FiberQ validation rules

FiberQ can audit a fibre design before it leaves your desk. **Plugins → FiberQ → Validate project** runs fourteen rules over the project and lists what it finds in a dockable panel; from there you can jump to any issue on the map or export the whole run as HTML, JSON or CSV.

This page documents every rule: what it checks, why that matters in the field, and what to do about a finding.

- **User guide:** `validation-guide.md` — running a validation, reading the panel, exporting a report, and repairing stored lengths
- **Sample report:** `samples/demo-validation.html`
- **Demo project:** `samples/demo_project.qgz` — open it and run the validator yourself
- **Schema reference:** `schema.md`

How to read a finding

Every issue carries a **rule id** (`A1`, `D3`, …), a **severity**, the **layer** and **feature** it came from, and a message.

Severity	Meaning	What it should stop
Error	The project is internally inconsistent. Something references what is not there, or a geometry is unusable.	Handover, export, and anything downstream.
Warning	Probably wrong, or incomplete. A human needs to look.	A final sign-off, but not your work in progress.
Info	Worth knowing. Often a near-miss that may be deliberate.	Nothing on its own.

A design still being drawn is expected to produce warnings. The number that matters at handover is the error count.

Rule ids are stable and untranslated, in the panel and in every export, so you can filter on them and reference them in a report or a ticket.

A — Topology and connectivity

Whether the network actually joins up. These rules use a snap tolerance (default **5 map units**) that you can think of as "close enough to be the same point".

> **A note on tolerance and your CRS.** The tolerance is in map units. In a national grid those are true metres. In Web Mercator (EPSG:3857) they are not: distances there are inflated by 1/cos(latitude), so a 5-unit tolerance behaves like 3.5 m in Serbia and about 3.1 m in the Netherlands — the A-rules are stricter than the number suggests. The tolerance is a setting the project keeps (see Configuring the rules). (E1 reports a geographic CRS, where the effect is drastic enough to break the rules outright; it does not report Web Mercator, where they still work.)

A1 — Cable endpoints are connected

Warning · every cable layer

Each end of every cable should land on an element (pole, manhole, joint closure, ODF, termination box) or on another cable's end. An end that reaches nothing is either an unfinished run or a drawing slip.

Fires once per unconnected endpoint, so a completely stranded cable produces two findings — it has two ends.

Typical causes: the run was left half-drawn; snapping was off while digitising; an element was moved after the cable was laid.

What to do: connect the end, or accept it if the run is genuinely still in progress.

A2 — Cable endpoints are not near-misses

Info · every cable layer

An endpoint that sits just outside tolerance — close enough that it was clearly meant to connect, far enough that it does not. These are the ones that look right on screen and are wrong in the data.

A2 refines A1 rather than replacing it: an endpoint reported here is also reported by A1.

What to do: snap it. This is the single most common real defect in a hand-digitised design.

A3 — Elements are attached to the network

Warning · all point-element layers

A pole, manhole, closure or termination that lies near no cable and no route. It exists in the project but takes part in nothing.

Typical causes: elements dropped in during planning and never connected; elements left behind after a route was re-drawn.

What to do: connect it, delete it, or leave it if the design is unfinished — this is a warning precisely because "planned but not yet connected" is a legitimate state.

B — Referential integrity and identity

Whether the links between features still resolve.

B1 — Optical slack references an existing cable

Error · Optical slack

Every slack loop records which cable it belongs to (`cable_layer_id` + `cable_fid`). This checks the reference resolves and points at a cable: the layer is in the project, it is a cable layer, and the feature is in it.

The layer check is not pedantry. A reference to a route or a duct resolves perfectly well, so without it a slack loop recorded against a trench reads as a valid link and the whole bill of materials inherits the error.

Typical causes: the cable was deleted; the layer was removed or replaced; the project was rebuilt from parts; the feature was placed by clicking near a trench rather than the cable.

What to do: re-link the slack, or delete it. A slack whose cable is gone inflates the bill of materials for a cable that no longer exists.

B2 — Fiber break references an existing cable

Error · Fiber break

The same check for fibre-break records. A break recorded against the Route layer is reported here: a fibre break is a break in a fibre, and before v1.4.0 the break tool would accept a click near any line — including a trench or a duct.

B3 — Cable references are spatially coherent

Warning · Optical slack, Fiber break

The reference resolves, but the feature sits far from the cable it names — beyond tolerance. Usually the cable was re-routed after the slack or break was placed, so the link is stale even though it still points somewhere.

What to do: move the feature onto its cable, or re-link it to the cable it is actually on now.

B4 — Feature identity present and unique

Error · every FiberQ layer

Every feature carries a `fiberq_uuid`: a stable identity that survives export, re-import and round trips through other tools. This checks each feature has one and that no two share it.

Typical causes: features copied and pasted (which duplicates the id); features created by another tool that does not know about the field; a project older than the identity migration that has not been opened and migrated yet.

What to do: open and re-save the project so the migration runs. Duplicate ids need the copy deleted and redrawn.

C — Completeness

C1 — Required attributes present

Warning · every FiberQ layer

Fields the schema marks as required are filled in. Missing values are exactly what turn up as blanks in a bill of materials or an as-built document.

What to do: fill them in. A single unnamed cable is what makes a whole schedule unusable.

C2 — Project contains FiberQ layers

Warning · project-wide

There is at least one FiberQ layer to check. Every other rule works by iterating layers, so a project with none sails through all of them and reports nothing — which reads as a clean bill of health for a project that was never checked at all.

What to do: open a FiberQ project, or create the layers from the FiberQ toolbar. If you meant to validate a project made in another tool, its layers need FiberQ's canonical names before the rules can see them.

D — Value domains and attribute sanity

Whether the values recorded are values the schema allows, and whether the numbers are physically plausible.

D1 — Attribute values within allowed domain

Warning · every layer with a controlled vocabulary

Fields with a fixed set of choices (cable type, subtype, state, route type, slack location and side) hold one of them.

Both spellings are accepted: FiberQ stores some vocabularies as English display labels and some as their Serbian equivalents, depending on when and how the feature was created. Both are as-built and both pass; only a value that is in neither set is reported.

What to do: correct the value using the layer's dropdown rather than by typing.

D2 — Numeric attributes within plausible ranges

Warning · every layer with numeric fields

Lengths, fibre counts, tube counts, attenuation and dispersion figures fall inside sane bounds — positive where they must be, and below a physical ceiling (fibre count, for instance, cannot exceed 1152).

Typical causes: a typo; a unit mix-up; a placeholder value never replaced.

D3 — Stored lengths agree with geometry

Warning · Route, all cable and pipe layers

The length recorded on a feature matches the length of the line drawn. Lengths are measured **on the project ellipsoid**, the same way the QGIS measure tool does — so the figure is what a surveyor would find on site, not what the map projection makes it look like.

Two things make a stored length wrong:

1. **The geometry was edited where FiberQ could not follow.** From FiberQ 1.6.0 a stored length follows the edit: move a vertex of a saved line and the length moves with it, in the same undo step, and other edits made in QGIS (a line not saved yet, a split, a tool such as Route correction) are re-measured when the layer is saved. A line edited in another program, with the plugin switched off, or in a FiberQ older than 1.6.0 keeps its old value. In a project that uses buffered transaction groups (Project Properties → Data Sources) there is no re-measuring at save, so FiberQ follows only the vertices you move on lines already saved: a feature you add there (including the new part of a split, or a paste), or a line that a tool such as Route correction changes, keeps its old value.
2. **It was written in map units.** Versions of FiberQ before 1.4.0 stored the projected length for routes and cables. Until 1.6.0, **Cut infrastructure** did the same for the two parts it creates whenever the project had no ellipsoid set; it also wrote the metres into `duzina_km` and reset `total_len_m` to the bare length, dropping the slack. In Web Mercator a projected length is inflated by 1/cos(latitude) — about 41% at Serbian latitudes, 62% in the Netherlands, 100% in southern Finland. A bill of materials built on those numbers over-orders by the same margin.

What to do: run **Plugins → FiberQ → Recalculate lengths…**. It shows you exactly what will change before writing anything, and rewrites `duzina`, `duzina_m`, `duzina_km` and `total_len_m` from the geometry. Slack values are read but never changed.

Lengths are measured on the project's ellipsoid if one is set, and otherwise on the ellipsoid the CRS itself names — so D3 works in a geographic CRS too. Only a CRS that names no ellipsoid at all is skipped, and D3 says so rather than comparing metres to map units.

E — Coordinate systems and geometry health

E1 — Coordinate reference systems are consistent

Warning · project-wide

The project has a CRS, all FiberQ layers share one, and that CRS expresses the connectivity tolerance meaningfully. Mixed CRSs make every tolerance in every other rule mean something different per layer.

A project with no CRS at all is reported separately. Every other rule reads each layer's CRS, so your existing data is unaffected and lengths stay correct — but the canvas has no projection, so a basemap cannot line up and the next feature you draw is placed in an undefined system.

This is not hypothetical: opening a project saved by QGIS 4.2 in QGIS 3.40 drops the project CRS (QGIS reports "could not be completely loaded"), while the layers keep theirs. QGIS 3 cannot read the project's other FiberQ settings either — relations, colour catalogues, latent elements, picture and drawing links, the auto-save path, the validation settings, and data an interchange import carried for other tools (the passthrough store) — and from 1.6.0 FiberQ warns on open that saving the project there removes them permanently. Reopen it in QGIS 4 to keep working. If you only need to look at it in QGIS 3, set the CRS in **Project → Properties → CRS** for that session and do not save.

A **geographic CRS** (EPSG:4326 and friends) is flagged separately. Lengths are fine there — they are measured on the ellipsoid — but the A-rule tolerance is in map units, and a map unit in a geographic CRS is a degree. The default 5 then means roughly 550 km, and A1/A2/A3 stop meaning anything.

What to do: reproject the odd layer out, and work in a projected CRS — ideally your national grid rather than Web Mercator.

E2 — Geometries are present and well formed

Error for a missing or empty geometry, Warning for a malformed one · every FiberQ layer

Every feature has a geometry and it is not empty (an Error otherwise). A line of zero length, a single-point line included, or one that crosses itself, and a polygon of zero area or with a self-intersecting boundary, are reported as Warnings.

Typical causes: an interrupted digitising session; an import that dropped geometry; a feature created by an attribute-table edit.

What to do: redraw the feature. A null geometry cannot be exported, measured or located.

Exporting a report

With a result in the panel, **Export report…** writes it in whichever format the filename asks for:

Format	Use it for
HTML	Handover and audit files. Self-contained — no internet needed, prints cleanly, opens in any browser years later.
JSON	Tooling. Stable keys, machine-readable, and the basis of the WP3 interchange format.
CSV	Spreadsheets and issue trackers. One row per issue.

Rule ids, severities and categories stay in English in every format so that downstream tools can key on them; only prose is translated.

The JSON carries a `format_version` — check it before parsing if you are building something on top.

Configuring the rules

The settings live in the project, so they travel with it: whoever opens the project validates it the same way. A project that has none uses the defaults:

Setting	Default	Meaning
`tol`	`5.0`	snap tolerance, in map units, for the A-rules and B3
`length_rel_tol`	`0.01`	a stored length agrees with the geometry within this fraction of it (D3, and **Recalculate lengths**)…
`length_abs_tol`	`0.5`	…or within this many metres, whichever is larger
`disabled_rules`	none	rule ids that do not run; the panel names them under its summary, as "Skipped by this project's settings: A2, B2", and a report lists them as **Rules skipped** (HTML) or `run.rules_skipped` (JSON)
`severity_overrides`	none	rule id → `error`, `warning` or `info`, for every issue that rule reports

There is no dialog for them in this release. Set them from the QGIS Python console (**Plugins → Python Console**), then save the project:

```
from qgis.core import QgsProject
from fiberq.core.validation_manager import Severity, ValidationConfig

project = QgsProject.instance()
config = ValidationConfig.from_project(project)
config.tol = 2.0                                    # map units
config.disabled_rules.add("A2")
config.severity_overrides["C1"] = Severity.INFO
config.save(project)
```

They are stored as one JSON object in the project entry `FiberQPlugin/validation`. A value FiberQ cannot use, or a rule id no rule has, is named in a warning when validation runs, and the default stands in for it; the other settings still apply. **Recalculate lengths** reads the same two length tolerances as D3, so recalculating and then validating clears every stored-length mismatch D3 reports.

A rule that fails to run is reported as such rather than passing silently — if a rule crashes on your data, the run continues and the panel says how many rules gave up. Which ones, and why, are in **View → Panels → Log Messages**, on the FiberQ tab, and in the HTML and JSON reports. Please open an issue if you see one.

Part of the FiberQ QGIS plugin, funded by the NLnet NGI Zero Commons Fund.