Cal.diy webhook security and replay

This reference defines the reviewed Cal.diy inbound webhook boundary. It covers route and header validation, raw-body HMAC, version and time checks, deterministic event mapping, bounded replay state, idempotent duplicate handling, durable l

SECTION	Connectors
TYPE	Connector
PROTOCOL	AIP 1.0
SOURCE	docs/connectors/cal-diy/reference/webhook-security-and-replay.md

Use the webhook capability pages to create or change subscriptions. This page starts when an external HTTP delivery reaches the standalone connector host.

Status and route

Item	Reviewed value
Connector source revision	`97be86e9efedf07ecf1783b03800f683f107fb04`
Method	`POST`
Internal host route	`/webhooks/cal-diy/{subscription_id}`
Required signature header	`x-cal-signature-256`
Required version header	`x-cal-webhook-version`
Supported payload version	`2021-10-20`
Body limit	1 MiB
Signature	HMAC-SHA256 over the exact raw body
Replay retention	86,400 seconds
Replay capacity	20,000 delivery digests per scope

The route is registered only when at least one webhook secret mapping is configured. Enabling it also requires the connector-host event endpoint. When no secret mapping is configured, the reviewed host does not mount this route.

An external reverse proxy may expose a different HTTPS origin and prefix. It must preserve the final `subscription_id` path segment, both required headers, and the request body byte for byte when forwarding to the internal route.

What the signature authenticates

The signature proves possession of the secret selected by the validated `subscription_id` route and authenticates the exact raw body bytes. The verification equation is:

```
TEXT
hex(HMAC-SHA256(selected_secret, raw_body_bytes)) == x-cal-signature-256
```

The path selector and `x-cal-webhook-version` header are not HMAC inputs. The connector validates them separately. Subscription creation binds the allowed HTTPS destination to the same opaque secret reference, but ingress must still preserve the trusted route and host configuration.

The signature is not proof that Cal.diy generated a particular business fact. It proves that the delivery matches one configured secret and body. Provider account ownership, subscription configuration, and secret custody remain deployment responsibilities.

Ingress pipeline

The standalone host processes one delivery in this order:

1. apply the 1 MiB HTTP body limit;
2. require non-empty signature and version headers;
3. require the request body to be valid UTF-8 without reserializing it;
4. require version `2021-10-20`;
5. validate the route selector and resolve its startup-loaded secret;
6. verify the 64-character hexadecimal HMAC in constant time;
7. derive a deterministic digest from the verified delivery;
8. parse the signed payload, validate its timestamp, and map its trigger;
9. observe the digest in the bounded shared replay store;
10. append the deterministic event to the local AIP event log;
11. durably enqueue the event for signed central publication;
12. return HTTP `200` after local durable acceptance.

Central acknowledgement can happen during step 11, but it is not required for HTTP success. A durably queued batch belongs to the background publisher after the response.

Route selector and secret lookup

`subscription_id` is an opaque secret reference, not secret material. Its exact grammar is:

- 1 to 128 ASCII bytes;
- first byte is alphanumeric;
- remaining bytes are alphanumeric or `.`, `_`, `~`, or `-`;
- no slash, whitespace, empty value, or duplicate configured reference.

The path value selects one secret from the deployment mapping loaded at host startup. An unknown or malformed selector and an invalid signature share the same public authentication failure, so the response does not reveal whether a particular reference exists.

Raw-body HMAC verification

`x-cal-signature-256` must contain exactly 64 hexadecimal characters. Uppercase and lowercase hex are both decodable. The connector computes HMAC-SHA256 over the exact UTF-8 bytes received and compares the 32-byte values in constant time.

Whitespace, object-key order, escaping, or a trailing newline changes the signed bytes. A proxy, middleware, or application must not parse and reserialize the JSON before the connector verifies it.

Do not log the raw body, signature, resolved secret, or mapped event payload. Webhook bodies can contain attendee, booking, meeting, recording, calendar, or form data and must remain under restricted PII controls.

Payload forms

Standard payload

When the top-level `payload` field is present, the signed JSON must deserialize to all three fields below:

```
JSON
{
  "triggerEvent": "BOOKING_CREATED",
  "createdAt": "2030-01-03T10:00:00Z",
  "payload": {
    "uid": "provider-booking-uid"
  }
}
```

`triggerEvent` and `createdAt` must be strings. `createdAt` must parse as RFC 3339. `payload` may contain any provider JSON shape, including `null`; the connector does not publish one exhaustive trigger-specific payload schema.

Managed webhook creation rejects a custom `payloadTemplate`, so newly managed subscriptions use the standard provider shape.

Flat compatibility payload

If the top-level `payload` field is absent, the connector accepts a compatibility form. It removes `triggerEvent` and optional `createdAt` and treats every remaining field as the event payload:

```
JSON
{
  "triggerEvent": "BOOKING_CANCELLED",
  "createdAt": "2030-01-03T10:05:00Z",
  "uid": "provider-booking-uid",
  "reason": "Attendee request"
}
```

If this flat form omits `createdAt`, or supplies it with a non-string value, the connector substitutes its trusted receipt time. The resulting event currently still records `timestampProvenance: "signed_body"`. Do not use that label alone as evidence that the original flat body contained a timestamp. Record a redacted marker when the compatibility fallback is used and timestamp provenance matters.

Version and timestamp checks

The required `x-cal-webhook-version` value is exactly `2021-10-20`. The header is not covered by HMAC, but its accepted value is fixed in source and is also forced into managed webhook create and update requests.

For a parsed `createdAt`, define:

```
TEXT
delta = trusted_received_at - signed_created_at
```

The accepted inclusive interval is `-300 <= delta <= 86400` seconds. A delivery may therefore be at most five minutes in the future or 24 hours old relative to the host clock. Keep connector-host clocks synchronized; the connector does not contact the provider to resolve clock disagreement.

Trigger-to-event mapping

Only these 19 provider triggers are accepted:

triggerEvent	AIP event kind
`BOOKING_CREATED`	`cal_diy.booking.created`
`BOOKING_PAYMENT_INITIATED`	`cal_diy.booking.payment_initiated`
`BOOKING_PAID`	`cal_diy.booking.paid`

triggerEvent	AIP event kind
BOOKING_RESCHEDULED	cal_diy.booking.rescheduled
BOOKING_REQUESTED	cal_diy.booking.requested
BOOKING_CANCELLED	cal_diy.booking.cancelled
BOOKING_REJECTED	cal_diy.booking.rejected
BOOKING_NO_SHOW_UPDATED	cal_diy.booking.no_show_updated
FORM_SUBMITTED	cal_diy.form.submitted
MEETING_ENDED	cal_diy.meeting.ended
MEETING_STARTED	cal_diy.meeting.started
RECORDING_READY	cal_diy.recording.ready
RECORDING_TRANSCRIPTION_GENERATED	cal_diy.recording.transcription_generated
OOO_CREATED	cal_diy.out_of_office.created
AFTER_HOSTS_CAL_VIDEO_NO_SHOW	cal_diy.meeting.host_no_show
AFTER_GUESTS_CAL_VIDEO_NO_SHOW	cal_diy.meeting.guest_no_show
FORM_SUBMITTED_NO_EVENT	cal_diy.form.submitted_without_event
DELEGATION_CREDENTIAL_ERROR	cal_diy.delegation.credential_error
WRONG_ASSIGNMENT_REPORT	cal_diy.booking.wrong_assignment_reported

An unknown trigger is an invalid delivery. The connector does not derive a fallback event kind from arbitrary provider text.

Deterministic delivery and event identity

After HMAC verification, the connector computes:

```
TEXT
delivery_digest = SHA-256(
  subscription_id || NUL ||
  webhook_version || NUL ||
  lowercase(signature_hex) || NUL ||
  raw_body_bytes
)

event_id = "evt_cal_diy_" || hex(delivery_digest)
```

Lowercasing the hexadecimal signature makes equivalent uppercase and lowercase representations produce the same event id. The digest is a replay and identity key; it is not a second authentication mechanism.

The mapped event has:

Field	Value
id	evt_cal_diy_<64 lowercase hex characters>
kind	Fixed value from the trigger table
occurred_at	Parsed createdAt or flat-form receipt fallback
session_id, action_id, correlation_id	Absent
Local upstream actor	Service principal service:cal_diy:webhook
data.triggerEvent	Original supported trigger
data.createdAt	Parsed or substituted timestamp string
data.payload	Provider payload JSON
data.webhookVersion	2021-10-20
data.subscriptionId	Validated route selector

Field	Value
`data.deliveryDigest`	Lowercase delivery digest
`data.timestampProvenance`	Current fixed value `signed_body`

Before central publication, the connector-host publisher moves the local provider actor into `data.upstream_actor`. The authenticated central ingress binds the protocol-level event actor to the connector-host signing identity. This preserves provider provenance without letting an external webhook assert a trusted AIP actor.

Replay state and exact duplicates

The standalone host uses a cluster-safe profile-state replay store. Its scope is `instance:`, so replicas for the same logical account must share the same durable runtime database and account id.

Store implementation	Intended boundary
Profile-state CAS	Standalone host and shared multi-replica scope
Atomic-replacement file	Durable single-process library use only
In-memory map	Tests or ephemeral single-process use

The store removes entries older than 86,400 seconds and retains at most 20,000 digests. At capacity it removes the oldest retained entry before inserting a new one. Profile-state admission retries a contended compare-and-set at most 32 times.

The replay store is not the final deduplication boundary. Its `false` result for a seen digest is intentionally not an ingestion error. The connector returns the same deterministic event, and the local AIP event log resolves the repeated event id idempotently.

Consequently, an exact provider retry that still passes the current version, secret, body, and timestamp checks receives normal success after durable handling. It is not rejected with HTTP `409`. The central event path can also see the same event id after a retry or crash and must deduplicate it while the event identity is retained.

Different raw bytes produce a different digest even when they describe the same provider object. This boundary deduplicates exact signed deliveries, not every semantic duplicate the provider could emit.

Durable acknowledgement and crash windows

The host acknowledges after two local durability steps: event-log append and event-outbox enqueue. The outbox attempts central delivery immediately, then a background worker owns any retained batch.

Interruption or duplicate point	Reviewed behavior
Crash before replay admission	Provider retry repeats full verification
Replay admitted, event append fails	Host returns `503`; retry recreates the same event id
Event appended, outbox enqueue fails	Host returns `503`; retry reuses the event and retries local enqueue
Outbox durable, central endpoint unavailable	Host returns `200` with `durably_queued`; worker retries
Central acknowledgement succeeds	Host returns `200` with `centrally_acknowledged`
Crash after central acknowledgement but before outbox deletion	Exact event id may be sent again; central log deduplicates
Exact provider delivery repeats within current validation windows	Same event id and deterministic outbox batch are used

This design reduces loss and duplicate effects, but it is not an exactly-once transport claim. Retention, storage health, identity preservation, and central event-log behavior remain part of the delivery guarantee.

HTTP responses

Success

The reviewed mapper produces one event. A successful response is HTTP `200`:

```
JSON
{
  "accepted_events": 1,
  "central_delivery": "durably_queued"
}
```

`central_delivery` is either `centrally_acknowledged` or `durably_queued`. Both mean the event was accepted durably by the connector host. Only the first means the central ingress acknowledged it before the HTTP response.

Failures

Condition	HTTP status	Error code
Missing or empty required header	`400`	`webhook.missing_header`
Body is not valid UTF-8	`400`	`webhook.invalid_utf8`
Unknown selector, malformed signature, or HMAC mismatch	`401`	`webhook.authentication_failed`
Unsupported version, invalid JSON, time window, or trigger	`400`	`webhook.invalid_delivery`
Replay-state, event-log, or pre-enqueue outbox storage failure	`503`	`webhook.storage_unavailable`
Connector unexpectedly maps a non-event envelope	`500`	`webhook.invalid_mapping`

A body rejected by the HTTP body-limit layer may not reach the handler's JSON error mapping. Do not depend on a connector error code for an oversized body.

Errors intentionally do not echo signatures, secret references, raw payloads, or provider PII.

Operational review checklist

Verify that:

- the external HTTPS route maps one final path segment to `subscription_id`;
- the proxy preserves the exact body and both required headers;
- each configured selector maps to one non-empty owner-controlled secret file;
- the managed subscriber URL and selector use the same reviewed binding;
- every replica for one account shares PostgreSQL profile state and event state;
- the connector-host event endpoint is configured before webhook ingress starts;
- host clocks stay within the five-minute future-skew boundary;
- raw bodies, signatures, secrets, and mapped payloads are absent from logs;
- local event-log, replay-state, outbox depth, and central delivery failures are

monitored separately;

- a recovery test covers valid, invalid-signature, stale, exact-duplicate, local

storage failure, and central-unavailable deliveries.

Passing this checklist validates deployment handling. It does not prove that a current Cal.diy account is sending every configured trigger.

Diagnose an ingress failure

Symptom	First evidence to inspect	Safe response
`webhook.missing_header`	Proxy forwarding and exact header names	Restore both headers; do not synthesize a signature
`webhook.invalid_utf8`	Raw request encoding before any JSON parser	Reject the delivery and preserve redacted transport evidence
`webhook.authentication_failed`	Route selector, secret revision, signature length, and raw-byte preservation	Compare configuration without logging secret material
`webhook.invalid_delivery`	Version, JSON form, `createdAt`, host clock, and trigger enum	Correct the subscription or compatibility mismatch
`webhook.storage_unavailable`	Replay CAS, local event log, and durable outbox	Restore local durability; allow the provider to retry `503`
Success with `durably_queued`	Outbox depth, active host lease, central route, and network	Do not ask the provider to replay; let the durable worker deliver
Repeated central event id	Local outbox recovery and central idempotent append	Treat it as expected retry evidence, not a new business event
Same business object with different event ids	Raw-body and signature differences	Handle semantic duplication at the consumer if required

Security and evidence boundaries

- HMAC covers the raw body, not the route or version header.
- The route selector chooses a deployment secret but is not itself a secret.
- A bounded replay cache does not provide permanent semantic deduplication.
- A deterministic event id does not prove exactly-once transport.
- HTTP `200` can mean durable local queueing rather than central acknowledgement.
- Event payloads remain restricted even after signature verification.
- Source and deterministic tests do not prove current external delivery.

Related documentation

- User webhook capabilities
- Event-type webhook capabilities
- Cal.diy configuration
- Security model
- Cal.diy isolated-live evidence