RECEIVE YOUR AUDIT TRAIL IN

HTTPS endpoint

Receive the REVIEWIT audit trail on a collector you operate: one signed HTTPS POST per batch, gzip NDJSON, an HMAC-SHA256 signature with a secret REVIEWIT generates and shows once — the generic contract for any collector.

VERSION
1.0 · October 2026

DURATION
Thirty minutes, in three steps

DISTRIBUTION
Confidential · REVIEWIT customers

WHAT YOU NEED

01 **An operator of the collector** able to expose an HTTPS endpoint on a public address, on port 443 or 8088, with a publicly trusted certificate or its fingerprint, and to implement the signature check below.

02 **A subscription administrator** of your REVIEWIT subscription. Creating, activating and replacing a credential ask for a strong authentication of less than ten minutes: the console sends you to your identity provider and brings you back.

03 **The order of this guide:** first the target on your side, then the destination in REVIEWIT, then the test. Nothing is delivered before a real write under your credentials has succeeded.

THE PATH, IN THREE STEPS

STEP 1

Prepare the target

Create the receiving endpoint and the credential REVIEWIT will use — minimal rights, nothing else.

STEP 2

Declare the destination

Four screens in the Subscription Manager: target, residence, content, review.

STEP 3

Test, then activate

A probe write under your credentials; a refusal says why. Delivery starts within the minute.

IN THIS GUIDE

1 How BEACON delivers your audit trail

2 What you receive

3 Prepare the target at HTTPS endpoint

4 Declare the destination in REVIEWIT

5 Test and activate

6 Verify offline with beacon-verify

7 Operate

8 Troubleshooting and checklist

01 How BEACON delivers your audit trail

BEACON is the delivery channel of the REVIEWIT audit trail. It does not produce the evidence and does not own the events: **the audit trail proves, REVIEWIT SEAL seals, BEACON delivers.** Every audit event becomes a link — a strictly increasing sequence, the SHA-256 of the record, the hash of the previous link, and the original record verbatim — and the head of every chain is signed by the sealing authority. What arrives at your target is a chain you can recompute and a seal you can check, with no call back to REVIEWIT.

YOUR REGION

Audit trail

Events chained per domain and retention class, sealed every hour by REVIEWIT SEAL.

→

YOUR REGION

BEACON

A pass every minute: batches of up to 500 events or 5 MB, a manifest per batch, a heartbeat every 15 minutes.

→

YOUR SIDE

HTTPS endpoint

Your collector: a POST per batch and per heartbeat, body gzip NDJSON, headers `X-Beacon-*`, HMAC signature.

GUIDING PRINCIPLE

Nothing is lost and nothing is pretended. Delivery advances chain by chain, only on a strict acknowledgement; a failure resumes exactly where it stopped; after twelve failed attempts the destination is paused and an alert is raised. A retention purge is delivered as a signed cut, a filter leaves stubs that keep the chain verifiable, and silence is covered by the heartbeat: never an unsigned gap.

SUPPORTED THROUGH

The **HTTPS endpoint** card of the destination wizard (HTTP egress profile). REVIEWIT **generates** the HMAC secret and shows it once; only its sealed envelope is kept, and every batch is signed by the sealing authority's office — never by a process that holds the secret in clear. Acknowledgement: any **2xx**; a 3xx is a failure, redirects are never followed. Residence is declarative.

02 What you receive

Three kinds of record travel to your target, all as UTF-8 JSON whose property order is the contract. A **link** is one audit event: `metadata.uid` (the stable event identifier), `reviewit.chain` (key, genesis), `reviewit.integrity` (`sequence`, `previous_hash`, `hash`, `raw_data`). A **manifest** closes each batch: chain, range, entry and exit hashes, counts, `batch_id`, and the reference of the latest seal of the chain. A **heartbeat** carries, every 15 minutes, the head and acknowledged positions of every chain.

RECORD	WHERE IT LANDS	WHAT IT IS FOR
`POST body: links, then the manifest line`	One request per batch, `Content-Type: application/x-ndjson`, `Content-Encoding: gzip`	The links as NDJSON, then `{"reviewit_manifest/v1":{…}}` on the last line
`X-Beacon-Batch-Id · X-Beacon-Chain · X-Beacon-Sequence-Range`	Request headers of a batch	Identify the batch, its chain and its `from-to` range without reading the body
`X-Beacon-Timestamp · X-Beacon-Signature`	On every request	Unix seconds, and `v1=<base64>[,v1=<base64>]` — two signatures during an open rotation
`X-Beacon-Kind: heartbeat`	One request every 15 minutes, body = one gzipped JSON line	State of every chain — alert on its absence

AT-LEAST-ONCE, DEDUPLICATED BY YOU

After a network doubt, the same batch may be presented again with the same `batch_id` and the same event identifiers. Dedupe on `metadata.uid`, and order across chains by occurrence time. A record that never arrives is a gap your SIEM can see: every link is numbered, and the heartbeat says where the head stands.

THE PULL, SAME CONTRACT

Your SIEM can also read the trail itself through the REVIEWIT public API, with the scope `audit.read`: `GET /api/v1/audit/chains` lists the chains of your subscription with their start cursor, `GET /api/v1/audit/stream?cursor=…&limit=500&manifest=true` returns the same links and the same manifests as the push. Push and pull verify with the same tool.

03 Prepare the target at HTTPS endpoint

Your collector exposes one HTTPS endpoint that accepts a POST, verifies the signature, stores the body as received, and answers 2xx only once the batch is durably written. REVIEWIT's egress guard requires a public address, port 443 or 8088, and never follows a redirect.

01 Expose the endpoint

`https://<host>[:8088]/<path>` on a publicly routable address, HTTPS only. A publicly trusted certificate needs nothing; a private-CA certificate can be **pinned** by your administrator from the destination panel (section 7).

02 Verify the signature before storing

Decode `X-Beacon-Signature` (`v1=…`, possibly two values during a rotation), recompute `HMAC-SHA256(secret, "{timestamp}.{subject}.{sha256}")` where `timestamp` is `X-Beacon-Timestamp`, `subject` is `X-Beacon-Batch-Id` (or the literal `heartbeat` when `X-Beacon-Kind: heartbeat`), and `sha256` is the lowercase hex SHA-256 of the **body bytes as they travel — the gzipped bytes**. Accept if any `v1=` value matches; reject a timestamp older than a few minutes.

03 Store the body as received, then acknowledge

Keep the gzipped body (or the decompressed NDJSON, unaltered): `raw_data` must stay verbatim for the offline verification. Answer **2xx only after the write is durable**: an acknowledged batch is never sent again, except after a network doubt — dedupe on `X-Beacon-Batch-Id`, then on `metadata.uid`.

04 Alert on silence

A heartbeat arrives every 15 minutes with the head and acknowledged positions of every chain. Raise an alert when none has arrived for 30 minutes: silence is not information.

SIGNATURE CHECK (PYTHON, REFERENCE)

```
import base64, hashlib, hmac

def verify(secret: str, headers: dict, body_bytes: bytes) → bool:
    subject = headers.get("X-Beacon-Batch-Id") or ("heartbeat" if headers.get("X-Beacon-
Kind") == "heartbeat" else None)
    digest = hashlib.sha256(body_bytes).hexdigest()             # the gzipped bytes, as
they travel
    message = f'{headers["X-Beacon-Timestamp"]}.{subject}.{digest}'.encode("utf-8")
    expected = hmac.new(secret.encode("utf-8"), message, hashlib.sha256).digest()
    for part in headers["X-Beacon-Signature"].split(","):     # "v1=…" or "v1=new,v1=ol
d" during a rotation
        if hmac.compare_digest(base64.b64decode(part.split("=", 1)[1]), expected):
            return True
    return False
```

WHAT YOU WILL ENTER IN THE WIZARD

WIZARD FIELD	VALUE
Transport	**HTTPS endpoint** (NDJSON, HMAC signature)
Name	How the destination will read in your list — "Collector — SOC".
Target URL	`https://<host>[:8088]/<path>`. HTTPS only, port 443 or 8088, a publicly routable host, no credentials in the URL, no redirect.
HMAC secret	Nothing to enter: REVIEWIT **generates** it at creation and shows it **once** (`bcsec_…`). Copy it into your collector's configuration at that moment; it is never shown again.
Target geography	Where your collector keeps the data. Declarative.

04 Declare the destination in REVIEWIT

Subscription Manager › **Settings** › **Integrations & AI** › **BEACON** › **New destination**. The wizard has four screens; nothing is created before the last one, and the target is tested against the egress guard before the destination exists.

01 Target

Name the destination, choose the **HTTPS endpoint** card, enter the target URL and the credential from section 3. The credential is opened in memory for this action only, then sealed under REVIEWIT SEAL custody: the screen never shows it again.

02 Residence

Declare the **geography where the collector stores the trail**. For an HTTP endpoint it cannot be observed, so it is declarative. If it differs from the geography of your REVIEWIT subscription, a written justification is required and recorded in the audit trail.

03 Content

Start: from the beginning of each chain (full history) or from activation (only what follows).
Events sent in full: leave empty for the full flow. A filter replaces every excluded event with a stub — sequence and hashes, no business data — and the sealed proof stops at the first stub; the screen says so before you confirm.

04 Review

Check the summary and click **Create**. The egress guard judges the target first: HTTPS only, no credentials in the URL, port 443 or 8088, a publicly routable host. A refused target names the failing check; nothing is created.

THE EGRESS GUARD, BEFORE EVERY SEND

Every target passes the same seven checks at creation, at every test and before every delivery: valid URL · HTTPS · no credentials in the URL · port 443 or 8088 · no internal REVIEWIT zone · host resolves · host is not a private or reserved address. Redirects are never followed, and TLS verification is never disabled.

05 Test and activate

A destination is activated only after a real write under your credentials has succeeded. Open the destination panel and click **Test the target**: REVIEWIT writes one probe record (a signed heartbeat-like request). "The target passes the egress guard" means the host, the credential and its rights are proven. A refusal says its cause — never the secret.

01 Test the target

REVIEWIT sends one signed request to your endpoint and expects a 2xx. A refusal names its cause: an HTTP status (401/403 — your collector rejected the signature: check the secret and the gzipped-body digest; 404 — wrong path; 3xx — a redirect, never followed), or a TLS refusal (pin the certificate, section 7).

02 Activate

From the panel, **Activate** (or tick "Activate as soon as it is created" in the wizard). Delivery begins at the next pass, within the minute; the first heartbeat follows.

03 Read the panel

Per chain: the acknowledged checkpoint, the age of the oldest undelivered event, the margin before the next retention cut, the latest seal and what lies beyond it. The batch history lists every manifest; the channel journal lists every administrative gesture with its audit code.

06 Verify offline with beacon-verify

`beacon-verify` is a command-line tool distributed with its public specification. It reads what your target received and the keyring REVIEWIT publishes at `/.well-known/seal-keyring.json`, anchored on the root fingerprint printed in your contract's security annex, and renders a verdict per chain and per range — with no network and no call to REVIEWIT.

THE COMMAND

```
beacon-verify <export> --keys seal-keyring.json --root <root fingerprint> --namespace <y
our sealing namespace>
```

Your **sealing namespace** is shown in the destination panel (`t-` followed by 26 characters). Pass it with `--namespace`: a seal from another namespace is, for you, a break. Without `--root` and `--namespace`, ranges are reported as sealed *with reserve*.

01 **Keep what arrived**

Your collector stored each body as received: gzipped NDJSON (or the clear NDJSON). Gather the bodies of a chain in a folder; the verifier reads gzip (1F 8B) and clear files alike, a folder in full.

02 **Run the verifier**

`beacon-verify ./collector-store --keys seal-keyring.json --root <root> --namespace <t-…>`. Exit 0 when everything before the last seal is sealed; 3 when a chain has no seal yet under your namespace.

03 **Verify a transport signature from a captured request**

If you kept the request headers with the body: `beacon-verify hmac <body> --timestamp <X-Beacon-Timestamp> --subject <X-Beacon-Batch-Id|heartbeat> --signature <X-Beacon-Signature>`, with the secret in `BEACON_HMAC_SECRET`. The transport signature proves origin and integrity on the wire; only the SEAL seal proves the trail.

VERDICT	MEANING
`SEALED`	Chained up to a head whose signature is valid, keyring anchored, namespace pinned.
`SEALED, WITH RESERVE`	Same, but the root or the namespace were not pinned on the command line.
`CHAIN ONLY`	Intact, beyond the last seal or behind a stub — the next seal will cover it.
`NOT VERIFIABLE`	A seal without namespace, an unknown algorithm or a key absent from the keyring — never called false.
`BROKEN`	A wrong hash, a gap, a divergent duplicate, an invalid signature or a foreign namespace — with the cause and the rank.

Exit codes: **0** nothing broken and everything before the last seal is sealed · **1** a break or a foreign namespace · **2** unusable input (a duplicated JSON property, a value both raw and wrapped, a record truncated at indexing) · **3** nothing broken, proof incomplete.

07 Operate

Every gesture on a destination is made by a subscription administrator, traced in the audit trail under their name, and the sensitive ones ask for a strong authentication of less than ten minutes. The channel that exports the trail is itself under the trail.

Rotate the HMAC secret	Panel › **Rotate the secret**: the new secret is shown once; both secrets sign every request (`v1=new,v1=old`) until you click **Close the rotation**, once your collector verifies with the new one. A second rotation is refused while one is open.
Pin the collector's certificate	Panel › Transport › "Certificate presented" shows the subject, issuer, expiry and SHA-256 fingerprint the egress guard read. Compare it with the one your operator gave you, then **Trust this TLS certificate** (strong authentication, traced). Deliveries then accept publicly trusted certificates or exactly this one. A renewed certificate must be pinned again; the pin goes away with the target when the URL changes.
Pause for failures	A persistent refusal — a revoked credential, a removed right, a target gone — pauses the destination after twelve attempts and raises an alert in the Alerts Center. Nothing is lost: resumption restarts at the acknowledged checkpoint of every chain, automatically when the hourly probe sees the target again, or with **Resume**.
Change the target	**Edit** › new URL, same form. The secret is kept; a pinned certificate is dropped and must be pinned again.
Retire	**Disable**, then remove or archive. A destination that delivered is archived, never deleted: its manifests are delivery proofs. Revoke the credential on your side afterwards: it serves no purpose any more.

08 Troubleshooting

WHAT YOU SEE	CAUSE	WHAT TO DO
Your collector rejects every signature	The digest was computed on the decompressed body, or the subject is wrong	Hash the gzipped bytes as received; subject = `X-Beacon-Batch-Id`, or the literal `heartbeat`.
HTTP 3xx on every send	Your endpoint redirects (http → https, trailing slash)	Enter the final URL; REVIEWIT never follows a redirect.
The egress guard refuses the target	Port other than 443/8088, private address, credentials in the URL	Expose the collector on 443 or 8088 on a public address; keep the URL clean.
A TLS error on every send	Private-CA or self-signed certificate	Pin it from the panel (section 7). Verification is never disabled.
The same batch arrives twice	A network doubt after a 2xx	Normal (at-least-once): dedupe on `X-Beacon-Batch-Id`, then `metadata.uid`.
Destination paused, "after failures"	Twelve consecutive non-2xx answers	Read the cause in the panel, fix the collector, **Test the target**, then **Resume**. Nothing was lost.

Checklist before you start

8 ITEMS

- [ ] An HTTPS endpoint on a public address, port 443 or 8088, no redirect
- [ ] A publicly trusted certificate, or its SHA-256 fingerprint ready to pin
- [ ] The signature check implemented on the gzipped bytes, both `v1=` values accepted
- [ ] A durable write before the 2xx, deduplication on batch and event identifiers
- [ ] An alert on 30 minutes without heartbeat
- [ ] The geography of the collector and its justification if it differs from your subscription's
- [ ] A subscription administrator with their second factor ready, ready to copy the secret once
- [ ] The root fingerprint from your contract annex, for beacon-verify

This guide is provided to REVIEWIT customers who connect their SIEM or storage to BEACON. Menu paths are those of the vendors' consoles at publication date; they may change. In case of discrepancy, REVIEWIT support will assist you.