LawnPro Software

API Reference

Every endpoint, field, filter and enumeration.

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

LawnPro Public REST API v1 — Developer Reference

The complete reference for the LawnPro public REST API. For a step-by-step walkthrough of building an integration see `GUIDE.md`; for a fast overview see `README.md`; for the machine-readable contract see `openapi.yaml` (OpenAPI 3.1). This document is the authoritative, human-readable description of every endpoint, field, and behavior.

- **Base URL:** `https://secure.lawnprosoftware.com/api/v1`
- **Content type:** `application/json` (request and response)
- **Auth:** Bearer API key (see Authentication)
- **Versioning:** the version is in the path (`/api/v1`). Breaking changes ship under a new path (`/api/v2`); v1 stays stable.

Table of contents

1. Conventions

Topic	Rule
Field names	The public field names in this document are the contract. Internal database column names are **never** exposed and may differ.
IDs	All resource IDs are integers.
Dates/times	Timestamps are ISO-8601 **UTC** (`2026-06-29T13:45:00Z`). Date-only fields are `YYYY-MM-DD`. A zeroed/empty date is returned as `null`.
Money	JSON numbers rounded to 2 decimals (e.g. `125.00`). Never strings.
Booleans	Real JSON `true`/`false` in responses, even though some are stored internally as `1`/`2`.
Unknown fields	Unknown fields in a request body are ignored. Only documented fields are written.
Partial updates	`PUT`/`PATCH` are treated as partial updates: only the fields you send are changed. Omitted fields are left untouched.
Request size	Request bodies are capped at **1 MB**. Larger bodies get `413 payload_too_large`.

2. Authentication

Every request must send an API key as a Bearer token:

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

A key has two parts joined by a dot:

- `<prefix>` — a public identifier (stored, indexed, safe to log).
- `<secret>` — the secret half. Only a bcrypt hash of it is stored, so a database leak cannot reveal usable keys.

How keys are issued

Keys are **requested by the merchant and approved by LawnPro staff**:

1. The **account owner** (`user_role = 1`) submits a request in **Settings → API Access → Request an API key**, giving it a label and describing what the integration does. Only one open request per company at a time; the owner can cancel a pending one.
2. **LawnPro staff approve** the request in the Control Center, setting its scopes and rate limit. The request shows as *Awaiting approval* until then, and staff may decline it.
3. The owner clicks **Generate key** on the approved row. The full key is displayed **once**, at that moment, and can never be retrieved again. If it's lost, request a new key and revoke the old one.

Key material is generated in the owner's own browser session at claim time, so **LawnPro staff never see the secret** — they approve the request, not the key. Only the public prefix and a bcrypt hash of the secret

are stored.

> **Availability:** the API Access page is gated by the `api_access` feature flag, not by plan tier — it is a pilot surface. If the page isn't visible, the account hasn't been enabled; contact support (see Support).

Auth failures

Situation	Response
No `Authorization` header / not a Bearer token	`401 unauthorized`
Malformed key (no `prefix.secret` shape)	`401 unauthorized`
Unknown prefix, wrong secret, or revoked key	`401 unauthorized`
More than 30 failed auth attempts from one IP in 60s	`429 rate_limited` + `Retry-After`
Company account suspended	`403 forbidden`
Plain HTTP in production	`403 https_required`

Failed authentication is throttled **per client IP** (30 per 60-second window) so a known prefix can't be brute-forced. A retry loop around a bad key will therefore start receiving `429` instead of `401` — never retry a `401`.

> Keys do not expire on a timer; they remain valid until revoked. Only keys in `active` status with no revocation timestamp authenticate.

---

3. Scopes & authorization

Each key is granted a set of **scopes** shaped as `resource:action`. A read scope never implies write.

Resource	Read scope	Write scope
Customers	`customers:read`	`customers:write`
Properties	`properties:read`	`properties:write`
Items & Services	`items:read`	`items:write`
Invoices	`invoices:read`	*(read-only — no write)*
Estimates	`estimates:read`	*(read-only — no write)*
Payments	`payments:read`	*(read-only — no write)*
Schedule	`schedule:read`	*(read-only — no write)*
Webhooks	`webhooks:read`	`webhooks:write`

> Invoices, estimates, payments, and schedule are **read-only** in v1. The `:write` scopes for those resources are reserved for future use and currently grant nothing.

A call that lacks the required scope returns `403 insufficient_scope` with a message naming the missing scope, e.g.:

```
{ "data": null, "meta": { "request_id": "req_..." },
  "errors": [ { "code": "insufficient_scope",
                "message": "This API key is missing the required scope: customers:write" } ] }
```

Least privilege: request only the scopes the integration actually needs. A read-only reporting tool should get only `*:read` scopes.

---

4. Tenant isolation

A key is permanently bound to exactly one company. Every query is automatically and irrevocably scoped to that company:

- You can only read or modify your own company's records.
- Cross-resource references are validated against your account. For example, creating a property with a `customer_id` that isn't yours returns `422 validation_failed`; fetching another company's record returns `404 not_found` (it simply doesn't exist as far as your key is concerned).

There is no way to widen a key's scope to another company.

---

5. The response envelope

Every response — success or error — uses the same envelope:

```
{
  "data":   <object | array | null>,
  "meta":   { "request_id": "req_..." },
  "errors": []
}
```

- `data` — the resource object, an array of objects (lists), or `null` on error.
- `meta.request_id` — a unique id for this call, also returned as the `X-Request-Id` response header. Include it when contacting support.
- `meta.pagination` — present on list responses (see Pagination).
- `errors` — empty `[]` on success; one or more error objects on failure.

Success (single object)

```
{
  "data": { "id": 42, "first_name": "Jane", "last_name": "Doe" },
  "meta": { "request_id": "req_2f1c9a7b" },
  "errors": []
}
```

Success (collection)

```
{
  "data": [ { "id": 42 }, { "id": 41 } ],
  "meta": {
    "request_id": "req_2f1c9a7b",
    "pagination": { "page": 1, "per_page": 50, "total": 213, "total_pages": 5 }
  },
  "errors": []
}
```

---

6. Errors

On failure, `data` is `null` and `errors` contains one or more objects:

```
{
  "data": null,
  "meta": { "request_id": "req_2f1c9a7b" },
  "errors": [
    { "code": "validation_failed", "message": "Invalid email address.", "fields": ["email"] }
  ]
}
```

Field	Description
`code`	Stable machine-readable error code (table below). Branch on this, not on `message`.
`message`	Human-readable explanation. May change wording over time.
`fields`	*(optional)* The request fields that failed validation.

Error codes

code	HTTP	Meaning
`invalid_json`	400	Body was not valid JSON.
`unauthorized`	401	Missing, malformed, invalid, or revoked API key.
`forbidden`	403	Account suspended.
`https_required`	403	Request was plain HTTP in production.
`insufficient_scope`	403	Key lacks the scope required for this call.
`not_found`	404	Resource doesn't exist in your account.

code	HTTP	Meaning
`method_not_allowed`	405	HTTP method not supported for this resource (e.g. creating a read-only resource).
`idempotency_conflict`	409	A create with this `Idempotency-Key` is still in flight. Retry shortly.
`payload_too_large`	413	Request body exceeded 1 MB.
`validation_failed`	422	Input failed validation (see `fields`).
`rate_limited`	429	Rate limit exceeded, or too many failed auth attempts from your IP. See `Retry-After`.
`server_error`	500	Unexpected server error. Safe to retry idempotently.
`rate_limiter_unavailable`	503	The rate limiter itself errored; the request was rejected (fail-closed). Honor `Retry-After` and retry.

---

7. HTTP status codes

Code	When
`200 OK`	Successful read, update, or delete.
`201 Created`	Successful create.
`400 Bad Request`	Invalid JSON body.
`401 Unauthorized`	Authentication failed.
`403 Forbidden`	Suspended account, missing scope, or HTTPS required.
`404 Not Found`	Resource not in your account.
`405 Method Not Allowed`	Unsupported method for the resource.
`409 Conflict`	Idempotency key still processing.
`413 Payload Too Large`	Body over 1 MB.
`422 Unprocessable Entity`	Validation error.
`429 Too Many Requests`	Rate limited, or auth-failure throttle tripped.
`500 Internal Server Error`	Server-side failure.
`503 Service Unavailable`	Rate limiter unavailable (fail-closed). Retry after `Retry-After`.

---

8. Rate limiting

Limits are enforced **per key, per minute** (fixed window). The default is **120 requests/minute**; support can raise it per key (up to 6000/min).

Every response includes:

```
X-RateLimit-Limit: 120
X-RateLimit-Remaining: 118
```

When you exceed the limit you get `429 rate_limited` plus:

```
Retry-After: 37
```

Wait the indicated number of seconds, then retry. Build clients to honor `Retry-After` and back off rather than hammering the endpoint.

> **The limiter fails closed.** If its counter hits a transient error, the request is **rejected**, not allowed through — you get `503` with code `rate_limiter_unavailable` and `Retry-After: 5`. The limiter is a security control, so it is never silently disabled. Clients must treat `503 rate_limiter_unavailable` the same as `429`: honor `Retry-After` and retry.

---

9. Pagination

List endpoints accept:

Query param	Default	Max	Notes
`page`	`1`	—	1-indexed.
`per_page`	`50`	`100`	Values above 100 are clamped to 100.

The `meta.pagination` block tells you how to page through results:

```
"pagination": { "page": 2, "per_page": 50, "total": 213, "total_pages": 5 }
```

Iterate until `page` reaches `total_pages`.

---

10. Idempotency (safe create retries)

Network failures make it unsafe to blindly retry a `POST` — you might create the same record twice. To retry safely, send an **`Idempotency-Key`** header on create calls with a unique value you generate (a UUID is ideal):

```
Idempotency-Key: 7c2b9f04-2f1a-4e3a-9a1d-6f5e2b8c1d22
```

Behavior:

- **First request** runs normally and the result is stored against the key.
- **A retry with the same key** replays the original stored response instead of creating a duplicate.

Replays carry the header `Idempotent-Replayed: true`.

- **Scope:** the key is scoped to the **endpoint** (method + path). The same key value used on a *different* endpoint is treated as a separate operation, not a replay.
- **Concurrency:** if two requests with the same key arrive at once, only one performs the create. The other receives `409 idempotency_conflict` while the first is still in flight (retry shortly), and a normal replay once it finishes.
- **Failures are not cached.** If a create returns an error (4xx/5xx), the key is released so you can retry the operation cleanly — you won't be stuck replaying a failure. Only successful (2xx) responses are stored for replay.

> Keep the same `Idempotency-Key` for all retries of one logical create; generate a fresh key for a genuinely new create.

---

11. Security & transport

- **HTTPS only.** Production rejects plain HTTP with `403 https_required` and sends `Strict-Transport-Security`. Always call `https://`.
- **Keep keys secret.** Treat the key like a password. Store it in a secrets manager / environment variable, never in client-side code or a public repo.
- **Rotate on exposure.** If a key may have leaked, have support revoke it and issue a new one. Revocation is immediate.
- **Scope minimally.** Use read-only keys wherever possible.
- **Audit trail.** Every API call is logged server-side with its `request_id`, so support can trace a specific request you reference.

---

12. Resources

Common conventions for all resources:

- List: `GET /{resource}` — paginated, returns a collection envelope.
- Get one: `GET /{resource}/{id}` — `404 not_found` if not yours.
- Create: `POST /{resource}` — `201` on success; supports `Idempotency-Key`.
- Update: `PUT /{resource}/{id}` — partial update; only sent fields change.
- Delete: `DELETE /{resource}/{id}` — soft delete; returns `{ id, deleted: true }`.

Read-only resources (invoices, estimates, payments, schedule) support only the List and Get operations.

Customers

Scopes: `customers:read`, `customers:write` · **Operations:** list, get, create, update, delete

Fields (response)

Field	Type	Notes
`id`	integer	
`company_name`	string	
`first_name`	string	
`last_name`	string	
`email`	string	
`phone`	string	
`mobile`	string	
`address`	string	
`address2`	string	
`city`	string	
`state`	string	
`postal_code`	string	
`county`	string	
`country`	string	ISO country code (defaults to `US`).
`customer_number`	string	Human-facing customer number.
`type_id`	integer|null	Customer type.
`status`	integer|null	See Customer status.
`tax_exempt`	boolean	
`created_at`	datetime|null	
`updated_at`	datetime|null	

Writable fields (create/update)

`company_name`, `first_name`, `last_name`, `email`, `phone`, `mobile`, `address`, `address2`, `city`, `state`, `postal_code`, `county`, `country`, `type_id`, `status`, `tax_exempt`, plus:

- `trigger_automations` *(boolean, default `true`)* — set `false` on bulk imports to suppress the "customer added" automations for that record.

Validation:

- Provide **at least one** of `company_name`, `first_name`, or `last_name`.
- `email`, if provided, must be a valid address.
- Subject to your plan's customer limit; exceeding it returns `403 forbidden`.

List filters

Query param	Notes
`status`	Filter by customer status (e.g. `1` active, `6` lead). `0` is honored.
`email`	Exact email match.

Examples

Create:

```
curl -s -X POST https://secure.lawnprosoftware.com/api/v1/customers \
  -H "Authorization: Bearer lp_live_<prefix>.<secret>" \
  -H "Content-Type: application/json" \
  -H "Idempotency-Key: 8f14e45f-ceae-4d3b-9a1e-1a2b3c4d5e6f" \
  -d '{
        "first_name": "Jane",
        "last_name": "Doe",
        "email": "jane@example.com",
        "phone": "205-555-0100",
        "address": "123 Oak St",
        "city": "Birmingham",
        "state": "AL",
        "postal_code": "35203"
      }'
```

Update (partial):

```
curl -s -X PUT https://secure.lawnprosoftware.com/api/v1/customers/42 \
  -H "Authorization: Bearer lp_live_<prefix>.<secret>" \
  -H "Content-Type: application/json" \
  -d '{ "phone": "205-555-0199" }'
```

Properties

Scopes: `properties:read`, `properties:write` · **Operations:** list, get, create, update, delete

A property is a service location belonging to a customer.

Fields (response)

Field	Type	Notes
`id`	integer	
`customer_id`	integer	Owning customer.
`name`	string	Optional label.
`address`	string	
`address2`	string	
`city`	string	
`state`	string	

Field	Type	Notes
`postal_code`	string	
`county`	string	
`country`	string	
`latitude`	string|null	
`longitude`	string|null	
`size`	number|null	Lot/turf size.
`size_type`	integer|null	Unit of the size value.
`notes`	string	
`status`	integer|null	
`created_at`	datetime|null	
`updated_at`	datetime|null	

Writable fields (create/update)

`customer_id` *(required on create)*, `name`, `address`, `address2`, `city`, `state`, `postal_code`, `county`, `country` (default `US`), `size`, `size_type`, `notes`, `status`, `latitude`, `longitude`.

Validation: `customer_id` must reference a customer in your account.

List filters

Query param	Notes
`customer_id`	List only properties for a given customer.

Example

```
curl -s -X POST https://secure.lawnprosoftware.com/api/v1/properties \
  -H "Authorization: Bearer lp_live_<prefix>.<secret>" \
  -H "Content-Type: application/json" \
  -d '{ "customer_id": 42, "address": "123 Oak St", "city": "Birmingham",
        "state": "AL", "postal_code": "35203", "size": 7500 }'
```

Items & Services

Scopes: `items:read`, `items:write` · **Operations:** list, get, create, update, delete

Your catalog of line items / services.

Fields (response)

Field	Type	Notes
`id`	integer	
`name`	string	

Field	Type	Notes
`category_id`	integer|null	Service category.
`price`	number	
`quantity`	number|null	
`type`	integer|null	
`unit`	string	
`description`	string	
`tax1`	number	
`tax2`	number	
`material_cost`	number	

Writable fields (create/update)

`name` *(required on create)*, `category_id`, `price`, `quantity`, `type`, `unit`, `description`, `tax1`, `tax2`, `material_cost`.

Validation:

- `name` is required on create.
- `category_id`, if provided, must belong to your account. On create, if omitted, it defaults to your account's first category (a create fails with `422 validation_failed` if no category exists yet).
- On update you may change `category_id` alone.

List filters

Query param	Notes
`category_id`	List only items in a given category.

Invoices (read-only)

Scope: `invoices:read` · **Operations:** list, get

Fields (response)

Field	Type	Notes
`id`	integer	
`number`	integer|null	Invoice number.
`customer_id`	integer	
`property_id`	integer|null	
`po_number`	string	
`title`	string	
`date`	date|null	
`status`	integer|null	See Invoice status.

Field	Type	Notes
`subtotal`	number	
`discount`	number	
`tax`	number	
`total`	number	
`paid_amount`	number	
`balance_due`	number	`total − paid_amount`.
`notes`	string	
`terms`	string	
`updated_at`	datetime|null	

List filters

Query param	Notes
`customer_id`	Invoices for one customer.
`status`	Filter by invoice status (`0` honored).

Estimates (read-only)

Scope: `estimates:read` · **Operations:** list, get

Fields (response)

Field	Type	Notes
`id`	integer	
`number`	integer|null	
`customer_id`	integer	
`title`	string	
`date`	date|null	
`status`	integer|null	See Estimate status.
`subtotal`	number	
`discount`	number	
`tax`	number	
`total`	number	
`paid_amount`	number	Deposit/amount applied.
`invoice_id`	integer|null	Set once converted to an invoice.
`notes`	string	
`terms`	string	
`created_at`	datetime|null	

Field	Type	Notes
`updated_at`	datetime|null	

List filters

Query param	Notes
`customer_id`	Estimates for one customer.
`status`	Filter by estimate status (`0` honored).

Payments (read-only)

Scope: `payments:read` · **Operations:** list, get

Fields (response)

Field	Type	Notes
`id`	integer	
`customer_id`	integer|null	
`invoice_id`	integer|null	
`method_id`	integer|null	See Payment method.
`date`	date|null	
`amount`	number	
`applied_to_invoice`	number	
`credit_amount`	number	
`tip_amount`	number	
`is_online`	boolean	
`is_refunded`	boolean	
`notes`	string	

List filters

Query param	Notes
`customer_id`	Payments for one customer.
`invoice_id`	Payments applied to one invoice.

Schedule / Visits (read-only)

Scope: `schedule:read` · **Operations:** list, get

Returns visits/events. Cancelled visits are not exposed and cannot be fetched by id.

Fields (response)

Field	Type	Notes
`id`	integer	
`customer_id`	integer|null	
`property_id`	integer|null	
`type`	integer|null	See Event type.
`title`	string	
`description`	string	
`date`	date|null	Visit date.
`end_date`	date|null	For multi-day events.
`has_time`	boolean	Whether a specific time is set.
`time_start`	string|null	
`time_end`	string|null	
`is_completed`	boolean	
`is_recurring`	boolean	
`recurring_id`	integer|null	The recurring series, if any.
`invoice_id`	integer|null	Linked invoice, if any.
`cost`	number	

List filters

Query param	Notes
`customer_id`	Visits for one customer.
`type`	Event type (`1`=Visit, `2`=Task, `3`=Meeting, `4`=Maintenance).
`from`	Start date (`YYYY-MM-DD`), inclusive.
`to`	End date (`YYYY-MM-DD`), inclusive.

Webhooks (management)

Scopes: `webhooks:read`, `webhooks:write` · **Operations:** list, get, create, update, delete

Manage the HTTPS endpoints that receive real-time event notifications. For the delivery format your endpoint must handle, see Webhooks (receiving events).

Fields (response)

Field	Type	Notes
`id`	integer	
`url`	string	The HTTPS destination.

Field	Type	Notes
`events`	string[]	Subscribed event names, or `["*"]` for all events except the customer-scoring pair, which must be named outright.
`description`	string	Optional label.
`status`	string	`active` or `paused`. Paused endpoints receive nothing.
`consecutive_failures`	integer	Resets to 0 after any success.
`last_success_at`	datetime|null	
`last_failure_at`	datetime|null	
`created_at`	datetime|null	
`updated_at`	datetime|null	

On **create only**, the response also includes `secret` — the signing secret, shown **once**. Store it; it is never returned again.

Writable fields (create/update)

Field	Notes
`url`	*(required on create)* Public HTTPS URL. Private/loopback/reserved addresses are rejected with `422`.
`events`	*(required on create)* Array or CSV of event names, or `"*"` for all events. `"*"` does **not** cover `customer_score_at_risk` or `customer_churn_risk_flagged` — name those explicitly, optionally alongside `"*"` (e.g. `["*", "customer_score_at_risk"]`). Other names sent beside `"*"` are redundant and dropped. See the event catalog.
`description`	Optional, ≤ 255 chars.
`status`	*(update only)* `active` or `paused`.

Extra endpoint

- `GET /webhooks/{id}/deliveries` — the 20 most recent delivery attempts for one endpoint (a health view). Each item: `event`, `status`, `attempts`, `last_code`, `last_error`, timestamps.

Example — create

```
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; data.secret is present ONCE — save it.
```

---

13. Webhooks (receiving events)

Webhooks push events to **your** server the moment they happen, so you don't have to poll the API. Register one or more HTTPS endpoints (via the management API or **Settings → Webhooks** in the app), choose the events you care about, and LawnPro will `POST` a signed JSON message to each endpoint when a matching event fires.

Delivery request

- Method: `POST` with `Content-Type: application/json`.
- Fired for every company that has a matching active endpoint — independent of whether the company uses Automations/Sequences.

Headers

Header	Description
`X-LawnPro-Event`	The event name (e.g. `invoice_paid_in_full`).
`X-LawnPro-Delivery`	Unique id for this delivery (stable across retries). Use it to de-duplicate.
`X-LawnPro-Timestamp`	Unix timestamp the signature was computed at.
`X-LawnPro-Signature`	`sha256=<hex>` — see Verifying.

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, "...": "..." }
}
```

- `id` — unique event id, stable across retries of the same delivery.
- `event` — the event name (same as `X-LawnPro-Event`).
- `object` / `object_id` — the type and id of the affected record.
- `data` — the affected object in the **same shape the REST API returns** for it (a Customer, Invoice, Estimate, Payment, or Event). If the object can't be fully loaded at send time, `data` degrades to a minimal `{ "id": ..., "customer_id": ... }` — always enough to fetch full detail from the REST API.

Verifying the signature

Every request is signed with your endpoint's secret so you can prove it came from LawnPro and wasn't tampered with. The signature covers the **exact raw body**:

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

Compare it (constant-time) against the hex in `X-LawnPro-Signature` (after the `sha256=` prefix). Reject the request if it doesn't match, or if the timestamp is older than a few minutes (guards against replay).

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');
}
http_response_code(200); // acknowledge
```

Node

```
const crypto = require("crypto");
function verify(rawBody, timestamp, sigHeader, secret) {
  const expected = "sha256=" + crypto.createHmac("sha256", secret)
    .update(timestamp + "." + rawBody).digest("hex");
  const ok = crypto.timingSafeEqual(Buffer.from(expected), Buffer.from(sigHeader));
  return ok && Math.abs(Date.now() / 1000 - Number(timestamp)) < 300;
}
```

Responding, retries & ordering

- **Respond fast with any `2xx`** (within 10 seconds) to acknowledge. The body is ignored.
- Any non-2xx, timeout, or network error is **retried with backoff**: 1 min, 5 min, 30 min, 2 h, 6 h — **6 attempts total** (the first plus five retries), spanning about 8½ hours. After the sixth attempt fails the delivery is **dead-lettered** (visible in the deliveries health view) and never retried.
- Deliveries may arrive **out of order** and, rarely, **more than once**. Treat handling as idempotent and de-duplicate on `X-LawnPro-Delivery` (or the body `id`).
- Deleting or pausing an endpoint stops (and retires) its pending deliveries.

Security

- Endpoints **must be HTTPS**. Private, loopback, and reserved addresses are rejected at save time and re-checked at delivery time (SSRF protection).
- The signing secret is shown once at creation. If it leaks, delete the endpoint and create a new one.

Event catalog

Event names mirror the automation triggers. Common ones:

Group	Events
Estimates	`estimate_create`, `estimate_sent`, `estimate_accept`, `estimate_declined`, `estimate_responded`, `estimate_not_responded`, `estimate_expiring`

Group	Events
Invoices	`invoice_created`, `invoice_sent`, `invoice_past_due`, `before_invoice_past_due`, `invoice_paid_in_full`, `invoice_partially_paid`
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`
Customers	`customer_added`, `customer_reactivated`, `customer_cancelled`, `customer_added_credit_card`, `credit_card_expiring`
Reviews	`customer_left_review`, `customer_left_good_review`, `customer_left_bad_review`
Work Requests	`work_request_received`, `work_request_converted`

Subscribe to `"*"` to receive **all** events. The authoritative, always-current list is the event picker in **Settings → Webhooks**.

---

14. Enumerations

Integer code fields map to the following values.

Customer status

Value	Meaning
0	Inactive
1	Active
2	Written Off (balance remaining)
5	Pending Approval
6	Lead

Invoice status

Value	Meaning
1	Draft
2	Paid
3	Partial
4	Past Due
5	Pending
6	Written Off

Estimate status

Value	Meaning
1	Draft

Value	Meaning
2	Accepted
3	Declined
4	Invoiced
5	Pending
6	Changes Requested
7	Changes Updated

Event type

Value	Meaning
1	Visit to Customer
2	Task
3	Meeting
4	Maintenance

Payment method

Value	Method	Value	Method
1	Cash	11	Venmo
2	Check	12	Zelle
3	Credit Card	13	CashApp
4	Customer Credit Balance	14	Apple Pay
5	ACH Bank Transfer	15	Electronic Funds Transfer
6	Paypal	16	Debit
7	Other	…	(additional internal methods exist)
8	Debit Card		
9	Direct Debit		
10	BACS		

---

15. Code samples

curl

```
# List active customers (page 1)
curl -s "https://secure.lawnprosoftware.com/api/v1/customers?status=1&per_page=50" \
  -H "Authorization: Bearer lp_live_<prefix>.<secret>"
```

PHP

```php
<?php
$key = getenv('LAWNPRO_API_KEY'); // lp_live_<prefix>.<secret>
$ch  = curl_init('https://secure.lawnprosoftware.com/api/v1/customers');
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER     => [
        'Authorization: Bearer ' . $key,
        'Content-Type: application/json',
        'Idempotency-Key: ' . bin2hex(random_bytes(16)),
    ],
    CURLOPT_POST       => true,
    CURLOPT_POSTFIELDS => json_encode([
        'first_name' => 'Jane',
        'last_name'  => 'Doe',
        'email'      => 'jane@example.com',
    ]),
]);
$response = json_decode(curl_exec($ch), true);
curl_close($ch);

if (!empty($response['errors'])) {
    throw new RuntimeException($response['errors'][0]['message']);
}
echo "Created customer #" . $response['data']['id'];
```

Python (requests)

```python
import os, uuid, requests

BASE = "https://secure.lawnprosoftware.com/api/v1"
key  = os.environ["LAWNPRO_API_KEY"]  # lp_live_<prefix>.<secret>
headers = {"Authorization": f"Bearer {key}"}

# Create a customer (idempotent)
resp = requests.post(
    f"{BASE}/customers",
    headers={**headers, "Idempotency-Key": str(uuid.uuid4())},
    json={"first_name": "Jane", "last_name": "Doe", "email": "jane@example.com"},
    timeout=30,
)
body = resp.json()
if body["errors"]:
    raise RuntimeError(body["errors"][0]["message"])
customer_id = body["data"]["id"]

# Page through all properties for that customer
page = 1
while True:
    r = requests.get(f"{BASE}/properties",
                     headers=headers,
                     params={"customer_id": customer_id, "page": page, "per_page": 100},
                     timeout=30).json()
    for prop in r["data"]:
        print(prop["id"], prop["address"])
    pg = r["meta"]["pagination"]
    if pg["page"] >= pg["total_pages"]:
        break
    page += 1
```

JavaScript (Node, fetch)

```
const BASE = "https://secure.lawnprosoftware.com/api/v1";
const key  = process.env.LAWNPRO_API_KEY; // lp_live_<prefix>.<secret>

async function createCustomer(data) {
  const res = await fetch(`${BASE}/customers`, {
    method: "POST",
    headers: {
      "Authorization": `Bearer ${key}`,
      "Content-Type": "application/json",
      "Idempotency-Key": crypto.randomUUID(),
    },
    body: JSON.stringify(data),
  });

  if (res.status === 429) {
    const wait = Number(res.headers.get("Retry-After") || 1);
    await new Promise(r => setTimeout(r, wait * 1000));
    return createCustomer(data); // retry after backoff
  }

  const body = await res.json();
  if (body.errors.length) throw new Error(body.errors[0].message);
  return body.data;
}
```

---

16. Legacy API & deprecation

The older `/api` surface (which transmitted the account email/password in request headers) is **deprecated**. It still functions for now, but:

- Responses to legacy password-auth calls carry `Deprecation: true` and `Link: </api/v1>; rel="successor-version"`.
- It transmits credentials on every call and lacks scopes, rate limiting, and idempotency.

All new integrations must use `/api/v1`. Migrate existing ones to key-based auth as soon as practical.

---

17. Support & changelog

- **Get a key / raise limits / report an issue:** contact LawnPro support.
  - Phone: **205-369-7052** (7am–7pm CST, 7 days a week)
- When reporting a problem with a specific call, include its `request_id` (from the response body or the `X-Request-Id` header).

Changelog

Version	Notes
`v1` (1.0.0)	Initial public release: customers, properties, items (read/write); invoices, estimates, payments, schedule (read-only). Bearer key auth, scopes, per-key rate limiting, idempotent creates.
`v1` (1.1.0)	Added **Webhooks**: manage subscriptions via `/api/v1/webhooks` (scopes `webhooks:read`/`webhooks:write`) and receive signed, retried event deliveries at your own HTTPS endpoints. Also configurable in **Settings → Webhooks**.