

BlueDot QuantumTouch IPoE

Theory of Operations — The Operator Plane

An operator- and NOC-facing theory of operations for QuantumTouch, the management plane that sits over the headless BlueDot Quantum IPoE BNG. It explains how QuantumTouch onboards a box, watches it, drives its configuration, and keeps an audit and recovery trail — and what an operator sees, sets, and relies on while the box is running. It is the companion to the Quantum IPoE BNG Theory of Operations, which covers the forwarding path itself. This document is for the person who runs the BNG, not the person who builds it.

Document Type	BlueDot Theory of Operations
Component	QuantumTouch — Operator Plane
Edition	Community Edition — Alpha
Audience	ISP Operators & NOC
Companion To	Quantum IPoE BNG — Theory of Operations (Data & Control Plane)
Source Tree	<code>ipoe/quantumtouch-bng</code> (<code>main</code>)
Classification	BlueDot Proprietary
Date	July 2026

Contents

Introduction 3

The System at a Glance 4

What QuantumTouch Does 6

Onboarding a Box: The Setup Wizard 7

How QuantumTouch Sees the Box 8

Analytics & Observability 10

The Life of an Operator Action 12

Operations Dashboards & Health 13

Audit, Events & Compliance 14

Configuration: Surfaces & Protection 16

Security Posture 18

Failure & Recovery 19

Scope and Caveats 20

Introduction

This document explains how **QuantumTouch** works in operation — what it does to get a BNG running, how it turns a live box into something you can watch and reason about, and what you, the operator, see and control while the system runs. It is a theory of operations, not an installation guide or an API reference. The goal is that after reading it you can look at QuantumTouch's dashboards, its health verdict, its event feed, and its charts and know what the numbers mean, why they move, and what to do about them.

QuantumTouch is the **operator plane** for the BlueDot Quantum IPoE BNG. The forwarding stack — the VPP data plane, the vpp_bridge translator, and the controller — is deliberately headless. It forwards packets, runs the subscriber lifecycle, and speaks REST, but it has no human interface, no historical metrics, no notion of an "operator," and no memory beyond its own configuration database. QuantumTouch is the layer that makes that stack **operable**: it onboards a fresh box, harvests data from it on a fixed cadence, presents analytics and dashboards, drives every configuration change, aggregates an audit trail, and protects configuration with point-in-time snapshots.

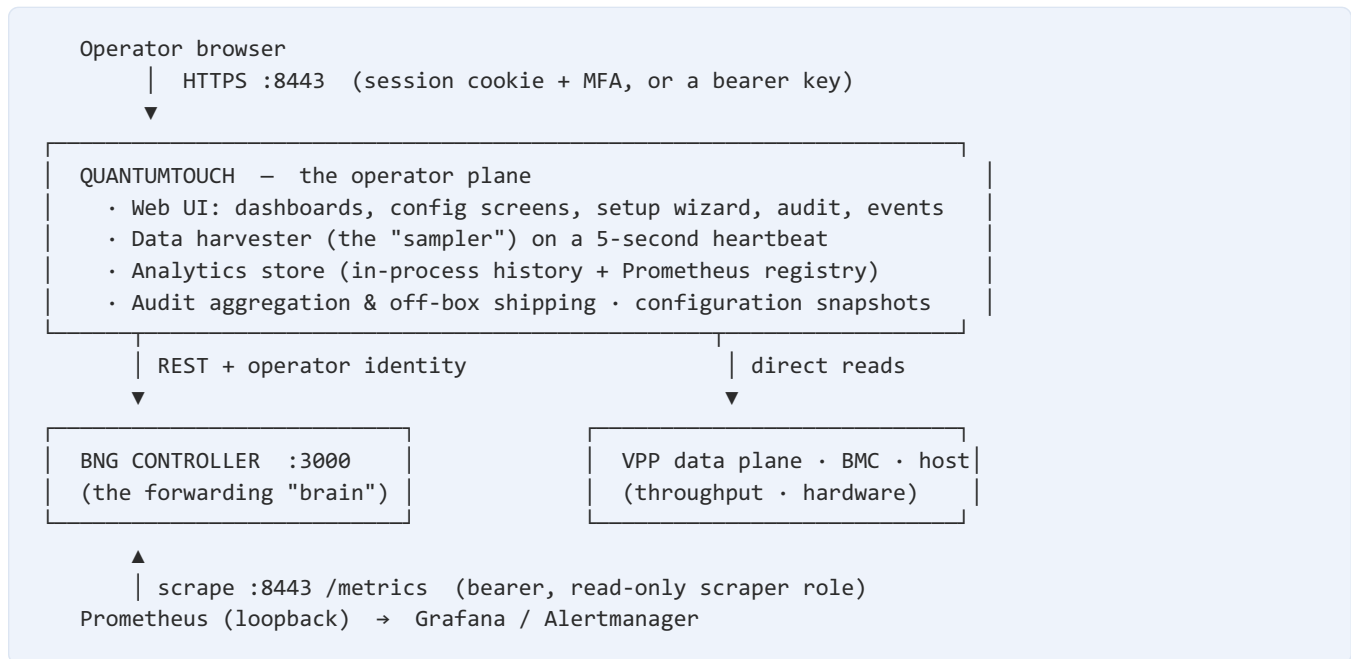
The single most important idea in this document — the one everything else hangs on — is that **QuantumTouch owns operator experience, not the forwarding path**. It never touches a packet, never programs a session, never makes a routing decision. Every change it makes to the BNG is a REST call to the controller; every number it shows you is something it polled. That boundary is what makes it safe: nothing QuantumTouch does can take the data plane down, and when QuantumTouch itself is unavailable the BNG keeps forwarding, unaffected. Knowing which side of that line a given behaviour lives on — the forwarding stack or the operator plane above it — is the key to operating and troubleshooting the box.

How this fits the companion document. The *Quantum IPoE BNG Theory of Operations* covers the forwarding path: DHCP onboarding, the subscriber lifecycle state machine, green/yellow/red policing, accounting, and recovery. This document is its mirror on the operator side. Where that one walks the *life of a subscriber*, this one walks the *life of an operator action*. Read together, they describe the whole system: the plane that moves packets, and the plane that lets a human run it.

Scope and one caveat. QuantumTouch is delivered as a single management application composed over the BNG. This Alpha, Community-Edition build ships the full operator experience — onboarding, harvesting, analytics, dashboards, audit, events, and snapshots are all real and running — but a handful of designed capabilities are not yet wired, and this document flags each one where it matters rather than describing an aspiration as reality. Two are worth naming up front and are revisited in *Scope and Caveats*: the Prometheus-first analytics store is designed but not yet the shipping read path (an in-process time-series store is), and a small number of metric families are registered but still report zero until a port-role map is populated.

The System at a Glance

QuantumTouch is not a separate box. It is a management process that runs on the same host as the BNG, presents a single web interface, and reaches everything else through well-defined seams. Knowing what talks to what is the foundation for everything that follows.



One web front door. Operators reach everything at <https://<box>:8443>. That single interface presents the whole operator plane: the dashboards, the typed configuration screens, the first-run setup wizard, the audit timeline, and the event feed. It authenticates humans with a session cookie (optionally with MFA) and machines with a bearer key.

Two ways to reach the controller. Every configuration change an operator makes reaches the forwarding stack through QuantumTouch, in one of two ways. Most config screens use a **transparent proxy**: the browser makes a call, and QuantumTouch forwards it to the controller *with the operator's identity attached*, so the controller logs the change under the real operator rather than as "anonymous." A smaller set of screens use **purpose-built routes** when QuantumTouch needs to do host-level work (read live throughput straight from VPP, read the hardware sensors, tail the system journal) or wants to cache a slow answer. Section *Configuration* returns to why the split exists.

It reads three sources. To show you what the box is doing, QuantumTouch polls three places: the **controller** (sessions, config, health), **VPP directly** (live throughput, because the controller's own counters refresh too slowly to drive a five-second chart), and the **BMC** (hardware — power supplies, fans, temperatures). It never asks these sources for anything it can compute itself, and it never writes to the data plane.

One process owns operator state. QuantumTouch runs as a single management process, and that is deliberate: the live history buffer, the sticky event feed, and the audit shipper are all held by exactly one owner, which is what keeps them consistent. The practical consequence for an operator is simple — this is a single-node operator plane in this release; horizontal scale-out of the management layer is a roadmap item, not a supported configuration today.

Nothing here is on the fast path. Worth repeating because it drives every troubleshooting decision: QuantumTouch is an observer and a controller-of-config, never a forwarder. If QuantumTouch is slow, restarting, or down, subscriber traffic is completely unaffected — VPP keeps forwarding, the controller keeps running the lifecycle, and the bearer keys QuantumTouch minted keep working because the controller validates them independently.

What QuantumTouch Does

At operator altitude, QuantumTouch has eight jobs. The rest of this document takes them one at a time; this is the map.

Onboard a fresh box. A first-run setup wizard walks an operator from a bare install to a forwarding BNG — pools, plans, uplinks, access, and authentication — gated by a one-time bootstrap token for the first admin account.

Harvest data. A single data collector on a five-second heartbeat polls the controller, VPP, and the hardware, turning raw counters into rates, histograms, and health verdicts. Every number on screen has a clock behind it.

Present analytics. Real-time and historical throughput, session, and QoS-band charts, backed by an in-process time-series store and a Prometheus metrics surface for your existing monitoring stack.

Drive configuration. Typed forms over every BNG configuration resource — pools, plans, edge routers, RADIUS, captive portal, access interfaces, NIC and driver selection — each change proxied to the controller with the operator's identity attached.

Show operations dashboards. A single-screen "is anything wrong, and how loaded are we" view, plus deep health, hardware, and topology pages, all driven from one shared health poll so every widget agrees with every other.

Aggregate audit and events. Four independent audit sources merged into one searchable timeline, shipped off-box to your SIEM, and distilled into a sticky operator event feed that keeps a recovered fault visible until someone acknowledges it.

Protect configuration. Point-in-time snapshots across five configuration domains, content-hashed for change detection, with structured diff, a locked restore, and signed export/import for provenance.

Authenticate humans and machines. Session cookies with MFA for operators, bearer keys with roles for integrations — the surfaces and storage live in QuantumTouch, while the enforcement is delegated to the platform and the controller.

Onboarding a Box: The Setup Wizard

A freshly installed box has a forwarding stack but no configuration and no operator accounts. The setup wizard turns it into a running BNG.

Getting in the first time. Before any admin exists, the box cannot simply be logged into. The install prints a one-time **bootstrap token**; presenting that token unlocks a single action — creating the first administrator (a username and a password that must meet a minimum strength). Once the first admin exists, the token is spent and normal authentication takes over. This is the only way to create the first account, and it closes the window in which a fresh box might be claimed by someone else on the network.

The wizard is a guided, resumable flow. From the first admin, the wizard walks a sequence of steps:

welcome → platform → dataplane → plans → pools → auth → *[radius]* → captive → tls → review

The RADIUS step appears only if you chose a non-local authentication mode. Each step gates *Next* behind a quick validity check — a dataplane needs a distinct uplink and access interface; you need at least one plan with exactly one marked default; each pool needs a CIDR and a gateway; local auth needs a username and password. You can jump backward to revise an earlier step, and the whole draft auto-saves as you go, so a browser refresh mid-wizard resumes where you left off rather than starting over.

Applying is per-section, and partial success is a first-class outcome. The *review* step is where the wizard actually writes to the box. It applies each section in order — plans, then pools, then captive portal, then RADIUS — and records the platform, dataplane, and auth choices. Crucially, a section that fails does not abort the rest: the review screen shows a per-section ✓ / ✕ grid and only reports "Setup Complete" when every applied section succeeded. If, say, the captive-portal write fails but plans and pools succeed, you see exactly that, and you fix the one thing rather than re-running everything.

A self-heal on first boot. The BNG depends on one low-level data-plane setting — a punt path that lets DHCP from unknown subscribers reach the controller — that the base platform does not ship enabled. QuantumTouch seeds it idempotently on startup and regenerates the data-plane configuration so the next VPP restart picks it up. Without this, onboarding would silently break: DHCP Discovers would never reach the controller and no subscriber could come online. You do not have to know this setting exists; QuantumTouch keeps it in place for you, including after a configuration save that would otherwise have wiped it.

How QuantumTouch Sees the Box

Everything QuantumTouch shows you comes from polling. A single data collector — informally, the **sampler** — is the engine behind every chart, gauge, and health verdict. Understanding its rhythm explains why the numbers look the way they do.

One heartbeat, three sources. The sampler runs on a fixed **five-second tick**. On each tick it gathers the current session list and configuration from the controller, derives live throughput straight from VPP, checks a set of reachability probes, and publishes a fresh set of metrics. The whole tick is defensive: a momentarily unreachable controller or a VPP hiccup yields zeros for that tick, never a crash, and the next tick recovers.

Why throughput comes straight from VPP. The obvious source for throughput would be the controller's counters — but they refresh too slowly to drive a five-second chart. So the primary throughput harvester bypasses the controller and reads the data plane's interface and node counters directly, differencing them tick-over-tick to derive bits-per-second and packets-per-second in each direction. The direction convention matches the forwarding doc: *up* is subscriber→core, *down* is core→subscriber. The green band is delivered traffic within contract; the red band (traffic shed at the policer) is estimated from the policer's drop counter, since the data plane exposes a red *packet-drop* signal rather than a red byte count. These parsers are written to degrade gracefully — a wording change in a future VPP release makes a number go stale, not silently wrong.

Reachability probes. Alongside raw counters, each tick runs a few cheap liveness checks that feed the health verdict and the event feed:

Probe	What it checks	How
RADIUS	AAA server reachability	via the controller's health view
Uplink gateway	The path to core is up	a ping <i>originated from VPP</i> (a kernel ping would exit the wrong NIC)
Captive portal	The portal answers	an HTTP HEAD to the portal
Service journal	The BNG services are running	the host's service manager
Hardware	PSUs, fans, temperatures	the BMC, via IPMI

A deliberate touch: probes that depend on optional features stay quiet when the feature is not configured. A box with no RADIUS servers configured never alarms about RADIUS being unreachable.

A stall detector on the wire. Because the sampler is the only component that actually watches the packet counters second by second, it carries a cheap proxy for "are DHCP replies getting back out?" — it compares downstream to upstream packet rates under load, and flags a stall only when the box is busy *and* the ratio stays bad for several consecutive ticks. A brief blip or an idle box never trips it. This signal is folded into the health view.

Every number has a clock — and a cache. The guiding principle is to match each cache to its source's cost and never poll faster than new data actually appears. The result is a layered set of refresh rates you can reason about:

What you're looking at	Refresh cadence
Throughput / session / QoS charts (live tier)	5 s
Health & dashboard (one shared poll)	5 s, with a 15 s staleness grace
Events feed	5 s
VPP port statistics	~7 s
Hardware / IPMI	~10 s
Version / "about" information	~30 s
Historical charts (week / month views)	up to 60 s

Two things the polling layer deliberately does *not* do, because operators ask: there is no WebSocket push — five-second polling is the model throughout this release — and most dashboard loops keep polling even on a hidden browser tab (the historical charts are the exception; they pause when not visible).

Analytics & Observability

This is how the evidence of what the box did becomes charts you can read. There are two audiences for the same underlying data: the QuantumTouch UI, and your own monitoring stack.

A dual-resolution history, in memory. QuantumTouch keeps a rolling time-series history at two resolutions: a **fast** buffer (five-second samples, about one hour deep) and a **slow** buffer (five-minute samples, about thirty days deep). A chart request picks the resolution automatically from the span you asked for — anything wider than roughly an hour reads the slow buffer, anything narrower reads the fast one. The history is periodically written to disk, so a QuantumTouch restart replays the last hour rather than starting blank. Per-subscriber second-by-second history is kept only in the fast buffer and only for a few hours — a month-old subscriber's per-second trace is not interesting, and keeping it would waste memory to no purpose.

What the charts show.

- **Throughput** — two stacked-area charts, upstream and downstream, with bands stacked green → yellow → red and a shared vertical scale across both directions (so a busy upstream doesn't visually dwarf a quiet downstream). Packets-per-second rides as a dashed overlay on its own axis.
- **QoS bands** — a live proportion bar of green / yellow / red for the whole box this second, read straight from the current counters rather than from history.
- **Sessions over time** — connected, captive, and offering counts against a moving time window, with a reference line at the Community-Edition session ceiling.
- **Per-session traffic** — a drawer, per subscriber, showing that subscriber's green + yellow + red history up and down.

One-tick lag, by design. The whole chain — the sampler tick, the metrics scrape interval, the chart's own step, and the browser's poll — is pinned to the same five seconds. The point is that the only lag you ever see is *one tick*, not the sum of four independent clocks drifting against each other. An honest consequence: across an outage the charts show a real *gap*, not a flat line stretched over missing data, because the query refuses to carry a stale sample forward across a hole.

A Prometheus surface for your stack. QuantumTouch also publishes roughly 65 metric series in Prometheus format at `/metrics` (throughput, QoS bytes and packets, sessions by state, RADIUS reachability, reply-path signals, port counters). A loopback Prometheus scrapes it with a path-scoped, read-only scraper key, so you can drive Grafana dashboards and Alertmanager rules from the same data the UI shows. Cardinality is kept deliberately aggregate-only — about five megabytes at thirty-day retention — with per-subscriber detail staying in the in-memory history and the controller's on-demand endpoints rather than inflating the metric count. Counter families are reset-aware: when a source counter resets (a controller restart, say), the metric re-baselines and bumps a visible "baseline reset" counter rather than emitting a nonsensical negative jump, so a reset shows up as an annotation in Grafana instead of a spike.

A caveat worth knowing. The historical read path that ships today is the in-memory time-series store described above — it *is* the analytics database in this release. A Prometheus-first model, in which Prometheus becomes the historical store and the in-memory buffer becomes a fallback (with a UI badge telling you which source a chart is reading), is designed but not yet built. Two metric families (`bng_uplink_*` and `bng_access_*`) are registered but report zero until a port-role map is populated, and the reply-path-stall gauge reads zero for the same wiring reason even though the detector itself works and feeds the health view. These are listed in *Scope and Caveats*.

The Life of an Operator Action

The companion document walks the life of a subscriber packet through the forwarding path. This is its operator-plane mirror: the concrete flow behind the things you actually do and see. Each of these ties the pieces above together.

A counter becomes a chart. Subscriber traffic increments the data plane's policing counters. On its next five-second tick the sampler reads the interface and node counters from VPP, differences them against the previous tick to get per-band bits-per-second, and appends the result to the fast history (and, every twelfth tick, to the slow history). It also publishes the value as a Prometheus gauge. When your browser's throughput panel polls, it reads that recent slice of history and stacks the bands into the area chart. Because every stage is pinned to the same five seconds, the chart is one tick behind the wire — no more.

An operator edits a plan. You open a plan, change its committed or peak rate, and save. QuantumTouch sends the change to the controller through the proxy, with your identity attached; the controller re-rates every connected subscriber on that plan in place — no session drop, no re-DHCP. If you also changed which plan is the default, that one fact is saved on the QuantumTouch side (the controller's plan list doesn't carry a "default" marker). The plan editor does two operator- friendly things before you commit: it tells you *exactly how many connected sessions a save will re-rate*, and if you try to delete a plan that live sessions still reference, it pre-warns you of the controller's refusal rather than letting the delete fail cryptically.

A service goes unreachable. A probe finds the RADIUS auth server down. QuantumTouch synthesizes a critical event, adds it to the feed as *active*, the dashboard renders it red, and the health verdict drops to *Fault*. When the server later recovers, the event is not simply deleted — it flips to *recovered*, its summary is rewritten "RESOLVED — ...", and it sorts to the top of the feed so you can see that something broke and healed. It stays visible until you acknowledge it. A flapping service re-fires onto the *same* incident rather than generating a storm of near-identical events.

A snapshot and a restore. You capture a snapshot; QuantumTouch reads all five configuration domains, content-hashes the result, and stores it. Later you restore it: QuantumTouch takes an exclusive lock (so two restores can't collide), writes the data-plane startup config, restarts VPP only if that config actually changed, pushes the controller configuration back in a safe order, rewrites the host-level settings, and reports whether a reboot is now required. The result is explicit — fully applied, partially applied, or "needs a deeper reset" — never a silent half-restore.

An audit query. You filter the audit timeline by time, source, or text. QuantumTouch fans out to the four audit sources in parallel, each with its own timeout, merges the results into one time-ordered list, and returns it. One slow or failing source degrades to a small **X** marker on that source; it does not fail the whole query. Every merged row gets a stable, deterministic ID, so a dashboard event can deep-link straight to its exact audit row.

Operations Dashboards & Health

This is the section to keep by the NOC screen. Everything here is available in the UI and, where noted, as Prometheus gauges your Alertmanager can page on.

One poll, one truth. Every health-consuming widget — the dashboard, the health page, the top-bar status pills, the topology view, the hero verdict — reads from a **single shared health poll**, not its own fetch. This exists because an earlier per-widget model let loops diverge: a top-bar pill could show green while the dashboard showed the box down. Now they cannot disagree. The shared poll refreshes every five seconds and holds the last good result through a **fifteen-second grace window**: on a transient error it keeps showing the last snapshot, and only after fifteen seconds of failure does it blank the data and flag itself stale — so a single missed poll never flickers the whole dashboard red.

The one-glance verdict. A single **Healthy / Degraded / Fault** verdict is the first thing an operator reads, and its rule is deliberately *monotonic under events* — events can only make the verdict worse, never better:

- The base comes from the BNG's own health: healthy, degraded, or fault.
- Any active critical event forces **Fault**; any active events at all pull a Healthy base down to **Degraded**.
- The verdict is accompanied by a plain-English reason sentence naming the concrete problems ("Secondary RADIUS auth unreachable; 2 active events, 1 critical") — because reading *why* beats reading *what*.

When the controller itself is unreachable, the health view still returns a real "fault" state rather than spinning, so the pill renders a definite answer instead of an ambiguous loading state.

What the dashboards show.

- **KPI row** — provisioned / captive / in-flight session counts, plus RADIUS and VPP status pills. An **orphan reconciliation** flags any sessions the controller counts but the lifecycle state machine hasn't accounted for — a quiet correctness check on the session table.
- **Pool pressure** — per-pool utilisation, colored and sorted by the controller's own ok / warn / full classification, with allocation failures called out in red.
- **Gauges** — radial rings for CPU, memory, workers, disk, and RX/TX, each with its own warning and critical thresholds; the network gauges fill as a percentage of configured link speed.
- **Workers & hardware** — a per-worker thread strip (pinned core, utilisation, pps), a hugepage breakdown, and hardware cards for power supplies, fans, and temperatures from the BMC.
- **Topology** — a live CPE → edge → BNG → core diagram whose flow animation speeds up with delivered throughput. A neat outage tell: when the controller stalls, the "as-of" timestamp stops advancing and the flow dots visibly freeze, surfacing the problem without a separate indicator.

Audit, Events & Compliance

Audit and events are two different things built from overlapping data. Audit is the *forensic record* of what happened; the event feed is the *live operational state* of what is wrong right now. Keeping them distinct is worth the paragraph.

A unified audit timeline from four sources. QuantumTouch assembles one searchable timeline by merging four independent sources on demand:

Source	What it carries
QuantumTouch events	Configuration changes, exports, and lifecycle events QuantumTouch itself stamps
Controller log	The forwarding stack's own record of REST changes
System journal	The host's service logs for the BNG units
Portal access	The captive-portal access log (opt-in)

The four are queried in parallel, filtered by source / actor / text, sorted newest-first with a stable tiebreak, and truncated to your limit. One failing source degrades to a marker, never a failed query. Every row gets a deterministic ID derived from its content, so the same audit entry always has the same link — a dashboard event can point straight at it.

Export, defensively. You can export the merged timeline (CSV or JSON) or pull a larger bulk export spanning up to a year. Both defend against spreadsheet formula injection — a cell that begins with a character a spreadsheet would treat as a formula is neutralised — and every bulk export is itself audited, so the act of exporting is on the record.

Off-box shipping. For long-term retention and SIEM integration, QuantumTouch can ship its event log off-box to **syslog** (RFC 5424, with RFC 6587 framing over TCP/TLS), **HTTP**, or **S3**. The shipping model is built for reliability under stress:

- **At-least-once delivery** — each destination has its own spool; a batch is removed only after the destination confirms it, and a failed send is retried with backoff and stays spooled until it succeeds.
- **Drop-oldest back-pressure** — if a destination stays down and its spool fills its cap, the *oldest* records are dropped first (and counted), so the newest data still gets off-box. Log shipping prefers losing old data to losing new.
- **Isolation** — a slow destination cannot stall the others; each ships independently.
- Destination secrets (an HTTP bearer, an S3 key) are encrypted at rest.

The live event feed. Events are not stored as they happen — they are *synthesized* from current state on every read, then reconciled against a sticky history so a recovered fault stays visible until an operator clears it. The lifecycle is small and worth internalising:

Transition	Meaning
new fault appears	added as active
still active	refreshed in place
was recovered, fault returns	revived as one incident — not a new event per flap
active, fault clears	flipped to recovered , resolved-time stamped
recovered, still gone	stays until an operator acknowledges it

Acknowledgement is separate from recovery: acknowledging removes an event from the feed, but a still-active condition will simply re-appear on the next read. The feed sorts recovered-but-unacknowledged items to the top and rewrites their summary with the fault's active duration, so the first thing you see is "here is what broke and how long it lasted, clear it when you've noted it."

Configuration: Surfaces & Protection

Everything the controller owns has a typed screen in QuantumTouch, and everything QuantumTouch changes is protected by snapshots. This section covers both — how config gets written, and how you get it back.

Typed screens over every resource. Pools, subscriber plans, edge routers, RADIUS, captive portal, access interfaces, and NIC/driver selection each have a purpose-built form. Writable resources go to the controller through the proxy with your identity attached; read-only inventories (access interfaces, driver lists) are cached briefly to protect the box from dashboard poll storms. A few patterns recur across the screens and are worth knowing:

- **Live re-rate awareness.** The plan editor names how many connected sessions a save will re-rate, and pre-warns the refusal you'll get if you try to delete a plan live sessions still use.
- **No-clobber drafts.** Pools, the plan editor, and edge routers seed their edit form once from the first poll and then leave it alone, so a five-second background refresh never overwrites what you're in the middle of typing.
- **Wire units vs. display units.** Rates are edited in kbps at the wire level; the tables and wizard show Mbps for readability. The value you commit is the wire value.

A local correction for a controller counting bug. In this release the controller mis-counts pool occupancy on certain prefix sizes (an integer-overflow bug that can report a wildly inflated pool total). Rather than display nonsense, QuantumTouch recomputes pool occupancy locally from the true host count and the live session addresses, so the dashboard's pool-utilisation bars are correct regardless of the controller's arithmetic. You do not need to work around this; it is already handled.

RADIUS is read-only in this build. The RADIUS screen shows the current configuration, but writing it back through the API is not yet supported by the controller. Instead of pretending a save happened, a save opens a fallback modal showing the exact configuration diff to hand-apply — an honest surface over a known API gap, revisited in *Scope and Caveats*.

Snapshots: the defense against a config reset. The controller's configuration database is reset on every package upgrade, so QuantumTouch keeps its own point-in-time snapshots of the whole box's configuration, across five domains:

Domain	What it captures
QuantumTouch settings	The operator-plane configuration
Controller config	Pools, plans, RADIUS, captive, uplinks, DNS
VPP startup config	The data-plane boot configuration
GRUB command line	Kernel boot parameters
Sysctl / hugepages	Host memory tuning

Each snapshot is **content-hashed** over what it captured (not its name), so renaming it doesn't change its identity, two identical configs share a hash, and a single changed byte flips it — and QuantumTouch re-verifies the hash on read, flagging any mismatch as possible corruption or tampering. You can **diff** live config against a snapshot, or one snapshot against another — a structured field- by-field diff for the settings domains and a text diff for the file domains — and the diff tells you whether applying it would require a reboot.

Restore is careful. A restore runs under an exclusive lock (a second restore attempt gets a clear "held by ..." refusal), applies the domains in a safe order, restarts the data plane only if its config actually changed, and reports its outcome explicitly — fully applied, partial, or needs-a-deeper-reset. A dry-run takes no lock and changes nothing, reporting what *would* happen step by step. Exports are **signed** with a box-local key: importing a snapshot tells you whether it came from this box (verified), another box (a warning), or an unsigned legacy export. An auto-prune keeps the snapshot set bounded but never touches one you've tagged **golden**.

Security Posture

QuantumTouch surfaces and stores; the platform and the controller enforce. This section is the operator's view of the resulting posture.

Ports and exposure.

Port	Service	Exposure	Auth
8443/ tcp	QuantumTouch web UI	all interfaces	Session cookie (with optional MFA) or bearer key
3000/ tcp	BNG controller	all interfaces	Bearer; intended to sit behind QuantumTouch's HTTPS in production
9090/ tcp	Prometheus	loopback only	Bearer (scraper key, scoped to <code>/metrics</code>)
8088/ tcp	Captive portal demo	all interfaces	none (demo, plaintext)

Only `:8443` (and the host's own SSH) is legitimate external attack surface; the controller's `:3000` should be firewalled to localhost and trusted operators, with QuantumTouch's HTTPS as the intended front door.

Two audiences, both delegated. Human operators authenticate with a session cookie backed by strong password hashing, optional TOTP MFA, progressive lockout, and — for larger shops — SAML SSO and LDAP/AD with group-to-role mapping. Machine callers use **bearer keys**. Both are delegated to the platform and controller for enforcement; QuantumTouch owns the surfaces and the key lifecycle.

Three roles, and a reveal-once key model. Every key carries exactly one of three roles — read-only, read-write, admin — with read-write implying read-only. An operator with the admin role can mint, reveal-once, revoke, rotate, and delete keys. A minted key's secret is shown **exactly once**, behind an "I've copied this" gate; only a hash is stored, so a lost key is rotated, never recovered. Revoking keeps the entry for audit; deleting removes it. Optional per-key restrictions (an IP allow-list, a path scope, an expiry) can be attached. The UI disables controls you lack the role for, but that is a convenience — the server-side role check is the real gate.

Secrets and TLS. Secrets at rest — MFA seeds, an LDAP bind password, shipping credentials — are encrypted; operator passwords and bearer keys are hashed, not reversible; snapshot exports are signed. TLS is fully managed from the UI: import a PEM certificate, keep a previous-cert history with one-click rollback, or issue via in-product ACME, and set a protocol policy (minimum version, cipher policy, HSTS) applied to the front door. Finally, a set of security webhooks (a burst of failed logins, an admin role granted, a certificate expiring, an audit export, and more) fire with a signed body so your security tooling can react in real time.

Failure & Recovery

The forwarding stack's failure modes are in the companion document. These are QuantumTouch's own, and the guiding principle is **fail-open observability**: the operator plane degrades to "I can't see X right now," never to "X is down."

If this happens	QuantumTouch does this	Subscriber impact
Controller unreachable	Harvesters return zeros; health shows a real fault state; a critical event is raised	None — forwarding is independent
A bad sampler tick	The tick is swallowed; the next one recovers; the last hour of history is replayed on restart	None
Prometheus outage	Charts fall back to the in-memory history (the primary store today anyway); the scrape target simply stops advancing	None
QuantumTouch restarts	Brief 502s on the web UI; the controller, sessions, and VPP are untouched; the keys QuantumTouch minted keep working	None
A shipping destination is down	Batches stay spooled (at-least-once), capped and dropping oldest-first; other destinations are unaffected	None (audit only)
A state file is corrupted	Corruption drops to an empty state for that feature rather than failing the box; every state file is written atomically, so a torn write is impossible	None

The through-line: because QuantumTouch is never on the forwarding path, its worst failure is a temporary loss of visibility or management, not a loss of service. Its durable state — operator accounts, audit history, snapshots — lives in a directory deliberately preserved across reinstalls, so a package upgrade that resets the controller's config database leaves your operator identities and history intact.

Scope and Caveats

Two different kinds of boundary apply, and it helps to keep them apart. **Scope** decisions are deliberate choices about what the operator plane *is* — they are stable and will not change from release to release. **Caveats** are specific to this Community-Edition Alpha build — they describe where the running software stops short of the full design today, and each has a defined path to being filled.

Scope — product-line decisions. These are intentional edges, not gaps to be closed:

- **Single-node operator plane.** One QuantumTouch process owns the operator state for one box. Horizontally scaling the *management* layer (multiple workers sharing state) is out of scope for this line; the forwarding stack's own horizontal scale is unaffected and covered in the companion document.
- **Poll, don't push.** QuantumTouch exposes state your systems read on their own schedule (REST, and Prometheus at `/metrics`); it does not push to a NMS. The five-second polling model is the v1 design throughout — there is no WebSocket push.
- **Aggregate-only metrics cardinality.** Per-subscriber detail stays in the in-memory history and the controller's on-demand endpoints; the Prometheus surface is kept aggregate to bound cardinality by design.
- **One plugin per deploy.** QuantumTouch is composed into the management shell as the single plugin for a BNG deployment; multi-plugin coexistence in one instance is not a supported shape.

Caveats — specific to this release. Each describes where this build stops short of the full design, with a defined path forward:

- **Analytics store is the in-memory history, not Prometheus (yet).** The shipping historical read path is the in-process time-series store. The Prometheus-first model — Prometheus as the store, the in-memory buffer as a fallback, with a per-chart source badge — is designed but not built.
- **Two metric families report zero.** The per-port `bng_uplink_*` and `bng_access_*` families are registered but emit zero until a port-role map is populated; the reply-path-stall *gauge* reads zero for the same wiring reason, though the detector itself works and feeds the health verdict.
- **RADIUS / captive / pool config writes from the UI are pending.** These read cleanly today; writing them back awaits a controller configuration-write surface. Until then the UI shows the exact diff to hand-apply rather than faking the write.
- **In-product ACME issuance is partial.** The configure-and-validate flow ships; the automated DNS-01 publish/poll/finalize path is a later item.

None of these caveats is a gap in the *running* operator plane so much as the seam between a focused Alpha and the full product design: a single-pane, deeply- observable, audit-complete operator plane that lets a small or mid-size ISP stand up a BlueDot BNG on commodity hardware, watch it in real time, protect its configuration, and integrate it into an existing monitoring and SIEM stack — today, standalone, with a clear path to the production build as each seam closes.

Prepared from the QuantumTouch source tree ([ipoe/quantumtouch-bng](#) , [main](#)) as the operator-plane companion to the Quantum IPoE BNG Theory of Operations. Field-level request and response shapes are authoritative in the generated OpenAPI spec and the source, which change faster than this document. © 2026 BlueDot Insight LLC. BlueDot Proprietary.