# xmllint XML check and query

**START HERE**

`xmllint <file>`
Parse and print it

`xmllint --noout <file>`
Check well-formedness

`xmllint --format <file>`
Pretty-print

`xmllint --xpath "<expr>" <file>`
Query nodes

`xmllint --schema <xsd> --noout <file>`
Validate against XSD

`xmllint --c14n <file>`
Canonical form

`xmllint --html <file>`
Parse HTML

`xmllint --shell <file>`
Browse interactively

**Parse** well-formed? `--xinclude` → **Process** XInclude, entities `--valid / --schema` → **Validate** DTD, XSD, RNG `--xpath / --format` → **Output** tree, nodes, c14n

## CONCEPTS

With no flags xmllint parses the file and reprints it; add --noout when you only want the exit code.

Exit code tells the story: 0 ok, 3 invalid, 4 not well-formed, 10 XPath error, 11 XPath empty.

The docs call --format and --recover unreliable; do not build pipelines that depend on their exact output.

## CHECK AND FORMAT

`xmllint <file>`
Parse and print the document

`xmllint --noout <file>`
Check well-formedness only

`xmllint --format <file>`
Reindent output

`xmllint --format <file> --output <out>`
Write result to a file

`xmllint --format - < <file>`
Read from stdin

`curl -s <url> | xmllint --format -`
Pretty-print a download

`XMLLINT_INDENT="    " xmllint --format <file>`
Choose the indent string
*Default is two spaces.*

`xmllint --pretty 2 <file>`
Format keeping significant whitespace
*0 = none, 1 = same as --format, 2 = XML_SAVE_WSNONSIG.*

`xmllint --encode UTF-8 <file>`
Output in another encoding

`xmllint --nocdata <file>`
Turn CDATA into plain text

`xmllint --quiet <file>`
Hide informational messages

`xmllint --version`
Show libxml2 version

## XPATH QUERIES

`xmllint --xpath "//book" <file>`
Select all matching nodes

`xmllint --xpath "//book/title/text()" <file>`
Get text content

## XPATH QUERIES (cont.)

`xmllint --xpath "string(//book[1]/title)" <file>`
Get one value as a string

`xmllint --xpath "count(//book)" <file>`
Count nodes

`xmllint --xpath "//book/@id" <file>`
Select attributes

`xmllint --xpath "//book[@id='b2']" <file>`
Filter by attribute

`xmllint --xpath "//book[price>10]/title" <file>`
Filter by child value

`xmllint --xpath "//*[local-name()='item']" <file>`
Match ignoring namespaces
*Needed for any document with a default xmlns; plain //item finds nothing.*

`xmllint --xpath "boolean(//book)" <file>`
Test existence

`xmllint --xpath "//book[last()]" <file>`
Pick the last node

`xmllint --xpath "//book[position()<=3]" <file>`
Pick the first three

A node-set result prints as XML, a string or number as bare text. Wrap with string() for a clean value.

An empty result prints 'XPath set is empty' and exits 11, so scripts can test with if.

### XPath quick reference

| Syntax | Meaning |
|---|---|
| `/a/b` | child b of root a |
| `//b` | b anywhere |
| `.  ..` | this node, parent |
| `@id` | attribute id |
| `b[2]` | second b |
| `b[@id='x']` | b with id x |
| `text()` | text children |
| `*` | any element |
| `a/following-sibling::b` | later sibling b |
| `contains(@c,'x')` | substring test |
| `starts-with(n,'x')` | prefix test |
| `normalize-space(n)` | trim whitespace |

## VALIDATION

`xmllint --noout --schema <xsd> <file>`
Validate against an XSD

`xmllint --noout --relaxng <rng> <file>`
Validate against RelaxNG

`xmllint --noout --dtdvalid <dtd> <file>`
Validate against a DTD file

`xmllint --noout --valid <file>`
Validate with the document's own DOCTYPE

`xmllint --noout --schematron <sch> <file>`
Validate against Schematron

`xmllint --noout --schema <xsd> <file1> <file2>`
Check several files at once

`xmllint --noout --stream --valid <file>`
Validate big files with low memory

`xmllint --noout --strict-namespace <file>`
Treat namespace errors as errors

`xmllint --noout --schema <xsd> <file> && echo ok`
Use the exit code in scripts

`xmllint --noout --path "<dir1> <dir2>" <file>`
Add search paths for DTDs and entities

Validation success prints '<file> validates' to stderr; failures return exit 3 (or 5 if the schema itself is broken).

## PARSING OPTIONS

`xmllint --recover <file>`
Salvage the parsable parts

`xmllint --recover --noout <file>`
Show errors but keep going

`xmllint --nonet <file>`
Never fetch over the network
*No effect from libxml2 2.15 on.*

`xmllint --xinclude <file>`
Resolve XInclude elements

`xmllint --noent <file>`
Substitute entity references
*Also loads external entities, which is a security risk on untrusted input.*

## PARSING OPTIONS (cont.)

`xmllint --huge <file>`
Lift built-in parser limits

## HTML, CANONICAL, SHELL

`xmllint --html <file>`
Parse as HTML

`xmllint --html --format <file>`
Tidy up HTML

`xmllint --html --xmlout <file>`
Convert HTML to XHTML-style XML

`xmllint --html --xpath "//a/@href" <file>`
Scrape links from HTML

`xmllint --c14n <file>`
Canonical XML 1.0

`diff <(xmllint --c14n <a>) <(xmllint --c14n <b>)`
Compare ignoring formatting noise

`xmllint --shell <file>`
Open the navigation shell

`ls [<path>]`
List children

`cd <path>`
Move to a node (empty goes to root)

`cat [<path>]`
Print a node

`quit / bye / exit`
Leave the shell

## RECIPES

**Pretty-print an API response**

```
curl -s <url> | xmllint --format - | less
```

**Validate every XML file in a tree**

```
find . -name '*.xml' -print0 \
  | xargs -0 -n1 xmllint --noout --schema schema.xsd
```

**Extract values for a shell loop**

```
for id in $(xmllint --xpath "//book/@id" f.xml \
  | grep -o '"[^"]*"' | tr -d '"'); do
  echo "$id"
done
```

**Read a namespaced value, fail if absent**

```
v=$(xmllint --xpath "string(//*[local-name()='version'])" pom.xml) \
  || exit 1
echo "$v"
```