LawnPro Software

Webhooks Reference

Delivery format, signature verification, retries, event catalog.

Public REST API v1 · https://secure.lawnprosoftware.com/api/v1
Generated August 3, 2026 · Support 205-369-7052, 7am–7pm CST

LawnPro Webhooks — Developer Reference

Outbound webhooks push LawnPro business events to your own HTTPS endpoint the moment they happen, so an integration never has to poll `/api/v1`. This is the complete, self-contained technical spec: how to register an endpoint, the delivery format, how to verify authenticity, and the retry/health model.

> This document is the consolidated, shareable version of the webhook material in `REFERENCE.md` (§12 management, §13 receiving) and the machine-readable `openapi.yaml`. If any of these disagree, `openapi.yaml` and the running code are the source of truth.
>
> For a step-by-step walkthrough that builds a receiver from scratch — including the four rules that keep webhooks boring — see `GUIDE.md`.

- App UI: **Settings → Webhooks** (owner-only)
- Management API base: `https://secure.lawnprosoftware.com/api/v1/webhooks`
- Availability: by request, any plan (its own `webhooks` gate — you do **not** need API Access enabled, though having it also grants webhooks)

> **Two different secrets — don't confuse them:**
>
> - The **API key** (`lp_live_<prefix>.<secret>`) authenticates *your* calls to the management API (the `Authorization: Bearer …` header in every curl example below). See **Getting an API key**.
> - The **signing secret** (`whsec_…`) is per-endpoint and is used to *verify* the webhook deliveries LawnPro sends you. See §4.
>
> If you only manage webhooks in the UI (**Settings → Webhooks**), you don't need an API key at all — just create the endpoint and copy its `whsec_` secret.

---

0. Getting an API key

The webhook **management API** is part of LawnPro's REST API, so it uses the same Bearer key as everything else under `/api/v1`:

```
Authorization: Bearer lp_live_<prefix>.<secret>
```

Keys are **requested by the account owner and approved by LawnPro staff**. The owner submits a request in **Settings → API Access** naming the integration, staff approve it with the scopes it needs — for webhooks that's `webhooks:read` and/or `webhooks:write` — and the owner then clicks **Generate key**. The full key is shown **once** and can't be retrieved later; if it's lost, request a new one and revoke the old. The secret is generated in the owner's own session, so staff never see it.

Full details, auth failures, and scope rules: see REFERENCE.md — §2 Authentication and REFERENCE.md — §3

Scopes & authorization. To reach support, see REFERENCE.md — §17 Support & changelog — phone **205-369-7052**, 7am–7pm CST.

---

1. Concepts

An **endpoint** is one HTTPS URL you register, plus the list of events it subscribes to and a per-endpoint **signing secret**. When a subscribed event fires for your company, LawnPro enqueues a **delivery** (one per matching endpoint) and a background worker `POST`s the signed JSON envelope to your URL, retrying on failure.

- Fan-out is **per company** and tenant-isolated — an endpoint only ever receives events from the company that owns it.
- Webhooks fire for the business event **regardless** of whether the company uses Automations/Sequences.
- Delivery is **at least once**, best-effort in-order but **not guaranteed** in order. Design consumers to be idempotent.

---

2. Registering an endpoint

Two equivalent surfaces:

- **UI:** Settings → Webhooks → **Add endpoint** (account owner only).
- **API:** `POST /api/v1/webhooks` with scope `webhooks:write`.

Either way, the **signing secret is returned exactly once** at creation (prefix `whsec_`). It is never retrievable again; store it immediately. To rotate, delete the endpoint and create a new one.

Create (API)

```
curl -s -X POST https://secure.lawnprosoftware.com/api/v1/webhooks \
  -H "Authorization: Bearer lp_live_<prefix>.<secret>" \
  -H "Content-Type: application/json" \
  -d '{
        "url": "https://example.com/hooks/lawnpro",
        "events": ["invoice_paid_in_full", "customer_added"],
        "description": "Sync to our CRM"
      }'
```

```
// 201 Created — data.secret is present ONCE.
{
  "data": {
    "id": 7,
    "url": "https://example.com/hooks/lawnpro",
    "events": ["invoice_paid_in_full", "customer_added"],
    "description": "Sync to our CRM",
    "status": "active",
    "consecutive_failures": 0,
    "last_success_at": null,
    "last_failure_at": null,
    "created_at": "2026-07-12T15:00:00Z",
    "updated_at": "2026-07-12T15:00:00Z",
    "secret": "whsec_...48 hex chars..."
  },
  "meta": { "request_id": "req_..." },
  "errors": []
}
```

Writable fields

Field	Rules
`url`	**Required on create.** Public **HTTPS** URL. Private/loopback/reserved addresses → `422 validation_failed`.
`events`	**Required on create.** Array or CSV of event names, or `"*"` for all. Unknown names → `422`.
`description`	Optional, ≤ 255 chars.
`status`	Update only: `active` or `paused`. Paused endpoints receive nothing.

Management operations

Operation	Call	Scope
List	`GET /api/v1/webhooks`	`webhooks:read`
Get one	`GET /api/v1/webhooks/{id}`	`webhooks:read`
Recent deliveries	`GET /api/v1/webhooks/{id}/deliveries`	`webhooks:read`
Create	`POST /api/v1/webhooks`	`webhooks:write`
Update	`PUT /api/v1/webhooks/{id}`	`webhooks:write`
Delete (soft)	`DELETE /api/v1/webhooks/{id}`	`webhooks:write`

Endpoint object fields (all reads): `id`, `url`, `events[]`, `description`, `status`, `consecutive_failures`, `last_success_at`, `last_failure_at`, `created_at`, `updated_at`. `secret` appears **only** in the create response.

---

3. The delivery request

- Method: `POST`, `Content-Type: application/json`.

- Timeout: your endpoint must respond within **10 seconds**.

Headers

Header	Description
`X-LawnPro-Event`	Event name, e.g. `invoice_paid_in_full`.
`X-LawnPro-Delivery`	Unique delivery id, **stable across retries**. De-duplicate on this.
`X-LawnPro-Timestamp`	Unix timestamp the signature was computed at.
`X-LawnPro-Signature`	`sha256=<hex>` HMAC — see §4.
`User-Agent`	`LawnPro-Webhooks/1.0`

Body envelope

```
{
  "id": "evt_9f8c1b2a3d4e5f60718293a4b5c6d7e8",
  "event": "invoice_paid_in_full",
  "created_at": "2026-07-11T18:03:22Z",
  "company_id": 1234,
  "object": "invoice",
  "object_id": 987,
  "data": { "id": 987, "customer_id": 42, "total": 250.00, "balance_due": 0.00 }
}
```

Field	Meaning
`id`	Unique event id; identical across retries of the same delivery.
`event`	Event name (matches `X-LawnPro-Event`).
`created_at`	ISO-8601 UTC event time.
`company_id`	The company the event belongs to.
`object` / `object_id`	Type and id of the affected record.
`data`	The affected object in the **same shape the REST API returns** (Customer / Invoice / Estimate / Payment / Event). Degrades to a minimal `{ "id": ..., "customer_id": ... }` if the record can't be fully loaded at send time — always enough to fetch full detail from the API.

`object` values map to: `customer`, `invoice`, `estimate`, `payment`, `event`/`visit`. Unrecognized types yield the minimal `data` shape.

---

4. Verifying the signature

Every request is signed with your endpoint's secret over the **exact raw body**:

```
signature = HMAC_SHA256(secret, "<X-LawnPro-Timestamp>.<raw request body>")
```

Steps your receiver MUST perform:

1. Read the **raw** body bytes (do not re-serialize parsed JSON — whitespace differences will break the HMAC).
2. Compute `sha256=` + `hash_hmac('sha256', timestamp . '.' . rawBody, secret)`.
3. Compare to `X-LawnPro-Signature` in **constant time**.
4. Reject if it doesn't match, or if `abs(now - timestamp) > 300` seconds (replay guard).
5. Only then process, and return `2xx`.

PHP

```
$payload   = file_get_contents('php://input');
$timestamp = $_SERVER['HTTP_X_LAWNPRO_TIMESTAMP'] ?? '';
$sigHeader = $_SERVER['HTTP_X_LAWNPRO_SIGNATURE'] ?? '';
$expected  = 'sha256=' . hash_hmac('sha256', $timestamp . '.' . $payload, $YOUR_SECRET);

if (!hash_equals($expected, $sigHeader) || abs(time() - (int) $timestamp) > 300) {
    http_response_code(400);
    exit('invalid signature');
}

$event = json_decode($payload, true);
// ...handle $event idempotently, keyed on $_SERVER['HTTP_X_LAWNPRO_DELIVERY']...

http_response_code(200);
```

Node (Express, raw body)

```
const crypto = require("crypto");

// Capture the raw body for HMAC (express.json's `verify` hook):
app.use(express.json({ verify: (req, _res, buf) => { req.rawBody = buf; } }));

function verify(req, secret) {
  const ts  = req.get("X-LawnPro-Timestamp") || "";
  const sig = req.get("X-LawnPro-Signature") || "";
  const expected = "sha256=" + crypto
    .createHmac("sha256", secret)
    .update(ts + "." + req.rawBody.toString("utf8"))
    .digest("hex");
  let ok = false;
  try { ok = crypto.timingSafeEqual(Buffer.from(expected), Buffer.from(sig)); } catch (_) {}
  return ok && Math.abs(Date.now() / 1000 - Number(ts)) < 300;
}

app.post("/hooks/lawnpro", (req, res) => {
  if (!verify(req, process.env.LAWNPRO_WEBHOOK_SECRET)) return res.sendStatus(400);
  // ...handle req.body idempotently on req.get("X-LawnPro-Delivery")...
  res.sendStatus(200);
});
```

Python (Flask)

```
import hmac, hashlib, time
from flask import request, abort

def verify(secret: str) -> bool:
    ts  = request.headers.get("X-LawnPro-Timestamp", "")
    sig = request.headers.get("X-LawnPro-Signature", "")
    raw = request.get_data()  # bytes, unparsed
    expected = "sha256=" + hmac.new(
        secret.encode(), f"{ts}.".encode() + raw, hashlib.sha256
    ).hexdigest()
    if not hmac.compare_digest(expected, sig):
        return False
    try:
        return abs(time.time() - int(ts)) < 300
    except ValueError:
        return False
```

---

5. Responses, retries & delivery semantics

- **Acknowledge with any `2xx`** within 10s. The response body is ignored.
- **Failure** = non-2xx, timeout, connection error, or DNS failure. Failed deliveries are retried on an increasing backoff:

Attempt	Sent after the previous attempt failed
1	(immediately, when the event fires)
2	1 minute
3	5 minutes
4	30 minutes
5	2 hours
6	6 hours

- **Max 6 attempts** (so 5 retries, ~8½ hours of backoff end to end). After the 6th attempt fails the delivery is **dead-lettered** (terminal `dead` status, visible in the deliveries health view) and is not retried again.
- **Ordering is not guaranteed.** Handle events by their own `created_at` / object state, not arrival order.
- **At least once.** A delivery can arrive more than once (e.g. your 2xx was lost, or a worker crashed after sending). **De-duplicate on `X-LawnPro-Delivery`** (or the body `id`).
- **Pausing** an endpoint defers its pending deliveries; **deleting** it retires them. Deliveries to an inactive/deleted endpoint are dropped.

Delivery health

`GET /api/v1/webhooks/{id}/deliveries` (or the UI) returns the 20 most recent attempts:

Field	Meaning
`id`	Delivery row id.
`endpoint_id`	The endpoint.
`event`	Event name.
`delivery_id`	The `X-LawnPro-Delivery` value.
`status`	`pending` | `processing` | `delivered` | `dead`.
`attempts`	Attempts made so far.
`last_code`	Last HTTP status received (0 = no response).
`last_error`	Last error string, if any.
`created_at` / `updated_at`	Timestamps.

The endpoint object also carries `consecutive_failures` (resets to 0 on any success), `last_success_at`, and `last_failure_at` for at-a-glance health.

---

6. Security

- **HTTPS only.** Non-HTTPS URLs are rejected.
- **SSRF protection.** The destination is validated at save time *and* re-resolved and re-checked at delivery time; any host resolving to a private, loopback, or reserved address (IPv4 or IPv6, including embedded/mapped IPv4) is refused. Unresolvable hosts are refused, not risked. The resolved public IP is pinned for the request.
- **Per-endpoint secret**, shown once at creation (`whsec_` + 48 hex chars). Treat it like a password; rotate by recreating the endpoint.
- **Tenant isolation.** Endpoints and deliveries are scoped to the owning company on every read and write.
- **Management is owner-only** in the UI, and requires the `webhooks:write` scope via the API.

---

7. Event catalog

Subscribe by exact event name, or `"*"` for all events — with the one exception noted under the table. The authoritative, always-current list is the event picker in **Settings → Webhooks** (it mirrors the Automation trigger catalog).

Group	Events
Estimates	`estimate_create`, `estimate_draft`, `estimate_sent`, `estimate_service`, `estimate_accept`, `estimate_declined`, `estimate_changes`, `estimate_responded`, `estimate_not_responded`, `estimate_expiring`
Invoices	`invoice_created`, `invoice_sent`, `invoice_past_due`, `before_invoice_past_due`, `invoice_paid_in_full`, `invoice_partially_paid`, `invoice_service`
Payments	`payment_to_account`, `payment_with_tip`, `payment_on_portal_by_customer`, `payment_added_to_invoice`, `payment_receipt_sent`, `credit_card_failed`
Visits	`visit_added`, `visit_completed`, `visit_skipped`, `visit_cancelled`, `before_visit`, `visit_note_added`, `property_not_serviced`
Customers	`customer_added`, `customer_reactivated`, `customer_cancelled`, `customer_has_tag`, `customer_added_credit_card`, `customer_added_bank_account`, `customer_removed_credit_card`, `customer_removed_bank_account`, `customer_inactive_30_days`, `customer_inactive_60_days`, `customer_inactive_90_days`, `credit_card_expiring`
Reviews	`customer_left_review`, `customer_left_good_review`, `customer_left_bad_review`, `customer_no_review`
Work Requests	`work_request_received`, `work_request_converted`
Tasks & Meetings	`todo_created`, `todo_completed`, `todo_has_category`, `before_task`, `meeting_created`, `meeting_completed`, `meeting_has_category`, `before_meeting`
Timer	`timer_started`, `timer_stopped`
Tags	`tag_added`, `tag_removed`
Contracts & Recurring	`contract_expiring`, `recurring_job_created`, `recurring_job_cancelled`
Customer scoring *(Plus, explicit subscription only)*	`customer_score_at_risk`, `customer_churn_risk_flagged`

The two customer-scoring events are the exception to `"*"`. They are the only events that come from a nightly model rather than something a person did, so an endpoint must name them outright — a bare `"*"` will not receive them. This keeps a long-standing catch-all endpoint from suddenly firing whatever it feeds on a score change its owner never subscribed to. They are also emitted only for Plus accounts.

To get everything *and* the scoring events, send both — e.g. `["*", "customer_score_at_risk"]`. The scoring name is kept alongside the wildcard rather than collapsed into it. Any other name sent beside `"*"` is redundant and dropped, since `"*"` already covers it.

Some events are scan-based (e.g. "inactive N days", "expiring") and are evaluated on a schedule rather than in direct response to a user action. LawnPro suppresses duplicate deliveries for the same endpoint/event/object in those cases, but consumers should still de-duplicate on `X-LawnPro-Delivery`.

8. Testing your endpoint

Send a test event

In **Settings → Webhooks**, the paper-plane button on an endpoint row queues a one-off **`ping`** delivery. It goes through the real pipeline — same signing secret, same SSRF and timeout rules, same delivery log — so a `ping` that shows as **delivered** is genuine proof the endpoint works, not a simulation. Watch the result in **Recent deliveries** with the real HTTP status code.

Details worth knowing:

- Account **owner** only, and only on **active** (not paused) endpoints.
- Limited to **one test per endpoint per minute**.
- **One attempt, no retries** (unlike real events, which retry six times). A failed test stays failed — fix your receiver and send another.
- It arrives on the worker's next run, so allow up to about a minute.

The `ping` event

`ping` is **not** in the event catalog and no business logic ever emits it — it is only ever sent when someone presses the test button. It's delivered **regardless of your subscription list**, so your handler must tolerate an event name it didn't subscribe to (good practice anyway: ignore unknown events rather than erroring on them).

```
{
  "id": "evt_9f2c…",
  "event": "ping",
  "created_at": "2026-07-27T18:20:11Z",
  "company_id": 1234,
  "object": "",
  "object_id": 0,
  "data": {
    "test": true,
    "source": "settings",
    "message": "Test event from LawnPro. …"
  }
}
```

Because it carries a real signature, a `ping` is the easiest way to exercise your signature-verification code (§4) before any live data flows.

While you're still building

Point the endpoint at a hosted request bin — webhook.site is the quickest (it hands you a URL and shows the full headers and body), and Svix Play will also check an HMAC signature for you.

You **cannot** point an endpoint at `localhost` or any private/reserved address: the worker re-resolves the host at delivery time and refuses non-public destinations (§6). For local development, expose your machine through a tunnel such as ngrok and register that HTTPS URL.

9. Implementation checklist

- ☐ Endpoint is public **HTTPS**.
- ☐ Reads the **raw** body before parsing.
- ☐ Verifies `X-LawnPro-Signature` (constant-time) and the timestamp window.
- ☐ Returns `2xx` in **< 10s**; does heavy work **after** acknowledging (queue it).
- ☐ Is **idempotent**, keyed on `X-LawnPro-Delivery`.
- ☐ Tolerates **out-of-order** and **duplicate** deliveries.
- ☐ Stores the `whsec_` secret securely at creation (it's shown once).
- ☐ Monitors `consecutive_failures` / the deliveries view for health.
- ☐ Ignores unknown event names (including `ping`) instead of erroring.