value="EI"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:attribute>
                                        <xs:attribute name="codeSystemOID" type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="selectCodeSystem">
                                    <xs:annotation>
                                        <xs:documentation>Modify the contents of an existing code
system.</xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus" type="ballotStatusType"
minOccurs="0"/>
                                            <xs:choice minOccurs="0" maxOccurs="unbounded">
                                                <xs:element name="addCodesToCodeSystem"
type="addCodesToCodeSystemType"/>
                                                <xs:element name="addPrintNameToCode"
type="addPrintNameToCodeType"/>
                                                <xs:element name="addPropertyToCode"
type="addPropertyToCodeType"/>
                                                <xs:element name="addConceptRelationship"
type="addConceptRelationshipType"/>
                                                <xs:element name="modifyCodeSystem">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                                            <xs:element ref="description"
minOccurs="0"/>
                                                        </xs:sequence>
                                                        <xs:attribute name="codeSystemName"

type="xs:string"/>
                                                    </xs:complexType>
                                                </xs:element>
                                                <xs:element name="updateCodePrintName">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                                        </xs:sequence>
                                                        <xs:attribute name="conceptCode"
type="xs:string" use="required"/>
                                                        <xs:attribute name="oldPrintName"
type="xs:string" use="required"/>
                                                        <xs:attribute name="newPrintName"
type="xs:string"/>
                                                        <xs:attribute name="languageCode"
type="xs:string" default="en"/>
                                                        <xs:attribute name="isPreferred"
type="xs:boolean" default="true"/>
                                                    </xs:complexType>
                                                </xs:element>
                                                <xs:element name="updateCodeDescription">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                                            <xs:sequence>
                                                                <xs:element ref="oldDescription"
minOccurs="0"/>
                                                                <xs:element ref="newDescription"/>
                                                            </xs:sequence>
                                                        </xs:sequence>
                                                        <xs:attribute name="conceptCode"
type="xs:string" use="required"/>
                                                        <xs:attribute name="conceptName"
type="xs:string"/>
```

```
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="moveCode">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="conceptCode"
type="xs:string" use="required"/>
                                        <xs:attribute name="conceptName"
type="xs:string"/>
                                        <xs:attribute name="fromParentCode"
type="xs:string"/>
                                        <xs:attribute name="fromParentName"
type="xs:string"/>
                                        <xs:attribute name="toParentCode"
type="xs:string"/>
                                        <xs:attribute name="toParentName"
type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="retireCode">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="conceptCode"
type="xs:string" use="required"/>
                                        <xs:attribute name="conceptName"
type="xs:string" use="required"/>
                                        <xs:attribute name="replacementCode"
type="xs:string"/>
                                        <xs:attribute name="replacementName"
type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="removeConceptRelationship">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="parentCode" type="xs:string"
use="required"/>
                                        <xs:attribute name="parentName"
type="xs:string"/>
                                        <xs:attribute name="relationship" type="xs:string"
use="required"/>
                                        <xs:attribute name="childCode" type="xs:string"
use="required"/>
                                        <xs:attribute name="childName"
type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
```

```
                                            </xs:choice>
                                        </xs:sequence>
                                        <xs:attribute name="codeSystemMnemonic" type="xs:string"
use="required"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="valueSetRevision">
                        <xs:annotation>
                            <xs:documentation>An addition or modification to a value
set.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:choice>
                                <xs:element name="createValueSet">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus" type="ballotStatusType"
minOccurs="0"/>
                                            <xs:element ref="description" minOccurs="0"/>
                                            <xs:element name="underValueSet" minOccurs="0">
                                                <xs:complexType>
                                                    <xs:attribute name="setName" type="xs:string"
use="required"/>
                                                    <xs:attribute name="addAsType" use="required">
                                                        <xs:simpleType>
                                                            <xs:restriction base="xs:NMTOKEN">
                                                                <xs:enumeration value="abstract"/>
                                                                <xs:enumeration value="specializable"/>
                                                            </xs:restriction>
                                                        </xs:simpleType>
                                                    </xs:attribute>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:choice minOccurs="0" maxOccurs="unbounded">
                                                <xs:element name="addCodesToValueSet"
type="addCodesToValueSetType"/>
                                                <xs:element name="addValueSetReferences"
type="addValueSetReferencesType"/>
                                                <xs:element name="addToVocabularyDomain"
type="addToVocabularyDomainType"/>
                                            </xs:choice>
                                        </xs:sequence>
                                        <xs:attribute name="setName" type="xs:string"
use="required"/>
                                        <xs:attribute name="codeSystemName" type="xs:string"/>
                                        <xs:attribute name="allCodes" type="xs:boolean"
default="false"/>
                                        <xs:attribute name="headCode" type="xs:string"/>
                                        <xs:attribute name="headCodePrintName" type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="selectValueSet">
                                    <xs:complexType>
                    <xs:sequence>
                      <xs:element name="ballotStatus" type="ballotStatusType"
minOccurs="0"/>
                      <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element name="addCodesToValueSet"
type="addCodesToValueSetType"/>
                        <xs:element name="addValueSetReferences"
type="addValueSetReferencesType"/>
                        <xs:element name="addToVocabularyDomain"
type="addToVocabularyDomainType"/>
                        <xs:element name="deleteValueSet">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="modifyValueSet">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                              <xs:element ref="oldDescription"
minOccurs="0"/>
                              <xs:element ref="newDescription"
minOccurs="0"/>
                            </xs:sequence>
                            <xs:attribute name="newName" type="xs:string"/>
                            <xs:attribute name="codeSystem"
type="xs:string"/>
                            <xs:attribute name="allCodes"
type="xs:boolean"/>
                            <xs:attribute name="headCode"
type="xs:string"/>
                            <xs:attribute name="headCodeName"
type="xs:string"/>
                          </xs:complexType>
                        </xs:element>
                        <xs:element name="removeCodesFromValueSet">
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                              <xs:element name="codeToRemove"
maxOccurs="unbounded">
                                <xs:complexType>
                                  <xs:attribute name="conceptCode"
type="xs:string" use="required"/>
                                  <xs:attribute name="conceptName"
type="xs:string"/>
                                </xs:complexType>
                              </xs:element>
                            </xs:sequence>
                          </xs:complexType>
                        </xs:element>
```

```
                                                <xs:element name="removeValueSetReferences">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element name="ballotStatus"
type="ballotStatusType" minOccurs="0"/>
                                                            <xs:element name="removeReferenceTo"
maxOccurs="unbounded">
                                                                <xs:complexType>
                                                                    <xs:attribute name="valueSet"
type="xs:string" use="required"/>
                                                                </xs:complexType>
                                                            </xs:element>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                            </xs:choice>
                                        </xs:sequence>
                                        <xs:attribute name="setName" type="xs:string"
use="required"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="vocabularyDomainRevision">
                        <xs:annotation>
                            <xs:documentation>An addition or modification to a vocabulary
domain.</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="createVocabularyDomain"
maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:sequence>
                                            <xs:element name="ballotStatus" type="ballotStatusType"
minOccurs="0"/>
                                            <xs:element ref="description" minOccurs="0"/>
                                        </xs:sequence>
                                        <xs:attribute name="vocabularyDomain" type="xs:string"
use="required"/>
                                        <xs:attribute name="restrictsDomain" type="xs:string"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="addCodesToCodeSystemType">
        <xs:sequence>
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
            <xs:choice>
                <xs:element name="underCode">
```

```
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="newCode" type="newCodeType"
maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="conceptCode" type="xs:string" use="required"/>
                        <xs:attribute name="conceptName" type="xs:string"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="newCode" type="newCodeType" maxOccurs="unbounded"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="addCodesToValueSetType">
        <xs:sequence>
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
            <xs:element name="codeAddition" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="conceptCode" type="xs:string" use="required"/>
                    <xs:attribute name="conceptName" type="xs:string"/>
                    <xs:attribute name="relationship" type="xs:string"/>
                    <xs:attribute name="relInclusion" default="inclusive">
                        <xs:simpleType>
                            <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="exclusive"/>
                                <xs:enumeration value="inclusive"/>
                                <xs:enumeration value="leafOnly"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="addConceptRelationshipType">
        <xs:sequence>
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="parentCode" type="xs:string" use="required"/>
        <xs:attribute name="parentName" type="xs:string"/>
        <xs:attribute name="relationship" type="xs:string" use="required"/>
        <xs:attribute name="childCode" type="xs:string" use="required"/>
        <xs:attribute name="childName" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="addPrintNameToCodeType">
        <xs:sequence>
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="conceptCode" type="xs:string" use="required"/>
        <xs:attribute name="conceptName" type="xs:string"/>
        <xs:attribute name="newPrintName" type="xs:string" use="required"/>
        <xs:attribute name="languageCode" type="xs:string" default="en"/>
        <xs:attribute name="isPreferred" type="xs:boolean" default="true"/>
    </xs:complexType>
    <xs:complexType name="addPropertyToCodeType">
        <xs:sequence>
```

```
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
            <xs:element name="property" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="conceptCode" type="xs:string" use="required"/>
        <xs:attribute name="conceptName" type="xs:string"/>
        <xs:attribute name="propertyId" type="xs:string" use="required"/>
        <xs:attribute name="language" type="xs:string" default="en"/>
    </xs:complexType>
    <xs:complexType name="addToVocabularyDomainType">
        <xs:sequence>
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="vocabularyDomain" type="xs:string" use="required"/>
        <xs:attribute name="context" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="addValueSetReferencesType">
        <xs:sequence>
            <xs:element name="ballotStatus" type="ballotStatusType" minOccurs="0"/>
            <xs:element name="listEntry" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="setName" type="xs:string" use="required"/>
                    <xs:attribute name="setType" use="required">
                        <xs:simpleType>
                            <xs:restriction base="xs:NMTOKEN">
                                <xs:enumeration value="abstract"/>
                                <xs:enumeration value="specializable"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ballotStatusType">
        <xs:annotation>
            <xs:documentation>The status of a given change within the HL7 organizational
structure.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="note" type="xs:string" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Notes, etc. about the current ballot state, decisions and the
like.</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="action" default="Proposed">
            <xs:annotation>
                <xs:documentation>The current state of the proposition</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="Proposed"/>
                    <xs:enumeration value="Passed"/>
                    <xs:enumeration value="PassedWithChanges"/>
                    <xs:enumeration value="Tabled"/>
                    <xs:enumeration value="Withdrawn"/>
                    <xs:enumeration value="NonVotingItem"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="vote">
            <xs:annotation>
                <xs:documentation>Vote is in the format yy-nn-aa where:
                                   yy is yes votes
                                   nn is no votes
                                   aa is abstention votes
                </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:pattern value="[0-9]+-[0-9]+-[0-9]+"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="description" type="xs:string">
        <xs:annotation>
            <xs:documentation>A textual definition or description</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="newCodeType">
        <xs:sequence>
            <xs:element ref="description" minOccurs="0"/>
            <xs:element name="newCode" type="newCodeType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="conceptCode" type="xs:string" use="required"/>
        <xs:attribute name="conceptName" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:element name="newDescription" type="xs:string"/>
    <xs:element name="oldDescription" type="xs:string"/>
</xs:schema>
```

Appendix E – Instructions for Using the VocabularyMaint Validation and Verification Tool

There is a Java-based tool available that validates and processes HL7 Vocabulary Submissions. The tool, which consists of one rather large .jar file, named *VocMaint_and_dependencies.jar*, can be downloaded from the HL7 Vocabulary Technical Committee Documents and Presentations web page. This program requires Java 1.3 or 1.4 to run.

You will also need to download and unzip the latest RIM Model Repository File in Access format from the HL7 Data models home page. Make a copy of this file, as you will need to restore it if the update has errors:

```
C:\Work> Copy rim0123d.mdb rim0123_master.mdb
```

The update file should first be validated by an XML parser such as XMLSpy.

The Java program is invoked via:

```
java –jar VocMaint_and_dependencies.jar <RIM Repository DB>
<XML Update File>
```

As an example, to install the complete beer classification file we would say:

```
C:\Work> copy rim0123_master.mdb rim0123d.mdb
     1 file(s) copied.
C:\Work> java –jar VocMaint_and_dependencies.jar
rim0123d.mdb beerComplete.xml
```

Which should generate the following output:

```
Release version: 147    22 May 2003 13:51:19,475
-----> Warning - inserting PROPOSED item  22 May 2003 13:51:19,485

Registering new code system: BEER (Castello Beer Classification)  22 May
2003 13:51:19,485
Adding Internal code system BEER (Castello Beer Classification) to OID
registry    22 May 2003 13:51:19,485
Created new OID_root entry: 2.16.840.1.113883.19.5.1  22 May 2003
13:51:19,495
(STUB) Adding new registry entry for 2.16.840.1.113883.19.5.1     22 May
2003 13:51:19,495

Selecting code system: CodeSystem   22 May 2003 13:51:19,505
OID is: 2.16.840.1.113883.5.22      22 May 2003 13:51:19,525
Adding BEER (Castello Beer Classification) to 2.16.840.1.113883.5.22
      22 May 2003 13:51:19,525
Adding property OID to BEER   22 May 2003 13:51:19,605
-----> Warning - inserting PROPOSED item  22 May 2003 13:51:19,615
```

```
Adding ALE (Ale) to 2.16.840.1.113883.19.5.1    22 May 2003 13:51:19,615
Adding PALE ALE (Pale Ale) to 2.16.840.1.113883.19.5.1      22 May 2003
13:51:19,685
Adding PALE ALE as a subtype of ALE (Ale) 22 May 2003 13:51:19,765
Adding BITTER ALE (Bitter Ale) to 2.16.840.1.113883.19.5.1  22 May 2003
13:51:19,776
Adding BITTER ALE as a subtype of ALE (Ale)     22 May 2003 13:51:19,856
Adding LIGHT ALE (Light Ale) to 2.16.840.1.113883.19.5.1    22 May 2003
13:51:19,866
Adding LIGHT ALE as a subtype of ALE (Ale)      22 May 2003 13:51:19,936
Adding MILD ALE (Mile Ale) to 2.16.840.1.113883.19.5.1      22 May 2003
13:51:19,946
Adding MILD ALE as a subtype of ALE (Ale) 22 May 2003 13:51:20,026
Adding BROWN ALE (Brown Ale) to 2.16.840.1.113883.19.5.1    22 May 2003
13:51:20,026
Adding BROWN ALE as a subtype of ALE (Ale)      22 May 2003 13:51:20,116
Adding STRONG ALE (Strong Ale) to 2.16.840.1.113883.19.5.1  22 May 2003
13:51:20,136
Adding STRONG ALE as a subtype of ALE (Ale)     22 May 2003 13:51:20,206
Adding INDIAN PALE ALE (Indian Pale Ale) to 2.16.840.1.113883.19.5.1
      22 May 2003 13:51:20,216
Adding INDIAN PALE ALE as a subtype of ALE (Ale)      22 May 2003
13:51:20,296
Adding SCOTCH ALE (Scotch Ale) to 2.16.840.1.113883.19.5.1  22 May 2003
13:51:20,296
Adding SCOTCH ALE as a subtype of ALE (Ale)     22 May 2003 13:51:20,376
Adding BARLEY WINE (Barley Wine) to 2.16.840.1.113883.19.5.1      22 May
2003 13:51:20,386
Adding BARLEY WINE as a subtype of ALE (Ale)    22 May 2003 13:51:20,466
Adding PORTER (Porter) to 2.16.840.1.113883.19.5.1    22 May 2003
13:51:20,466
Adding STOUT (Stout) to 2.16.840.1.113883.19.5.1      22 May 2003
13:51:20,536
Adding BITTER STOUT (Bitter Stout) to 2.16.840.1.113883.19.5.1    22 May
2003 13:51:20,617
Adding BITTER STOUT as a subtype of STOUT (Stout)     22 May 2003
13:51:20,687
Adding SWEET STOUT (Sweet Stout) to 2.16.840.1.113883.19.5.1      22 May
2003 13:51:20,697
Adding SWEET STOUT as a subtype of STOUT (Stout)      22 May 2003
13:51:20,777
Adding MILK STOUT (Milk Stout) to 2.16.840.1.113883.19.5.1  22 May 2003
13:51:20,777
Adding MILK STOUT as a subtype of STOUT (Stout) 22 May 2003 13:51:20,877
Adding IMPERIAL STOUT (Imperial Stout) to 2.16.840.1.113883.19.5.1
      22 May 2003 13:51:20,877
Adding IMPERIAL STOUT as a subtype of STOUT (Stout)   22 May 2003
13:51:20,967
Adding TRAPPIST (Trappist) to 2.16.840.1.113883.19.5.1      22 May 2003
13:51:20,967
Adding BIRRE D'ABBAZIA (Birre D'Abbazia) to 2.16.840.1.113883.19.5.1
      22 May 2003 13:51:21,047
Adding SPECIALI BELGHE (Speciali Belghe) to 2.16.840.1.113883.19.5.1
      22 May 2003 13:51:21,117
Adding BIRRE BIANCHE BELGHE (Birre Bianche Belghe) to
2.16.840.1.113883.19.5.1      22 May 2003 13:51:21,197
Adding BIERE DE GARDE (Biere De Garde (French)) to
2.16.840.1.113883.19.5.1      22 May 2003 13:51:21,267
Adding KRIEK (Kriek) to 2.16.840.1.113883.19.5.1      22 May 2003
13:51:21,348
Adding ALT BIER (Alt Bier) to 2.16.840.1.113883.19.5.1      22 May 2003
13:51:21,428
Adding KÖLSCH (Kölsch) to 2.16.840.1.113883.19.5.1    22 May 2003
13:51:21,508
```

```
Adding BIRRE TEDESCHE DI FRUMENTO (Birre Tedesche Di Frumento) to
2.16.840.1.113883.19.5.1      22 May 2003 13:51:21,588
Adding BERLINER WEISSE (Berliner Weisse) to 2.16.840.1.113883.19.5.1
      22 May 2003 13:51:21,678
Adding BERLINER WEISSE as a subtype of BIRRE TEDESCHE DI FRUMENTO (Birre
Tedesche Di Frumento)   22 May 2003 13:51:21,818
Adding WEIZENBIER (Weizenbier) to 2.16.840.1.113883.19.5.1  22 May 2003
13:51:21,828
Adding WEIZENBIER as a subtype of BIRRE TEDESCHE DI FRUMENTO (Birre
Tedesche Di Frumento)   22 May 2003 13:51:22,008
Adding HEFE WEIZEN (Hefe Weizen) to 2.16.840.1.113883.19.5.1      22 May
2003 13:51:22,018
Adding HEFE WEIZEN as a subtype of BIRRE TEDESCHE DI FRUMENTO (Birre
Tedesche Di Frumento)   22 May 2003 13:51:22,149
Adding DUNKEL WEIZEN (Dunkel Weizen) to 2.16.840.1.113883.19.5.1  22 May
2003 13:51:22,159
Adding DUNKEL WEIZEN as a subtype of BIRRE TEDESCHE DI FRUMENTO (Birre
Tedesche Di Frumento)   22 May 2003 13:51:22,279
-----> Warning - inserting PROPOSED item  22 May 2003 13:51:22,279
Create new vocabulary domain: Beer  22 May 2003 13:51:22,289
-----> Warning - inserting PROPOSED item  22 May 2003 13:51:22,289

Creating value set Beer using  all codes from  code system BEER   22 May
2003 13:51:22,289
      Value set identifier is 2.16.840.1.113883.1.11.19364  22 May 2003
13:51:22,319
Checking whether allCodes needs to be expanded...     22 May 2003
13:51:22,329
Adding concept KÖLSCH to value set Beer   22 May 2003 13:51:23,891
Adding concept BIERE DE GARDE to value set Beer 22 May 2003 13:51:23,901
Adding concept SPECIALI BELGHE to value set Beer      22 May 2003
13:51:23,921
Adding concept PORTER to value set Beer   22 May 2003 13:51:23,941
Adding concept ALT BIER to value set Beer 22 May 2003 13:51:23,951
Adding concept KRIEK to value set Beer    22 May 2003 13:51:23,971
Adding concept BIRRE D'ABBAZIA to value set Beer      22 May 2003
13:51:23,991

Creating value set Stout(STOUT) using  code system BEER with head code
STOUT 22 May 2003 13:51:24,011
      Value set identifier is 2.16.840.1.113883.1.11.19365  22 May 2003
13:51:24,031
Adding specialized reference to Stout(STOUT) under Beer     22 May 2003
13:51:24,051
Adding concept BITTER STOUT to value set Stout(STOUT) 22 May 2003
13:51:24,061
Adding concept SWEET STOUT to value set Stout(STOUT)  22 May 2003
13:51:24,081
Adding concept MILK STOUT to value set Stout(STOUT)   22 May 2003
13:51:24,111
Adding concept IMPERIAL STOUT to value set Stout(STOUT)     22 May 2003
13:51:24,131

Creating value set Birre Tedesche Di Frumento(BIRRE TEDESCHE DI
FRUMENTO) using  code system BEER with head code BIRRE TEDESCHE DI
FRUMENTO    22 May 2003 13:51:24,151
      Value set identifier is 2.16.840.1.113883.1.11.19366  22 May 2003
13:51:24,181
Adding specialized reference to Birre Tedesche Di Frumento(BIRRE
TEDESCHE DI FRUMENTO) under Beer    22 May 2003 13:51:24,201
Adding concept BERLINER WEISSE to value set Birre Tedesche Di
Frumento(BIRRE TEDESCHE DI FRUMENTO)      22 May 2003 13:51:24,221
Adding concept WEIZENBIER to value set Birre Tedesche Di Frumento(BIRRE
TEDESCHE DI FRUMENTO)   22 May 2003 13:51:24,241
```

```
Adding concept HEFE WEIZEN to value set Birre Tedesche Di Frumento(BIRRE
TEDESCHE DI FRUMENTO)   22 May 2003 13:51:24,251
Adding concept DUNKEL WEIZEN to value set Birre Tedesche Di
Frumento(BIRRE TEDESCHE DI FRUMENTO)      22 May 2003 13:51:24,271
Adding concept BIRRE BIANCHE BELGHE to value set Beer 22 May 2003
13:51:24,291
Adding concept TRAPPIST to value set Beer 22 May 2003 13:51:24,311

Creating value set Ale(ALE) using  code system BEER with head code ALE
      22 May 2003 13:51:24,321
      Value set identifier is 2.16.840.1.113883.1.11.19367  22 May 2003
13:51:24,351
Adding specialized reference to Ale(ALE) under Beer   22 May 2003
13:51:24,371
Adding concept PALE ALE to value set Ale(ALE)   22 May 2003 13:51:24,392
Adding concept BITTER ALE to value set Ale(ALE) 22 May 2003 13:51:24,412
Adding concept LIGHT ALE to value set Ale(ALE)  22 May 2003 13:51:24,432
Adding concept MILD ALE to value set Ale(ALE)   22 May 2003 13:51:24,442
Adding concept BROWN ALE to value set Ale(ALE)  22 May 2003 13:51:24,462
Adding concept STRONG ALE to value set Ale(ALE) 22 May 2003 13:51:24,482
Adding concept INDIAN PALE ALE to value set Ale(ALE)  22 May 2003
13:51:24,502
Adding concept SCOTCH ALE to value set Ale(ALE) 22 May 2003 13:51:24,522
Adding concept BARLEY WINE to value set Ale(ALE)      22 May 2003
13:51:24,532
      allCodes was set to false and code system was expanded      22 May
2003 13:51:24,552
-----> Warning - inserting PROPOSED item  22 May 2003 13:51:24,562
Adding value set Beer to domain Beer      22 May 2003 13:51:24,562


=============== Successful load! ===============      22 May 2003
13:51:24,562
```

If all went well, the last output line should be “Successful load!” If any errors were encountered, the message:

```
********** Update session terminated!   22 May 2003 13:53:26,820
```

will appear instead. Refer back to the listing to determine the source of the problem. Once corrected, you can repeat the database copy and run steps above.

Once the update program has run successfully, the output can be reviewed using the latest version of RoseTree. Note that RoseTree only lists vocabulary domains, which means that a code system will not show up intil it is connected with a value set which, in turn, is associated with a vocabulary domain.