Abstract:
This document describes the OASIS Code List Representation model and W3C XML Schema, known collectively as “genericode”1.

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

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

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

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

1Genericode can be written starting either with an upper-case or lower-case “g”. It depends whether genericode is at the start of the sentence or not.
Table of Contents

1 Introduction...........................................................................................................................................5
  1.1 Terminology..................................................................................................................................5
  1.2 Normative References..................................................................................................................5
  1.3 Non-normative References..........................................................................................................5
2 What is a Code List?................................................................................................................................6
3 Genericode Model & XML Format........................................................................................................9
  3.1 Tabular Structure........................................................................................................................9
  3.2 Genericode Document Types.......................................................................................................9
  3.3 Column Sets – Columns and Keys...............................................................................................10
  3.4 Code lists....................................................................................................................................18
  3.5 Code list sets............................................................................................................................23
4 Genericode XML Schema Reference ..............................................................................................26
  4.1 Notation.......................................................................................................................................26
  4.2 Table of Schema Definitions.......................................................................................................26
  4.3 Global Schema Definitions in Alphabetic Order.........................................................................28
Appendix A. Acknowledgments............................................................................................................51
Appendix B. Example Code Lists in Genericode Format.................................................................52
  B.1. UBL Example – Country Codes...............................................................................................52
  B.2. FpML Example – Business Centers.........................................................................................55
  B.3. Multiple Key Example...............................................................................................................57
  B.4. Undefined Values Example........................................................................................................59
  B.5. Complex (XML) Values Example.............................................................................................62
  B.6. External Column Set Example...................................................................................................64
  B.7. Code List Set Example..............................................................................................................66
1 Introduction

Code lists, or enumerated values, have been with us since long before computers. They should be well understood and easily dealt with by now. Unfortunately, they are not. As is often the case, if you take a fundamentally simple concept, you find that everyone professes to understand it with complete clarity. When you look more closely, you find that everybody has their own unique view of what the problem is and how it should be solved.

If code lists were really so simple and obvious, there would already be a single, well-known and accepted way of handling them in XML. There is no such agreed solution, though. The problem is that while code lists are a well understood concept, people don't actually agree exactly on what code lists are, and how they should be used.

The OASIS Code List Representation format, “genericode”, is a single model and XML format (with a W3C XML Schema) that can encode a broad range of code list information. The XML format is designed to support interchange or distribution of machine-readable code list information between systems.

This version 1.0 of genericode implements the “Version 1.0 Requirements” from the OASIS Code List Representation Requirements document, version 1.0.1 (http://docs.oasis-open.org/codelist/codelist-genericode-1.0/doc/oasis-code-list-representation-requirements-1.0.1.pdf). The requirements document also lists requirements for future versions of genericode, which will not be discussed further in this version of this document.

1.1 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in IETF RFC 2119.

1.2 Normative References


http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

1.3 Non-normative References

http://www.idealliance.org/proceedings/xml04/abstracts/paper86.html
2 What is a Code List?

This section is non-normative.

What is a code list, then? Most people would agree that the following is a code list:

\{
'SUN', 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT'
\}

*Example 1: Days of the week: English, uppercase*

This is a perfectly reasonable set of alphabetic codes for representing days of the week. However, so is:

\{
'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'
\}

*Example 2: Days of the week: English, mixed case*

These two code lists are similar, but certainly not identical. That said, they can both be used to represent the days of the week. Of course, you could also use:

\{
'Dim', 'Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam'
\}

*Example 3: Days of the week: French, mixed case*

which is created from abbreviations for the days of the week in French. Then again, you could use:

\{0, 1, 2, 3, 4, 5, 6\}

*Example 4: Days of the week: numeric*

which is suitable as a computer representation, e.g. for a database column. On the other hand:

\{'S', 'M', 'T', 'W', 'T', 'F', 'S'\}

*Example 5: Days of the week: English, single character*

is not suitable as a code list for the days of the week, because the values are not unique.

Now suppose that you are using codes to represent days of the week in an application, and you are displaying the days of the week using 3-letter abbreviations in English or French. In that context, should *Example 2* and *Example 3* be considered to be code lists, or should they be considered to be display values that would be keyed to either the *Example 1* or *Example 4* codes? The fact is, they could be either code lists or display values. A value which is a code in one context might only be an associated value for that code in another context. Nothing privileges any of these code lists over the others in terms of ability or suitability to be the code list (except the *Example 5* values which are not suitable). There is a choice of code lists that can be used, and the answer to the question “which choice is the best?” depends on the needs of each particular situation.

What the above examples show is that for each *distinct entry* in a code list, there are many possible associated values (we use the term *distinct entry* to express the idea that we are talking a single item that needs to be represented in the code list, rather than about the code value(s) that can be used to identify that item). Some of those associated values are suitable for use in code lists, some are not. This leads to a tabular model, where each row of the table represents a conceptual code, and each column represents an associated value (code list metadata), as follows:
Table 1: Days of the week

Notice that the first 4 of the 5 columns have been labeled as "key" columns. This means that the values in those columns can be used to uniquely identify the rows, and hence they can be used as code list values. The term key is used here similarly to a relational database table.

This is the most common case, where a single column can be used as a key. However, consider the following modification:

Table 2: Days of the week, version 2

Here, the first two columns are each a key column. The last two columns are not individually key columns, but together they form a compound key, i.e. while the individual columns do not contain unique values, the pair of values is unique within each row. This is again similar to what happens in some relational databases, that a key for the rows need not be constructed from a single column, but instead may be constructed by combining two or more columns.

Finally, there is no reason why a column should only contain simple values like strings or numbers. A column could also contain a complex compound group of data, such as a fragment of XML:
Table 3: Days of the week, version 3

Notice that the final XHTML column is not marked as a key column. The values are unique, so it certainly could be used as a key column. However, sometimes you may not wish to mark a column as a key column, even if the values are unique. The values in the column may not make particularly suitable keys. They might be too long to process quickly and conveniently, or they might not be able to be used in a particular context, such as for an XML attribute value. Also, it may be that while the values in a particular column are unique now, there is no guarantee or expectation that they will remain unique as the code list grows or changes in the future.

Once you see the tabular nature that underlies the information that can be associated with code lists, it becomes clear why they can be a source of so much debate. Different users need different subsets of the code list information, and people often assume that the information they need is all the information that anyone needs.

That kind of thinking doesn't work well with code lists, because code lists are sufficiently generic a concept that they are used across messages/documents, applications, and databases. The code list details that you need for the XML schemas often will not be exactly the same as the details that you need for your database or your application. If the code list information cannot be shared easily across these different areas, the result is duplication of effort and potential loss of synchronization between different implementations of the same code list.

The XML schema may only require a set of 3-letter codes to represent the code list. The database may require a set of numeric codes, plus display labels (possibly in different languages). The application may need to know which 3-letter code corresponds to which numeric code, so that it can process the XML and update the database. Also, some information related to a code list might not be appropriate for the XML format. For example, if you have a different image file for each code, it isn't ideal to include this image inline in the code list XML, since it vastly increases the size of the XML, and makes it more difficult to read. So in an XML representation, you are more likely to include some reference (e.g. a URL) to the image. For a database, however, it may be feasible to store the image in a BLOB column in a database.

One last piece of experience from databases is that support for undefined values will be required. Sometimes users will have values that need to be associated with some of the codes in a code list, but won’t have values to associate with every code. In that case, the concept of a undefined (nil or null) value is needed.

---

1Binary Large OBject.
3 Genericode Model & XML Format

The text of this section is normative. The illustrations, Schema subsets and examples are not normative.

This section describes the genericode model and XML format for representing code lists. The "gc" XML prefix refers to the namespace URI

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

If you are reading this section for the first time, you may find it helpful to review the examples in the appendices before continuing.

3.1 Tabular Structure

Genericode has a tabular structure for code list information. Each row in the table represents a single distinct entry in the code list, i.e. each row represents a single uniquely identifiable item in the code list.

Each column in the table represents a metadata value that can be defined for each distinct entry in the code list. Each column is either required or optional. A required column does not allow any row to have an undefined (nil or null) value. An optional column allows undefined values.

A genericode key is a set of one or more required columns that together uniquely identify each distinct entry in the code list. Optional columns cannot be used for keys. Each code list must have at least one key. Genericode keys are equivalent to what people usually mean when they talk about the "codes" in a code list. However, genericode allows multiple keys for each code list, and there is no single preferred key. For code lists that have multiple keys, it is assumed that the choice of which key to use is a late binding choice that is specific to the application, technology and/or context in which the code list is used.

3.2 Genericode Document Types

There are 3 kinds of genericode documents, all supported by the one W3C XML Schema:

- Column Set documents;
- Code List documents;
- Code List Set documents.
Column Set Documents

A column set document has the root element `<gc:ColumnSet>`. It contain definitions of genericode columns or keys that can be imported into code list documents or into other column set documents.

Code List Documents

A code list document has the root element `<gc:CodeList>`. It contains metadata describing the code list as a whole, as well as explicit code list data – codes and associated values.

Code List Set Documents

A code list set document has the root element `<gc:CodeListSet>`. It contains references to particular versions of code lists, and can also contain version-independent references to code lists. A code list set document can be used to define a particular configuration of versions of code lists that are used by a project, application, standard, etc.

3.3 Column Sets – Columns and Keys

A column set is a set of definitions of genericode columns and/or keys. A column defines a particular metadata value that can be defined for each distinct entry in a code list. A key defines a set of one or more columns.

Extract 1: Genericode Schema - Global Element Declarations

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
>
  ...
  <xsd:element
    name="CodeList"
    type="gc:CodeListDocument">
    <xsd:annotation>
      <xsd:documentation>
        Top-level (root) element for a genericode code list definition.
        A code list definition defines the details of a particular (version of) a code list.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  ...
  <xsd:element
    name="CodeListSet"
    type="gc:CodeListSetDocument">
    <xsd:annotation>
      <xsd:documentation>
        Top-level element for the definition of a code list set.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  ...
  <xsd:element
    name="ColumnSet"
    type="gc:ColumnSetDocument">
    <xsd:annotation>
      <xsd:documentation>
        Top-level element for the definition of a column set.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  ...
</xsd:schema>
```
It is not necessary to use separate column set documents. A genericode code list document can contain all of the required column and key definitions. Column set documents are provided as a convenience mechanism for sharing column and/or key definitions between multiple code lists.

This figure is in UML notation. Each column set must have a unique ID. For a column set defined within a code list document, the code list document's unique identifier is used. A column set can define any number of columns. It can also reference any number of columns from other column sets (in column set documents or code list documents). A column set can also define any number of keys. Each key is defined by one or more of the columns in the column set (either defined or imported). Keys are used to uniquely identify the rows (distinct entries) of code lists. Columns and keys are uniquely named within the column set that defines them, and each can also be uniquely identified using a specific URI if required additionally.

The matching genericode W3C XML Schema (WXS) representation of column set content is:
Illustration 2: Structure of a column set document

This figure is in XML Spy® notation. A default datatype library URI can be provided to identify which datatype library should be used for columns which do not explicitly specify a datatype library. If this URI is not provided, the datatype library defaults to the W3C XML Schema (WXS) datatype library.

A column set definition contains optional user annotation information (Annotation), and then identification and location information (Identification). A column set has a short name, any number of long names and a version.
A column set is uniquely identified by a canonical URI. Particular versions of the column set are uniquely identified by a canonical version URI. Location URIs can also be provided to suggest URLs from which an XML genericode column set instance may be retrieved (at the discretion of an application). Alternative location URIs can be provided to suggest URLs from which non-genericode representations of the column set can be retrieved. Canonical URIs and canonical version URIs must not be used as de facto location URIs for retrieving column set instances (nor anything else). The column set definition can also list the details of the agency which is responsible for publishing and/or maintaining the column set information.
A column definition (Column) contains a unique ID for the column and its use (required or optional). It also contains a short name (token) for the column, any number of long names, and optional extra canonical identification URIs. The datatype information for the column is contained in its Data element.
The data structure is based on the data element in RELAX NG. The datatype is specified as a Type from a DatatypeLibrary. If the datatype library is not specified, it is inherited from the DatatypeLibrary attribute of the enclosing column set definition. It otherwise defaults to the W3C XML Schema (WXS) datatype library.

If the data is XML (complex valued), the DatatypeLibrary is set to the namespace URI for the XML (or to "*" if any namespace¹ is allowed), and the Type is set to the root element name for the XML data (or to "*" if any root element is allowed).

Data definitions can contain Parameter elements which define facets that refine the datatype. When using the WXS datatype library, these are just the usual WXS datatype facets.

---

¹Any namespace except the genericode namespace.
If a column is defined in an external column set or code list document, it is referenced using a ColumnRef. The column reference must have an ID just as a column definition would, but it also has an ExternalRef which contains the column's ID in the external document. The external column set or code list is identified by a CanonicalVersionUri and/or by any LocationUri information that is provided.
A key definition (Key) contains an ID for the key. It also contains a short name (token) for the key, any number of long names, and optional extra canonical identification URIs. The columns which together form the key are referenced using one or more ColumnRef elements. The Ref attribute of each contains the ID of either a Column or ColumnRef in the column set. Only required (not optional) columns may be used within a key (note that this rule is not able to be enforced using the genericode WXS Schema alone).
If a key is defined in an external column set or code list document, it can be referred to using a KeyRef. The key reference must have an ID, and also has an ExternalRef which contains the key’s ID in the external document. The external column set or code list is identified by a CanonicalVersionUri and/or by any LocationUri information that is provided.

### 3.4 Code lists

A code list can contain its own embedded column set definition. It can also import columns and keys from any number of external column sets (in column set documents and/or code list documents). In the simplest case, what a code list provides is information (metadata) about the code list and (optionally) a set of rows, where each row defines a distinct entry in the code list.

A code list document that contains only information (metadata) about the code list as a whole is known as a CodeList Metadata document. If the code list document defines (zero or more) row, it is a Simple CodeList. These are the only kinds of code list that are supported in this version of the specification.
There is an important difference between a **CodeList Metadata** document and **Simple CodeList** that contains zero rows (zero distinct entries). The former does not provide information on how many distinct entries are contained in the code list. The latter explicitly indicates that a particular version of the code list contains zero distinct entries, i.e. the particular version of the code list is empty. A CodeList Metadata document does not provide any indication about whether a code list is empty or not.

### 7.2.1 Simple CodeLists and CodeList Metadata

A **CodeList Metadata** document is a special case of a **Simple CodeList** document. The differences will be discussed explicitly where appropriate.
A *Simple CodeList* is modeled as follows:

![Diagram of Simple CodeList](image)

**Illustration 10: Simple code lists**

A *Simple CodeList* contains zero or more rows (it is necessary to support empty code lists to allow for code lists that are empty now, but will be populated in future versions). Each *Row* defines a single distinct entry in the code list.

A *Row* contains one or more *Values*, where each of those values corresponds to a distinct column in the code list. At least one value is required, because a code list has to have at least one key, and each key requires at least one column. As a consequence, a *Row* must have at least one *Value*. Additionally, a *Row* must contain a defined Value for each of the required columns in the code list, i.e. for those columns for which a *Value* must be defined (non-null) for each *Row* (distinct entry) in the code list.

Each *Value* is associated with a single distinct column of the code list. For each *Key* in the code list, the values associated with the columns for that key must form a unique set, i.e. no two rows are allowed to have the same set of values for the same key columns. Note that this uniqueness requirement cannot be enforced using the genericode WXS Schema for code list documents, which is structured as follows:
Many of these elements and types have appeared already in section 3.3, so the explanations will not be repeated here. A code list document can either define its own embedded ColumnSet, or refer to an externally defined column set using a ColumnSetRef.

Illustration 11: Structure of a code list document

A ColumnSetRef contains the canonical version URI which uniquely identifies a referenced column set or code list document which contains the column set. It can also contain suggested URLs from which to retrieve the column set or code list. Canonical version URIs must not be used as de facto location URIs for retrieving column set instances (nor anything else).

Illustration 12: Structure of a column set reference

A code list document that contains a SimpleCodeList element is a Simple CodeList. If the code list document does not contain a SimpleCodeList element, then it is a CodeList Metadata document.

The genericode WXS Schema representation of a SimpleCodeList is
A SimpleCodeList contains zero or more Row elements. Each Row contains one or more Value elements.

The Value container element is needed to allow optional user annotations of individual values in the code list. It has a ColumnRef attribute which contains the unique document ID of the associated column. A Value element can contains either a SimpleValue containing a textual value, or a ComplexValue.
containing a balanced (well-formed) XML fragment from a namespace other than the genericode namespace.

If a Value element does not contain either a SimpleValue element or a ComplexValue element, then the value is undefined. Only optional columns are allowed to have undefined values. Also, if a Row element does not contain a Value element corresponding to a particular column, then the row's value for that column is undefined.

Note that the ColumnRef attribute of a Value is optional. If it is not provided, it is assumed that the column is the one which follows the column associated with the previous value in the row. If the first Value in a Row does not have a ColumnRef, it is assumed to be associated with the first column in the column set. It is an error if a row contains more than one value for the same column, or if it does not contain a value for a required column.

The genericode WXS Schema is not able to validate that the contents of a Value match the datatype of the associated column. Other validation mechanisms should be used to perform datatype validation.

### 3.5 Code list sets

A CodeList Set lists a configuration of code lists and/or codelist versions. CodeList Sets can be used to provide lists of the code lists or code list versions that are associated with a particular version of an application or specification. The genericode WXS Schema structure for CodeList Set documents is:
Many of these elements and types have appeared already in section 3.3, so the explanations will not be repeated here. A code list set document contains a series of zero or more CodeListRef elements. Each of these is a reference to a code list or to a version of a code list. If the CanonicalVersionUri is defined, then the LocationUri elements (if any) contain retrieval URIs for genericode CodeList documents. If the CanonicalVersionUri is not defined, then the LocationUri elements (if any) contain retrieval URIs for genericode CodeList Metadata documents. Note that canonical URIs and canonical version URIs must not be used as de facto location URIs for retrieving code list instances (nor anything else).

A CodeList Set does not contain definitions of code lists, it only refers to the code list and code list versions which are a part of the particular version of the CodeList Set. It should also be noted that a code list set may contain a reference to a code list without specifying a particular version of the code list, and it may contain a reference to a code list or code list version without specifying a location for retrieving a genericode definition of that code list (metadata) or code list version. This is to support situations where
● the code list definition is known to the users, and no location needs to be published. This may be because users have an application which maps the canonical URI or canonical version URI to an explicit set of codes;

● the code list is sufficiently well-known (e.g. ISO 3-letter country codes) that users only need to have it uniquely identified, and do not need to have it enumerated for them.
4 Genericode XML Schema Reference

Schema version: 1.0

Target namespace: http://docs.oasis-open.org/codelist/ns/genericode/1.0/

4.1 Notation

<table>
<thead>
<tr>
<th>Multiplicity</th>
<th>Minimum Occurrence</th>
<th>Maximum Occurrence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>?</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>+</td>
<td>1</td>
<td>unbounded</td>
</tr>
<tr>
<td>*</td>
<td>0</td>
<td>unbounded</td>
</tr>
<tr>
<td>{m,n}</td>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td>{m,}</td>
<td>m</td>
<td>unbounded</td>
</tr>
</tbody>
</table>

ANY – any element in any namespace.

ANY[#other] – any element in any namespace other than the target namespace of the genericode XML Schema.

(A, B, C, ...) – sequence of items (A, B, C, etc.).

(A | B | C | ...) – choice between items (A, B, C, etc.).

4.2 Table of Schema Definitions

Global Elements

- CodeList (page 29)
- CodeListSet (page 31)
- ColumnSet (page 34)

Global Complex Types

- Agency (page 28)
- Annotation (page 28)
- AnyOtherContent (page 29)
- AnyOtherLanguageContent (page 29)
- CodeListDocument (page 29)
- CodeListRef (page 30)
- CodeListSetDocument (page 31)
Global Simple Types

- UseType (page 48)

Global Model Groups

- ColumnChoice (page 32)
- ColumnSetChoice (page 36)
- ColumnSetContent (page 36)
- DocumentHeader (page 40)
- IdentificationRefUriSet (page 42)
- IdentificationVersionUriSet (page 42)
- KeyChoice (page 44)
4.3 Global Schema Definitions in Alphabetic Order

Agency (Complex Type)

Details of an agency which produces code lists or related artifacts.

**Content Model:** (ShortName?, LongName*, Identifier*)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortName</td>
<td>ShortName (page 47)</td>
<td>Short name (without whitespace) for the agency.</td>
</tr>
<tr>
<td>LongName</td>
<td>LongName (page 45)</td>
<td>Human-readable name for the agency.</td>
</tr>
<tr>
<td>Identifier</td>
<td>GeneralIdentifier (page 41)</td>
<td>Identifier for the agency.</td>
</tr>
</tbody>
</table>

Annotation (Complex Type)

User annotation information.

**Content Model:** (Description*, AppInfo?)

**Mixed Content:** No
Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>AnyOtherLanguageContent</td>
<td>Human-readable information.</td>
</tr>
<tr>
<td></td>
<td>(page 29)</td>
<td></td>
</tr>
<tr>
<td>ApplInfo</td>
<td>AnyOtherContent</td>
<td>Machine-readable information.</td>
</tr>
<tr>
<td></td>
<td>(page 29)</td>
<td></td>
</tr>
</tbody>
</table>

**AnyOtherContent (Complex Type)**

Container for any XML content which is in a different namespace to the Schema's target namespace.

**Content Model:** (ANY[##other]*)

**Mixed Content:** No

**AnyOtherLanguageContent (Complex Type)**

Container for any human-readable XML content which is in a different namespace to the Schema's target namespace.

**Extension of:** AnyOtherContent (page 29)

**Content Model:** (ANY[##other]*)

**Mixed Content:** No

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml:lang</td>
<td>optional</td>
<td></td>
<td>Language code for the human-readable XML content.</td>
</tr>
</tbody>
</table>

**CodeList (Element)**

Top-level (root) element for a genericode code list definition.

A code list definition defines the details of a particular (version of a) code list.

**Complex Type:** CodeListDocument (page 29)

**CodeListDocument (Complex Type)**

Document type for genericode code list definitions.

**Rules:**

A code list must have at least one key, unless it is a metadata-only definition without a 'SimpleCodeList' element.

**Content Model:** ((Annotation? , Identification) , (ColumnSet | ColumnSetRef) , ((SimpleCodeList))?)

**Mixed Content:** No
Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Annotation (from DocumentHeader on page 40) | Annotation (page 28) | User annotation information. **DocumentHeader:** General information (metadata) for the code list.
| Identification (from DocumentHeader on page 40) | Identification (page 41) | Identification and location information (metadata). **DocumentHeader:** General information (metadata) for the code list.
| ColumnSet (from ColumnSetChoice on page 36) | ColumnSet (page 34) | Definition of a column set (columns and keys for the code list). **ColumnSetChoice:** A choice between a column set definition and a column set reference.
| ColumnSetRef (from ColumnSetChoice on page 36) | ColumnSetRef (page 38) | Reference to a column set defined in an external column set or code list document. **ColumnSetChoice:** A choice between a column set definition and a column set reference.
| SimpleCodeList (from SimpleCodeListSequence on page 47) | SimpleCodeList (page 47) | Simple (explicit) code list definition. **SimpleCodeListSequence:** Details of a simple code list definition. **OuterCodeListChoice:** The only choice is a simple (explicit) code list definition. Not used if the code list definition contains code list metadata only. |

**CodeListRef (Complex Type)**

Reference to a code list, possibly defined in an external document.

**Content Model:** (Annotation?, CanonicalUri, CanonicalVersionUri?, LocationUri*)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation  (page 28)</td>
<td>User annotation for the referenced code list.</td>
</tr>
<tr>
<td>CanonicalUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies all versions (collectively) of the referenced code list.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CanonicalVersionUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies a specific version of the referenced code list.</td>
</tr>
<tr>
<td>LocationUri</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this code list, in genericode format. If the CanonicalVersionUri has been defined, the location URI must point to a genericode CodeList document. If the CanonicalVersionURI is undefined, the location URI must point to a genericode CodeList Metadata document.</td>
</tr>
</tbody>
</table>

**CodeListSet (Element)**

Top-level element for the definition of a code list set.

**Complex Type:** CodeListSetDocument (page 31)

**CodeListSetDocument (Complex Type)**

Document type for the definition of a set of code lists.

**Content Model:** (Annotation?, Identification, CodeListRef*)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation (from DocumentHeader on page 40)</td>
<td>Annotation (page 28)</td>
<td>User annotation information. <strong>DocumentHeader:</strong> General document information for the code list set.</td>
</tr>
<tr>
<td>Identification (from DocumentHeader on page 40)</td>
<td>Identification (page 41)</td>
<td>Identification and location information (metadata). <strong>DocumentHeader:</strong> General document information for the code list set.</td>
</tr>
<tr>
<td>CodeListRef</td>
<td>CodeListRef (page 30)</td>
<td></td>
</tr>
</tbody>
</table>

**Column (Complex Type)**

Definition of a column.

Each column of a code list defines a piece of metadata that can be specified for each item in the code list.
**Content Model:** (Annotation?, (ShortName, LongName*), (CanonicalUri, CanonicalVersionUri?)?, Data)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User information about the column.</td>
</tr>
<tr>
<td>ShortName (from NameSet on page 45)</td>
<td>ShortName (page 47)</td>
<td>Short name (without whitespace). &lt;br&gt;<strong>NameSet:</strong> Name(s) of the column.</td>
</tr>
<tr>
<td>LongName (from NameSet on page 45)</td>
<td>LongName (page 45)</td>
<td>Human-readable name. &lt;br&gt;<strong>NameSet:</strong> Name(s) of the column.</td>
</tr>
<tr>
<td>CanonicalUri (from IdentificationVersionUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies all versions collectively. &lt;br&gt;<strong>IdentificationVersionUriSet:</strong> URIs used to identify the column and/or the version of the column.</td>
</tr>
<tr>
<td>CanonicalVersionUri (from IdentificationVersionUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies this version. &lt;br&gt;<strong>IdentificationVersionUriSet:</strong> URIs used to identify the column and/or the version of the column.</td>
</tr>
<tr>
<td>Data</td>
<td>Data (page 39)</td>
<td>Data type of the column.</td>
</tr>
</tbody>
</table>

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (from IdDefinition on page 41)</td>
<td>required</td>
<td>xsd:ID</td>
<td>Unique ID within the document. &lt;br&gt;<strong>IdDefinition:</strong> ID which identifies the column within the document.</td>
</tr>
<tr>
<td>Use (from RequiredUseDefinition on page 46)</td>
<td>required</td>
<td>UseType (page 48)</td>
<td>Whether the usage is required or optional. &lt;br&gt;<strong>RequiredUseDefinition:</strong> Whether the column is required or optional.</td>
</tr>
</tbody>
</table>

**ColumnChoice (Model Group)**

A choice between a column definition and a column reference.
**Content Model:** (Column | ColumnRef)

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>Column (page 31)</td>
<td>Definition of a column.</td>
</tr>
<tr>
<td>ColumnRef</td>
<td>ColumnRef (page 33)</td>
<td>Reference to a column defined in an external column set or code list.</td>
</tr>
</tbody>
</table>

**ColumnRef (Complex Type)**
Reference to a column defined in an external column set or code list.

**Content Model:** (Annotation? , (CanonicalVersionUri , LocationUri*) , Data?)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the referenced column.</td>
</tr>
<tr>
<td>CanonicalVersionUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which serves as a unique identifier for this version.</td>
</tr>
<tr>
<td>IdentificationRefUriSet</td>
<td></td>
<td>Identification of the external column set or code list document which contains the column set definition.</td>
</tr>
<tr>
<td>LocationUri</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this version, in genericode format.</td>
</tr>
<tr>
<td>IdentificationRefUriSet</td>
<td></td>
<td>Identification of the external column set or code list document which contains the column set definition.</td>
</tr>
<tr>
<td>Data</td>
<td>DataRestrictions (page 39)</td>
<td>Restrictions to the data type of the referenced column.</td>
</tr>
</tbody>
</table>

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (from IdDefinition on page 41)</td>
<td>required</td>
<td>xsd:ID</td>
<td>Unique ID within the document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IdDefinition: ID which identifies the column within this document.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Usage</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ExternalRef (from ExternalReference on page 41)</td>
<td>required</td>
<td>xsd:NCName</td>
<td>Unique ID within the external document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Rules:</strong> The external reference must not be prefixed with a ‘#’ symbol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>ExternalReference:</strong> ID which identifies the column within the external</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>column set or code list.</td>
</tr>
<tr>
<td>Use (from OptionalUseDefinition on page 46)</td>
<td>optional</td>
<td>UseType (page 48)</td>
<td>Whether the usage is required or optional.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>OptionalUseDefinition:</strong> Whether the column is required or optional.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If specified, this overrides the usage specified in the external column set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>or code list document.</td>
</tr>
</tbody>
</table>

**ColumnReference (Attribute Group)**

Attribute set for referring to a column definition.

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnRef</td>
<td>optional</td>
<td>xsd:IDREF</td>
<td>Reference to a column ID in the document.</td>
</tr>
</tbody>
</table>

**ColumnSet (Element)**

Top-level element for the definition of a column set.

**Complex Type:** ColumnSetDocument (page 36)

**ColumnSet (Complex Type)**

Definition of a column set (columns and keys for a code list).

**Content Model:** ((Column | ColumnRef)* , (Key | KeyRef)*)

**Mixed Content:** No
### Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Column (from ColumnChoice on page 32) | Column (page 31) | Definition of a column.  
  **ColumnChoice:**  
  A choice between a column definition and a column reference.  
  **ColumnSetContent:**  
  Column set definitions. |
| ColumnRef (from ColumnChoice on page 32) | ColumnRef (page 33) | Reference to a column defined in an external column set or code list.  
  **ColumnChoice:**  
  A choice between a column definition and a column reference.  
  **ColumnSetContent:**  
  Column set definitions. |
| Key (from KeyChoice on page 44) | Key (page 43) | Definition of a key.  
  **KeyChoice:**  
  A choice between a key definition and a key reference.  
  **ColumnSetContent:**  
  Column set definitions. |
| KeyRef (from KeyChoice on page 44) | KeyRef (page 44) | Reference to a key defined in an external column set or code list.  
  **KeyChoice:**  
  A choice between a key definition and a key reference.  
  **ColumnSetContent:**  
  Column set definitions. |

### Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| DatatypeLibrary (from DefaultDatatypeLibrary on page 40) | optional | xsd:anyURI | URI which uniquely identifies the default datatype library for the column set. If not provided, defaults to the URI for W3C XML Schema datatypes.  
  **DefaultDatatypeLibrary:**  
  Identification of the default datatype library for the column set. |
ColumnSetChoice (Model Group)

A choice between a column set definition and a column set reference.

Content Model: (ColumnSet | ColumnSetRef)

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnSet</td>
<td>ColumnSet (page 34)</td>
<td>Definition of a column set (columns and keys for the code list).</td>
</tr>
<tr>
<td>ColumnSetRef</td>
<td>ColumnSetRef (page 38)</td>
<td>Reference to a column set defined in an external column set or code list document.</td>
</tr>
</tbody>
</table>

ColumnSetContent (Model Group)

Specific details of a column set.

Content Model: ((Column | ColumnRef)* , (Key | KeyRef)*)

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column (from ColumnChoice on page 32)</td>
<td>Column (page 31)</td>
<td>Definition of a column.</td>
</tr>
<tr>
<td>ColumnChoice:</td>
<td></td>
<td>A choice between a column definition and a column reference.</td>
</tr>
<tr>
<td>ColumnRef (from ColumnChoice on page 32)</td>
<td>ColumnRef (page 33)</td>
<td>Reference to a column defined in an external column set or code list.</td>
</tr>
<tr>
<td>ColumnChoice:</td>
<td></td>
<td>A choice between a column definition and a column reference.</td>
</tr>
<tr>
<td>Key (from KeyChoice on page 44)</td>
<td>Key (page 43)</td>
<td>Definition of a key.</td>
</tr>
<tr>
<td>KeyChoice:</td>
<td></td>
<td>A choice between a key definition and a key reference.</td>
</tr>
<tr>
<td>KeyRef (from KeyChoice on page 44)</td>
<td>KeyRef (page 44)</td>
<td>Reference to a key defined in an external column set or code list.</td>
</tr>
<tr>
<td>KeyChoice:</td>
<td></td>
<td>A choice between a key definition and a key reference.</td>
</tr>
</tbody>
</table>

ColumnSetDocument (Complex Type)

Document type for the definition of a column set, which is a set of code list columns and/or keys.

Content Model: ((Annotation? , Identification) , ((Column | ColumnRef)* , (Key | KeyRef)*))
Mixed Content: No

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Annotation (from DocumentHeader on page 40) | Annotation (page 28) | User annotation information.  
DocumentHeader:  
General document information for the column set. |
| Identification (from DocumentHeader on page 40) | Identification (page 41) | Identification and location information (metadata).  
DocumentHeader:  
General document information for the column set. |
| Column (from ColumnChoice on page 32) | Column (page 31) | Definition of a column.  
ColumnChoice:  
A choice between a column definition and a column reference.  
ColumnSetContent:  
Details of the column set. |
| ColumnRef (from ColumnChoice on page 32) | ColumnRef (page 33) | Reference to a column defined in an external column set or code list.  
ColumnChoice:  
A choice between a column definition and a column reference.  
ColumnSetContent:  
Details of the column set. |
| Key (from KeyChoice on page 44) | Key (page 43) | Definition of a key.  
KeyChoice:  
A choice between a key definition and a key reference.  
ColumnSetContent:  
Details of the column set. |
<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyRef (from KeyChoice on page 44)</td>
<td>KeyRef (page 44)</td>
<td>Reference to a key defined in an external column set or code list. <strong>KeyChoice:</strong> A choice between a key definition and a key reference. <strong>ColumnSetContent:</strong> Details of the column set.</td>
</tr>
</tbody>
</table>

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatatypeLibrary (from DefaultDatatypeLibrary on page 40)</td>
<td>optional</td>
<td>xsd:anyURI</td>
<td>URI which uniquely identifies the default datatype library for the column set. If not provided, defaults to the URI for W3C XML Schema datatypes. <strong>DefaultDatatypeLibrary:</strong> Identification of the default datatype library for the column set.</td>
</tr>
</tbody>
</table>

**ColumnSetRef (Complex Type)**

Reference to a column set defined in an external column set or code list document.

**Content Model:** (Annotation? , (CanonicalVersionUri , LocationUri*))

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the referenced column set.</td>
</tr>
<tr>
<td>CanonicalVersionUri (from IdentificationRefUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td>Canonical URI which serves as a unique identifier for this version. <strong>IdentificationRefUriSet:</strong> Identification of the external column set or code list document which contains the column set definition.</td>
</tr>
<tr>
<td>LocationUri (from IdentificationRefUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this version, in genericode format. <strong>IdentificationRefUriSet:</strong> Identification of the external column set or code list document which contains the column set definition.</td>
</tr>
</tbody>
</table>
Data (Complex Type)

Data type definition.

Content Model: (Annotation?, Parameter*)

Mixed Content: No

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation</td>
<td>User annotation for the datatype.</td>
</tr>
<tr>
<td>Parameter</td>
<td>DatatypeFacet</td>
<td>Facet parameter which refines the datatype.</td>
</tr>
</tbody>
</table>

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>required</td>
<td>xsd:token</td>
<td>Unique ID for the datatype within its datatype library.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the data is complex (i.e. XML), this value is set to the root element</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>name for the XML value, or '*' if the root element name is not restricted.</td>
</tr>
<tr>
<td>DatatypeLibrary</td>
<td>optional</td>
<td>xsd:anyURI</td>
<td>URI which uniquely identifies the datatype library.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If not provided, the datatype library for the enclosing column set is used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the data is complex (i.e. XML), this value is set to the namespace URI for the XML, or '*' if the namespace URI is not restricted.</td>
</tr>
<tr>
<td>xml:lang</td>
<td>optional</td>
<td></td>
<td>Language from which the data is taken or derived.</td>
</tr>
</tbody>
</table>

DataRestrictions (Complex Type)

Restrictions to a data type.

Rules:

The 'xml:lang' attribute may be specified only if no language is set for the data type that is being restricted.

Content Model: (Parameter*)

Mixed Content: No
Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>DatatypeFacet (page 40)</td>
<td>Facet parameter which refines the datatype.</td>
</tr>
</tbody>
</table>

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml:lang</td>
<td>optional</td>
<td></td>
<td>Language from which the data is taken or derived.</td>
</tr>
</tbody>
</table>

**DatatypeFacet (Complex Type)**

Facet information for refining a datatype.

**Extension of:** xsd:string

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortName</td>
<td>required</td>
<td>xsd:token</td>
<td>Short name (token) for the datatype facet.</td>
</tr>
<tr>
<td>LongName</td>
<td>optional</td>
<td>xsd:normalizedString</td>
<td>Long name for the datatype facet.</td>
</tr>
</tbody>
</table>

**DefaultDatatypeLibrary (Attribute Group)**

Identification of the default datatype library for a column set.

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatatypeLibrary</td>
<td>optional</td>
<td>xsd:anyURI</td>
<td>URI which uniquely identifies the default datatype library for the column set. If not provided, defaults to the URI for W3C XML Schema datatypes.</td>
</tr>
</tbody>
</table>

**DocumentHeader (Model Group)**

General document information (metadata).

**Content Model:** (Annotation?, Identification)

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation information.</td>
</tr>
<tr>
<td>Identification</td>
<td>Identification (page 41)</td>
<td>Identification and location information (metadata).</td>
</tr>
</tbody>
</table>
**ExternalReference (Attribute Group)**

Attribute set used to identify a definition within an external document.

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalRef</td>
<td>required</td>
<td>xsd:NCName</td>
<td>Unique ID within the external document.</td>
</tr>
</tbody>
</table>

**Rules:**
The external reference must not be prefixed with a '#' symbol.

**GeneralIdentifier (Complex Type)**

An identifier value. Typically not a long or short name.

**Extension of:** xsd:normalizedString

**IdDefinition (Attribute Group)**

Attribute set used to identify a definition within the document.

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>required</td>
<td>xsd:ID</td>
<td>Unique ID within the document.</td>
</tr>
</tbody>
</table>

**Identification (Complex Type)**

Identification and location information (metadata).

**Content Model:** 
((ShortName , LongName*) , Version , CanonicalUri , (CanonicalVersionUri , LocationUri* , AlternateFormatLocationUri*) , Agency?)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortName (from NameSet on page 45)</td>
<td>ShortName (page 47)</td>
<td>Short name (without whitespace).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NameSet:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Various names.</td>
</tr>
<tr>
<td>LongName (from NameSet on page 45)</td>
<td>LongName (page 45)</td>
<td>Human-readable name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NameSet:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Various names.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>CanonicalUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies all versions (collectively).</td>
</tr>
<tr>
<td>CanonicalVersionUri (from VersionLocationUriSet on page 49)</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies this version.</td>
</tr>
<tr>
<td>LocationUri (from VersionLocationUriSet on page 49)</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this version, in genericode format.</td>
</tr>
<tr>
<td>AlternateFormatLocationUri (from VersionLocationUriSet on page 49)</td>
<td>MimeTypedUri (page 45)</td>
<td>Suggested retrieval location for this version, in non-genericode format.</td>
</tr>
<tr>
<td>Agency</td>
<td>Agency (page 28)</td>
<td>Agency that is responsible for publication and/or maintenance of the information.</td>
</tr>
</tbody>
</table>

**IdentificationRefUriSet (Model Group)**

Identification and location URIs.

**Content Model:** (CanonicalVersionUri, LocationUri*)

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanonicalVersionUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which serves as a unique identifier for this version.</td>
</tr>
<tr>
<td>LocationUri</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this version, in genericode format.</td>
</tr>
</tbody>
</table>

**IdentificationVersionUriSet (Model Group)**

URIs used as unique identifiers.

**Content Model:** (CanonicalUri, CanonicalVersionUri?)

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanonicalUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies all versions collectively.</td>
</tr>
<tr>
<td>CanonicalVersionUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies this version.</td>
</tr>
</tbody>
</table>
Key (Complex Type)

Definition of a key.

A key is a set of one or more columns whose values together provide a unique identification of each item in a code list.

Rules:

Only required columns can be used for keys.

Content Model: \((\text{Annotation?} , (\text{ShortName} , \text{LongName}^{*}) , (\text{CanonicalUri} , \text{CanonicalVersionUri}? , \text{ColumnRef}^{+})\)

Mixed Content: No

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation</td>
<td>(page 28)</td>
</tr>
<tr>
<td>ShortName (from NameSet on page 45)</td>
<td>ShortName</td>
<td>(page 47)</td>
</tr>
<tr>
<td>NameSet:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LongName (from NameSet on page 45)</td>
<td>LongName</td>
<td>(page 45)</td>
</tr>
<tr>
<td>NameSet:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CanonicalUri (from IdentificationVersionUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td></td>
</tr>
<tr>
<td>IdentificationVersionUriSet:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CanonicalVersionUri (from IdentificationVersionUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td></td>
</tr>
<tr>
<td>IdentificationVersionUriSet:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ColumnRef</td>
<td>KeyColumnRef</td>
<td>(page 44)</td>
</tr>
</tbody>
</table>

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (from IdDefinition on page 41)</td>
<td>required</td>
<td>xsd:ID</td>
<td>Unique ID within the document.</td>
</tr>
<tr>
<td>IdDefinition:</td>
<td></td>
<td></td>
<td>ID which identifies the key within the document.</td>
</tr>
</tbody>
</table>
**KeyChoice (Model Group)**

A choice between a key definition and a key reference.

**Content Model:** (Key | KeyRef)

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Key (page 43)</td>
<td>Definition of a key.</td>
</tr>
<tr>
<td>KeyRef</td>
<td>KeyRef (page 44)</td>
<td>Reference to a key defined in an external column set or code list.</td>
</tr>
</tbody>
</table>

**KeyColumnRef (Complex Type)**

Reference to a column which forms part of a key.

**Content Model:** (Annotation?)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the column.</td>
</tr>
</tbody>
</table>

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref</td>
<td>required</td>
<td>xsd:IDREF</td>
<td>Reference to the ID of the column within the document.</td>
</tr>
</tbody>
</table>

**KeyRef (Complex Type)**

Reference to a key defined in an external column set or code list.

**Content Model:** (Annotation?, (CanonicalVersionUri, LocationUri*))

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the referenced key.</td>
</tr>
<tr>
<td>CanonicalVersionUri (from IdentificationRefUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td>Canonical URI which serves as a unique identifier for this version.</td>
</tr>
</tbody>
</table>

**IdentificationRefUriSet:**

Identification of the external column set or code list which contains the key definition.
<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocationUri (from IdentificationRefUriSet on page 42)</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this version, in genericode format. IdentificationRefUriSet: Identification of the external column set or code list which contains the key definition.</td>
</tr>
</tbody>
</table>

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (from IdDefinition on page 41)</td>
<td>required</td>
<td>xsd:ID</td>
<td>Unique ID within the document. IdDefinition: ID which identifies the key within this document.</td>
</tr>
<tr>
<td>ExternalRef (from ExternalReference on page 41)</td>
<td>required</td>
<td>xsd:NCName</td>
<td>Unique ID within the external document. Rules: The external reference must not be prefixed with a ‘#’ symbol. ExternalReference: ID which identifies which identifies the key within the external column set or code list.</td>
</tr>
</tbody>
</table>

**LongName (Complex Type)**

A human-readable name. Extension of: xsd:normalizedString

**MimeTypedUri (Complex Type)**

URI for a resource, with support for specifying the MIME type. Extension of: xsd:anyURI

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MimeType</td>
<td>optional</td>
<td>xsd:normalizedString</td>
<td>MIME type of the resource which can be retrieved from the URI.</td>
</tr>
</tbody>
</table>

**NameSet (Model Group)**

Various names. Content Model: (ShortName , LongName*)
### Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ShortName</td>
<td>ShortName (page 47)</td>
<td>Short name (without whitespace).</td>
</tr>
<tr>
<td>LongName</td>
<td>LongName (page 45)</td>
<td>Human-readable name.</td>
</tr>
</tbody>
</table>

### OptionalUseDefinition (Attribute Group)

Attribute set which defines the usage (optional attribute).

### Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>optional</td>
<td>UseType (page 48)</td>
<td>Whether the usage is required or optional.</td>
</tr>
</tbody>
</table>

### OuterCodeListChoice (Model Group)

A choice which currently only allows a simple (explicit) code list definition.

**Content Model:** (SimpleCodeList)

### Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleCodeListSequence:</td>
<td></td>
<td>Details of a simple code list definition.</td>
</tr>
</tbody>
</table>

### RequiredUseDefinition (Attribute Group)

Attribute set which defines the usage (required attribute).

### Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use</td>
<td>required</td>
<td>UseType (page 48)</td>
<td>Whether the usage is required or optional.</td>
</tr>
</tbody>
</table>

### Row (Complex Type)

Row which represents an individual item in a code list.

**Content Model:** (Annotation?, Value+)

**Mixed Content:** No
### Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the row.</td>
</tr>
<tr>
<td>Value</td>
<td>Value (page 48)</td>
<td>Column value for the row. Column value must be provided for each required column. Column value does not need to be provided for a column if the column is optional. If a value does not have an explicit column reference, the column is taken to be the column following the column of the preceding value in the row, or the first column if the value is the first value of the row.</td>
</tr>
</tbody>
</table>

### ShortName (Complex Type)

A short name without whitespace that is suitable for use in generating names for software artifacts.

**Extension of:** xsd:token

**Attributes:**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml:lang</td>
<td>optional</td>
<td></td>
<td>The language from which the short name is taken or derived.</td>
</tr>
</tbody>
</table>

### SimpleCodeList (Complex Type)

Simple (explicit) code list definition.

**Rules:**

Applications must not have any dependency on the ordering of the rows.

**Content Model:** (Annotation?, Row*)

**Mixed Content:** No

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the code list.</td>
</tr>
<tr>
<td>Row</td>
<td>Row (page 46)</td>
<td>Row which represents an individual item in the code list.</td>
</tr>
</tbody>
</table>

### SimpleCodeListSequence (Model Group)

Details of a simple code list definition.

**Content Model:** (SimpleCodeList)
Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleCodeList</td>
<td>SimpleCodeList (page 47)</td>
<td>Simple (explicit) code list definition.</td>
</tr>
</tbody>
</table>

**SimpleValue (Complex Type)**

Simple textual value.

Extension of: xsd:string

**UseType (Simple Type)**

Indicates whether the usage is required or optional.

Restriction of: xsd:token

Allowed Values:
- optional
- required

**Value (Complex Type)**

An individual code list metadata value.

Content Model: (Annotation?, (SimpleValue | ComplexValue)?)

Mixed Content: No

Elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation</td>
<td>Annotation (page 28)</td>
<td>User annotation for the value.</td>
</tr>
<tr>
<td>SimpleValue (from ValueChoice on page 49)</td>
<td>SimpleValue (page 48)</td>
<td>Simple textual value. ValueChoice: A choice between a simple textual value and a complex (structured) XML value. If the value is undefined, then neither choice is used.</td>
</tr>
<tr>
<td>ComplexValue (from ValueChoice on page 49)</td>
<td>AnyOtherContent (page 29)</td>
<td>Complex (structured) XML value. ValueChoice: A choice between a simple textual value and a complex (structured) XML value. If the value is undefined, then neither choice is used.</td>
</tr>
</tbody>
</table>
Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColumnRef (from ColumnReference on page 34)</td>
<td>optional</td>
<td>xsd:IDREF</td>
<td>Reference to a column ID in the document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ColumnReference: Reference to the column with which this value is associated.</td>
</tr>
</tbody>
</table>

**ValueChoice (Model Group)**

A choice between a simple textual value and a complex (structured) XML value.

**Content Model:** (SimpleValue | ComplexValue)

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleValue</td>
<td>SimpleValue (page 48)</td>
<td>Simple textual value.</td>
</tr>
<tr>
<td>ComplexValue</td>
<td>AnyOtherContent (page 29)</td>
<td>Complex (structured) XML value.</td>
</tr>
</tbody>
</table>

**ValueIdentification (Attribute Group)**

Information which identifies one of a set of alternate values.

Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>optional</td>
<td>xsd:normalizedString</td>
<td>A string which identifies one of a set of alternate values.</td>
</tr>
<tr>
<td>xml:lang</td>
<td>optional</td>
<td></td>
<td>The language from which the value is taken or derived.</td>
</tr>
</tbody>
</table>

**VersionLocationUriSet (Model Group)**

Identification and location URIs for a version.

**Rules:**

The canonical version URI must not be used by applications as an implicit location URI from which to try and retrieve information.

**Content Model:** (CanonicalVersionUri , LocationUri* , AlternateFormatLocationUri*)

**Elements:**

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanonicalVersionUri</td>
<td>xsd:anyURI</td>
<td>Canonical URI which uniquely identifies this version.</td>
</tr>
<tr>
<td>Element</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LocationUri</td>
<td>xsd:anyURI</td>
<td>Suggested retrieval location for this version, in genericode format.</td>
</tr>
<tr>
<td>AlternateFormatLocationUri</td>
<td>MimeTypedUri (page 45)</td>
<td>Suggested retrieval location for this version, in a non-genericode format.</td>
</tr>
</tbody>
</table>
Appendix A. Acknowledgments

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

Participants:

- Jon Bosak, Sun Microsystems
- Anthony Coates, Miley Watts LLP
- Ray Denenberg, Library of Congress
- Marc Gratacos, Associate Member
- Jim Harris, National Center for State Courts
- G. Ken Holman, Associate Member
- Paul Spencer, Associate Member
Appendix B. Example Code Lists in Genericode Format

This section is non-normative.

B.1. UBL Example – Country Codes

This is an example of a single-key code list which includes agency metadata. Notice that only the top-level "CodeList" element requires a "gc:" namespace prefix. There are 3 columns and 1 key defined in the embedded ColumnSet:

- Code, column, normalized string, required;
- Name, column, string, optional;
- NumericCode, column, string, optional;
- CodeKey, key cased on "Code" column.

Note that the values in each row have an explicit reference (ColumnRef) to the matching column (some of these are highlighted in yellow).

```xml
<?xml version="1.0" encoding="ISO-8859-1"?
<gc:CodeList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
  xmlns:column="http://docs.oasis-open.org/codelist/ns/genericode/column/1.0/"
  xmlns:columnref="http://docs.oasis-open.org/codelist/ns/genericode/columnref/1.0/"/>
  <Identification>
      <ShortName>CountryIdentificationCode</ShortName>
      <LongName xml:lang="en">Country</LongName>
      <LongName Identifier="listID">ISO3166-1</LongName>
      <Version>0.3</Version>
      <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/default/CountryIdentificationCode-2.0.gc</LocationUri>
      <Agency>
        <LongName xml:lang="en">United Nations Economic Commission for Europe</LongName>
        <Identifier>6</Identifier>
      </Agency>
      </Identification>
      <ColumnSet>
        <Column Id="code" Use="required">
          <ShortName>Code</ShortName>
          <Data Type="normalizedString"/>
        </Column>
        <Column Id="name" Use="optional">
          <ShortName>Name</ShortName>
          <Data Type="string"/>
        </Column>
        <Column Id="numericcode" Use="optional">
          <ShortName>NumericCode</ShortName>
          <Data Type="string"/>
        </Column>
        <Key Id="codeKey">
          <ShortName>CodeKey</ShortName>
        </Key>
      </ColumnSet>
    </codeList>
```
<ColumnRef Ref="code"/>
</Key>
</ColumnSet>
<SimpleCodeList>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>AF</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>AFGHANISTAN</SimpleValue>
    </Value>
    <Value ColumnRef="numericcode">
      <SimpleValue>004</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>AL</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>ALBANIA</SimpleValue>
    </Value>
    <Value ColumnRef="numericcode">
      <SimpleValue>008</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>ZM</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>ZAMBIA</SimpleValue>
    </Value>
    <Value ColumnRef="numericcode">
      <SimpleValue>894</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="code">
      <SimpleValue>ZW</SimpleValue>
    </Value>
    <Value ColumnRef="name">
      <SimpleValue>ZIMBABWE</SimpleValue>
    </Value>
    <Value ColumnRef="numericcode">
      <SimpleValue>716</SimpleValue>
    </Value>
  </Row>
</SimpleCodeList>

This example could also have been implemented without explicit column references on the values, as the column can be inferred based on the position.
It is *not* an official UBL document.

```xml
<Description>
</Description>
<Identification>
  <ShortName>CountryIdentificationCode</ShortName>
  <LongName xml:lang="en">Country</LongName>
  <LongName Identifier="listID">ISO3166-1</LongName>
  <Version>0.3</Version>
  <CanonicalUri>http://www.example.com/ubl/codelist/genericode/CountryIdentificationCode</CanonicalUri>
  <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/default/CountryIdentificationCode-2.0.gc</LocationUri>
  <Agency>
    <LongName xml:lang="en">United Nations Economic Commission for Europe</LongName>
    <Identifier>6</Identifier>
  </Agency>
</Identification>

<ColumnSet>
  <Column Id="code" Use="required">
    <ShortName>Code</ShortName>
    <Data Type="normalizedString"/>
  </Column>
  <Column Id="name" Use="optional">
    <ShortName>Name</ShortName>
    <Data Type="string"/>
  </Column>
  <Column Id="numericcode" Use="optional">
    <ShortName>NumericCode</ShortName>
    <Data Type="string"/>
  </Column>
  <Key Id="codeKey">
    <ShortName>CodeKey</ShortName>
    <ColumnRef Ref="code"/>
  </Key>
</ColumnSet>

<SimpleCodeList>
  <Row>
    <Value>
      <SimpleValue>AF</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value>
      <SimpleValue>AFGHANISTAN</SimpleValue>
    </Value>
  </Row>
  <Row>
    <Value>
      <SimpleValue>004</SimpleValue>
    </Value>
  </Row>
  <!-- ... -->
  <Row>
    <Value>
      <SimpleValue>ZM</SimpleValue>
    </Value>
  </Row>
</SimpleCodeList>
```
B.2. FpML Example – Business Centers

This is an example of a single-key code list without agency metadata, but with user metadata. Note that the column for each value is inferred based on the position of the value in the row. Note also that only the top-level "CodeList" element requires a "gc1:" namespace prefix. There are 3 columns and 1 key defined in the embedded ColumnSet:

- Code, column, token, required;
- Source, column, string, optional;
- Description, column, string, optional;
- PrimaryKey, key cased on "Code" column.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gcl:CodeList
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:doc="http://www.fpml.org/coding-scheme/documentation"
    xmlns:gcl="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
>
    <Annotation/>
    <Description>
        A financial business center location
    </Description>
</gcl:CodeList>
```

In general, the codes are based on the ISO country code and the English name of the location. Additional location codes can be built according to the following rules. The first two characters represent the ISO country code, the next two characters represent a) if the location name is one word, the first two letters of the location b) if the location name consists of at least two words, the first letter of the first word followed by the first letter of the second word.

There are exceptions to this rule. For example, the TARGET (Trans-European Automated Real-time Gross settlement Express Transfer system) business center for Euro settlement has a code of EUTA.
This coding scheme is currently consistent with the S.W.I.F.T. Financial Center scheme used in the MT340/MT360/MT361 message definitions, although FpML controls the Business Center Scheme and it should not be assumed that both schemes will remain synchronized.
B.3. Multiple Key Example

This is an example of a code list with multiple alternative keys for the distinct entries in the code list. There are 3 alternative keys, one for each revision of the ISO639 standard. Note that the columns have data types and facets (Parameter element) defined. Note also the (optional) use of XHTML for human-readable annotations.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gc:CodeList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
xmlns:html="http://www.w3.org/1999/xhtml/">
  <Annotation>
    <Description>
    </Description>
  </Annotation>
  <Identification>
    <ShortName>iso639-1</ShortName>
    <Version>1.0</Version>
    <CanonicalUri>http://www.example.com/languages/iso639-1</CanonicalUri>
    <CanonicalVersionUri>http://www.example.com/languages/iso639-1/1.0</CanonicalVersionUri>
  </Identification>
  <ColumnSet>
    <Column Id="col-iso639-1" Use="required">
      <ShortName>ISO639-1</ShortName>
      <Data Type="string">
        <Parameter ShortName="pattern">[a-z]{2}</Parameter>
      </Data>
    </Column>
    <Column Id="col-iso639-2" Use="required">
      <ShortName>ISO639-2</ShortName>
      <Data Type="string">
        <Parameter ShortName="pattern">[a-z]{3}</Parameter>
      </Data>
    </Column>
    <Column Id="col-iso639-3" Use="required">
      <ShortName>ISO639-3</ShortName>
      <Data Type="string">
        <Parameter ShortName="pattern">[a-z]{3}</Parameter>
      </Data>
    </Column>
  </ColumnSet>
</gc:CodeList>
```
<table>
<thead>
<tr>
<th>Id</th>
<th>Use</th>
<th>ShortName</th>
<th>LongName</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>col-language-name</td>
<td>required</td>
<td>LanguageName</td>
<td>Language Name</td>
<td>string</td>
</tr>
<tr>
<td>col-scope</td>
<td>required</td>
<td>Scope</td>
<td></td>
<td>string</td>
</tr>
<tr>
<td>col-type</td>
<td>required</td>
<td>Type</td>
<td></td>
<td>string</td>
</tr>
</tbody>
</table>

<ColumnSet>
  <SimpleCodeList>
    <Row>
      <Value><SimpleValue>aa</SimpleValue></Value>
      <Value><SimpleValue>aar</SimpleValue></Value>
      <Value><SimpleValue>aar</SimpleValue></Value>
      <Value><SimpleValue>Afar</SimpleValue></Value>
      <Value><SimpleValue>Individual</SimpleValue></Value>
      <Value><SimpleValue>Living</SimpleValue></Value>
    </Row>
    <Row>
      <Value><SimpleValue>ab</SimpleValue></Value>
      <Value><SimpleValue>abk</SimpleValue></Value>
      <Value><SimpleValue>abk</SimpleValue></Value>
      <Value><SimpleValue>Abkhazian</SimpleValue></Value>
      <Value><SimpleValue>Individual</SimpleValue></Value>
      <Value><SimpleValue>Living</SimpleValue></Value>
    </Row>
    <Row>
      <Value><SimpleValue>ae</SimpleValue></Value>
      <Value><SimpleValue>ave</SimpleValue></Value>
      <Value><SimpleValue>ave</SimpleValue></Value>
      <Value><SimpleValue>Avestan</SimpleValue></Value>
      <Value><SimpleValue>Individual</SimpleValue></Value>
      <Value><SimpleValue>Ancient</SimpleValue></Value>
    </Row>
    <Row>
      <Value><SimpleValue>af</SimpleValue></Value>
      <Value><SimpleValue>afr</SimpleValue></Value>
      <Value><SimpleValue>afr</SimpleValue></Value>
      <Value><SimpleValue>Afrikaans</SimpleValue></Value>
      <Value><SimpleValue>Individual</SimpleValue></Value>
      <Value><SimpleValue>Living</SimpleValue></Value>
    </Row>
    <Row>
      <Value><SimpleValue>ak</SimpleValue></Value>
      <Value><SimpleValue>aka</SimpleValue></Value>
      <Value><SimpleValue></Value>
      <Value><SimpleValue>Ancient</SimpleValue></Value>
      <Value><SimpleValue>Living</SimpleValue></Value>
    </Row>
  </SimpleCodeList>
</ColumnSet>
B.4. Undefined Values Example

This example is a variation of the "Multiple Key Example" on page 57. Some of the columns are optional, meaning that the value for that column can be left undefined in a row. Some of the rows have defined values for all of the columns, some do not (see, for example, the final Row element).
<table>
<thead>
<tr>
<th>ISO 639-2</th>
<th>ISO 639-3</th>
<th>Language Name</th>
<th>Scope</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ach</td>
<td>ach</td>
<td>Acoli</td>
<td>Individual</td>
<td>Living</td>
</tr>
<tr>
<td>ada</td>
<td>ada</td>
<td>Adangme</td>
<td>Individual</td>
<td>Living</td>
</tr>
<tr>
<td>ady</td>
<td>ady</td>
<td>Adyghe</td>
<td>Individual</td>
<td>Living</td>
</tr>
<tr>
<td>afa</td>
<td></td>
<td>Afro-Asiatic (Other)</td>
<td>Collective</td>
<td></td>
</tr>
</tbody>
</table>
B.5. Complex (XML) Values Example

This example is a variation of the second example from "UBL Example – Country Codes" on page 52. An extra column has been added containing XHTML markup to display an image for each country in the code list.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<gc:CodeList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
    xmlns:html="http://www.w3.org/1999/xhtml/">
    <Annotation>
        <Description>
            This is an example based on a UBL code list. It is <html:strong>not</html:strong> an official UBL document.</Description>
    </Annotation>
</Identification>
<ColumnSet>
    <Column Id="code" Use="required">
        <ShortName>Code</ShortName>
        <Data Type="normalizedString"/>
    </Column>
    <Column Id="name" Use="optional">
        <ShortName>Name</ShortName>
        <Data Type="string"/>
    </Column>
    <Column Id="numericcode" Use="optional">
        <ShortName>NumericCode</ShortName>
        <Data Type="string"/>
    </Column>
    <Column Id="imagehtml" Use="required">
        <ShortName>ImageHTML</ShortName>
        <LongName xml:lang="en">Satellite Image (X)HTML</LongName>
        <Data Type="img" DatatypeLibrary="http://www.w3.org/1999/xhtml/"/>
    </Column>
    <Key Id="codeKey">
        <ShortName>CodeKey</ShortName>
        <ColumnRef Ref="code"/>
    </Key>
</ColumnSet>
</SimpleCodeList>
```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>Afghanistan</td>
<td>AL</td>
<td>Albania</td>
<td>ZM</td>
<td>Zambia</td>
</tr>
<tr>
<td>ZW</td>
<td>Zimbabwe</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B.6. External Column Set Example

This example is a variation of the second example from "UBL Example – Country Codes" on page 52. The embedded column set (column and key definitions) has been moved to a separate genericode file. First the column set:

```xml
<gc:ColumnSet xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:html="http://www.w3.org/1999/xhtml/">
    <Annotation>
        <Description>
            This is an example based on a UBL code list. It is <html:strong>not</html:strong> an official UBL document.
        </Description>
    </Annotation>
    <Identification>
        <ShortName>UBLCodeListColumnSet</ShortName>
        <LongName xml:lang="en">UBL Code List Column Set (Example)</LongName>
        <Version>0.1</Version>
        <CanonicalUri>http://www.example.com/ubl/codelist/genericode/columnset</CanonicalUri>
        <CanonicalVersionUri>http://www.example.com/ubl/codelist/genericode/columnset/1.0/</CanonicalVersionUri>
        <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/default/UBLColumnSet.gc</LocationUri>
    </Identification>
    <Column Id="code" Use="required">
        <ShortName>Code</ShortName>
        <Data Type="normalizedString"/>
    </Column>
    <Column Id="name" Use="optional">
        <ShortName>Name</ShortName>
        <Data Type="string"/>
    </Column>
    <Column Id="numericcode" Use="optional">
        <ShortName>NumericCode</ShortName>
        <Data Type="string"/>
    </Column>
    <Key Id="codeKey">
        <ShortName>CodeKey</ShortName>
        <ColumnRef Ref="code"/>
    </Key>
</gc:ColumnSet>
```

Now the code list that refers to the column set. The column/key references are highlighted in yellow:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<gc:ColumnSet xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:html="http://www.w3.org/1999/xhtml/">
    <Annotation>
        <Description>
            This is an example based on a UBL code list. It is <html:strong>not</html:strong> an official UBL document.
        </Description>
    </Annotation>
    <Identification>
        <ShortName>UBLCodeListColumnSet</ShortName>
        <LongName xml:lang="en">UBL Code List Column Set (Example)</LongName>
        <Version>0.1</Version>
        <CanonicalUri>http://www.example.com/ubl/codelist/genericode/columnset</CanonicalUri>
        <CanonicalVersionUri>http://www.example.com/ubl/codelist/genericode/columnset/1.0/</CanonicalVersionUri>
        <LocationUri>http://docs.oasis-open.org/ubl/os-ubl-2.0/cl/gc/default/UBLColumnSet.gc</LocationUri>
    </Identification>
    <Column Id="code" Use="required">
        <ShortName>Code</ShortName>
        <Data Type="normalizedString"/>
    </Column>
    <Column Id="name" Use="optional">
        <ShortName>Name</ShortName>
        <Data Type="string"/>
    </Column>
    <Column Id="numericcode" Use="optional">
        <ShortName>NumericCode</ShortName>
        <Data Type="string"/>
    </Column>
    <Key Id="codeKey">
        <ShortName>CodeKey</ShortName>
        <ColumnRef Ref="code"/>
    </Key>
</gc:ColumnSet>
```
This is an example based on a UBL code list. It is not an official UBL document.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Numeric Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF</td>
<td>AFGHANISTAN</td>
<td>004</td>
</tr>
</tbody>
</table>

United Nations Economic Commission for Europe
### B.7. Code List Set Example

This example is a code list set derived from the code lists distributed with UBL 2.0.

```xml
<Value>
  <SimpleValue>AL</SimpleValue>
</Value>
<Value>
  <SimpleValue>ALBANIA</SimpleValue>
</Value>
<Value>
  <SimpleValue>008</SimpleValue>
</Value>
</Row>
<!-- ... -->
</SimpleCodeList>
</gc:CodeList>

<table>
<thead>
<tr>
<th>Country</th>
<th>Code</th>
<th>Code List ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albania</td>
<td>AL</td>
<td>ALBANIA</td>
</tr>
<tr>
<td>Zambia</td>
<td>ZM</td>
<td>ZAMBIA</td>
</tr>
<tr>
<td>Zimbabwe</td>
<td>ZW</td>
<td>ZIMBABWE</td>
</tr>
</tbody>
</table>
```

It is **not** an official UBL document. © OASIS® 2007. All Rights Reserved.
The full version of the example was generated using the following XQuery and the Saxon XSLT processor.

```xml
declare namespace html = "http://www.w3.org/1999/xhtml/";
declare namespace cls = "http://www.example.com/xquery/genericode/codelistset/";
declare namespace gc = "http://docs.oasis-open.org/codelist/ns/genericode/1.0/";
(: UBL 2.0 uses an older, draft version of genericode. :) 
declare namespace ublgc = "http://genericode.org/2006/ns/CodeList/0.4/";
declare variable $UBLCodeListDirectoryUrl as xs:anyURI external;
declare function cls:list-code-lists() as node()* {
    for $file in collection(concat($UBLCodeListDirectoryUrl, '?recurse=yes;select=*.gc'))
        return cls:reference-code-list($file);}
declare function cls:reference-code-list($file as node()) as node()* {
    for $identification in $file/ublgc:CodeList/Identification
        return <CodeListRef>
            <CanonicalUri>
                {$identification/CannoticalUri/text() }
            </CanonicalUri>
        </CodeListRef>
    };
```
{ if (exists($identification/CanonicalVersionUri))
   then element CanonicalVersionUri {
       $identification/CanonicalVersionUri/text() }
   else ()
 }
{ if (exists($identification/LocationUri))
   then for $location in $identification/LocationUri
       return element LocationUri { $location/text() }
   else element LocationUri { document-uri($file) }
 }
</CodeListRef>

<gc:CodeListSet xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
    xmlns:xhtml="http://www.w3.org/1999/xhtml/
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://docs.oasis-open.org/codelist/ns/genericode/1.0/ ../xsd/CodeList.xsd">
<Annotation>
   <Description>
      <html:p>This is an example based on UBL 2.0. It is <html:strong>not</html:strong> an official UBL document.</html:p>
   </Description>
</Annotation>
<ShortName>UBL-CodeListSet-2-0</ShortName>
<Version>2.0</Version>
<CanonicalUri>http://www.example.com/ubl/codelist/genericode/codelistset</CanonicalUri>
<CanonicalVersionUri>http://www.example.com/ubl/codelist/genericode/code
listset/2.0</CanonicalVersionUri>
</Identification>
{ cls:list-code-lists() }
</gc:CodeListSet>