Code List Representation Requirements Version 1.0.1

2 May 2007

Document URIs:

This Version:

http://docs.oasis-open.org/codelist/cd-genericode-1.0/doc/oasis-code-list-representation-requirements-1.0.1.odt

http://docs.oasis-open.org/codelist/cd-genericode-1.0/doc/oasis-code-list-representation-requirements-1.0.1.pdf

Previous Version:

http://www.oasis-open.org/committees/download.php/20944/oasis-code-list-representation-requirements-1.0.pdf

Latest Version:

http://docs.oasis-open.org/codelist/genericode/doc/oasis-code-list-representation-requirements.odt

http://docs.oasis-open.org/codelist/genericode/doc/oasis-code-list-representation-requirements.pdf

Latest Approved Version:

http://docs.oasis-open.org/codelist/approved/genericode/doc/oasis-code-list-representation-requirements.odt

http://docs.oasis-open.org/codelist/approved/genericode/doc/oasis-code-list-representation-requirements.pdf

Technical Committee:

OASIS Code List Representation TC

Chair(s):

G. Ken Holman, Associate Member, gkholman@CraneSoftwrights.com

Editor(s):

Anthony B. Coates, Miley Watts LLP, abcoates@mileywatts.com

Abstract:

This document contains the current and future (outstanding) requirements for the OASIS Code List Representation format, known as “genericode”.

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/.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1 Introduction

This document contains the current and future (outstanding) requirements for the OASIS Code List Representation format, known as “genericode”[1]. This is the first part of the work of the OASIS Code List Representation TC. A key principle for accepting requirements for this work is that new requirements should not have a significant adverse impact on the implementation (and quality thereof) of existing requirements. For example, some potential future requirements that would be relatively straightforward to implement for explicitly defined code lists have not yet been accepted because of the difficulty in correctly integrating them with the planned support for derived code lists.

The OASIS Code List Representation format has a tabular model for code lists. The “rows” are individual entries in a code list, where an entry is a set of one or more codes, plus other metadata, that is associated with a single conceptual entry in the code list. The “columns” are individual (typed) pieces of metadata that can be applied to each entry in a code list. So columns define what kind of data can be in the code list, while rows define what actual data is in the code list.

The code list format also supports the concept of “keys”, where a key is a set of one or more columns which uniquely identifies each row in the code list. Where a key has more than one column, it is a compound key.

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

[RFC 2119] S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt.

1.3 Non-normative References

[XML 2004] Anthony B. Coates. *Why are simple code lists so complex?* XML 2004 conference proceedings. http://www.idealliance.org/proceedings/xml04/abstracts/paper86.html.

[1]Genericode 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.

2 Version 1.0 Requirements

These are the requirements for version 1.0 of the OASIS Code List Representation format.

2.1 R1.1 The code list representation format should be a pure code list representation that is not tied to any particular code list validation process or software

2.2 R1.2 The code list metamodel should be expressed as a UML logical model, not just as a physical model (e.g. an XML schema)

2.3 R1.3 The code list format should support complex code list definitions, but it should not be complex to define simple code lists

2.4 R1.4 Support for multiple, alternative codes

It must be possible to have multiple, alternative codes for the same code list. For example, there are both 2-letter and 3-letter ISO country codes. It should be possible to include both as part of the same representation of the ISO country code list.

2.5 R1.5 No particular choice of code is the preferred choice

Where a code list has multiple, alternative codes, there must be no “preferred” choice of code. The choice of code is a “late binding” decision that is made by users of the code list, not by publishers of the code list.

2.6 R1.6 Arbitrary metadata can be added at all levels in the code list representation

This includes column sets, code lists, code list sets, rows, columns, keys, and values.

2.7 R1.7 Codes are part of the metadata for the code list entries

The codes in a code list (whether multiple or not) are part of the metadata for the entries in the code list. They may be used as codes in some contexts, but they be used as non-code metadata in other contexts.

2.8 R1.8 Only unique metadata can be used for codes

Columns can be part of a key (a “code” in common parlance) only if the entries (rows) in those columns are unique, i.e. no two rows in the code list have the same key value(s).

2.9 R1.9 Every code list must have at least one key

A code list must have at least one key, since the keys are the “codes” in common parlance.

2.10 R1.10 Code list metadata does not need to be unique

Columns are not required to contain unique metadata values unless they are used in a key. In a compound key, individual columns can contain non-unique metadata values, so long as the compound key value is unique.

2.11 R1.11 Unique metadata does not have to be defined as an alternative code

Metadata columns (or sets thereof) with unique values do not have to be defined as keys for a code list. For example, some columns may contain "gratuitously unique" data – data that is currently unique, but which is not guaranteed to be unique over the life of the code list (and its versions).

2.12 R1.12 The order of codes in a code list is not important

The order of codes (rows) in a code list (in an XML file or other ordered representation) should not be used to convey any meaning (semantic information). The code list metadata itself (any set of columns) should be used to define any ordering that is appropriate, in a way that is independent of the ordering in any particular code representation.

2.13 R1.13 The order of metadata in a code list is not important

The order of metadata (columns) in a code list should not be used to convey any meaning. Column identifiers and/or column metadata (which applies to the column, not to any of the rows) should be used to identify columns.

2.14 R1.14 Metadata can be simple or complex in structure

Metadata columns can contain "simple" data values (in the sense of schema simple types), or they can contain "complex" data values (XML fragments).

2.15 R1.15 Metadata can have a particular simple type

Metadata columns can have a defined data type. The W3C XML Schema simple types are the default set, but a different datatype library can be specified (through the use of an identifying URI).

Facets can be defined to restrict the data type (e.g. length, minimum/maximum, or pattern restrictions).

Data typing only applies to "simple valued" (text) values, not to complex (XML) values. Complex values using XML namespaces rather than datatypes.

2.16 R1.16 Metadata can be left undefined

Particular metadata columns can be defined as "nillable", meaning that they can contain undefined (nil) values. Nillable columns cannot be used as part of a key.

2.17 R1.17 Column sets can be represented

Sets of code list columns can be defined independently of any particular code lists, and any number of the columns from a column set can be used in the definition of a code list or another column set.

2.18 R1.18 Column sets can contain keys

Column sets can contain keys which can be used in the definition of a code list or another column set. Where a key from a column set is used in a code list or another column set, all of the columns in that key must also be used.

2.19 R1.19 Code lists can use columns and keys defined in other code lists or in column sets

Where a key from a code list is used in another code list, all of the columns in that key must also be used.

References to externally defined columns or keys must use the unique identifier for the column/key, and may optionally include one or more location URIs for the column set or code list in which the column/key is defined.

2.20 R1.20 Metadata-only code lists can be represented

Code lists which contain metadata only (no rows) can be represented. These are code lists for which the row data is not published. The representation must be different from that of code lists which are empty (zero rows), but which nonetheless contain a complete publication of the data in the code list.

2.21 R1.21 Each code list has a unique identifier, independent of its individual versions

2.22 R1.22 Each code list version has a unique identifier, different to the version-independent identifier for the code list

The code list definition contains the version number (or string) as well as the unique identifier (a URI). This applies even if the code list representation is a metadata-only representation (which does not define the code list values in the code list).

2.23 R1.23 Each column or key in a code list or column set can have a unique identifier

2.24 R1.24 Location URIs are distinct from identification URIs

It should not be assumed that URIs used for identification correspond to any retrievable asset (even if the URI is a URL). There must be explicit support for (multiple) location URIs for retrievable representations of code list sets, column sets, or code lists.

2.25 R1.25 Sets of code list versions can be represented

It must be possible to specify a “configuration” of versions of code lists that together form a coherent set for some purpose. It must be possible to refer to either a version-independent code list URIs or a version-specific code list URI for each code list in the set.

2.26 R1.26 Documentation and annotations can be applied to definitions

Documentation and arbitrary annotation data (in the sense of W3C XML Schema) must be able to be specified for code list sets, column sets, code lists, columns, keys, rows, and individual values in rows.

2.27 R1.27 Documentation has a language identifier, and there can be documentation in multiple languages

2.28 R1.28 Short and long names are supported

A code list set, column set, code list, column, or key must have a short name (token name) which can be used for naming software artefacts. It can also have any number of long names. These can be in different languages. Where necessary, a long name can have an “identifier” attribute, which is a string that sufficiently identifies a particular long name from a set of long names.

Columns and keys must have short names that are unique within the column set or code list in which they are defined.

3 Version 2.0 Requirements

These are the requirements that have been accepted to date for version 2.0 of the OASIS Code List Representation format. It is a requirement on version 1.0 that it must be designed to that these version 2.0 requirements can be implemented without causing a loss of backwards compatibility with version 1.0 code lists.

3.1 R2.1 A code list can be derived from an existing code list by adding/removing rows/columns/keys

New rows cannot be added if they violate the uniqueness of the keys. Columns cannot be removed if they are used as part of a key, and if a key is removed, at least one other key must remain.

3.2 R2.2 A code list can be derived from existing code lists by aggregating their rows

Aggregation of rows can only occur if the source code lists have the same columns, or if the columns which are not common to all source code lists are allowed nillable columns.

3.3 R2.3 A code list can be derived from existing code lists by aggregating their columns

Aggregation of columns can only occur if the source code lists have at least one key in common (which also implies they have one or more columns in common). Where any of the source code lists have the same column, the values in that column must be the same in each source code list (which means that the values for any common key must be the same across the source code lists).

3.4 R2.4 A code list can be derived from existing code lists by removing all common rows before aggregating the remaining rows

3.5 R2.5 A code list can be derived from existing code lists by removing all common columns before aggregating the remaining columns

It should also be possible to remove common keys as well as common columns. A column cannot be removed unless all keys that it is part of are removed, and there aggregate must contain at least one key.

> Note: support for keys as well is columns is not yet implemented.

3.6 R2.6 A derived code list can be required to contain a source code list as a row-wise subset

> Note: there is an open question about whether it should be possible to specify that only keys are compared, or that only particular keys are compared.

3.7 R2.7 A derived code list can be required to contain a source code list as a column-wise subset

It should be possible to specify the subset via keys as well as via columns.

Note: support for keys as well is columns is not yet implemented.

3.8 R2.8 The basic code list operations can be composed arbitrarily and to any depth to create a derived code list from a set of source code lists

3.9 R2.9 The operations used to derive particular code lists must be unambiguously encodable so that they are repeatable and auditable

4 Possible Future Requirements

4.1 F.1 It should be possible to represent code lists that cannot be enumerated

Some code lists cannot be explicitly enumerated, e.g. because they are too large to practically enumerate, or because they are proprietary (so they can be matched against, but not made available in their entirety).

4.2 F.2 Support for multiple alternate code values for the same code list entry

For example, if a country changes its name, and hence changes its ISO country code, it should be possible to have a code list representation where both the old and new codes for that country are included, and where those old and new codes are represented as two alternative values in the same code list for the same country.

The difficulty with this requirement is how to specify the addition and/or removal of alternate code (or metadata) values in derived code lists, without undue complexity.

4.3 F.3 Start/expiry dates/times for code lists, code list sets, and individual codes

The difficulty with this requirement is how to specify the addition, removal, or modification of start/expiry dates/times in derived code lists, without undue complexity.

4.4 F.4 Support for pattern or range restrictions for code lists that cannot be enumerated

For code lists that cannot be explicitly enumerated, it would sometimes be useful if a pattern or range restriction for the code list can be specified, to allow a degree of early sanity checking of codes before they are fully validated against the code list.

Appendix A. Revision History

Date	Version	Comments
2006-10-31	1.0	Initial published version of requirements.
2006-12-19	1.0.1	Added requirement F.4.
2007-04-30	1.0.1	Updated to use latest OASIS template.
2007-05-02	1.0.1	Updated the document URLs on the first page.