

XMLSTARLET USER'S GUIDE

see also <http://xmlstar.sourceforge.net/>

1. BASIC COMMAND LINE OPTIONS

```
xml
XMLStarlet Toolkit: Command line utilities for XML
Usage: xml [<options>] <command> [<cmd-options>]
where <command> is one of:
  ed (or edit) - Edit/Update XML document(s)
  sel (or select) - Select data or query XML document(s) (XPATH, etc)
  tr (or transform) - Transform XML document(s) using XSLT
  val (or validate) - Validate XML document(s) (well-formed/DTD/XSD/RelaxNG)
  fo (or format) - Format XML document(s)
  el (or elements) - Display element structure of XML document
  c14n (or canonic) - XML canonicalization
  ls (or list) - List directory as XML
  esc (or escape) - Escape special XML characters
  unesc (or unescape) - Unescape special XML characters
  pyx (or xmln) - Convert XML into PYX format (based on ESIS - ISO 8879)
  p2x (or depyx) - Convert PYX into XML
```

<options> are:

- version - show version
- help - show help

Wherever file name mentioned in command help it is assumed
that URL can be used instead as well.

Type: xml <command> --help <ENTER> for command help

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

2. Select/Query XML documents

```
xml sel --help
XMLStarlet Toolkit: Select from XML document(s)
Usage: xml sel <global-options> {<template>} [<xml-file> ... ]
where
<global-options> - global options for selecting
<xml-file> - input XML document file name/uri (stdin is used if missing)
<template> - template for querying XML document with following syntax:
```

<global-options> are:

- C or --comp - display generated XSLT
- R or --root - print root element <xsl-select>
- T or --text - output is text (default is XML)
- I or --indent - indent output
- D or --xml-decl - do not omit xml declaration line
- B or --noblanks - remove insignificant spaces from XML tree
- N <name>=<value> - predefine namespaces (name without 'xmlns:')
- ex: xsql:urn:oracle-xsql
- Multiple -N options are allowed.
- net - allow fetch DTDs or entities over network
- help - display help

Syntax for templates: -t|--template <options>

where <options>

- c or --copy-of <xpath> - print copy of XPATH expression
- v or --value-of <xpath> - print value of XPATH expression
- o or --output <string> - output string literal
- n or --nl - print new line
- f or --inp-name - print input file name (or URL)
- m or --match <xpath> - match XPATH expression
- i or --if <test-xpath> - check condition <xsl:if test="test-xpath">
- e or --elem <name> - print out element <xsl:element name="name">
- a or --attr <name> - add attribute <xsl:attribute name="name">

-b or --break - break nesting
 -s or --sort op xpath - sort in order (used after -m) where
 op is X:Y:Z,
 X is A - for order="ascending"
 X is D - for order="descending"
 Y is N - for data-type="numeric"
 Y is T - for data-type="text"
 Z is U - for case-order="upper-first"
 Z is L - for case-order="lower-first"

There can be multiple --match, --copy-of, --value-of, etc options in a single template. The effect of applying command line templates can be illustrated with the following XSLT analogue

```
xml sel -t -c "xpath0" -m "xpath1" -m "xpath2" -v "xpath3" \
-t -m "xpath4" -c "xpath5"
```

is equivalent to applying the following XSLT

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:call-template name="t1"/>
  <xsl:call-template name="t2"/>
</xsl:template>
<xsl:template name="t1">
  <xsl:copy-of select="xpath0"/>
  <xsl:for-each select="xpath1">
    <xsl:for-each select="xpath2">
      <xsl:value-of select="xpath3"/>
    </xsl:for-each>
  </xsl:for-each>
</xsl:template>
<xsl:template name="t2">
  <xsl:for-each select="xpath4">
    <xsl:copy-of select="xpath5"/>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

XMLStarlet is a command line toolkit to query/edit/check/transform XML documents (for more information see <http://xmlstar.sourceforge.net/>)

Current implementation uses libxslt from GNOME codebase as XSLT processor (see <http://xmlsoft.org/> for more details)

3. Editing XML documents

```
xml ed --help
XMLStarlet Toolkit: Edit XML document(s)
Usage: xml ed <global-options> {<action>} [ <xml-file-or-uri> ... ]
where
<global-options> - global options for editing
<xml-file-or-uri> - input XML document file name/uri (stdin is used if missing)
```

<global-options> are:
 -P (or --pf) - preserve original formatting
 -S (or --ps) - preserve non-significant spaces
 -O (or --omit-decl) - omit XML declaration (<?xml ...?>)
 -N <name>=<value> - predefine namespaces (name without 'xmlns:')

ex: xsqI=urn:oracle-xsqI

Multiple -N options are allowed.

-N options must be last global options.

--help or -h - display help

where <action>
 -d or --delete < xpath >
 -i or --insert <xpath> -t (--type) elem|text|attr -n <name> -v (--value) <value>
 -a or --append <xpath> -t (--type) elem|text|attr -n <name> -v (-value) <value>
 -s or --subnode <xpath> -t (--type) elem|text|attr -n <name> -v (-value) <value>
 -m or --move <xpath1> <xpath2>
 -r or --rename <xpath1> -v <new-name>
 -u or --update <xpath> -v (-value) <value>
 -x (-expr) <xpath> (-x is not implemented yet)

XMLStarlet is a command line toolkit to query/edit/check/transform XML documents (for more information see <http://xmlstar.sourceforge.net/>)

4. Using XSLT to transform XML documents

```
xml tr --help
XMLStarlet Toolkit: Transform XML document(s) using XSLT
Usage: xml tr [<options>] <xsl-file> {-p|-s <name>=<value>} [<xml-file-or-uri> ... ]
where
< xsl-file >           - main XSLT stylesheet for transformation
<xml-file>              - input XML document file name (stdin is used if missing)
<name>=<value>          - name and value of the parameter passed to XSLT processor
-p                      - parameter is XPATH expression ("string" to quote string)
-s                      - parameter is a string literal
<options> are:
--omit-decl            - omit xml declaration <?xml version="1.0"?>
--show-ext              - show list of extensions
--val                   - allow validate against DTDs or schemas
--net                   - allow fetch DTDs or entities over network
--xinclude              - do XInclude processing on document input
--maxdepth val          - increase the maximum depth
--html                  - input document(s) is(are) in HTML format
--docbook               - input document(s) is(are) in SGML docbook format
--catalogs              - use SGML catalogs from $SGML_CATALOG_FILES
otherwise XML catalogs starting from
file:///etc/xml/catalog are activated by default
```

XMLStarlet is a command line toolkit to query/edit/check/transform XML documents (for more information see <http://xmlstar.sourceforge.net/>)

Current implementation uses libxslt from GNOME codebase as XSLT processor
(see <http://xmlsoft.org/> for more details)

5. Formatting XML documents

```
xml fo --help
XMLStarlet Toolkit: Format XML document(s)
Usage: xml fo [< options >] < xml-file >
where <options> are
-n or --noindent         - do not indent
-t or --indent-tab       - indent output with tabulation
-s or --indent-spaces <num> - indent output with <num> spaces
-o or --omit-decl        - omit xml declaration <?xml version="1.0"?>
-R or --recover          - try to recover what is parsable
-e or --encode <encoding> - output in the given encoding (utf-8, unicode...)
-H or --html              - input is HTML
-h or --help              - print help
```

XMLStarlet is a command line toolkit to query/edit/check/transform XML documents (for more information see <http://xmlstar.sourceforge.net/>)

6. Validating XML documents

```
xml val --help
XMLStarlet Toolkit: Validate XML document(s)
Usage: xml val < options > [< xml-file-or-uri > ... ]
where <options>
-w or --well-formed      - validate well-formedness only (default)
-d or --dtd <dtd-file>   - validate against DTD
-s or --xsd <xsd-file>   - validate against XSD schema
-r or --relaxng <rng-file> - validate against Relax-NG schema
-e or --err               - print verbose error messages on stderr
-b or --list-bad          - list only files which do not validate
-g or --list-good         - list only files which validate
-q or --quiet             - do not list files (return result code only)
```

NOTE: XML Schemas are not fully supported yet due to its incomplete

support in libxml (see <http://xmlsoft.org>)

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

7. Displaying structure of XML documents

```
xml el --help
XMLStarlet Toolkit: Display element structure of XML document
Usage: xml el [<options>] <xml-file>
where
<xml-file> - input XML document file name (stdin is used if missing)
<options>:
-a - show attributes as well
-v - show attributes and their values
-u - print out sorted unique lines
```

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

8. Escape/Unescape special XML characters

```
xml esc --help
XMLStarlet Toolkit: Escape special XML characters
Usage: xml esc [<options>] [<string>]
where <options> are
--help - print usage
(TODO: more to be added in future)
if <string> is missing stdin is used instead.
```

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

9. List directory as XML

```
xml ls --help
XMLStarlet Toolkit: List directory as XML
Usage: xml ls
Lists current directory in XML format.
```

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

10. XML canonicalization

```
xml c14n --help
XMLStarlet Toolkit: XML canonization
Usage: xml c14n <mode> <xml-file> [<>xpath-expr>] [<inclusive-ns-list>]
where
<xml-file> - input XML document file name (stdin is used if '-')
<mode> is one of following:
--with-comments    XML file canonization w comments
--without-comments  XML file canonization w/o comments
--exc-with-comments Exclusive XML file canonization w comments
--exc-without-comments Exclusive XML file canonization w/o comments
```

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

11. Convert XML into PYX format (based on ESIS - ISO 8879)

```
xml pyx --help
XMLStarlet Toolkit: Convert XML into PYX format (based on ESIS - ISO 8879)
Usage: xml pyx {<xml-file>}
where
<xml-file> - input XML document file name (stdin is used if missing)
```

The PYX format is a line-oriented representation of XML documents that is derived from the SGML ESIS format.
(see ESIS - ISO 8879 Element Structure Information Set spec,
ISO/IEC JTC1/SC18/WG8 N931 (ESIS))

A non-validating, ESIS generating tool originally developed for
pyxie project (see <http://pyxie.sourceforge.net/>)
ESIS Generation by Sean Mc Grath <http://www.digitome.com/sean.html>

XMLStarlet is a command line toolkit to query/edit/check/transform
XML documents (for more information see <http://xmlstar.sourceforge.net/>)

12. Examples:

Input1

<examples/xml/table.xml>

```
<?xml version="1.0"?>
<xml>
<table>
<rec id="1">
<numField>123</numField>
<stringField>String Value</stringField>
</rec>
<rec id="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</rec>
<rec id="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>
```

Input2

<examples/xml/tab-obj.xml>

```
<?xml version="1.0"?>
<xml>
<table>
<rec id="1">
<numField>123</numField>
<stringField>String Value</stringField>
<object name="Obj1">
<property name="size">10</property>
<property name="type">Data</property>
</object>
</rec>
<rec id="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</rec>
<rec id="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>
```

Input3

<examples/html/hello1.html>

```
<html>
<head>
<title>Hello World</title>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
</head>
<body>
<div align="center">Hello World!<br></div>
</body>
</html>
```

Input4
examples/sgml/docbook1.sgml

```
<!DOCTYPE book
PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
"http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd">
<book>

<bookinfo>
<title>DocBook document example</title>
<author>
<firstname>Mikhail</firstname>
<surname>Grushinskiy</surname>
</author>

<copyright>
<year>2002</year>
<holder>Mikhail Grushinskiy</holder>
</copyright>
</bookinfo>

<preface>
<title>Sample document</title>
<para>A simple DocBook example document.</para>
</preface>

<chapter>
<title>XMLStarlet Example</title>
<para>The <emphasis>XMLStarlet</emphasis> command line toolkit
allows querying/checking/editing/transforming/formatting XML documents
from command line</para>
<para>To find out more on how to use the
<emphasis>XMLStarlet</emphasis> for XML processing, point
your browser to <ulink
url="http://xmlstar.sourceforge.net/">http://xmlstar.sourceforge.net/</ulink>.
</ para >
</chapter>
</book>
```

Stylesheet1
examples/xsl/sum1.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:param name="inputFile">-</xsl:param>
<xsl:template match="/">
<xsl:call-template name="t1"/>
</ xsl:template >
<xsl:template name="t1">
<xsl:value-of select="sum(/xml/table/rec/numField)"/>
<xsl:value-of select="
" />
</xsl:template>
</xsl:stylesheet>
```

Stylesheet2

examples/xsl/hello1.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:param name="inputFile">-</xsl:param>
<xsl:template match="/">
  <xsl:call-template name="t1"/>
</xsl:template>
<xsl:template name="t1">
  <xsl:for-each select="/">
    <xsl:value-of select="/html/body/div"/>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Stylesheet3

examples/xsl/param1.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:param name="Text"/>
<xsl:param name="Count"/>
<xsl:template match="/">
  <xsl:call-template name="t1"/>
</xsl:template>
<xsl:template name="t1">
  <xsl:for-each select="/xml">
    <xsl:value-of select="$Text"/>
    <xsl:value-of select="$Count"/>
    <xsl:value-of select="
"/>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Command:

XML canonicalization
xml c14n --with-comments ..//examples/xml/structure.xml ; echo \$?

Result Output:

```
<a1>
  <a11>
    < a111 >
      <a1111></a1111>
    </a111>
  <a112>
    <a1121></a1121>
  </a112>
  </a11>
  <a12></a12>
  <a13>
    <a131></a131>
  </a13>
</a1>
0
```

Command:

Count elements matching XPath expression
xml sel -t -v "count(/xml/table/rec/numField)" xml/table.xml

Result Output:

3

Command:

Count all nodes in XML document
xml sel -t -f -o " " -v "count(//node())" xml/table.xml xml/tab-obj.xml

Result Output:

xml/table.xml 32
xml/tab-obj.xml 41

Command:

Delete elements matching XPath expression

```
xml ed -d /xml/table/rec[@id='2'] xml/table.xml
Result Output:
<?xml version="1.0"?>
<xml>
<table>
<rec id="1">
<numField>123</numField>
<stringField>String Value</stringField>
</rec>
<rec id="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>
```

Command:

```
# Generate HTML from given SGML docbook document
xml tr --omit-decl --docbook /usr/share/sgml/docbook/yelp/docbook/html/docbook.xsl sgml/docbook1.sgml | \
  xml fo --html --indent-spaces 2
Result Output:
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
<meta content="text/html; charset=ISO-8859-1" http-equiv="Content-Type"/>
<title>DocBook document example</title>
<meta name="generator" content="DocBook XSL Stylesheets V1.48"/>
</head>
<body bgcolor="white" text="black" link="#0000FF" vlink="#840084" alink="#0000FF">
<div class="book">
<div class="titlepage">
<div>
<h1 class="title"><a name="id2765244">DocBook document example</a></h1>
</div>
<div>
<h3 class="author">Mikhail Grushinskiy</h3>
</div>
<div>
<p class="copyright">Copyright □ 2002 Mikhail Grushinskiy</p>
</div>
<hr/>
</div>
<div class="toc">
<p>
<b>Table of Contents</b>
</p>
</div>
<dl>
<dt>
<a href="#id2765482">Sample document</a>
</dt>
<dt>1. <a href="#id2767329">XMLStarlet Example</a></dt>
</dl>
</div>
<div class="preface">
<div class="titlepage">
<div>
<h2 class="title"><a name="id2765482">Sample document</a></h2>
</div>
</div>
<p>A simple DocBook example document.</p>
</div>
<div class="chapter">
<div class="titlepage">
<div>
<h2 class="title"><a name="id2767329">Chapter 1. XMLStarlet Example</a></h2>
</div>
</div>
<p>The <span class="emphasis"><i>XMLStarlet</i></span> command line toolkit
allows querying/checking/editing/transforming/formatting XML documents
from command line</p>
<p>To find out more on how to use the
<span class="emphasis"><i>XMLStarlet</i></span> for XML processing, point
your browser to <a href="http://xmlstar.sourceforge.net/" target="_top">http://xmlstar.sourceforge.net/</a>.
</p>
</div>
```

```
</div>
</body>
</html>
```

Command:

```
# Validate XML document against DTD
xml val --dtd dtd/table.dtd xml/tab-obj.xml >/dev/null 2>&1; echo $?
Result Output:
1
```

Command:

```
# Validate XML document against DTD
xml val --dtd dtd/table.dtd xml/table.xml >/dev/null 2>&1; echo $?
Result Output:
0
```

Command:

```
# Display element structure of XML document
xml el ./xml/tab-obj.xml
Result Output:
xml
xml/table
xml/table/rec
xml/table/rec/numField
xml/table/rec/stringField
xml/table/rec/object
xml/table/rec/object/property
xml/table/rec/object/property
xml/table/rec
xml/table/rec/numField
xml/table/rec/stringField
xml/table/rec
xml/table/rec/numField
xml/table/rec/stringField
```

Command:

```
# Display element structure of XML document (including attributes)
xml el -a ./xml/tab-obj.xml
Result Output:
xml
xml/table
xml/table/rec
xml/table/rec/@id
xml/table/rec/numField
xml/table/rec/stringField
xml/table/rec/object
xml/table/rec/object/@name
xml/table/rec/object/property
xml/table/rec/object/property/@name
xml/table/rec/object/property
xml/table/rec/object/property/@name
xml/table/rec
xml/table/rec/@id
xml/table/rec/numField
xml/table/rec/stringField
xml/table/rec
xml/table/rec/@id
xml/table/rec/numField
xml/table/rec/stringField
```

Command:

```
# Display element structure of XML document (including attribute values)
xml el -v ./xml/tab-obj.xml
Result Output:
xml
xml/table
xml/table/rec[@id='1']
xml/table/rec/numField
xml/table/rec/stringField
xml/table/rec/object[@name='Obj1']
xml/table/rec/object/property[@name='size']
xml/table/rec/object/property[@name='type']
```

```
xml/table/rec[@id='2']
xml/table/rec/numField
xml/table/rec/stringField
xml/table/rec[@id='3']
xml/table/rec/numField
xml/table/rec/stringField
```

Command:

```
# Escape special XML characters
cat xml/structure.xml | xml esc
```

Result Output:

```
&lt;a1&gt;
&lt;a11&gt;
&lt;a111&gt;
  &lt;a1111/&gt;
&lt;/a111&gt;
&lt;a112&gt;
  &lt;a1121/&gt;
&lt;/a112&gt;
&lt;/a11&gt;
&lt;a12/&gt;
&lt;a13&gt;
  &lt;a131/&gt;
&lt;/a13&gt;
&lt;/a1&gt;
```

Command:

```
# Calculate EXSLT (XSLT extention) XPath value
echo "<x/>" | xml sel -t -v "math:abs(-1000)"
```

Result Output:

1000

Command:

```
# Find XML files matching XPath expression (containing 'object' element)
xml sel -t -m //object -f xml/table.xml xml/tab-obj.xml
```

Result Output:

xml/tab-obj.xml

Command:

```
# Generate XML document using command line xml sel
echo "<x/>" | xml sel -t -m / -e xml -e child -a data -o value
```

Result Output:

<xml><child data="value"/></xml>

Command:

```
# Apply XSLT stylesheet to HTML input file
xml tr -html xsl/hello1.xsl html/hello1.html
```

Result Output:

Hello World!

Command:

```
# Use local-name() XSLT function in XPath expression
xml sel -t -v "//*[local-name()='query']" xsql/jobserv.xsql
```

Result Output:

```
SELECT substr(title,1,26) short_title, title, location, skills
FROM job
WHERE UPPER(title) LIKE '%ORACLE%'
ORDER BY first_posted DESC
```

Command:

```
# Select text value of an XML element mathing given XPath expression
xml sel -t -m "/xml/table/rec[@id='2']" -v numField xml/table.xml
```

Result Output:

346

Command:

```
# Format XML document disabling indent
cat xml/tab-obj.xml | xml fo --noindent
```

Result Output:

```
<?xml version="1.0"?>
<xml>
<table >
<rec id="1">
<numField>123</numField>
<stringField>String Value</stringField>
<object name="Obj1">
<property name="size">10</property>
<property name="type">Data</property>
</object>
</rec>
<rec id="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</rec>
<rec id="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>
```

Command:

```
# Predefine namespaces for XPath expressions
xml sel -N xsql=urn:oracle-xsql -t -v /xsql:query xsql/jobserve.xsql
```

Result Output:

```
SELECT substr(title,1,26) short_title, title, location, skills
FROM job
WHERE UPPER(title) LIKE '%ORACLE%'
ORDER BY first_posted DESC
```

Command:

```
# Recover malformed XML document
xml fo -R xml/malformed.xml 2>/dev/null
```

Result Output:

```
<?xml version="1.0"?>
<test_output>
<test_name>foo</test_name>
<subtest>...</subtest>
</test_output>
```

Command:

```
# Rename attributes
xml ed -r "/*[@id]" -v ID xml/tab-obj.xml
```

Result Output:

```
<?xml version="1.0"?>
<xml>
<table>
<rec ID="1">
<numField>123</numField>
<stringField>String Value</stringField>
<object name="Obj1">
<property name="size">10</property>
<property name="type">Data</property>
</object>
</rec>
<rec ID="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</rec>
<rec ID="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>
```

Command:

```
# Rename elements
xml ed -r "/xml/table/rec" -v record xml/tab-obj.xml
```

```

Result Output:
<?xml version="1.0"?>
<xml>
< table >
<record id="1">
<numField>123</numField>
<stringField>String Value</stringField>
<object name="Obj1">
<property name="size">10</property>
<property name="type">Data</property>
</object>
</record>
<record id="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</record>
<record id="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</record>
</table>
</xml>
```

Command:

```
# Validate against XSD schema
xml val -b -s xsd/table.xsd xml/table.xml xml/tab-obj.xml 2>/dev/null; echo $?
```

Result Output:

```
xml/tab-obj.xml
1
```

Command:

```
# xsl:copy-of in xml sel command
xml sel -B -t -m /xml/table/rec -c . -n xml/table.xml
```

Result Output:

```
<rec id="1"><numField>123</numField><stringField>String Value</stringField></rec>
<rec id="2"><numField>346</numField><stringField>Text Value</stringField></rec>
<rec id="3"><numField>-23</numField><stringField>stringValue</stringField></rec>
```

Command:

```
# Query XML document and produce sorted text table
xml sel -T -t -m /xml/table/rec -s D:N:- "@id" -v "concat(@id,'|',numField,'|',stringField)" -n xml/table.xml
```

Result Output:

```
3|-23|stringValue
2|346|Text Value
1|123|String Value
```

Command:

```
# Print structure of XML element using xml sel (advanced XPath expressions and xml sel command usage)
xml sel -T -t -m '//*[@'
-m 'ancestor-or-self::*' -v 'name()' -i 'not(position()=last())' -o . -b -b -n \
```

xml/structure.xml

Result Output:

```
a1
a1.a11
a1.a11.a111
a1.a11.a111.a1111
a1.a11.a112
a1.a11.a112.a1121
a1.a12
a1.a13
a1.a13.a131
```

Command:

```
# Calculating running sum on XML document
xml sel -t -v "sum(/xml/table/rec/numField)" xml/table.xml
```

Result Output:

```
446
```

Command:

```
# Indent XML document with tabs
```

```
cat xml/tab-obj.xml | xml fo --indent-tab
Result Output:
<?xml version="1.0"?>
<xml>
<table>
  <rec id="1">
    <numField>123</numField>
    <stringField>String Value</stringField>
    <object name="Obj1">
      <property name="size">10</property>
      <property name="type">Data</property>
    </object>
  </rec>
  <rec id="2">
    <numField>346</numField>
    <stringField>Text Value</stringField>
  </rec>
  <rec id="3">
    <numField>-23</numField>
    <stringField>stringValue</stringField>
  </rec>
</table>
</xml>
```

Command:

```
# Generate plain text table from XML document
xml sel -T -t -m /xml/table/rec -v "@id" -o " | " -v numField -o " | " -v stringField -n xml/table.xml
```

Result Output:

```
1|123|String Value
2|346|Text Value
3|-23|stringValue
```

Command:

```
# Generate plain text table from XML document
xml sel -T -t -m /xml/table/rec -v "concat(@id,'|',numField,'|',stringField)" -n xml/table.xml
```

Result Output:

```
1|123|String Value
2|346|Text Value
3|-23|stringValue
```

Command:

```
# Generate plain text table from XML document
```

```
xml sel -T \
  -t -o "======" -n \
  -m xml/table/rec -v "concat(@id,'|',numField,'|',stringField)" -n \
  -t -o "======" -n xml/table.xml
```

Result Output:

```
=====
1|123|String Value
2|346|Text Value
3|-23|stringValue
=====
```

Command:

```
# Select from XML document containing unicode characters
xml sel -T -t -m "//test[@lang='français']/@lang" -v . -n xml/unicode.xml
```

Result Output:

```
français
français
français
```

Command:

```
# Update value of an attribute
xml ed -u '/xml/table/rec[@id=3]/@id' -v 5 xml/tab-obj.xml
```

Result Output:

```
<?xml version="1.0"?>
<xml>
<table>
  <rec id="1">
    <numField>123</numField>
    <stringField>String Value</stringField>
    <object name="Obj1">
```

```

<property name="size">10</property>
<property name="type">Data</property>
</object>
</rec>
<rec id="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</rec>
<rec id="5">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>

```

Command:

```
# Update value of an element
xml ed -u '/xml/table/rec[@id=1]/numField' -v 0 xml/tab-obj.xml
```

Result Output:

```

<?xml version="1.0"?>
<xml>
<table>
<rec id="1">
<numField>0</numField>
<stringField>String Value</stringField>
<object name="Obj1">
<property name="size">10</property>
<property name="type">Data</property>
</object>
</rec>
<rec id="2">
<numField>346</numField>
<stringField>Text Value</stringField>
</rec>
<rec id="3">
<numField>-23</numField>
<stringField>stringValue</stringField>
</rec>
</table>
</xml>

```

Command:

```
# Validate XML documents using well-formedness/DTD/XSD/RelaxNG checks
echo "=====
echo "Well-Formedness Validation Tests"
echo "- 1 -----"
xml val xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 2 -----"
xml val -g xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 3 -----"
xml val -b xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 4 -----"
xml val -q xml/table.xml xml/tab-obj.xml 2>/dev/null; echo $?

echo "=====
echo "DTD Validation Tests"
echo "- 1 -----"
xml val -d dtd/table.dtd xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 2 -----"
xml val -g -d dtd/table.dtd xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 3 -----"
xml val -b -d dtd/table.dtd xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 4 -----"
xml val -q -d dtd/table.dtd xml/table.xml 2>/dev/null; echo $?

echo "=====
echo "Schema Validation Tests"
echo "- 1 -----"
xml val -s xsd/table.xsd xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 2 -----"
xml val -g -s xsd/table.xsd xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 3 -----"
xml val -b -s xsd/table.xsd xml/table.xml xml/tab-obj.xml xml/tab-bad.xml 2>/dev/null; echo $?
echo "- 4 -----"
xml val -q -s xsd/table.xsd xml/table.xml 2>/dev/null; echo $?
```

```
echo "=====
echo "RelaxNG Schema Validation Tests"
echo "- 1 -----"
xml val -r relaxng/address.rng relaxng/address.xml relaxng/address-bad.xml 2>/dev/null; echo $?
echo "- 2 -----"
xml val -g -r relaxng/address.rng relaxng/address.xml relaxng/address-bad.xml 2>/dev/null; echo $?
echo "- 3 -----"
xml val -b -r relaxng/address.rng relaxng/address.xml relaxng/address-bad.xml 2>/dev/null; echo $?
echo "- 4 -----"
xml val -q -r relaxng/address.rng relaxng/address.xml 2>/dev/null; echo $?
Result Output:
=====
Well-Formedness Validation Tests
- 1 -----
xml/table.xml - valid
xml/tab-obj.xml - valid
xml/tab-bad.xml - invalid
1
- 2 -----
xml/table.xml
xml/tab-obj.xml
1
- 3 -----
xml/tab-bad.xml
1
- 4 -----
0
=====
DTD Validation Tests
- 1 -----
xml/table.xml - valid
xml/tab-obj.xml - invalid
xml/tab-bad.xml - invalid
1
- 2 -----
xml/table.xml
1
- 3 -----
xml/tab-obj.xml
xml/tab-bad.xml
1
- 4 -----
0
=====
Schema Validation Tests
- 1 -----
xml/table.xml - valid
xml/tab-obj.xml - invalid
xml/tab-bad.xml - invalid
1
- 2 -----
xml/table.xml
1
- 3 -----
xml/tab-obj.xml
xml/tab-bad.xml
1
- 4 -----
0
=====
RelaxNG Schema Validation Tests
- 1 -----
relaxng/address.xml - valid
relaxng/address-bad.xml - invalid
1
- 2 -----
relaxng/address.xml
1
- 3 -----
relaxng/address-bad.xml
1
- 4 -----
0
```

Command:

```
# Include one XML document into another using XInclude  
xml tr --xinclude xsl/cat.xsl xml/document.xml  
Result Output:  
<?xml version="1.0" encoding="utf-8"?>  
<document xmlns:xi="http://www.w3.org/2001/XInclude">  
  <p>120 Mz is adequate for an average home user.</p>  
  <disclaimer xml:base="xml/disclaimer.xml">  
    <p>The opinions represented herein represent those of the individual  
    and should not be interpreted as official policy endorsed by this  
    organization.</p>  
  </disclaimer>  
</document>
```

Command:

```
# Passing parameters to XSLT stylesheet  
xml tr xsl/param1.xsl -p Count='count(/xml/table/rec)' -s Text="Count=" xml/table.xml  
Result Output:  
Count=3
```

Command:

```
# Applying XSLT stylesheet to XML document  
xml tr xsl/sum1.xsl xml/table.xml  
Result Output:  
446
```