FiberQ data schema

This document describes the FiberQ project data model: the layer types, their fields, units and value domains, the CRS expectations, and the project **schema version** marker. It describes the single source of truth in `fiberq/models/schema.py` and is kept in step with it by hand; the interchange field mapping is the page generated from it (and from `fiberq/core/interchange_fields.py`) and checked against them in CI.

> **Status:** this is the as-built schema for FiberQ **1.x**, schema version **1.0**. It faithfully reflects what the plugin creates today. A small number of known inconsistencies are listed under Notes & limitations; they are preserved as-is in 1.0 and will be reconciled by the open interchange-format work, where changing stored data is safe to coordinate.

Conventions

- **Field names are the stored schema.** They are mostly the original (Serbian) database names. The English text shown in the QGIS attribute table is a display **alias** only — renaming a stored field is a data migration, not a cosmetic change.
- `fiberq_uuid` is the per-feature **identity** field. It is present on every FiberQ layer (appended automatically when the layer is created) and is what links features across operations and exports.
- **Geometry & CRS.** Layers are created in memory. Most layers use the map canvas destination CRS (falling back to `EPSG:3857`); the polygon layers Service Area and Objects use the **project** CRS.
- **Type legend:** `text` (string), `int` (integer), `double` (real), `enum` (string with a fixed option list), `year` (4-digit, stored as integer), `bool` (boolean).
- **Value maps** let a field show an English label while storing a different (often Serbian) value — noted per field below.

Schema versioning

The project schema version is the string `SCHEMA_VERSION` in `fiberq/models/schema.py` (currently `1.0`). It is:

- **tracked separately from the plugin version** in `metadata.txt` — the two bump independently;
- **persisted into the project** when you save (a QGIS project entry), and **mirrored into the GeoPackage** `_fiberq_metadata` table on a full project export, so the version travels with the data file;
- **absent in pre-1.0 projects** — those are treated as the pre-marker baseline (version `0`) and are upgraded by the migration framework.

Migrations (on load)

When a project is opened, FiberQ runs `fiberq/core/migrations.py:run_migrations`:

1. It reads the stored version. A **malformed / foreign** marker (anything that is not a dotted run of integers, e.g. `v2.0`) is left untouched — never coerced, so a genuinely newer project can't be silently downgraded.
2. If the project is **older**, it runs the ordered chain of migration steps in the half-open range `(stored, current]`. The only step today, `uuid-identity` (baseline `0` → `1.0`), adds and backfills the `fiberq_uuid` field.
3. The current version is stamped **only if every step succeeded**. If a step cannot persist (a locked or read-only GeoPackage), the marker is left unchanged so the next load retries — the marker never claims a migration that didn't reach disk.
4. A project stamped **newer** than the running plugin is left untouched (never downgraded).

Each step is **idempotent**: re-running on an up-to-date project does nothing. Because a project entry only becomes durable when the project is saved (or exported to GeoPackage, which also stamps the marker), an on-load upgrade that isn't saved simply re-runs — harmlessly — next time.

The `fiberq_uuid` identity invariant

`fiberq_uuid` must exist and be populated on **every** FiberQ feature. Adding it is the `0 → 1.0` migration, but keeping it filled is a standing invariant, not a one-time step: an idempotent backfill runs on **every** load regardless of the marker, so features introduced later (imports, external edits, interchange round-trips) are healed too.

Layer types

Point elements (shared roster)

These 12 element layers share one 15-field roster (one QGIS layer per type): **ODF, TB, Patch panel, OTB, Indoor OTB, Outdoor OTB, Pole OTB, TO, Indoor TO, Outdoor TO, Pole TO, Joint Closure TO**. Geometry: Point. The only per-layer variation: an "OD cabinet" (`od ormar`) layer defaults `kapacitet` to 24.

field	type	units	domain / default	attribute alias
naziv	text		""	Name
proizvodjac	text		""	Manufacturer
oznaka	text		""	Label
kapacitet	int	ports	0 (24 for OD cabinets)	Capacity
ukupno_kj	int		0	Total SCs
zahtev_kapaciteta	int		0	Required capacity
zahtev_rezerve	int		0	Reserve capacity
oznaka_izvoda	text		""	Port label
numeracija	text		""	Numbering
naziv_objekta	text		""	Site name
adresa_ulica	text		""	Street
adresa_broj	text		""	Street No.
address_id	text		""	Address ID
stanje	enum		[Planned, Built, Existing], default Planned	Status
godina_ugradnje	year		2025	Install year
fiberq_uuid	text		identity	FiberQ UUID

Joint Closures

Point. Legacy name: `Nastavci`.

field	type	alias
naziv	text	Name
fiberq_uuid	text	FiberQ UUID

Poles

Point. Legacy name: `Stubovi`.

field	type	units	alias
tip	text		Type
podtip	text		Subtype
visina	double	m	Height (m)
materijal	text		Material
fiberq_uuid	text		FiberQ UUID

Manholes

Point. Legacy name: `OKNA`.

field	type	units	alias
broj_okna	text		Manhole ID
tip_okna	text		Manhole type
vrsta_okna	text		Construction type
polozaj_okna	text		Position
adresa	text		Address
stanje	text		Status (free text)
god_ugrad	int		Installation year
opis	text		Description
dimenzije	text	cm	Dimensions (cm)
mat_zida	text		Wall material
mat_poklop	text		Cover material
odvodnj	text		Drainage
poklop_tes	bool		Heavy cover
poklop_lak	bool		Light cover
br_nosaca	int		Number of steps
debl_zida	double	cm	Wall thickness (cm)
lestve	text		Ladder
fiberq_uuid	text		FiberQ UUID

Route

LineString. Legacy name: `Trasa`.

field	type	units	value map	alias
naziv	text			Route name
duzina	double	m		Length (m)
duzina_km	double	km		Length (km)
tip_trase	text		Aerial→vazdusna, Underground→podzemna, Through the object→kroz objekat	Route type
fiberq_uuid	text			FiberQ UUID

Optical slack

Point. Canonical name `Optical slack`. FiberQ's slack tools create the layer as `Optical slacks` (plural); an interchange import creates it as `Optical slack`; `Opticke_rezerve` is the pre-1.0 name. All three are recognised.

field	type	units	value map	alias
tip	text			Type
duzina_m	double	m		Length (m)
lokacija	text		Manhole→OKNO, Pole→Stub, Object→Objekat	Location
cable_layer_id	text		FK → cable layer	Cable layer ID
cable_fid	int		FK → cable feature	Cable feature ID
strana	text		FROM→od, TO→do, MID SPAN→sredina	Side
napomena	text			Note
fiberq_uuid	text			FiberQ UUID

PE pipes / Transition pipes

LineString. Same roster. Legacy names `PE cevi` / `Prelazne cevi`.

field	type	units	alias
materijal	text		Material
kapacitet	text		Capacity
fi	int	mm	Diameter (mm)
od	text		From
do	text		To
duzina_m	double	m	Length (m)
fiberq_uuid	text		FiberQ UUID

Service Area

Polygon, **project CRS**. Legacy names `Service area`, `Rejon`. Fields are already English.

field	type	units	notes
name	text		
created_at	text		timestamp
area_m2	double	m²	ellipsoidal area
perim_m	double	m	
count	int		source feature count
fiberq_uuid	text		identity

Objects

Polygon, **project CRS**. Legacy name `Objekti`.

field	type	alias
tip	text	Type
spratova	int	Floors above ground
podzemnih	int	Floors below ground
ulica	text	Street
broj	text	Number
naziv	text	Name
napomena	text	Note
fiberq_uuid	text	FiberQ UUID

Aerial cables / Underground cables

LineString. Both share one roster. Legacy names `Kablovi_vazdusni` / `Kablovi_podzemni`. (`branch_index` is part of the roster but created on demand: FiberQ adds the column the first time **Branch cables (offset)** is run on the layer, so a cable layer made by Lay cable does not carry it. A cable layer created by importing an interchange bundle does.)

field	type	units	value map	alias
tip	text		Optical→Optical, Copper→Copper (older projects: opticki, bakarnI)	Cable type
podtip	text		Backbone→glavni, Distribution→distributivni, Drop→razvodni	Segment type
color_code	text			Color code
broj_cevcica	int			Number of ducts
broj_vlakana	int			Number of fibers
tip_kabla	text			Cable model
vrsta_vlakana	text			Fiber type
vrsta_omotaca	text			Sheath type
vrsta_armature	text			Armour type
talasno_podrucje	text			Wavelength band
naziv	text			Name
slabljenje_dbkm	double	dB/km		Attenuation [dB/km]
hrom_disp_ps_nmxkm	double	ps/nm/km		Chromatic dispersion [ps/nm/km]
stanje_kabla	text		Planned→Projektovano, Existing→Postojeće, Under construction→U izgradnji	Status
cable_laying	text		Underground→Podzemno, Aerial→Vazdusno	Installation type
vrsta_mreze	text			Network type
godina_ugradnje	int	year		Installation year
konstr_vlakna_u_cevcicama	int		construction flag (0/1)	Fibers in ducts
konstr_sa_uzlepljenim_elementom	int		0/1	With bonded element
konstr_punjeni_kabl	int		0/1	Gel-filled cable
konstr_sa_arm_vlaknima	int		0/1	Aramid yarn armouring
konstr_bez_metalnih	int		0/1	Non-metallic
od	text			From
do	text			To

field	type	units	value map	alias
duzina_m	double	m		Length [m]
slack_m	double	m		Slack [m]
total_len_m	double	m		Total length [m]
fibers_per_tube	int		fiber-schema (interchange)	Fibers per tube
total_fibers	int		fiber-schema (interchange)	Total fibers
color_standard	text		fiber-schema (interchange), e.g. TIA-598-C	Color standard
branch_index	int		on demand: created the first time a cable is branched	—
fiberq_uuid	text		identity	FiberQ UUID

Fiber break

Point. Legacy name `Prekid vlakna`.

field	type	units	alias
naziv	text		Name
cable_layer_id	text		Cable layer ID
cable_fid	int		Cable feature ID
distance_m	double	m	Distance (m)
segments_hit	int		Segments hit
vreme	text		Time
fiberq_uuid	text		FiberQ UUID

Colour catalogues

Fibre/tube colours follow an ordered 12-colour ANSI/TIA-598-C sequence (`position → colour = (position − 1) mod tube_count`, default `tube_count = 12`). The seeded catalogue is **TIA-598-C**; catalogues are stored as JSON in a project entry and referenced by name per cable (`color_standard`). See `fiberq/models/color_catalogs.py`.

Layer-name aliases (legacy → canonical)

Older projects use the original names; FiberQ recognises them as the same type:

canonical	legacy / alternate
Poles	Stubovi
Manholes	OKNA
Patch panel	Patch Panel
Joint Closures	Nastavci
Optical slack	Optical slacks (the name the slack tools create), Opticke_rezerve
Fiber break	Prekid vlakna
Route	Trasa
Aerial cables	Kablovi_vazdusni
Underground cables	Kablovi_podzemni
PE pipes	PE cevi
Transition pipes	Prelazne cevi
Objects	Objekti
Service Area	Service area, Rejon

Notes & limitations

These are known as-built inconsistencies, preserved unchanged in schema 1.0 and scheduled for reconciliation by the interchange-format work (so that any change to stored values is coordinated and lossless):

- **Status vocabulary differs by layer:** element `stanje` = [Planned, Built, Existing] (stored English); cable `stanje_kabla` = [Planned, Existing, Under construction] (stored Serbian via value map); manhole `stanje` is free text.
- **`kapacitet` type differs:** integer on elements, text on pipes.
- **Manhole `poklop_tes` / `poklop_lak` are boolean** — booleans may not round-trip cleanly through some GeoPackage/OGR paths.
- **Cable `tip` storage direction (corrected in 1.6.0):** FiberQ stores `Optical` / `Copper`, and every cable restyle (at project open, after Lay cable and after each slack recompute) rewrites an older project's `opticki` / `bakarnI` to them. Until 1.6.0 the value map still expected the older values, so the attribute form showed every cable's type in brackets, "(Optical)"; it now maps to what is stored. Validation rule D1 still accepts the older values, and an interchange import writes `Optical` / `Copper`.
- **Mixed field naming:** a few keys are English in an otherwise Serbian roster (e.g. element `address_id`, the cable fibre-schema fields). These are kept as-is.