# xmllint XML check and query

## CONCEPTS

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

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

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

## 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

## XPATH QUERIES

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

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

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

## XPATH QUERIES (cont.)

`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 "//*[local-name()='item']" <file>`
Match ignoring namespaces

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.

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

## PARSING OPTIONS

`xmllint --nonet <file>`
Never fetch over the network

## HTML, CANONICAL, SHELL

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