RECEIVE YOUR AUDIT TRAIL IN

Microsoft Sentinel

Receive the REVIEWIT audit trail in your Log Analytics workspace through the Azure Monitor Logs Ingestion API: a data collection rule, a table, an app registration with a certificate — no Function, no Logic App, no agent, no client secret.

VERSION
1.0 · October 2026

DURATION
Forty-five minutes, in three steps

DISTRIBUTION
Confidential · REVIEWIT customers

WHAT YOU NEED

01 **An Azure administrator** able to create a Log Analytics workspace, a data collection endpoint and rule, an app registration in your Microsoft Entra tenant, and to assign a role on the rule.

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 Microsoft Sentinel

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

Microsoft Sentinel

The Logs Ingestion API of Azure Monitor, into the table `REVIEWIT_Audit_CL` of your workspace, under your app registration's certificate.

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 **Microsoft Sentinel** card of the destination wizard (Logs Ingestion profile). Authentication is the **certificate of your app registration**: REVIEWIT obtains an Entra token by a signed assertion (RS256) with the private key you entrust — PFX or PEM — sealed under REVIEWIT SEAL custody. No client secret, ever. Acknowledgement is **HTTP 204** from Azure Monitor. Proven end to end on 19 September 2026 on an Azure trial (France Central).

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
`Kind == "link"`	One row per link in `REVIEWIT_Audit_CL`; `TimeGenerated` is the occurrence time; `Event` is the link	The audit event, with its proof fields and `raw_data` verbatim
`Kind == "manifest"`	One row per batch, bare manifest	Range, hashes, counts, `batch_id`, seal reference
`Kind == "heartbeat"`	One row every 15 minutes	State of every chain — alert on its absence
`Kind == "probe"`	One row at creation, at every test and at activation	The probe write that proves the rule, the stream and the role

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 Microsoft Sentinel

An Azure administrator creates, once, the workspace and its Sentinel binding, then the table and its data collection rule, then the app registration that is REVIEWIT's identity in your tenant. Labels below are those of the Azure portal; the portal wizard requires a data collection endpoint even though a direct rule would not.

01 Log Analytics workspace, then Sentinel

Create a resource › **Log Analytics workspace**: resource group, name (`law-reviewit-beacon`), region — the geography you will declare in REVIEWIT. Then Microsoft Sentinel › **Add** the workspace.

02 Data collection endpoint

Monitor › Data collection endpoints › **Create**: name `dce-reviewit-beacon`, same resource group and region. Note its **Logs ingestion URI** (`https://dce-…-xxxx.<region>-1.ingest.monitor.azure.com`).

03 Table and data collection rule

Workspace › Settings › **Tables** › **Create** › **New custom log (DCR-based)**. Table name `REVIEWIT_Audit` (the portal adds `_CL`), plan Analytics, create the rule `dcr-reviewit-beacon` from the link, endpoint `dce-reviewit-beacon`. Schema: upload the sample below as a file; the portal infers `TimeGenerated` (datetime), `Kind` (string), `Event` (dynamic), transformation `source`. Create.

04 Note the rule's immutable ID

Rule › Overview › **Immutable ID** (`dcr-` + 32 hexadecimal characters), copied with the copy button: one wrong character and REVIEWIT answers "unknown stream" (404). The ingestion URL to enter in REVIEWIT is `https://<logs ingestion URI host>/dataCollectionRules/<immutable id>/streams/Custom-REVIEWIT_Audit_CL?api-version=2023-01-01`, one line.

05 App registration with a certificate

Microsoft Entra ID › App registrations › **New registration**: name `reviewit-beacon`, single tenant, no redirect URI. Copy the **Directory (tenant) ID** and the **Application (client) ID**. Certificates & secrets › Certificates › **Upload certificate**: the public `.cer` of a key your PKI issued for this application (RSA 2048 at least, digital signature). Keep the export **with the private key** (PFX + password, or PEM): it is entrusted to REVIEWIT in section 4. Create **no** client secret.

06 **Before entrusting the PFX, read it back with the password you will type**

This is exactly the check REVIEWIT performs; a password that does not match is refused with no further diagnosis. The fingerprint must be the one Entra shows.

SAMPLE FILE FOR THE TABLE SCHEMA (TWO ROWS)

```
[
  { "TimeGenerated": "2026-09-19T12:00:00.0000000Z", "Kind": "link",
    "Event": { "metadata": { "uid": "00000000-0000-0000-0000-000000000001" }, "raw_dat
a": "{\"sample\":true}",
               "reviewit": { "chain": { "key": "…##BUSINESS#G1", "genesis_at": "2026-09-
19T00:00:00.000000Z" },
                            "integrity": { "sequence": 1, "hash": "0000", "integrity_ve
rsion": 2, "hash_algorithm": "SHA-256" } } } },
  { "TimeGenerated": "2026-09-19T12:00:01.0000000Z", "Kind": "manifest",
    "Event": { "batch_id": "00000000-0000-0000-0000-000000000002", "from_sequence": 1,
"to_sequence": 1 } }
]
```

SELF-SIGNED CERTIFICATE FOR A TRIAL, AND THE READ-BACK CHECK (POWERSHELL)

```
$c = New-SelfSignedCertificate -Subject "CN=reviewit-beacon-client, O=<your organization
>" -KeyAlgorithm RSA -KeyLength 2048 -HashAlgorithm SHA256 -KeyExportPolicy Exportable -
KeyUsage DigitalSignature -CertStoreLocation Cert:\CurrentUser\My -NotAfter (Get-Date).A
ddYears(1)
Export-PfxCertificate -Cert $c -FilePath .\reviewit-beacon.pfx -Password (Read-Host -AsS
ecureString "PFX password")
Export-Certificate  -Cert $c -FilePath .\reviewit-beacon.cer     # upload this one to En
tra

$p = Read-Host -AsSecureString "PFX password"; $plain = [Runtime.InteropServices.Marsha
l]::PtrToStringUni([Runtime.InteropServices.Marshal]::SecureStringToGlobalAllocUnicode
($p))
$t = [System.Security.Cryptography.X509Certificates.X509Certificate2]::new(".\reviewit-b
eacon.pfx", $plain, "EphemeralKeySet"); $plain = $null
"$($t.Subject) | private key: $($t.HasPrivateKey) | $($t.Thumbprint)"   # the thumbprint
Entra shows
```

WHAT YOU WILL ENTER IN THE WIZARD

WIZARD FIELD	VALUE
Transport	**Microsoft Sentinel** (Azure Monitor Logs Ingestion API)
Name	How the destination will read in your list — "Sentinel — SOC".
Target URL	The full ingestion URL of the rule: `https://<endpoint>.<region>-1.ingest.monitor.azure.com/dataCollectionRules/dcr-…/streams/Custom-REVIEWIT_Audit_CL?api-version=2023-01-01`.
Entra directory (tenant) ID · Application (client) ID	From the app registration's overview page. Not secrets.
Certificate with its private key	The PFX (with its password) or PEM exported in step 05. REVIEWIT checks it carries a non-expired RSA private key, seals the key, keeps the public certificate for the fingerprint and the expiry. A `.cer` alone is refused, and says so.
Target geography	The workspace's region (France Central = FRA). 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 **Microsoft Sentinel** 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 of the workspace** (its Azure region). For an HTTP endpoint 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 (`Kind = probe` on the stream, after a token obtained by signed assertion). "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

Two things are proven in one gesture: REVIEWIT obtains an Entra token by a signed assertion (proves the certificate), then writes one `probe` row on the stream (proves the rule, the stream and the role). Before testing, give the role: rule `dcr-reviewit-beacon` › **Access control (IAM)** › **Add role assignment** › **Monitoring Metrics Publisher** › member `reviewit-beacon` (type Application). It is the only right REVIEWIT needs — on the rule alone, not on the workspace. The role is effective within a minute. A refusal names its cause: "certificate not registered", "role missing on the rule", "unknown stream". The destination is created as a draft: activate it from the panel once the probe passes.

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 **Query, over a wide window**

`REVIEWIT_Audit_CL | where Kind in ("link", "manifest") | project TimeGenerated, Kind, Event`, time range **last 7 days** or wider (`TimeGenerated` of a link is the occurrence time: a short window cuts batches apart), display limit **5 000** or maximum. The footer must announce the expected count.

02 **Share › Export to CSV (all columns)**

The portal saves `query_data.csv`. Do not edit it: the verifier reads the CSV as is (header, `Event` column as quoted JSON), as well as the JSON tables of the query API. Log Analytics rewrites dynamic-column dates to seven decimals; it has no effect on links (`raw_data` is a string) and the verifier accepts both forms for a cut's dates.

03 **Run the verifier**

`beacon-verify query_data.csv --keys seal-keyring.json --root <root> --namespace <t-…>`. The trial export of 19 September 2026 rendered exit 3 (the chain sealed since binding SEALED; earlier chains NOT VERIFIABLE until their next seal); one character altered in a `raw_data` rendered BROKEN at its rank, exit 1.

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.

Renew the certificate	It is your expiry and your gesture: thirty days before, the panel and REVIEWIT's health digest announce it. Upload the new certificate to the app registration (Entra accepts several), then panel › **Replace the certificate**: the form opens in the panel (PFX or PEM, password, Confirm). The previous key stops signing at once and the panel shows the new fingerprint; remove the old certificate from Entra afterwards. A refused file (no key, not RSA, expired, wrong password) changes nothing.
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 certificate and the identity are kept; assign the role on the new rule, then **Test the target**.
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
"Certificate not registered" at test	The `.cer` uploaded to Entra is not the one whose private key you entrusted	Compare the SHA-1 fingerprint in the panel ("View certificate") with Entra's; upload the right `.cer` or replace the certificate in the panel.
"Role missing on the rule" at test	Monitoring Metrics Publisher not assigned on the DCR, or assigned on the workspace instead	Assign it on the rule `dcr-reviewit-beacon`, member `reviewit-beacon` (Application); wait a minute; test again.
"Unknown stream" (404)	A wrong character in the immutable ID, or a stream name other than `Custom-REVIEWIT_Audit_CL`	Copy the immutable ID with the copy button; check the JSON view of the rule (`streamDeclarations`, `outputStream`).
The PFX is refused: "wrong password"	The password does not match, or the export has no private key	Run the read-back check of section 3; re-export from your certificate store with the private key.
Destination paused, "after failures"	Certificate removed or expired, role removed, application deleted	Fix on the Azure side, **Test the target**, then **Resume**. Nothing was lost.
Private Link / private endpoint	The ingestion endpoint is only reachable privately	REVIEWIT's egress guard reaches public addresses only; keep the public ingestion endpoint for this rule.

Checklist before you start

8 ITEMS

- [ ] A Log Analytics workspace, Sentinel added, in the region you will declare
- [ ] A data collection endpoint and rule, table `REVIEWIT_Audit_CL` with the three columns
- [ ] The rule's immutable ID and the endpoint's logs ingestion URI, assembled into the ingestion URL
- [ ] An app registration `reviewit-beacon` with a certificate uploaded, no client secret
- [ ] The PFX or PEM with the private key, read back with the password you will type
- [ ] Monitoring Metrics Publisher assigned on the rule to the application
- [ ] 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.