

BNG Telemetry API

Theory of Operations — Raw Telemetry Interface

The BlueDot BNG exposes its subscriber, traffic, and health telemetry through a single stable REST interface. This document presents that interface as raw telemetry: the endpoints, the exact fields they return, and the semantics an integrator must observe to turn them into information. It is the same API that QuantumTouch consumes — QuantumTouch holds no privileged access — so an ISP can use these identical endpoints to integrate the BNG with their own NMS or to build bespoke analytics.

Document Type	BlueDot Theory of Operations
Component	Quantum BNG — Controller REST API
Audience	Integrators — NMS / Analytics / OSS-BSS
Source Tree	<code>ipoe/bng</code> @ <code>v1.2.0</code> (<code>main</code>)
Interface	Northbound REST, base path <code>/bng</code>

Classification **BlueDot Proprietary**

Date July 2026

Contents

Purpose and Framing	4
The Data-Plane-to-API Path	5
Access Model and Conventions	6
Traffic and QoS Counters	7
Session Inventory and Lifecycle	9
Health and Capacity	11
Address-Pool Occupancy	12
Configuration Read-Back	13
Counter Semantics and the Two-Sample Rate Pattern	14
From Raw Telemetry to Information	15
Boundaries of the Telemetry API	16
Summary	17

Purpose and Framing

This document describes the BlueDot BNG's telemetry interface as exactly what it is: a **raw telemetry API**. The BNG captures an unusually broad and high-fidelity record of what is happening inside the box — every subscriber's traffic by QoS colour and direction, the distribution of sessions across their lifecycle, address-pool occupancy, DHCP protocol health, and the controller's own operational timing — and it publishes all of it through a single, stable, pull-oriented REST surface under the base path `/bng`. What the API deliberately does *not* do is analyse that data. It computes no trends, no key performance indicators, no forecasts, and no alert conditions. Those are the concern of a separate layer built on top of the interface.

The most important fact for anyone integrating with the BNG is that this raw telemetry interface is **the same interface that QuantumTouch uses**, and QuantumTouch has no privileged or private access path.

QuantumTouch — BlueDot's own operational "view" into the BNG — authenticates as an ordinary API client and reads the endpoints documented here, exactly as any third party would. There is no back channel, no hidden RPC, and no internal-only telemetry feed. The practical consequence is parity: an ISP that wants to integrate the BNG into its existing Network Management System (NMS), feed a data lake, or build custom subscriber analytics has access to precisely the same data QuantumTouch does, through precisely the same endpoints. This document is written to make that integration straightforward.

The framing throughout is a deliberate separation of two concerns. The BNG's job is to **harvest** raw operational data from the VPP data plane and the controller and expose it faithfully. The job of turning that raw data into **information** — a utilisation trend, a capacity forecast, an upsell candidate list, a health alarm — belongs to the consumer, whether that consumer is QuantumTouch or an ISP's own tooling. Keeping the two separate is what makes the interface stable and reusable: the BNG never couples itself to any particular analytics model or downstream schema, and every consumer is free to compute whatever its use case requires from the same raw operands.

The Data-Plane-to-API Path

Understanding where the numbers come from clarifies what the API can and cannot offer. The telemetry has two origins, joined at the controller and published as one REST surface.

The first origin is the **VPP data plane**. On the fast path, the IPoE plugin maintains a per-subscriber counter block — twelve 64-bit counters covering the cross-product of three policer colours (green, yellow, red), two directions (upstream and downstream), and two units (octets and packets) — plus a set of six aggregate band counters summed across all subscribers, and the plugin's compiled session-cap state. These live in the data plane, sharded per worker thread so that maintaining them costs nothing on the forwarding path. The controller pulls them across the `vpp_bridge` gRPC boundary on demand, through purpose-built calls (`BngCountersGet` for one session, `BngCountersGetBulk` for every session in one reply, `BngAggregateCountersGet` for the node-wide bands, and `BngCapStatus` for the cap), and re-exposes them as REST.

The second origin is the **controller** itself. It holds the authoritative session records (state, identity, plan, QoS, timers, accounting snapshot), the monotonic DHCP protocol counters, the address-pool occupancy computed live from the session table, the composite health signals, and its own persistence-path timing. None of this requires a data-plane round trip.

Two properties of this path shape every integration built on it. First, the interface is **pull only** — a consumer polls; the BNG never pushes, streams, or opens a webhook. Second, the BNG stores **no history**: every value returned is either *current state* (how many sessions are connected right now, how full a pool is right now) or *cumulative-since-start* (total octets a subscriber has passed since their session began). The API is a window onto the present and the running totals, not a time-series database. Deriving rates, trends, and history is therefore inherently the consumer's responsibility — the consumer samples the window repeatedly and stores what it needs.

Access Model and Conventions

Every endpoint in this document lives under the base path `/bng` and is served by the BNG controller over the management interface. A small number of conventions apply uniformly and are worth stating once.

Authentication and authorisation. Every call carries an API key. Endpoints are classified by role: read endpoints require a `read_only` key, and write endpoints require a `read_write` key. For a telemetry or NMS-harvesting integration this distinction is decisive and convenient — **the entire telemetry and configuration-read-back surface is `read_only`**, so a read-only key polls everything in this document. A read-write key is needed only if the integration also intends to *act* on the BNG (tear down a session, edit a plan), which is outside the scope of raw telemetry.

Connection gating. Endpoints that read live data-plane counters (`/bng/counters` and `/bng/counters/{ip}`) additionally require the VPP data plane to be connected and return **503 Service Unavailable** when it is not. Pure controller-state reads (`/bng/sessions` , `/bng/health` , `/bng/pools` , and the configuration read-backs) remain available even if the data plane is momentarily disconnected — `/bng/health` in particular is designed to report *that* VPP is down rather than fail.

Value encoding. Counter values that originate as unsigned 64-bit integers are serialised as **decimal strings**, not JSON numbers, so that byte counts remain exact beyond JavaScript's 2^{53} safe-integer limit (they stay precise into the petabyte range). Timestamps on session records (`started_at` , `session_expires_at` , `idle_expires_at`) are **milliseconds since the Unix epoch** as numbers. Telemetry snapshots carry an `as_of` field — an ISO-8601 timestamp marking when the snapshot was taken — which is the anchor for any rate calculation.

Polling cadence. The two counter endpoints are pure reads, deliberately kept off the controller's persistence/write path, and are therefore safe to poll at high frequency without perturbing session setup. The state reads (`/bng/sessions` , `/bng/health` , `/bng/pools`) are inexpensive but reflect the controller's live session map; a polling interval on the order of seconds is appropriate for dashboards, with counter endpoints polled as fast as the consumer's resolution requires.

Traffic and QoS Counters

This is the core of the telemetry surface and the data that most distinguishes the BlueDot BNG: per-subscriber, per-colour, per-direction, byte-and-packet traffic accounting. Two endpoints expose it — one node-wide, one per subscriber.

GET /bng/counters returns a node-wide snapshot: the aggregate QoS bands (summed across every subscriber) plus DHCP protocol counters, a session-state tally, and per-pool occupancy gauges. It is the single richest situational-awareness read on the box, answering "what is this node doing right now, by QoS band" in one call without iterating the session table.

```
{
  "as_of": "2026-07-13T12:00:00.000Z",
  "dhcp": {
    "discovers_received": 18342,
    "offers_emitted": 18201,
    "requests_received": 18150,
    "acks_emitted": 18150,
    "naks_emitted": 0
  },
  "sessions": {
    "total": 8421,
    "by_state": { "CONNECTED": 8102, "CAPTIVE": 274, "OFFERING": 45 }
  },
  "pools": {
    "subs-pool": { "size": 16382, "in_use": 8421, "utilization": 0.5140, "alloc_failures": "0" }
  },
  "up": {
    "green": { "packets": "480223311", "bytes": "512338290122" },
    "yellow": { "packets": "10233", "bytes": "14882201" },
    "red": { "packets": "88123", "bytes": "132184500" }
  },
  "down": {
    "green": { "packets": "990122344", "bytes": "1208334982001" },
    "yellow": { "packets": "22110", "bytes": "31200418" },
    "red": { "packets": "140233", "bytes": "210344100" }
  }
}
```

GET /bng/counters/{ip} returns the same colour-and-direction band structure for a single subscriber, identified by framed IP, read live from the data plane. It returns **404** if no session exists for that IP.

```
{
  "framed_ip": "100.64.12.30",
  "as_of": "2026-07-13T12:00:00.000Z",
  "up": {
    "green": { "packets": "88213", "bytes": "91002381" },
    "yellow": { "packets": "12", "bytes": "14880" },
    "red": { "packets": "3", "bytes": "4020" } },
  "down": {
    "green": { "packets": "140233", "bytes": "180344100" },
    "yellow": { "packets": "40", "bytes": "52200" },
    "red": { "packets": "9", "bytes": "12800" } }
}
```

The meaning of the three colours is the same in both endpoints and is the key to interpreting the data.

Green is traffic that conformed to the subscriber's committed information rate (CIR) — in-contract, forwarded. **Yellow** is burst traffic above the CIR but within the peak information rate (PIR) — also forwarded, but eligible to be shed first under downstream congestion. **Red** exceeded the PIR and was **dropped** by the policer. Because the counters preserve this distinction per subscriber and per direction, a consumer can see not merely how much a subscriber used, but how much was in-contract, how much was burst, and how much was discarded for exceeding their rate — a level of per-subscriber QoS visibility that coarser BNG telemetry does not provide. The packet counts alongside the octet counts expose traffic *shape*: a subscriber whose packet-to-byte ratio is unusually high is doing something structurally different (small-packet floods, scanning, VoIP) than one moving the same volume in large frames.

Session Inventory and Lifecycle

Where the counter endpoints describe traffic, the session endpoints describe *who* is on the box and *where they are* in their lifecycle. Two endpoints expose the session table.

GET /bng/sessions lists active sessions as compact summaries, wrapped with a total. An optional `?state=OFFERING|CONNECTED|CAPTIVE` query parameter filters by FSM state — useful for polling just the captive population, for instance.

```
{
  "sessions": [
    {
      "framed_ip":      "100.64.12.30",
      "mac":            "aa:bb:cc:dd:ee:01",
      "state":          "CONNECTED",
      "plan_name":      "Silver-20M",
      "giaddr":         "100.64.12.1",
      "started_at":     1752403200000,
      "session_expires_at": 1752489600000
    }
  ],
  "total": 1
}
```

GET /bng/sessions/{ip} returns the full record for one subscriber, including identity, applied QoS, and the accounting-cadence counter snapshot. It returns **404** if no session exists for the IP.

```
{
  "framed_ip":      "100.64.12.30",
  "mac":            "aa:bb:cc:dd:ee:01",
  "state":          "CONNECTED",
  "plan_name":      "Silver-20M",
  "giaddr":         "100.64.12.1",
  "circuit_id":     "0x0106000400020001",
  "remote_id":      "0x020600aabbccdde01",
  "acct_session_id": "9f8b2c14-6e3a-4a7d-9f2b-1c0e4d5a6b7c",
  "started_at":     1752403200000,
  "session_expires_at": 1752489600000,
  "idle_expires_at": 1752405000000,
  "qos": {
    "dn_cir_bps": 250000, "dn_pir_bps": 300000, "dn_cbs_byte": 256000, "dn_pbs_byte": 384000,
    "up_cir_bps": 125000, "up_pir_bps": 150000, "up_cbs_byte": 128000, "up_pbs_byte": 192000
  },
  "last_counters": { "green_in_octets": "91002381", "green_out_octets": "180344100", "...": "..." }
}
```

Two fields deserve emphasis for an integrator. `acct_session_id` is a stable UUID that identifies this accounting session; it is the correct key for correlating a subscriber across a reconnect, because the framed IP can be reused while the accounting session is distinct. `last_counters` is the counter snapshot as of the last accounting-interim cycle — it lags the live figures from `GET /bng/counters/{ip}` by up to one interim interval. An integrator should choose deliberately between the two: for billing reconciliation, the accounting snapshot on the session record is the aligned figure; for a real-time dashboard, the live counter endpoint is current. The `circuit_id` and `remote_id` are the DHCP Option-82 line identity (hex-encoded), which an NMS can map back to a physical port or subscriber line.

Health and Capacity

`GET /bng/health` composes several independently captured signals into one document and reduces them to a single status verdict. It is the natural endpoint for an NMS to poll for up/down and capacity alarms.

```
{
  "status": "ok",
  "vpp": { "connected": true },
  "radius": { "auth": { "reachable": true }, "acct": { "reachable": true } },
  "sessions": {
    "connected": 8102,
    "captive": 274,
    "offering": 45,
    "total": 8421,
    "onboard_refused": 0,
    "max": 100000,
    "at_cap": false
  },
  "pools": {
    "subs-pool": { "utilization": 0.5140, "alloc_failures": "0", "pressure": "ok" }
  },
  "persistence": {
    "save_ms": { "last": 6, "max": 41 },
    "write_lock_wait_ms": { "last": 0, "max": 12 }
  }
}
```

The `status` field rolls up to `ok`, `degraded`, or `fault` by a deliberate rule: `fault` when VPP is disconnected or both RADIUS endpoints are unreachable, `degraded` when exactly one RADIUS endpoint is unreachable, and pool pressure is reported but intentionally does *not* gate the status — a deliberately full pool is an operational condition to act on, not a service outage. The `sessions` block carries the capacity-edge signals an operator needs: `onboard_refused` is a monotonic count of onboards the data plane rejected (which climbs once a capped box is full), and `max` / `at_cap` report the plugin's effective session cap and whether the box is currently refusing new sessions. When VPP is down the cap is unknown and `max` / `at_cap` are omitted rather than guessed. The `persistence` gauges expose the controller's own write-path timing — the early-warning signal for the database-bound write ceiling that ultimately bounds sessions per node.

Address-Pool Occupancy

`GET /bng/pools` reports per-pool IPAM occupancy, computed live from the session table rather than a stored side-count that could drift.

```
{
  "pools": {
    "subs-pool": { "range": "100.64.0.0/18", "leasetime": 86400, "total": 16382, "allocated": 8421, "free": 7961 }
  }
}
```

Pool occupancy appears in three places with three shapes, all reflecting the same underlying truth:

`GET /bng/pools` gives `total / allocated / free` (the IPAM view), `GET /bng/counters` gives `size / in_use / utilization / alloc_failures` (the telemetry gauge), and `GET /bng/health` gives `utilization / alloc_failures / pressure` (the alarm view). An integrator can use whichever is most convenient for its purpose; they are consistent snapshots of the same pool state.

Configuration Read-Back

Beyond the live telemetry, a set of read-only configuration endpoints let an integrator resolve the identifiers that appear in the telemetry into meaningful reference data. These are the lookups that turn a `plan_name` into a rate tier, a `giaddr` into an edge router, or a pool name into a CIDR range:

- `GET /bng/config` — the full API-managed configuration as one document.
- `GET /bng/plans` — service rate tiers (CIR/PIR/CBS/PBS per plan).
- `GET /bng/edge_routers` — the `giaddr → gateway` mapping.
- `GET /bng/dns` — the DHCP option-6 DNS server list.
- `GET /bng/captive_portal` — the configured captive-portal IP.
- `GET /bng/session` — session-subsystem tunables (`max_recovery_age` , `offer_timeout` , `interim_default`).
- `GET /bng/access` and `GET /bng/uplinks` — the enabled data-plane interfaces.
- `GET /bng/aaa/local` and `GET /bng/aaa/local/subscribers` — the local-AAA policy and per-line subscriber records.

All are `read_only` , so the same read-only key that harvests telemetry also reads this reference data.

Counter Semantics and the Two-Sample Rate Pattern

Because the BNG returns cumulative and current values rather than rates, an integrator computes rates itself, and a handful of semantic rules must be observed to do so correctly. These are the same rules QuantumTouch obeys.

The aggregate bands and the DHCP counters are **monotonic since controller process start**. To obtain a rate, a consumer takes two samples and divides the difference in a counter by the difference in the `as_of` timestamps. A counter that *decreases* between two samples signals a controller restart, not negative traffic, and should be treated as a reset (the running baseline restarts from the new, lower value). Per-session counters are **cumulative since that session started** and begin again at zero when a subscriber reconnects under a new `acct_session_id` — which is why correlation across a reconnect must key on `acct_session_id`, not on the framed IP alone.

For volume and throughput, only **green plus yellow** represents traffic the subscriber actually received; red was dropped by the policer. A usage or billing computation sums green and yellow; the red band is a throttle-and-health signal, not consumption. Converting octets to a bit rate is the usual factor of eight.

As a concrete pattern, downstream committed throughput over a sampling interval is:

```
rate_bps(down, green) =
  ( bytes(down.green, t2) - bytes(down.green, t1) ) * 8
  / ( seconds(as_of_t2) - seconds(as_of_t1) )

# and the subscriber's *delivered* rate adds the yellow (burst) band:
delivered_bps(down) =
  ( (down.green.bytes + down.yellow.bytes) @ t2
    - (down.green.bytes + down.yellow.bytes) @ t1 ) * 8
  / ( seconds(as_of_t2) - seconds(as_of_t1) )
```

The same two-sample method applied to `sessions.by_state` yields lifecycle rates (onboarding throughput, captive-to-connected conversion), and applied to `pools.allocated` yields an address-consumption rate from which a time-to-exhaustion forecast follows.

From Raw Telemetry to Information

Everything above is inert until a layer on top derives meaning from it. That layer is where QuantumTouch adds its value, and it is exactly the layer an ISP builds when integrating the BNG with its own NMS. The BNG ships the raw operands; the consumer computes the information. The following are representative transforms — the raw signal on the left comes from the endpoints in this document; the information on the right is the consumer's to compute:

Raw signal (from the API)	Information the consumer derives
<code>up / down</code> green/yellow/red bytes, sampled over time	Per-subscriber and node throughput; in-contract vs burst ratio; red-drop rate → chronically throttled subscribers (upsell candidates); congestion detection
<code>sessions.by_state</code> , sampled over time	Onboarding funnel and captive→connected conversion; churn and dwell-time analysis
<code>pools.allocated</code> / <code>free</code> , over time	Address-consumption rate and time-to-exhaustion forecast per pool
<code>onboard_refused</code> + <code>at_cap</code> + <code>sessions.total</code>	Node capacity trend → "provision another node by date X"
<code>persistence.save_ms</code> / <code>write_lock_wait_ms</code>	Control-plane saturation trend → the database-bound scaling ceiling
<code>dhcp.*</code> protocol counters	DHCP health; Discover→Ack conversion; retransmit/failure ratios
per-session <code>packets</code> vs <code>bytes</code>	Traffic-shape fingerprinting — scan/flood/abuse detection

The BNG intentionally ships none of these derived quantities. It ships the raw counters, records, and gauges so that the analytics layer — QuantumTouch, an ISP's NMS, or a third party's custom software — is free to compute whatever its use case demands from the same, complete set of operands.

Boundaries of the Telemetry API

Two boundaries are worth stating plainly so an integration is scoped correctly.

First, **logs and Prometheus metrics are not part of the `/bng` REST surface**. The BNG's structured JSON logs are written to standard output and are consumed through the operator's log pipeline (Filebeat, Graylog, ELK, and similar), and a Prometheus-format `GET /metrics` endpoint is provided by the underlying controller framework for scrape-based monitoring. Neither is a `/bng` REST call. An NMS integration therefore has three distinct channels available — the structured REST telemetry documented here (for exact, queryable state), the Prometheus scrape (for standard metric collection), and the log stream (for events and diagnostics) — and should use each for what it is suited to.

Second, this document covers the **read/telemetry** surface. The same API also exposes a control surface — session lifecycle actions (`POST /bng/sessions/{ip}/disconnect|activate|revoke|plan`) and configuration writes (`PUT / DELETE` on plans, pools, edge routers, DNS, captive portal, session tunables, local AAA, and interfaces), all requiring a `read_write` key and all recorded in an audit trail. A pure analytics or monitoring integration needs none of these; an integration that also *manages* the BNG would use them, and they warrant their own reference.

For request and response shapes at field-level precision, the generated OpenAPI specification (produced from the controller code via `make dev_docs`) is the authoritative source; the payloads in this document are representative and correct in structure, and are intended for orientation and design.

Summary

The BlueDot BNG presents its telemetry through one stable, pull-oriented REST interface that returns raw operational data — per-subscriber traffic by QoS colour and direction in both bytes and packets, the session table and its lifecycle distribution, address-pool occupancy, DHCP protocol health, composite health and capacity signals, and the controller's own timing — and deliberately stops there, leaving analysis to the consumer. The single most important property for an integrator is parity: QuantumTouch, BlueDot's own operational view, is built entirely on this public interface and holds no privileged access, which means an ISP can use the identical endpoints to feed its NMS, populate a data lake, or build custom subscriber analytics, with the full fidelity of the data the BNG captures. The interface hands over complete raw operands and a small set of well-defined semantics — cumulative counters, `as_of` anchors, the green/yellow/red distinction, `acct_session_id` correlation, and the two-sample rate pattern — and from those, any consumer can construct the information its operation requires.