The xmltool command-line utility

Hussein Shafie, XMLmind Software <xmleditor-support@xmlmind.com>

November 22, 2023

Abstract

This document is the reference manual of the **xmltool** command-line utility. The **xmltool** command-line utility can be used to validate and pretty-print (i.e. indent) XML documents and also to automatically generate a reference manual in HTML format for a schema.

This utility, like all the other command-line utilities, is found in *XXE_install_dir*/bin/.

Table of Contents

1. Why use the **xmltool** command-line utility? 1
2. Synopsis 1
3. `validate` options 2
4. `indent` options 4
5. `schematron` options 7
6. `schemadoc` options 8
7. Common options 9
A. Implementation limits 10
    1. Limitations related to XML Schema Datatypes 10
    2. Limitations related to XML Schema Structures 10
    3. Limitations related to DTD support 13

1. Why use the xmltool command-line utility?

The **xmltool** command-line utility can be used to validate and pretty-print (i.e. indent) XML documents and also to automatically generate a reference manual in HTML format for a schema.

This utility, like all the other command-line utilities, is found in *XXE_install_dir*/bin/.

Mac users

If you install **XXE** on the Mac using the recommended `.dmg` distribution, you'll not find *XXE_install_dir*/bin/. The **xmltool** command-line utility, like all the other command-line utilities, is found in `XMLmind.app/Contents/Resources/xxe/bin/`.

However, unlike the XMLEditor app, `XMLmind.app/Contents/Resources/xxe/bin/xmltool` will *not* run unless you first install a Java™ 1.8+ runtime on your Mac.

If you plan to use **xmltool** a lot, may be you'll prefer to install the `.zip` distribution on your Mac rather the recommended `.dmg` distribution. Installing the `.zip` distribution on your Mac requires also installing a Java 1.8+ runtime.

2. Synopsis

Usage:

```
xmltool validate|indent|schematron|schemadoc ?options? arguments
```

The **xmltool** utility comprises 4 different processors:

validate [2]
: Checks the validity of a document conforming to a DTD, W3C XML Schema or RELAX NG schema.

  May also be used to check the validity of a DTD, W3C XML Schema or RELAX NG schema.

indent [4]
: Saves one or more documents after reformatting their XML contents.

  May be also used to *flatten* these documents, that is, to transclude XInclude elements or DITA conrefs.

schematron [7]
: Checks the validity of a document against a Schematron schema.

  May also be used to check the validity of a Schematron schema.

schemadoc [8]
: Generates a reference manual in HTML format for a DTD, W3C XML Schema or RELAX NG schema.

  The generated HTML reference manual, organized like "*DocBook: The Definitive Guide*" by Norman Walsh and al., lists all the elements and attributes specified in the schema.

  This manual is intended to help content authors create instances conforming to a given schema. This manual is not intended to help schema authors document their design.

  Note that, for now, this documentation generator cannot extract documentation contained in a schema (i.e. in annotation/documentation elements) and merge extracted documentation with automatically generated documentation.

3. `validate` options

Usage:

```
xmltool validate validate_options common_options [9] [ xml_file ]*
```

Validates specified XML files. If no XML files are specified, it is the schemas specified using the "`-s`" option which are validated.

`-s` *schema*
: Use specified schema file to validate all specified XML files.

  schema must be a DTD having a "`.dtd`" extension or a W3C XML Schema having a "`.xsd`", "`.xs`", "`.wxs`" extension or a RELAX NG schema having a "`.rng`", "`.rnc`" (compact syntax) extension.

  By default, the schema used for validation is found in each specified XML file (e.g. `<!DOCTYPE>`).

  It is possible to specify several `-s` options:
  - When no XML files are specified, each schema is individually validated.

- When several XML files are specified, the schemas are composed to form a compound schema (e.g. DocBook 5+MathML) and this compound schema is used to validate specified XML files.

`-l`

Give priority to the schemas locally specified in the file (e.g. `<!DOCTYPE>`) to be validated over those specified using the `-s` option. Ignored unless the `-s` option is used too.

`-f`

When validating a document, ignore false duplicate ID errors caused by multiple inclusions of the same element. When saving a document, replace such false duplicate IDs by automatically generated ones.

Example: Validate `docbook-image.xml` against the DTD specified in its `<!DOCTYPE>`.

```
/opt/xxe/demo$ xmltool validate docbook-image.xml
```

xmltool automatically uses all the XML catalogs of XMLmind XML Editor

The **xmltool** command-line utility automatically uses all the XML catalogs found in the two add-on directories of **XXE** (*`XXE_install_dir`*`/addon/` and *`XXE_user_preferences_dir`*`/addon/`).

Example: file `docbook-image.xml` starts with

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<article>
...
```

yet the validation is quick. Why that? Because **xmltool** uses *`XXE_install_dir`*`/addon/config/docbook/catalog.xml` to resolve `http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd` as *`XXE_install_dir`*`/addon/config/docbook/dtd/V4.2/docbookx.dtd`.

Example: Validate `docbook-table.xml` and `docbook-image.xml` against the DocBook 4.5 DTD.

```
/opt/xxe/demo$ xmltool validate -s ../addon/config/docbook/dtd/V4.5/docbookx.dtd \
    docbook-table.xml docbook-image.xml
```

Example: Validate the DocBook 4.5 DTD.

```
/opt/xxe/demo$ xmltool validate -s ../addon/config/docbook/dtd/V4.5/docbookx.dtd
```

Example: Validate `sample.xml` against the combined `docbook.rng` and `mathml2.rng` RELAX NG schemas.

```
/opt/xxe/addon/mathml_config/db5mml$ xmltool validate \
    -s ../../config/docbook5/rng/V5.0/docbook.rng \
    -s rng/mathml2.rng \
    sample.xml
```

Note that the combined schemas don't need to be of the same kind:

```
/opt/xxe/addon/mathml_config/db5mml$ xmltool validate \
    -s ../../config/docbook5/rng/V5.0/docbook.rng \
    -s ../standalone/xsd/mathml2.xsd \
    sample.xml
```

4. `indent` options

Usage:

```
xmltool indent indent_options validate_options [2] common_options [9] [ xml_file ]*
```

Save, possibly indenting and/or flattening[1], specified XML files.

Do not forget to use option `-s` when needed too

Unless the document to be indented contains a specification of which schema to use (e.g. `<!DOCTYPE>`, `xsi:noNamespaceSchemaLocation`, `xsi:schemaLocation` or `<?xml-model?>`), using the `-s` option [2] to explicitly specify this schema is required. See example below [7].

Failing to do so will cause **xmltool** to incorrectly indent the document after reporting the following warning: `WARNING: Cannot determine which schema to use for validating "DOCUMENT_PATH".`

-o *`save_file_or_dir`*

Specifies where to save all specified documents.

If a single document is to be saved, *`save_file_or_dir`* may specify a file or an existing directory.

If multiple documents are to be saved, *`save_file_or_dir`* must specify a directory. Such directory is automatically created if it does not already exist.

Default: reuse the original filename of each specified document after renaming this document using a "`.BAK`" filename.

-flat

Do not preserve inclusions. Instead ``flatten'' the document.

This option automatically generates `xml:base` attribute when needed to. Note that `xml:base` attributes are added even when this attribute is not allowed by the schema of the document being indented.

Default: preserve inclusions.

-indent *`integer`*

Specifies the number of space characters used to indent a child element relatively to its parent element.

- A positive or null value means: indent always.
- Value "`-1`" means: never indent.

[1] That is, transclude references.

- Any other negative value means indent, but only if the document to be saved has an actual document type. In such case, the number of space characters is: (-2 - *integer*).

Default: -4.

-maxlinelength *positive_integer*

Specifies the maximum line length for elements containing text interspersed with child elements.

Default: 78.

-noopenlines

Do not add open lines between the child elements of elements having an ``element-only" content model.

Default: add open lines.

-xhtml

Favor the interoperability with HTML as recommended in the XHTML spec.

In practice, if this option has been specified:
- Empty elements having a non empty content are saved as "`<tag></tag>`".
- Empty elements having an empty content are saved as "`<tag />`" (with a space after the tag).

Note that specifying this option for document types other than XHTML does not really make sense

Default: do *not* favor the interoperability with HTML.

-nocharentities

Do not save characters not supported by the encoding as entity references. Instead, save them as numeric references.

Default: when possible and when needed to, save characters as entity references.

-specialchars *list_of_chars_or_char_ranges*

Always save specified characters as entity references.

Example: `-specialchars "reg 174 0x00ae 0256 pound:yen 163:165 0xA3:0xA5 0243:0245"`.

No default.

-cdatasections *list_of_simple_XPaths*

Save the textual contents of specified elements as CDATA sections.

XHTML example: `-cdatasections "htm:script htm:style"`.

No default.

-prefix prefix *namespace*

Associates a prefix to a namespace.

Multiple "`-prefix`" options are allowed.

This may be needed to allow parsing the XPaths argument of the above "`-cdatasections`" option. Options "`-prefix`" must precede the "`-cdatasections`" option.

XHTML example: `-prefix htm http://www.w3.org/1999/xhtml`.

No default.

-nooriginalprefixes
: Do not use the namespace prefixes originally specified in the document. Instead, generate prefixes.

  Default: Reuse the original prefixes as much as possible.

-nodefaultnamespace
: Do not use the default namespace originally specified in the document.

  Default: Reuse the default namespace if any.

-xmlversion 1.0|1.1|original
: Specifies the "`version`" pseudo-attribute of the XML declaration. "original" means: reuse the XML version originally specified in the document.

  Default: `original`.

-encoding *`java_supported_encoding`*|original
: Specifies which encoding to use when saving a document. "original" means: reuse the encoding originally specified in the "`encoding`" pseudo-attribute of the XML declaration of the document.

  Default: `original`.

-standalone yes|no|original
: Specifies the "`standalone`" pseudo-attribute of the XML declaration. "`original`" means: reuse the "`standalone`" pseudo-attribute originally specified in the document.

  No default: do not add a "`standalone`" pseudo-attribute to the XML declaration.

-noinvalid
: Do not save specified documents if any of them is found to have validity errors (even harmless cross-reference errors).

  Default: save documents even if some of them are found to be invalid.

-script *`URL_or_filename`*
: Run specified `.xed` script in order to modify the document before saving it to disk. Note that it's possible to specify the `-script` option several times in order to use several scripts in turn.

Example: Indent `docbook-table.xml` using the default settings. The original `docbook-table.xml` is saved to `docbook-table.xml.BAK`.

```
/opt/xxe/demo$ xmltool indent docbook-table.xml
/opt/xxe/demo$ ls docbook-table.xml*
docbook-table.xml
docbook-table.xml.BAK
```

Example: Indent `docbook-table.xml` using specified settings. Save indented file to `out.xml`.

```
/opt/xxe/demo$ xmltool indent -indent 1 -noopenlines -nolegacy -o out.xml docbook-table.xml
```

Example: Indent `docbook5-mathml.xml`, a DocBook V5.0 document conforming to RELAX NG schema *`XXE_INSTALL_DIR`*`/addon/config/docbook5/rng/V5.0/docbook.rng`, using specified settings. Save indented file to `out.xml`.

```
/opt/xxe/demo$ xmltool indent \
  -s /opt/xxe/addon/config/docbook5/rng/V5.0/docbook.rng \
  -indent 2 -maxlinelength 78 \
  -o out.xml docbook5-mathml.xml
```

Example: Force the indentation of schema-less file `xhtml_strict.xxe`.

```
/opt/xxe/addon/config/xhtml$ xmltool indent -indent 2 -o indented.xxe xhtml_strict.xxe
WARNING: Cannot determine which schema to use for validating "xhtml_strict.xxe".
```

Example: Indent all `.xhtml` files contained in current directory. Create save files in directory `/tmp/out/`.

```
/opt/xxe/demo$ xmltool indent -v [9] -encoding Windows-1252 -o /tmp/out *.xhtml
```

Example: Transclude all XInclude elements contained `docbook-modular-book.xml` (`-f [3]` is needed otherwise `out.xml` would contain a number of duplicate ID errors).

```
/opt/xxe/demo$ xmltool indent -xi [9] -f [3] -flat -o out.xml docbook-modular-book.xml
```

Example: Modify `doc.xml` using `edit.xed` before saving it, indented, to `/tmp/out.xml`.

```
/opt/xxe/demo$ xmltool indent -script edit.xed -o /tmp/out.xml doc.xml
```

5. `schematron` options

Usage:

```
xmltool schematron schematron_options common_options [9] schematron [ xml_file ]*
```

Validate specified XML documents against specified Schematron.

Optionally validate the Schematron itself.

Unless the `-iso` option is used, the Schematron may be embedded in another type of XML document (e.g. a DocBook 5 RELAX NG grammar not using the compact syntax).

-iso

Fully validate the Schematron as an ISO Schematron schema.

Default: do not validate the Schematron, just load it.

Note that the Schematron loader is very lenient and accepts ISO Schematron as well as Schematron 1.5 schemas.

-o *`out_schematron_file`*

Save the Schematron to specified file. The written schema is in all cases an ISO Schematron schema using the minimal syntax.

-phase *phase_id*

Specifies the ID of the phase which is to be used for validation. May also be `#ALL` or `#DEFAULT`.

Default: `#DEFAULT`, if any, `#ALL` otherwise.

-var *name value*

Specify overrides for some of the let variables defined in the Schematron.

Note that value must be a valid XPath expression and not a plain string.

Example: Validate `docbook.sch` as an ISO Schematron schema. Additionally save a copy in `/tmp/out.sch`.

```
/opt/xxe/addon/config/docbook$ xmltool schematron -iso -o /tmp/out.sch docbook.sch
```

Example: Validate docbook-image.xml against `docbook.sch`.

```
/opt/xxe/demo$ xmltool schematron ../addon/config/docbook/docbook.sch \
    docbook-image.xml
```

Example: Validate `docbook-image.xml` against `docbook.sch`, using phase `#ALL`. Pass to the Schematron variable `foo` having XPath string literal "`bar`" as its value.

```
/opt/xxe/demo$ xmltool schematron -phase '#ALL' -var foo '"bar"' \
    ../addon/config/docbook/docbook.sch \
    docbook-image.xml
```

6. `schemadoc` options

Usage:

```
xmltool schemadoc schemadoc_options common_options [9] schema doc_dir
```

Generate in directory *`doc_dir`* a reference manual in HTML format for schema file *`schema`*.

`Schema` must be a DTD having a "`.dtd`" extension or a W3C XML Schema having a "`.xsd`", "`.xs`" or "`.wxs`" extension or a RELAX NG schema having a "`.rng`" or "`.rnc`" (compact syntax) extension.

`doc_dir` is automatically created if it does not exist.

-css *`css_url`*

Specifies the URL of the CSS style sheet used for the generated HTML.

Default: generated HTML does not have a `<link rel="stylesheet">`.

-charset *`encoding`*

Specifies the charset used for the generated HTML.

Default: use platform default encoding and generated HTML does not have a`<meta http-equiv="Content-Type">`.

-xxe
: Generate annotations which are meaningful when using the schema with XMLmind XML Editor.

Example:

```
/opt/xxe/demo$ xmltool schemadoc bugreport/bugreport.xsd /tmp/bugreport_doc
/opt/xxe/demo$ ls /tmp/bugreport_doc
a__6sv96.html
abbr__79rrv.html
...
index.html
...
workaround__jus1.html
```

7. Common options

-xi
: When loading a document, transclude XInclude elements.

  Default: XInclude elements are treated as any other element.

  This option is equivalent to "`-inclscheme com.xmlmind.xml.xinclude.XIncludeScheme`".

-inclscheme *`class_name`*
: When loading a document, transclude nodes specifying an inclusion directive belonging to specified inclusion scheme.

  Default: nodes specifying an inclusion directive are treated as any other node.

  Specifying several `-inclscheme` options is permitted. Mixing `-xi` and `-inclscheme` options is permitted.

-cache *`schema_cache_dir`*
: Specifies the directory to be used as a schema cache. This directory is automatically created if it does not exist.

  Default: do not cache schemas.

  RELAX NG schemas can be cached only in memory and not on disk.

-rncencoding *`encoding`*
: Specifies the encoding used for RELAX NG compact syntax schemas.

  Default: do not cache schemas.

-v
: Verbose.

Example: Transclude all XInclude elements contained `docbook-modular-book.xml` (`-f [3]` is needed otherwise `out.xml` would contain a number of duplicate ID errors).

```
/opt/xxe/demo$ xmltool indent -xi -f [3] -flat -o out.xml docbook-modular-book.xml
```

Example: Transclude all conref elements contained in `topic1.dita`.

```
/tmp$ xmltool indent -inclscheme "com.xmlmind.xmleditext.dita.ConrefScheme" \
    -f -flat -o out.dita topic1.dita
```

Example: Validate `docbook-table.xml`. Cache the DocBook DTD if it is not already cached. If it is already cached, use the cached copy.

```
/opt/xxe/demo$ xmltool validate -cache /tmp/cache docbook-table.xml
/opt/xxe/demo$ ls /tmp/cache
directory.txt
docbookx.ser
```

Example: Validate `mathml.pane` against `pane.rnc`, a RELAX NG schema using the compact syntax, encoded in ISO-8859-1.

```
/opt/xxe/addon/mathml_config/common/pane$ xmltool validate \
    -rncencoding ISO-8859-1 -s pane.rnc mathml.pane
```

A. Implementation limits

1. Limitations related to XML Schema Datatypes

Formal reference: XML Schema Part 2: Datatypes.

- A 32-bit signed integer is used rather than an arbitrary precision integer to implement the **length**, **minLength**, **maxLength**, **totalDigits** and **fractionDigits** facets.
- Similarly, the components of the **duration** datatype are implemented using 32-bit integers and double-precision floating-point numbers.
- The **length** facet of datatype **QName** is implemented as the number of characters in the local part of the name (that is, the prefix part is not taken into account by facet **length**).

2. Limitations related to XML Schema Structures

Formal reference: XML Schema Part 1: Structures.

Constraints on XML instances which are not checked:

- Entity Name: an attribute value of type **ENTITY** or **ENTITIES** must match the name of an unparsed entity declared in the DTD.

Constraints on XML schemas which are not checked:

- ``The {model group} of the model group definition which corresponds to it per XML Representation of Model Group Definition Schema Components (§3.7.2) must be a ·valid restriction· of the {model group} of that model group definition in I, as defined in Particle Valid (Restriction) (§3.9.6)." [src-redefine.6.2.2]

  In this case, the implementation simply overwrites the previously defined **group**.

- ``The {attribute uses} and {attribute wildcard} of the attribute group definition which corresponds to it per XML Representation of Attribute Group Definition Schema Components (§3.6.2) must be ·valid restrictions· of the {attribute uses} and {attribute wildcard} of that attribute group definition in I." [src-redefine.7.2.2]

  In this case, the implementation simply overwrites the previously defined **attributeGroup**.

- Attribute Group Definition Representation OK. [src-attribute_group.2] [src-attribute_group.3] Attribute Group Definition Properties Correct. [ag-props-correct.2] [ag-props-correct.3]

  attributeGroups are not validated as such. If something is wrong, it is detected when the the **attributeGroup** is actually used.

  Example 1: circular references are checked when the **attributeGroup** is actually used.

  Example 2: duplicate attribute and several ID attributes in the same **attributeGroup** are checked when the **attributeGroup** is actually used.

- Model Group Definition Representation OK. [mgd-props-correct]

  groups are not validated as such. If something is wrong, it is detected when the the **group** is actually used.

- Unique Particle Attribution. [cos-nonambig]

- Derivation Valid (Restriction, Simple). [cos-st-restricts.1.3] [cos-st-restricts.2.3.3] [cos-st-restricts.3.3.3]

  The implementation allows to add facets not defined by the base type.

- ``It must in principle be possible to derive the complex type definition in two steps, the first an extension and the second a restriction (possibly vacuous), from that type definition among its ancestors whose {base type definition} is the ·ur-type definition·." [cos-ct-extends.1.5]

- ``The {content type} of the {base type definition} must be a simple type definition of which the {content type} is a ·valid restriction· as defined in Derivation Valid (Restriction, Simple) (§3.14.6)." [derivation-ok-restriction.5.1.1]

  - See above the limitations related to Derivation Valid (Restriction, Simple) [cos-st-restricts]

  - In this case, the implementation does not check that the new facet value actually restricts the facet value of the base type.

- ``No element member of the ·key-sequence· of any member of the ·qualified node set· was assessed as ·valid· by reference to an element declaration whose {nillable} is true." [cvc-identity-constraint.4.2.3]

Other specificities:

- The algorithm used to check *Particle Valid (Restriction)* [cos-particle-restrict] is more powerful than the one described in the spec.

  Rationale: the schema for schemas is found invalid when the algorithm described in the spec is used.

- `<xs:complexType name="title" mixed="true" />` is mixed, not empty.

- Not being able to load a schema **include**-ed, **import**-ed or **redefine**-ed by another schema is considered to be a fatal error [x-src-include.1] [x-src-import.1] [x-src-redefine.1]. For the spec, it is just a warning.

- A **import** construct must almost always specify a **schemaLocation** [x-src-import].

  However, the validation engine supports `xs:import` elements *without* a `schemaLocation` attribute, if an `xs:import` element for the same namespace but this time having a `schemaLocation` attribute has previously been processed.

  Example:

```
  <xs:import namespace="foo"
             schemaLocation="http://foo.com/schema1.xsd" />

  <!-- Later, typically inside an included module. -->
  <xs:import namespace="foo" />
```

  Note that the other example below will not work because the validation engine cannot guess which of `schema1.xsd` or `schema2.xsd` contains the components to be imported.

```
  <xs:import namespace="foo"
             schemaLocation="http://foo.com/schema1.xsd" />

  <!-- Later, typically inside an included module. -->
  <xs:import namespace="foo"
             schemaLocation="http://foo.com/schema2.xsd" />

  <!-- Later, typically inside another included module. -->
  <xs:import namespace="foo" />
```

- Identity-constraint definition identities must be unique within an XML Schema [x-c-props-correct].

- A regular expression such as "`[a-zA-Z0-9-]`" is not supported as is. It must be rewritten like this: "`[a-zA-Z0-9\-]`".

- For readability, whitespace may be used in `selector` and `field` XPath expressions. But whitespace is only supported around '`|`' and not around all tokens as mandated by the W3C recommendation.

  That is, it is possible to specify this:

```
<xs:key name="truck1" >
  <xs:selector xpath="." />
  <xs:field xpath="truck/@number | truck/@plate" />
</xs:key>
```

  But not this:

```
<xs:key name="truck1" >
  <xs:selector xpath="." />
  <xs:field xpath="truck / @number | truck / @plate" />
</xs:key>
```

- The following *valid* element declaration is not supported.

```
<xs:element name="foo">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="bar" />
      <xs:element name="bar" form="qualified" type="xs:decimal" /> <!--NOT SUPPORTED-->
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="bar" type="xs:decimal"  />
```

An *implementation limit* error `x-cos-element-consistent` is reported in that case.

- The following *valid* element declaration is not supported.

```
<xs:element name="foo">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="bar" type="xs:token" />
      <xs:element name="bar" type="xs:token" nillable="true" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

An *implementation limit* error `x-cos-element-consistent` is reported in that case.

3. Limitations related to DTD support

Formal reference: Extensible Markup Language (XML) 1.1.

Constraints on XML instances which are not checked:

- Validity constraint: Root Element Type. The Name in the document type declaration must match the element type of the root element.
- Entity Name: an attribute value of type **ENTITY** or **ENTITIES** must match the name of an unparsed entity declared in the DTD.

Constraints on DTDs which are not checked:

- Notation Declared: in an unparsed entity, the name after **NDATA** must match the declared name of a notation.
- Standalone Document Declaration.
- Proper Declaration/PE Nesting.
- Proper Group/PE Nesting.
- Proper Conditional Section/PE Nesting.