FiberQ interchange — field mapping

The canonical field and value mapping for the FiberQ interchange format. Read that first: this page is the mapping table it refers to.

Generated from the code. The plugin side comes from `fiberq/models/schema.py` and the canonical side from `fiberq/core/interchange_fields.py`, with a test (`tests/test_interchange_mapping.py`) that fails if this page and the code drift apart. A field added to the plugin without a canonical name fails CI.

Why names change at all

FiberQ's stored field names are the original Serbian database names — `duzina_m`, `slabljenje_dbkm`, `vrsta_omotaca` — and several stored values are Serbian too (`vazdusna`, `glavni`). That is the plugin's schema, it is what is on disk in every existing project, and it is not changing: renaming a stored field is a data migration, not a cosmetic change.

A bundle carrying those names would not be a tool-neutral format. It would be FiberQ's schema in a GeoPackage, and every other tool would have to learn Serbian to read it — which is mapping directly to one tool's schema, exactly what rule 3 exists to prevent. So a bundle carries canonical English names and canonical values, and this page is the bijection.

Nothing here changes your project. The renaming happens on the copy written into the bundle. Your layers, their fields and their values are read and left exactly as they are.

The mapping is per element type

`tip` means cable type on a cable and pole type on a pole. `podtip` means subtype on one and segment type on the other. A single global rename table would quietly merge two different meanings into one canonical field, and nothing downstream could tell them apart afterwards. So the tables below are per **roster** — a group of element types that share a field set.

The four mismatch cases

Case	What it means	How it is handled
Both sides	The canonical field and a plugin field mean the same thing	Renamed on export, renamed back on import
Plugin-only	A field FiberQ has and other tools may not	Still gets a canonical name — the format is a superset, so nothing has to be dropped to fit
Format-only	Something the format models and the plugin does not (fibre splices, trays, duct occupancy)	Carried in the side-car tables and the passthrough store; the plugin never sees it and never destroys it
Structural	The same fact, a different shape on each side	Converted, not renamed — see below

The structural case

FiberQ records which cable an optical slack loop or a fibre break belongs to as `cable_layer_id` + `cable_fid` — a QGIS layer id plus a feature id. Both are local to one project file. They mean nothing in another tool, and nothing in the same project after the layer is rebuilt.

A bundle carries **`cable_uuid`** instead, resolved on export and resolved back on import. The local pair is not written. A reference that cannot be resolved — the cable was deleted — is **reported and left empty**, never pointed at the nearest plausible cable.

Plugin (stored)	Canonical	Rule
`cable_layer_id` + `cable_fid`	`cable_uuid`	Resolved through `fiberq_uuid`; unresolvable references are reported, not guessed

The one field that is never renamed

`fiberq_uuid` keeps its name everywhere. It is the join key every conformant tool must already know (section 4), so translating it would be renaming the thing that makes a round trip verifiable.

Field tables

Point elements — ODF, OTB, TO, TB, patch panel

Element types: `odf`, `otb`, `patch_panel`, `tb`, `to`

Canonical field	Plugin field (stored)	Type	Units	Notes
`name`	`naziv`	text	—	—
`manufacturer`	`proizvodjac`	text	—	—
`label`	`oznaka`	text	—	—
`port_capacity`	`kapacitet`	integer	ports	—
`total_connections`	`ukupno_kj`	integer	—	—
`required_capacity`	`zahtev_kapaciteta`	integer	—	—
`reserve_capacity`	`zahtev_rezerve`	integer	—	—
`port_label`	`oznaka_izvoda`	text	—	—
`numbering`	`numeracija`	text	—	—
`site_name`	`naziv_objekta`	text	—	—
`address_street`	`adresa_ulica`	text	—	—
`address_number`	`adresa_broj`	text	—	—
`address_id`	`address_id`	text	—	same name on both sides
`status`	`stanje`	text (domain)	—	one of `Planned`, `Built`, `Existing` (stored and carried as-is)
`install_year`	`godina_ugradnje`	integer	year	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Joint closures

Element types: `closure.joint`

Canonical field	Plugin field (stored)	Type	Units	Notes
`name`	`naziv`	text	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Poles

Element types: `pole`

Canonical field	Plugin field (stored)	Type	Units	Notes
`pole_type`	`tip`	text	—	—
`segment_type`	`podtip`	text	—	—
`height_m`	`visina`	real	m	—
`material`	`materijal`	text	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Manholes

Element types: `manhole`

Canonical field	Plugin field (stored)	Type	Units	Notes
`manhole_id`	`broj_okna`	text	—	—
`manhole_type`	`tip_okna`	text	—	—
`construction_type`	`vrsta_okna`	text	—	—
`position`	`polozaj_okna`	text	—	—
`address`	`adresa`	text	—	—
`status`	`stanje`	text	—	—
`install_year`	`god_ugrad`	integer	—	—
`description`	`opis`	text	—	—
`dimensions_cm`	`dimenzije`	text	cm	—
`wall_material`	`mat_zida`	text	—	—
`cover_material`	`mat_poklop`	text	—	—
`drainage`	`odvodnj`	text	—	—
`cover_heavy`	`poklop_tes`	boolean	—	—
`cover_light`	`poklop_lak`	boolean	—	—
`step_count`	`br_nosaca`	integer	—	—
`wall_thickness_cm`	`debl_zida`	real	cm	—
`ladder`	`lestve`	text	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Routes

Element types: `route`

Canonical field	Plugin field (stored)	Type	Units	Notes
`name`	`naziv`	text	—	—
`length_m`	`duzina`	real	m	—
`length_km`	`duzina_km`	real	km	—
`route_type`	`tip_trase`	text	—	controlled vocabulary — see below
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Cables — aerial and underground

Element types: `cable.aerial`, `cable.underground`

Canonical field	Plugin field (stored)	Type	Units	Notes
cable_type	tip	text	—	controlled vocabulary — see below
cable_subtype	podtip	text	—	controlled vocabulary — see below
color_code	color_code	text	—	same name on both sides
tube_count	broj_cevcica	integer	—	—
fiber_count	broj_vlakana	integer	—	—
cable_model	tip_kabla	text	—	—
fiber_type	vrsta_vlakana	text	—	—
sheath_type	vrsta_omotaca	text	—	—
armour_type	vrsta_armature	text	—	—
wavelength_band	talasno_podrucje	text	—	—
name	naziv	text	—	—
attenuation_db_km	slabljenje_dbkm	real	dB/km	—
chromatic_dispersion_ps_nm_km	hrom_disp_ps_nmxkm	real	ps/nm/km	—
status	stanje_kabla	text	—	controlled vocabulary — see below
installation_type	cable_laying	text	—	controlled vocabulary — see below
network_type	vrsta_mreze	text	—	—
install_year	godina_ugradnje	integer	year	—
constr_fibers_in_tubes	konstr_vlakna_u_cevcicama	integer	—	—
constr_bonded_element	konstr_sa_uzlepljenim_elementom	integer	—	—
constr_gel_filled	konstr_punjeni_kabl	integer	—	—
constr_aramid_armour	konstr_sa_arm_vlaknima	integer	—	—
constr_non_metallic	konstr_bez_metalnih	integer	—	—
from_label	od	text	—	—

Canonical field	Plugin field (stored)	Type	Units	Notes
`to_label`	`do`	text	—	—
`length_m`	`duzina_m`	real	m	—
`slack_m`	`slack_m`	real	m	same name on both sides
`total_length_m`	`total_len_m`	real	m	—
`fibers_per_tube`	`fibers_per_tube`	integer	—	same name on both sides
`total_fibers`	`total_fibers`	integer	—	same name on both sides
`color_standard`	`color_standard`	text	—	same name on both sides
`branch_index`	`branch_index`	integer	—	**on demand** — created only when a feature needs it, so a layer may not carry this column; same name on both sides
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Ducts — PE and transition

Element types: `duct.pe`, `duct.transition`

Canonical field	Plugin field (stored)	Type	Units	Notes
`material`	`materijal`	text	—	—
`capacity`	`kapacitet`	text	—	—
`diameter_mm`	`fi`	integer	mm	—
`from_label`	`od`	text	—	—
`to_label`	`do`	text	—	—
`length_m`	`duzina_m`	real	m	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Optical slack

Element types: `slack`

Canonical field	Plugin field (stored)	Type	Units	Notes
`slack_type`	`tip`	text	—	—
`length_m`	`duzina_m`	real	m	—
`location`	`lokacija`	text	—	controlled vocabulary — see below
`side`	`strana`	text	—	controlled vocabulary — see below
`note`	`napomena`	text	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Fibre breaks

Element types: `fiber_break`

Canonical field	Plugin field (stored)	Type	Units	Notes
`name`	`naziv`	text	—	—
`distance_m`	`distance_m`	real	m	same name on both sides
`segments_hit`	`segments_hit`	integer	—	same name on both sides
`recorded_at`	`vreme`	text	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Service areas

Element types: `service_area`

Canonical field	Plugin field (stored)	Type	Units	Notes
`name`	`name`	text	—	same name on both sides
`created_at`	`created_at`	text	—	same name on both sides
`area_m2`	`area_m2`	real	m²	same name on both sides
`perimeter_m`	`perim_m`	real	m	—
`source_part_count`	`count`	integer	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Buildings / objects

Element types: `building`

Canonical field	Plugin field (stored)	Type	Units	Notes
`building_type`	`tip`	text	—	—
`floors_above`	`spratova`	integer	—	—
`floors_below`	`podzemnih`	integer	—	—
`street`	`ulica`	text	—	—
`street_number`	`broj`	text	—	—
`name`	`naziv`	text	—	—
`note`	`napomena`	text	—	—
`fiberq_uuid`	`fiberq_uuid`	text	—	identity, never renamed

Every feature additionally carries the bundle's own columns: **`fq_type`**, **`placement`** and **`fq_extra_json`** (section 6.1).

Value domains

The fields below hold values from a controlled vocabulary, and FiberQ stores them in Serbian. A bundle carries the canonical form.

Real projects hold **both spellings** — the English display label on some features and the stored Serbian value on others, depending on how and when the feature was created (see validation rule D1). Both are as-built, and both map to the same canonical value.

A value in neither set is **carried through unchanged**. Tidying a vocabulary is not worth discarding what the user recorded.

`route_type` — routes

Canonical value	Stored by FiberQ	Also accepted
`aerial`	`vazdusna`	`Aerial`
`underground`	`podzemna`	`Underground`
`through_building`	`kroz objekat`	`Through the object`

`location` — optical slack

Canonical value	Stored by FiberQ	Also accepted
`manhole`	`OKNO`	`Manhole`
`pole`	`Stub`	`Pole`
`building`	`Objekat`	`Object`

`side` — optical slack

Canonical value	Stored by FiberQ	Also accepted
`from`	`od`	`FROM`
`to`	`do`	`TO`
`mid_span`	`sredina`	`MID SPAN`

`cable_type` — cables

Canonical value	Stored by FiberQ	Also accepted
`optical`	`Optical`	`opticki`
`copper`	`Copper`	`bakarnI`, `bakarni`

`cable_subtype` — cables

Canonical value	Stored by FiberQ	Also accepted
`backbone`	`glavni`	`Backbone`
`distribution`	`distributivni`	`Distribution`
`drop`	`razvodni`	`Drop`

`status` — cables

Canonical value	Stored by FiberQ	Also accepted
`planned`	`Projektovano`	`Planned`
`existing`	`Postojeće`	`Existing`
`under_construction`	`U izgradnji`	`Under construction`

`installation_type` — cables

Canonical value	Stored by FiberQ	Also accepted
`underground`	`Podzemno`	`Underground`
`aerial`	`Vazdusno`	`Aerial`

Mapping another tool

A second tool maps to the **canonical** column, never to FiberQ's stored column. That is what keeps adding a third tool from being a change to the first (rule 3).

If a tool has a field with no canonical equivalent, the honest options are, in order: propose it for the canonical set (the format is meant to be a superset), or write it as a column under its own name (what this plugin does, below; a reader must carry it, rule 1), or carry it in the feature's `fq_extra_json`, or carry the whole object in `fq_extension`. What is never acceptable is mapping it onto the nearest canonical field that nearly fits — that loses information irreversibly while looking like it worked.

A column with no canonical name

The two directions are not symmetrical, and a tool that assumes they are will lose data.

Writing. A column FiberQ has no canonical name for is written to the bundle **under the name it arrived with**, with its own type. It is not renamed and it is not moved into `fq_extra_json` — unless its name is taken. A GeoPackage compares column names regardless of letter case, so when a column's name, in any case, is one FiberQ writes on that layer — the canonical name another column is renamed to (a column of your own called `name` or `Name` beside FiberQ's `naziv`, say), `fq_type`, `placement` or `fq_extra_json` in another case, `cable_uuid` on a slack or fibre-break layer, `geom`, or a `fid` that is not the layer's own row id — or differs only in case from the name of a column before it, its values are written into the feature's `fq_extra_json` under its own name instead, and the export says which columns went there. Nothing is merged or dropped; where the feature's own `fq_extra_json` already has a key of that name, the column's value replaces it, and the export says so. A column named exactly `fq_type`, `placement` or `fq_extra_json` is the bundle's own column and keeps its values, and a GeoPackage layer's own `fid` stays the row id. A reader with no column for a column FiberQ wrote must still carry it (rule 1); this plugin keeps it in `fq_extra_json`, as described next. A reader that has one — the tool that wrote it, most often — finds it exactly where it left it.

Reading. A bundle column with no counterpart in the target layer is kept in that feature's `fq_extra_json`, keyed by the identity of the feature it belongs to, and is emitted again on the next write (rule 1, section 8). Nothing is dropped because this implementation has no field for it — an empty value included, and a `cable_uuid` on an element type the plugin does not link to a cable — except on a feature with no `fiberq_uuid`, which has nothing to keep it against; the import says how many.

One field in the table above is marked **on demand**: the plugin's own tools create that column only when a feature needs it, so a layer they create does not carry it (a cable layer created by a bundle import does), and a bundle written from such a project may not contain it. It is published here because it exists in real projects and a conformant tool must expect it.

Part of the FiberQ QGIS plugin, developed with support from the NLnet NGI Zero Commons Fund. The specification and this mapping are CC-BY-4.0.