RECEIVE YOUR AUDIT TRAIL IN

Amazon S3

Receive the REVIEWIT audit trail as objects in an Amazon S3 bucket — or in an S3-compatible object store in your own network (StorageGRID, MinIO…): batches and adjacent manifests, signed in the object metadata, verifiable at rest.

VERSION
1.0 · October 2026

DURATION
Twenty minutes, in three steps

DISTRIBUTION
Confidential · REVIEWIT customers

WHAT YOU NEED

01 **An AWS administrator** able to create a bucket and an IAM user with a single permission — or, for an S3-compatible store, the operator of that store, able to create a bucket and an access key limited to writing into it.

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 Amazon S3

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

Amazon S3

Your bucket: two objects per batch (data, then manifest), a heartbeat object, a probe object — under access keys that can only write.

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 **Amazon S3** card of the destination wizard (object-store profile). Two forms of target: the AWS regional virtual-host form, `https://{bucket}.s3.{region}.amazonaws.com`, where REVIEWIT **observes** the region and expects the owner account; and, since October 2026, an **S3-compatible store in your network**, `https://{host}[:8088]/{bucket}` (path style), where the residence is **declared**. Access keys are entrusted once and sealed under REVIEWIT SEAL custody. It is the only transport where the HMAC transport signature stays verifiable at rest.

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
`{from}-{to}-{batch_id}.ndjson.gz`	`beacon/{subscription}/{destination}/{class}/g{generation}/{chain}/` — `from` and `to` zero-padded to 20 digits	The links, NDJSON gzipped, the manifest on the last line
`….manifest.json`	The same key, suffixed; written **after** the data object	The line `{"reviewit_manifest/v1":{…}}` — a batch is delivered only once this object exists
`heartbeat.json`	`beacon/{subscription}/{destination}/`, rewritten every 15 minutes	State of every chain — alert on its absence
`probe.json`	Same prefix; at creation, at every test and at activation	The probe write that proves the bucket, the keys and the owner

INGEST ON THE MANIFEST, NEVER ON THE DATA OBJECT ALONE

A data object without its adjacent manifest is a batch in progress or failed: never ingest it. A retried batch is rewritten identically under the same key; a batch replanned on a wider range leaves the previous data object orphaned — a lifecycle rule may purge `.ndjson.gz` objects without manifest after a few days. Keys carry no date: they are ordered by chain and by rank. REVIEWIT never deletes anything: the objects are yours, your lifecycle policy applies.

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 Amazon S3

An administrator prepares the bucket and a credential that can **only write under the `beacon/` prefix**. Nothing else: REVIEWIT reads nothing, lists nothing, deletes nothing; on AWS the region is observed without a key.

01 Amazon S3: the bucket

A name **without a dot** (the virtual-host form is the only one accepted on AWS, and a dot breaks the TLS validation that REVIEWIT never disables), in the region you want. Default encryption on (SSE-S3 or SSE-KMS): REVIEWIT sends no encryption header, the bucket policy applies. Overwrite allowed: a redelivery rewrites the same key — a policy that forces conditional writes would fail redeliveries; versioning is compatible.

02 Amazon S3: the IAM user and its policy

A dedicated IAM user with the single statement below — `s3:PutObject` on `arn:aws:s3:::<bucket>/beacon/*`. Create an access key pair for it; note the **12-digit account ID** that owns the bucket: every write carries `x-amz-expected-bucket-owner`, so that a bucket name taken over one day by another account refuses the write instead of receiving your trail.

03 S3-compatible store in your network

StorageGRID, MinIO, Ceph RGW or any store exposing the S3 API over HTTPS on **443 or 8088**, addressed **by path**: `https://<host>[:8088]/<bucket>`. The certificate is the host's, so the bucket name may carry dots. Create the bucket and an access key limited to writing into it; signing uses the conventional region `us-east-1`. No owner account is expected; the residence is what you declare. An AWS host is never written in this form.

04 Keep the keys for the wizard only

The access key ID and secret are entered once in REVIEWIT (section 4). REVIEWIT writes `probe.json` under them before entrusting them: if the write fails — a key without right, a wrong owner account — nothing is created and nothing is entrusted.

IAM POLICY OF THE DEDICATED USER (AMAZON S3)

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::<bucket>/beacon/*"
  }]
}
```

WHAT YOU WILL ENTER IN THE WIZARD

WIZARD FIELD	VALUE
Transport	**Amazon S3** (object store)
Name	How the destination will read in your list — "S3 archive — compliance".
Target URL	AWS: `https://<bucket>.s3.<region>.amazonaws.com` (no dot in the name, the region is observed). S3-compatible: `https://<host>[:8088]/<bucket>` (you then declare the geography).
Bucket owner account	AWS only: the 12-digit account ID that owns the bucket.
Access key ID · Secret access key	The dedicated key pair. Tested by a probe write before being kept.
Target geography	AWS: observed from the bucket's real region, shown as "Bucket observed in … — geographies: …". S3-compatible: declared.

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 **Amazon S3** 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

On AWS the region is **observed**: a bucket in Paris is in France and in Europe, and the screen says whether it lies within your subscription's geography. Outside it, an acknowledgement with a written justification is required and recorded in the audit trail. On an S3-compatible store nothing can be observed: you **declare** where it is located, and the same rule applies.

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 (`probe.json` under your keys). "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 writes `probe.json` under the access keys, with the expected owner account on AWS. "The target passes the egress guard" means the bucket, the keys and the owner are proven. A refusal names its cause: "the bucket refuses this destination" — key without `s3:PutObject` on the `beacon/` prefix, wrong owner account, bucket name with a dot, or a path-style URL on an AWS host.

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 **Collect the objects of a chain**

Download, from the chain's prefix, the `.ndjson.gz` data objects whose `.manifest.json` exists, plus the manifests. A folder is read in full: `beacon-verify <folder>`. Never decompress and re-serialize: the verifier reads gzip (1F 8B) and clear NDJSON alike.

02 **Run the verifier**

`beacon-verify ./beacon-export --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 at rest**

S3 is the only transport that keeps the signed bytes. Each object carries `x-amz-meta-beacon-batch-id`, `-sequence-range`, `-timestamp` and `-signature` (`v1=base64(HMAC-SHA256(secret, "{timestamp}.{batch_id|heartbeat}.{sha256 hex lowercase of the object}"))`). `beacon-verify hmac <object> --timestamp <seconds> --subject <batch_id|heartbeat> --signature <v1=…>`, with the secret in the environment variable `BEACON_HMAC_SECRET`, never as an argument. During an open rotation the metadata carries two signatures, `v1=new,v1=old`: accept the object if either verifies.

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 access keys	Create the new key pair, then panel › **Replace the access keys**: the new keys must pass a probe write before being kept. Deactivate the old pair on your side afterwards. The HMAC secret rotates separately — **Rotate the secret** shows the new value once, both sign until **Close the rotation**.
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. A bucket is not changed in place: create a new destination for a new bucket, disable the old one.
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
"The bucket refuses this destination" at creation	No `s3:PutObject` on `beacon/*`, wrong owner account, or a bucket name with a dot	Fix the policy or the account ID; rename the bucket without a dot; try again — nothing was created.
Target URL refused on an S3-compatible store	A port other than 443 or 8088, more than one path segment, or an AWS host written in path style	Use `https://<host>[:8088]/<bucket>`; AWS buckets use the virtual-host form.
The egress guard refuses: "The host resolves to a private or reserved address"	The store is only reachable on your private network	In the cloud offer, expose it on a public address; on an on-premises REVIEWIT installation, declare the private network to the installation's egress guard.
Data objects without manifest accumulate	Batches replanned on a wider range between two attempts	Normal; ingest on the manifest only; a lifecycle rule may purge orphans after a few days.
Geography shown outside the subscription's	The bucket's real region is elsewhere	Acknowledge with a justification (traced), or create the bucket in the right region.
Destination paused, "after failures"	Keys deactivated, policy changed, bucket policy forcing conditional writes	Fix on the storage side, **Test the target**, then **Resume**. Nothing was lost.

Checklist before you start

7 ITEMS

- [ ] AWS: a bucket without a dot, encryption on, overwrite allowed; or an S3-compatible store on 443 or 8088
- [ ] A dedicated user or key limited to writing under `beacon/`
- [ ] AWS: the 12-digit owner account ID
- [ ] The access key ID and secret, used only in the wizard
- [ ] The bucket's geography and its justification if it differs from your subscription's
- [ ] A subscription administrator with their second factor ready
- [ ] 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.