# Twenty webhook security, replay, and events

This reference defines the reviewed Twenty inbound webhook boundary. It covers route activation, exact raw-body HMAC, clock and workspace checks, durable nonce replay fencing, deterministic event mapping, local acceptance, and central AIP p

| | |
|---|---|
| SECTION | Connectors |
| TYPE | Connector |
| PROTOCOL | AIP 1.0 |
| SOURCE | docs/connectors/twenty/reference/webhook-security-replay-and-events.md |

Use the metadata guide to create or rotate provider registration. This page starts when an external HTTP delivery reaches the standalone connector host.

## Status and route

| Item | Reviewed value | | | | |
|---|---|---|---|---|---|
| AIP source revision | `97be86e9efedf07ecf1783b03800f683f107fb04` | | | | |
| Method | `POST` | | | | |
| Internal host route | `/webhooks/twenty` | | | | |
| Timestamp header | `x-twenty-webhook-timestamp` | | | | |
| Signature header | `x-twenty-webhook-signature` | | | | |
| Nonce header | `x-twenty-webhook-nonce` | | | | |
| Body limit | 4 MiB | | | | |
| Signature | HMAC-SHA256 over `timestamp | | ":" | | raw_body` |
| Accepted clock skew | 300,000 ms in either direction | | | | |
| Replay scope | Logical connector instance | | | | |
| Live replay capacity | 100,000 nonces | | | | |
| Local event channel | `twenty-webhooks` | | | | |

The route exists only when `AIP_TWENTY_WEBHOOK_SECRET_FILE` is configured. Enabling it also requires `AIP_CONNECTOR_HOST_EVENT_ENDPOINT`; otherwise host startup fails.

An external reverse proxy may expose a different HTTPS origin and prefix. It must preserve all three headers and the request body byte for byte when forwarding to the internal route.

## What the signature authenticates

The verification equation is:

```
TEXT
expected = HMAC-SHA256(
  configured_webhook_secret,
  timestamp_header_bytes || ":" || exact_raw_body_bytes
)

hex_decode(signature_header) == expected
```

The connector decodes exactly 64 hexadecimal signature characters and uses the HMAC library's verification method. Uppercase and lowercase hex are accepted.

The nonce is not an HMAC input. It is validated and fenced separately. The HTTP route and header names are also outside the signature input.

Possession of the secret authenticates the timestamp and exact body. It does not prove that Twenty generated a business fact, that the configured provider registration belongs to the expected operator, or that the body is true.

## Ingress pipeline

The standalone host processes one delivery in this order:

1. apply the 4 MiB HTTP body limit;
2. require non-empty timestamp, signature, and nonce headers;
3. require a non-empty body no larger than 4 MiB;
4. validate 32 hexadecimal nonce characters;
5. validate 64 hexadecimal signature characters;
6. parse a 1-to-32-digit Unix-millisecond timestamp;
7. require absolute host-clock skew at most 300,000 ms;
8. verify HMAC over timestamp, colon, and exact raw body;
9. durably admit the nonce in the instance replay scope;
10. parse the signed body as a JSON object;
11. validate event name, configured workspace, webhook UUID, and event date;
12. derive a deterministic AIP event;
13. append it to the local event log;
14. durably enqueue it on channel `twenty-webhooks`;
15. attempt immediate central delivery;
16. return HTTP `200` after durable local acceptance.

Central acknowledgement is optional at step 15. When it is unavailable, the background publisher owns a durably queued batch.

## Validate the headers and clock

| Value | Constraint |
| --- | --- |
| Timestamp | Decimal Unix milliseconds, 1 through 32 digits |
| Signature | Exactly 64 ASCII hexadecimal characters |
| Nonce | Exactly 32 ASCII hexadecimal characters |
| Clock delta | Absolute difference from host time no greater than 300,000 ms |

The timestamp string itself is signed. Reformatting the same instant changes the HMAC input.

Keep host clocks synchronized. The connector does not contact Twenty or a time authority to resolve clock disagreement.

## Preserve exact raw-body bytes

The connector computes HMAC before parsing JSON. Whitespace, key order, escaping, number formatting, or a trailing newline changes the signed bytes.

A proxy or middleware must not parse and reserialize the body. It may enforce a smaller ingress limit if it preserves accepted bodies and headers exactly.

Do not log the secret, signature, full body, or mapped event data. The body can contain confidential and personal workspace data.

## Validate the signed payload

After HMAC and replay admission, the body must be a JSON object containing:

```
JSON
{
  "eventName": "company.updated",
  "workspaceId": "0190c42f-2d5a-7000-8000-000000000080",
  "webhookId": "0190c42f-2d5a-7000-8000-000000000081",
  "eventDate": "2030-01-03T10:00:00Z"
}
```

Each named field must be a non-empty string. Additional provider fields remain inside the event data.

`eventName` accepts 1 through 256 ASCII alphanumeric, `.`, `_`, or `-` characters. It is not restricted to a source-owned event-name enum.

`workspaceId` must equal the host's configured workspace string exactly. `webhookId` must be an RFC 4122 UUID version 1 through 5. `eventDate` must parse as RFC 3339 and becomes the AIP event's `occurred_at`.

A valid signature from another workspace is rejected. The HMAC secret does not replace workspace binding.

## Replay state

The standalone host uses a profile-state replay store with:

| Property | Value |
| --- | --- |
| Namespace | `aip.connector.twenty.webhook_replay.v1` |
| Scope | `instance:<connector_instance_id>` |
| Update | Profile-state compare-and-set |
| CAS attempts | 32 |
| Retained timestamps | Values at or after `now - 300000 ms` |
| Capacity | 100,000 live nonces |

Every replica of one logical instance must share the same durable runtime database and instance ID. Different instance IDs have independent replay scopes.

Before inserting a nonce, the store removes entries older than the cutoff. It rejects an existing nonce and rejects a new nonce when the live map already contains 100,000 entries.

CAS conflicts yield and retry. Continued contention after 32 attempts is a replay-store failure, not an authentication success.

The connector also contains an in-memory implementation for tests and single-process development. The standalone host constructs the profile-state implementation.

## Understand the admission-order boundary

The replay store admits a verified nonce before JSON parsing, payload-field validation, local event-log append, or outbox enqueue.

An authenticated malformed payload therefore consumes its nonce. A failure in the later local event-log or outbox step also occurs after nonce admission.

Repeating the exact delivery with the same nonce then receives a public authentication failure. Provider retry is not the recovery mechanism for a late local failure at this revision.

Preserve host storage evidence and inspect event log, replay state, and outbox before changing state. Escalate according to deployment policy when a verified delivery was fenced but not durably published.

## Map the deterministic event

The connector calculates:

```
TEXT
digest = SHA-256(webhook_id || ":" || nonce || ":" || signature_header)
event_id = "evt_" || hex(first_16_digest_bytes)
event_kind = "twenty." || eventName
```

The event records:

| Field | Value |
| --- | --- |
| `id` | `evt_` plus 32 lowercase hexadecimal characters |
| `kind` | `twenty.<validated_eventName>` |
| `occurred_at` | Parsed signed `eventDate` |
| `data` | Complete parsed provider object |

The event ID does not include raw-body bytes directly. Signature changes alter the ID, while the replay fence separately owns nonce uniqueness.

Before central publication, the common publisher projects a provider-originated actor into `data.upstream_actor`. Authenticated central ingress owns the trusted protocol-level event actor and binds it to connector-host signing identity.

## Local event log and central outbox

The host appends the deterministic event to its local AIP event log. Duplicate event IDs reuse the log's accepted event outcome rather than creating a new event identity.

It then durably enqueues one-event batch on channel `twenty-webhooks`. The publisher derives a stable batch key from normalized channel and event data.

The acknowledgement body reports one of two central outcomes:

| `central_delivery` | Meaning at HTTP response time |
|---|---|
| `centrally_acknowledged` | Central AIP acknowledged the batch |
| `durably_queued` | Local durable outbox owns delivery; background publication continues |

A crash after central acknowledgement but before local outbox deletion may resend the same event ID. The central event log is expected to deduplicate that identity.

HTTP `200` proves durable local acceptance or central acknowledgement. It does not always prove that central delivery completed before the response.

## Public HTTP responses

| Condition | Status | Body code or fields |
|---|---|---|
| Missing or invalid required header encoding | `400` | `webhook.missing_header` |
| Invalid body, timestamp, nonce, signature, HMAC, duplicate, JSON, payload field, workspace, or event mapping | `401` | `webhook.authentication_failed` |
| Replay-store, event-log, or durable-outbox failure | `503` | `webhook.storage_unavailable` |
| Accepted and centrally acknowledged | `200` | `accepted_events: 1`, `central_delivery: centrally_acknowledged` |
| Accepted and durably queued | `200` | `accepted_events: 1`, `central_delivery: durably_queued` |

The common HTTP body layer can reject an oversized request before the handler. Do not depend on one JSON error code for that proxy or framework-level failure.

The public `401` intentionally collapses validation details. Diagnose through bounded internal telemetry without logging secret or payload content.

## Operate and recover the boundary

| Symptom | First decision |
|---|---|
| Repeated `400` | Verify exact header names and non-empty HTTP values |
| Repeated `401` | Check clock, raw-byte preservation, configured workspace, secret revision, and nonce reuse |
| `503` before acceptance | Preserve replay, event-log, profile-state, PostgreSQL, and outbox evidence |
| Success reports `durably_queued` | Monitor the durable publisher and central acknowledgement; do not ask provider to resend |
| Replay capacity reached | Check clock, traffic volume, stale-state pruning, and abuse before changing instance identity |
| CAS contention persists | Check replica count, shared scope, database latency, and competing state ownership |
| Central event is absent | Search by deterministic event ID and outbox batch before changing registration |
| Event belongs to another workspace | Reject it and reconcile provider registration with host binding |

Do not clear replay state, rotate an instance ID, replace the event ID, or ask for same-nonce replay merely to make a delivery pass. Those actions can weaken deduplication or lose the original failure evidence.

## Security and evidence boundaries

Retain:

- host artifact, instance, replica, tenant, workspace, and secret revision;
- redacted provider registration identity;
- receipt time and bounded header validation outcomes;
- deterministic event ID and kind;
- replay-store scope and admission outcome;
- local event-log append identity;
- outbox batch identity and central outcome;
- recovery actions and final central observation.

Do not retain secret bytes, full raw payload, or unredacted personal data in ordinary logs. Store any necessary payload evidence under explicit tenant, access, and retention controls.

Source review establishes this implemented pipeline. It does not establish that a particular deployment, provider registration, artifact, or live event has passed qualification.

## Related documentation

- Twenty connector configuration
- Authentication and workspace isolation
- Manage views, layouts, and webhook metadata
- Connector fleet architecture
- Errors and retry decisions