BLUEDOT

PRIVACY DISCLOSURE

The RADIUS Exchange Timeline

What It Records About Your Subscribers

The subscriber identifiers that the `radius-exchanges` diagnostic writes into `event_log.jsonl`, where that file goes afterwards, and what the audit shipper does — and does not — send off the box. Read this before turning the capture on, and before opting a destination in to this source.

Document Type	BlueDot Privacy Disclosure
Component	QuantumBNG — `radius-exchanges` diagnostic
Audience	Operators, Deployers & Data-Protection Owners
Last Updated	23 August 2026
Classification	**BlueDot Proprietary**
Date	October 2026

© 2026 BlueDot · All rights reserved.

Contents

Read this before you turn the capture on 3
Exactly what is recorded 3
Exactly what is not recorded 4
Where the log goes 4
What the shipper does with this source 4
How to see that it is happening 5
If your deployment is subject to GDPR, CCPA or an equivalent regime 6
Is there a redaction mode? 7

What this covers: the subscriber identifiers that the `radius-exchanges` diagnostic writes into `event_log.jsonl`, where that file goes afterwards, and what the shipper does — and does not — send off the box.

Last updated: 2026-08-23

Read this before you turn the capture on

The RADIUS exchange timeline is a diagnostic. It reads capture segments off the tap's spool, checks them against the conformance register, and appends a short record per interesting exchange to the audit log. It is off unless you turn it on.

It puts subscriber identifiers in the audit log, which is what makes the timeline useful: pasting a subscriber's MAC into the search box is the whole feature. The audit log is also the file `audit_shipper` sends off the box, so those two facts used to add up to a decision the deployer had to make.

They no longer do, by default. `audit_shipper` **does not ship this source unless you opt a destination in** — see *What the shipper does with this source* below. The identifiers are on the box, in the log and on the timeline; they do not leave it on their own. This page is still the disclosure of what is recorded, and it is now also the description of where it stops.

Exactly what is recorded

Every entry the diagnostic writes carries a `session` field: the key the engine grouped the exchange under. It is one of

`key_kind`	`session` holds	On an IPoE BNG that is
`subscriber-arc`	`User-Name|NAS-Port-Id`	the subscriber's MAC address and the physical port they are on
`user-nas-port`	`User-Name|NAS-Port-Id`	the same, for an arc we could not see start to finish
`acct-session-id`	`Acct-Session-Id`	the NAS's own session handle, which your accounting records key on
`unmatched`	*(empty)*	an exchange we could not attribute to anyone

Measured over the committed capture corpus, **88% of entries carry a session key** — every `exchange`, `finding` and `unverifiable` entry does. Only the per-segment receipt does not.

An entry also carries: the RADIUS code, the segment filename, a capture timestamp, and — on a finding — the register requirement id, its tier, the RFC citation, and the engine's `observed` / `expected` prose. That prose quotes packet identifiers where a rule is about one (an Identifier, a frame number, an attribute that was present or absent).

Exactly what is *not* recorded

- **No packet bytes.** `Packet.raw` and decoded attribute values are never copied into an entry. A test over the whole capture corpus asserts that no datagram from any capture appears anywhere in the entries it produces.
- **No shared secret**, and no exception message that might have carried one: every foreign exception text is scrubbed against the secret before it is written. See `scrub_exception` in `radius/diagnostics/summarize.py`.
- **No passwords.** User-Password, CHAP-Password and EAP-Message never leave the decoder.

Be careful with the first bullet: *no packet bytes is not no subscriber data.* The session key is a subscriber identifier and it is on nearly every entry.

Where the log goes

`event_log.jsonl` is the box's append-only audit trail. Two things read it:

1. **The audit timeline** in the web UI, behind operator authentication. It sees the full identifier — that is the point of the feature.
2. `audit_shipper`, which tails the file from a persisted byte offset and ships new lines to whatever destinations you configured — `syslog`, `http` or `s3`. It reads the file directly, not through the timeline API, and it ships archives after a rotation as well as the live tail.

Those are two different places, and only the second one takes the identifier off the box. **So that is where the identifier stops.**

What the shipper does with this source

`radius-exchanges` **is not shipped off the box unless you opt a destination in.** Every other source ships exactly as it always has.

The shipper holds a short list of sources it withholds by default — `WITHHELD_SOURCES` in `quantumtouch_bng/helpers/audit_shipper.py`, which today contains `radius-exchanges` and nothing else. A destination ships a withheld source only if it names that source in `opt_in_sources`:

```
{
  "bng": {"audit": {"shipping": {"destinations": [
    {"name": "corp-siem", "type": "syslog", "host": "siem.example.com"},
    {"name": "local-archive", "type": "s3", "bucket": "audit-archive",
     "opt_in_sources": ["radius-exchanges"]}
  ]}}}
}
```

`corp-siem` above gets every source it got before — the qt event lines, the config changes, the snapshot operations — and no RADIUS entries. `local-archive` gets those plus `radius-exchanges`.

Four things follow, and they are worth being explicit about:

- `opt_in_sources` **is an opt-in list, not an allow-list.** Naming one source never stops another from shipping. A destination that does not have the key at all ships precisely what it shipped before this filter existed — the filter cannot change an existing deployment's behaviour for any source that is not on the withheld list.
- **It is per destination.** You can archive RADIUS entries to a bucket whose retention you control while your corporate SIEM never sees them.
- **The filter is at the shipper, not at the entry.** Nothing is hashed, truncated or masked. The identifier in the local log and on the operator's timeline is the real one, so searching the timeline for the subscriber you are on the phone with still works, and two boxes shipping to one SIEM stay correlatable because nothing was salted per box.
- **It fails closed.** A malformed `opt_in_sources` in a hand-edited `settings.json` withholds rather than ships. The admin API rejects a malformed one outright, including a source name that is not actually withheld — a typo like `radius_exchanges` is a 400, not a setting that looks configured and does nothing.

You can set it in the web UI instead — **Settings → Audit shipping → Edit a destination → "Ship radius-exchanges entries to this destination"**, unticked on every destination including new ones — or through the admin API:

```
PATCH /api/bng/audit/shipping/destinations/local-archive
{"opt_in_sources": ["radius-exchanges"]}
```

How to see that it is happening

Withholding is reported, not silent. `GET /api/bng/audit/shipping/status` returns the daemon's per-destination snapshot, and each destination carries:

field	means
`entries_withheld`	how many entries this destination did not get
`withheld_by_source`	the same, broken out per source
`opt_in_sources`	which withheld sources this destination *is* opted into
`withheld_note`	the plain-language reason, empty when nothing was held

So the operator asking "why does my SIEM have no RADIUS entries?" reads:

```
"corp-siem": {
  "entries_shipped": 412,
  "entries_withheld": 37,
  "withheld_by_source": {"radius-exchanges": 37},
  "opt_in_sources": [],
  "withheld_note": "37 from radius-exchanges: withheld because these entries
    carry subscriber identifiers and this destination has not opted in. They
    are still in the local audit log and on the operator timeline. To ship
    them, add \"radius-exchanges\" to this destination's opt_in_sources."
}
```

The same numbers are on the destination's row in **Settings** → **Audit shipping**: a destination that had entries held back carries a `N withheld` chip whose tooltip is the note above, and one that is opted in says so.

The counters run from the last time the destination list was loaded, and they count entries the tail handed to *that* destination — a destination added after a burst of RADIUS entries will report zero, because it was never offered them.

If your deployment is subject to GDPR, CCPA or an equivalent regime

Subscriber MAC addresses and session identifiers are personal data. With the default configuration they stay on the box, so turning the capture on is a change to what the box records, not to what it discloses to a processor.

Opting a destination in is the processing change. That is when the identifiers reach a destination you do not control, under whatever retention it applies, and that is what belongs in your record of processing. Two other bounds still apply and are still worth using:

- **Run it as a bounded session.** Turn the capture on while you are diagnosing and stop it afterwards. Nothing about it is meant to be left running, and everything else in the RADIUS diagnostics — the offline engine, the conformance report, the capture corpus — works with the tap stopped.

- **Size the exposure.** A segment contributes at most 51 entries (a receipt plus `max_entries`, default 50), whatever is in it — so a spool rotating every sixty seconds tops out near 3000 entries an hour even during an incident.

Is there a redaction mode?

No, and the filter above is why there does not need to be one. A redaction mode would have to hash or truncate the session key where the entry is built, in `radius/diagnostics/summarize.py`, and that costs more than it buys:

- A hashed or truncated session key is **pseudonymisation, not anonymisation**. It still correlates one subscriber's records to each other across the whole timeline — which is precisely the property the timeline exists for.
- It would break the operator's main workflow: searching the timeline for the subscriber they are on the phone with. A hash is not something you can type in from a trouble ticket.
- The salt becomes a second secret to manage, and a per-box salt makes entries from two boxes uncorrelatable in the SIEM that collects both.

Every one of those costs is paid to protect the identifier *on the box*, where it is behind operator authentication and is the reason the feature exists. The risk was the identifier leaving the box, and that is a different place. The shipper filter addresses it there, at full fidelity locally and nothing at all off-box by default.

If you need the on-box value redacted as well — a threat model where the operator themselves must not see the identifier — that is a real requirement and a different one. Tell us, and it goes in `summarize` with the trade-offs above accepted deliberately rather than by default.