

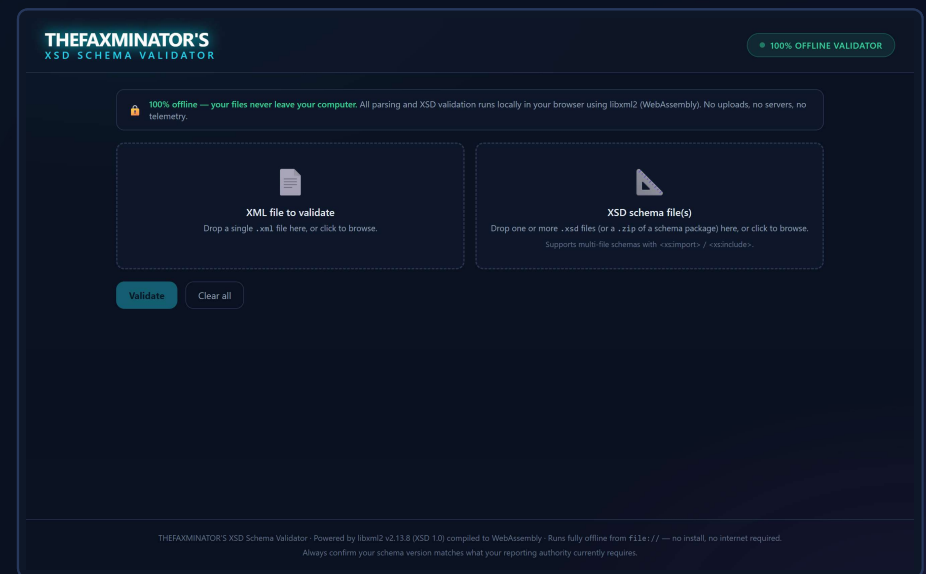
● 100% OFFLINE · libxml2 v2.13.8 · XSD 1.0

XSD Schema Validator

USER GUIDE

Validate XML against XSD schemas — a single file, a multi-file set, or a whole zip package — fully offline, right in your browser.

THEFAXMINATOR'S



What the validator does

Real XSD validation in the browser — not just a well-formedness check.



Real schema validation

Confirms your XML actually conforms to the XSD rules, powered by libxml2 — the engine behind xmllint.



100% offline

Runs entirely in your browser via WebAssembly. No internet, no server, nothing is uploaded.



Multi-file schemas

Resolves schemas split across many files with `<xs:import>` / `<xs:include>`, or a full .zip package.



Clear results

A prominent VALID / INVALID verdict, with every error's line number and a plain-language message.



XML file to validate

Drop a single .xml file here, or click to browse.



XSD schema file(s)

Drop one or more .xsd files (or a .zip of a schema package) here, or click to browse.
Supports multi-file schemas with `<xs:import>` / `<xs:include>`.

Two simple drop zones: your XML on the left, your schema(s) on the right.

What's in the download

Unzip once. Everything needed is inside — no install, no internet.



XSD-Schema-Validator-Offline.html

Double-click to open the app in any browser.



assets/

Engine, logic and styling — keep this folder next to XSD-Schema-Validator-Offline.html.



xmllint.js + xmllint-wasm-data.js

libxml2 (WebAssembly) — the validation engine. Don't rename.



fflate.min.js

Unzips schema packages you upload.



app.js · styles.css

App logic and styling.



XSD-Schema-Validator-User-Guide.pdf

The illustrated guide you are reading now.



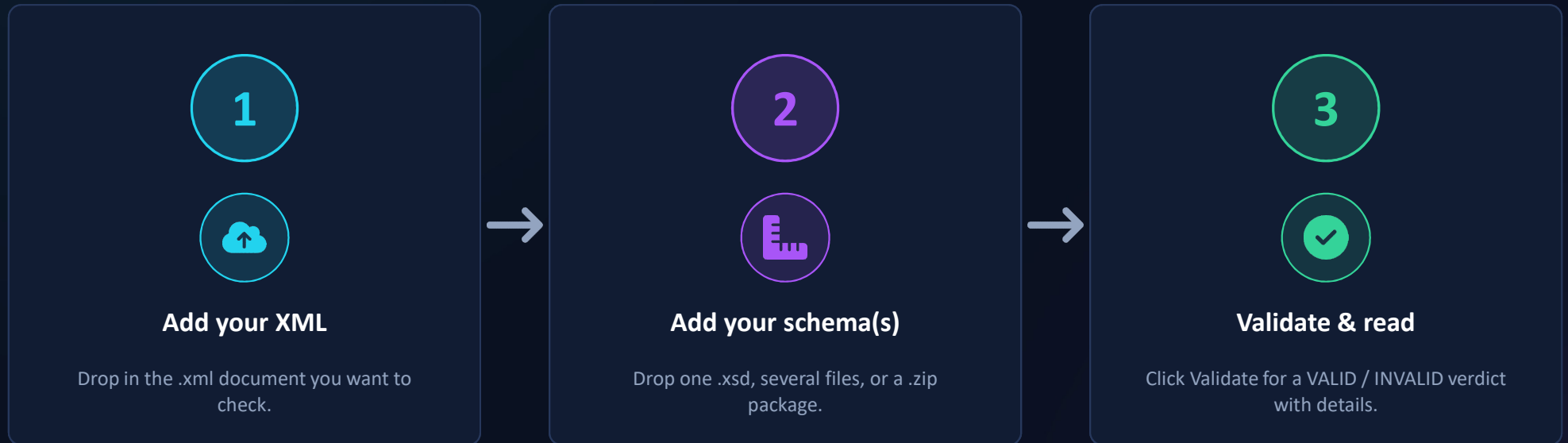
Keep it together

The assets folder must stay beside **XSD-Schema-Validator-Offline.html**.

xmllint-wasm-data.js holds the WebAssembly engine as text so the app works even when opened directly from **file://**. Don't delete or rename it.

How it works — three steps

The same simple flow, every time.



Step 1 — Add your XML file

The document you want to validate. Just one file.

1

Drop your .xml onto the left “XML file to validate” box — or click it to browse.

2

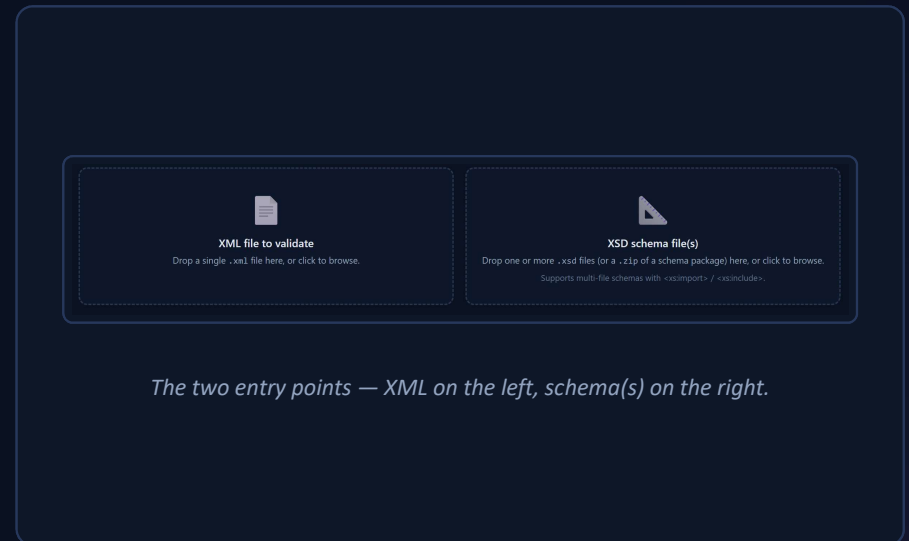
It appears under “XML document” with its file size, so you can confirm the right file loaded.

3

Only one XML at a time — adding another simply replaces it.



Private: the file is read locally in your browser — it never leaves your computer.



Step 2 — Add your schema file(s)

One file, many files, or a whole zip — the tool resolves the links between them.



A single .xsd

For a simple, self-contained schema with no imports or includes.



Multiple .xsd files

Select or drop them all together. Imports & includes resolve by filename.



A .zip package

Official packages unzip in your browser — folders and structure preserved.



Every file you add is listed under “Loaded schema files” — with its name and size — so you can confirm exactly what loaded before validating.

Pick the entry-point schema

When several schemas are loaded, one is the “root” that the others plug into.



Auto-detected for you

The tool reads your XML's `xsi:schemaLocation`, or picks the schema that no other schema imports (flagged “root?”).



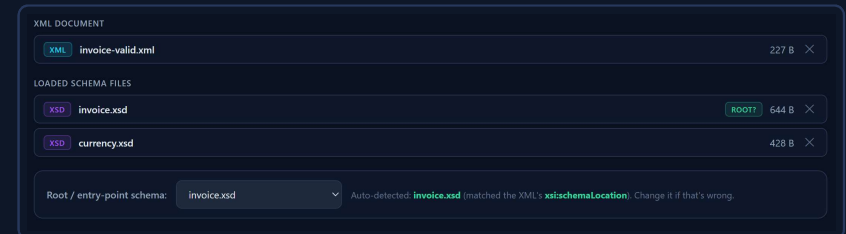
Override anytime

If the guess is wrong, choose the correct main schema from the Root / entry-point dropdown.



Single schema? Nothing to pick

With just one .xsd loaded, it's used automatically — no dropdown needed.



Loaded files, the “root?” hint, and the entry-point dropdown.

Step 3 — Validate & read the result

One click. A clear verdict.



Click Validate

The engine checks your document against the resolved schema set.



VALID (green)

The XML conforms to the schema. The panel confirms the document and every schema file used.



Full transparency

“Schema file(s) used” lists each .xsd involved, with the root clearly marked.

**VALID**

The XML document conforms to the schema.

Document: `invoice-valid.xml`

Schema file(s) used: `invoice.xsd (root)` `currency.xsd`

A passing document — clear, green, and auditable.

When it's INVALID — reading the errors

Every problem, pinpointed to the line.



Line number

Jump straight to the offending line in your XML document.



Plain-language message

libxml2's own description — e.g. a value outside the allowed set, or an unexpected element.



Raw output on demand

Expand “Show raw xmllint output” for the exact tool text.



Fix & re-validate

Each error is independent — correct them and run again.

INVALID

The XML document does not conform to the schema.

Document: `invoice-invalid.xml`

Schema file(s) used: `invoice.xsd (root)` `currency.xsd`

2 VALIDATION ERRORS

[SHOW RAW XMLLINT OUTPUT](#)

`invoice-invalid.xml`
1 Schemas validity error : Element 'amount', attribute 'currency': [facet 'enumeration'] The value 'XYZ' is not an element of the set {'USD', 'EUR', 'GBP', 'NOK'}.
line 3

`invoice-invalid.xml`
1 Schemas validity error : Element 'notes': This element is not expected.
line 4

Multi-file schemas — import vs include

Real-world schemas (OECD, IRS, ISO 20022, UBL) are rarely a single file.



<xs:import>

Pulls in a schema from a **DIFFERENT** namespace — e.g. a separate file just for currency codes (ISO 4217) or country codes (ISO 3166).



<xs:include>

Pulls in more of the **SAME** namespace — common complex types split across several files in one schema.



Missing a referenced file? The tool names the exact .xsd that's missing (and which file references it) instead of a vague parse error — just add it, or the full zip, and re-validate.

Loading an official schema package (.zip)

OECD CRS/CbC, IRS FATCA, ISO 20022, UBL/PEPPOL — drop the package as-is.

1

Download the official schema package (.zip) from your reporting authority.

2

Drop the .zip onto the schema box — it unzips in your browser, preserving the folder structure.

3

All .xsd files load together. If prompted, pick the main schema from the dropdown.



Check the version. Confirm the schema matches what your authority currently requires — official schemas get updated.



Client-side unzip

Powered by the bundled fflate library — the archive is expanded in memory. Nothing is sent anywhere, and __MACOSX / folder entries are ignored automatically.

Private by design — nothing leaves your computer

Built for sensitive compliance and tax data.



No uploads

Files are read in-browser. There are no servers and no network calls of any kind.



WebAssembly engine

libxml2 is compiled to WASM and embedded in the download — it runs locally.



Works with no internet

After downloading, you can disconnect entirely and it still validates.



No telemetry

No analytics, no tracking, no accounts. Ever.

Troubleshooting & tips

Quick fixes for the things people hit most.



“Validate” is greyed out

Add both an XML file and at least one schema — the button enables when both are present.



“Missing schema file”

An `<xs:import>/<xs:include>` target isn't loaded. Add that `.xsd` (or the whole `.zip`).



INVALID, but you expected valid

Check you picked the right entry-point schema, and that the schema version matches your data.



Wrong schema auto-selected

Use the Root / entry-point dropdown to choose the correct main `.xsd` yourself.



Blank page / nothing happens

Keep the assets folder (incl. `xmllint-wasm-data.js`) beside `XSD-Schema-Validator-Offline.html`; don't rename files.



Large file feels slow

Multi-MB XML validates in-browser — give it a moment on the first run while the engine loads.

QUICK REFERENCE

You're ready to validate



Open — Double-click XSD-Schema-Validator-Offline.html (keep the assets folder beside it).



Add XML — Drop the .xml document you want to check.



Add schema(s) — One .xsd, several files, or a .zip package.



Entry point — Confirm or choose the root schema if prompted.



Validate — Click Validate → read VALID / INVALID + line-level errors.