LEADHUNTER

The Manual

Find, score, and reach the right accounts — without the spreadsheet sprawl.

Generated 2026-07-11 · docs.leadhunter.humans2agents.com

Contents

START HERE

1. Welcome to LeadHunter
2. How LeadHunter works
3. Quick start
4. Glossary

CONCEPTS

5. Company and Product
6. Account
7. Campaign
8. Acquisition channels
9. Outreach reasons and targets
10. Target persona and scoring
11. Custom fields
12. Account value (LTV)
13. Research
14. Messaging

HOW-TO GUIDES

15. Import accounts
16. Track inbound leads (Adwords, Instagram, referrals)
17. Build a saved filter
18. Run your first campaign
19. Track campaign costs and CAC
20. Track customer value and ROI per channel
21. Log a conversation
22. Merge duplicates

23. Push events to your tools — webhooks

24. Stay close to your clients — Client Pulse

25. Install LeadHunter as an app

ACCOUNT & BILLING

26. Team and companies

27. Usage and costs

REFERENCE

28. FAQ

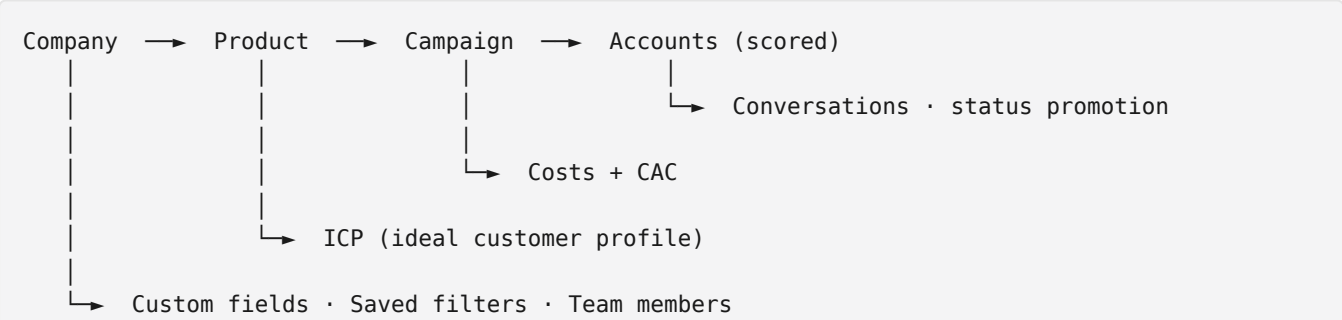
START HERE · CHAPTER 1

Welcome to LeadHunter

LeadHunter is an outbound research and outreach platform for small teams running their own go-to-market. You bring a product you want to sell and a rough idea of the customers you want to reach; LeadHunter turns that into:

- A **scored database of accounts** (companies, organisations, venues) that match your Ideal Customer Profile (ICP).
- **Per-product fit scores** with typed reasons — a 0–10 number, a category label, and 2–4 concrete justifications, so you know *why* the AI thinks an account is a 9/10 vs a 4/10.
- **Campaigns** you run against those accounts, with built-in safeguards so you don't accidentally cold-pitch a customer, a competitor, or your investor.
- A **manual communication log** across email / LinkedIn / Instagram / X / WhatsApp / phone, with AI drafting and auto-translation between your language and the account's.
- **Inbound attribution** when the lead came in through Google Ads, Instagram DMs, referrals, or any non-outbound channel — start them at the right lifecycle status, with the UTM parameters and thread URLs preserved for later analysis.
- **Cost-of-acquisition tracking** per campaign — ad spend, agency fees, labor hours go in; CAC per outreached / responded / closed customer comes out.

How the pieces fit together



- A **Company** is your tenant — your team, your data, your accounts. One user can have multiple companies.
- Inside each company you define one or more **Products**: what you're selling, who it's for. The Product carries the ICP.
- Each Product can run multiple **Campaigns** — same product, different audience or angle.
- **Accounts** are the organisations you care about. The same account can live in multiple campaigns, but is **scored once per product** — adding it to a second campaign of the same product doesn't re-score it.

Outbound, inbound, and the same database

LeadHunter started life as an **outbound** tool — you research, you find, you reach out. The same database now also tracks accounts that came in through **inbound** channels (Adwords clicks, Instagram DMs,

referrals, event introductions). Inbound accounts start at the `contacted` lifecycle status automatically since they've already reached out to you, and the marketing-channel attribution carries through to the funnel and CAC stats. See Track inbound leads.

Who LeadHunter fits

The product was built around a few specific kinds of operator:

- **Small teams running their own outbound** — one to five people, doing the actual reach-out themselves, with strong AI assistance but real human judgment on every account.
- **Outbound agencies running multiple clients** — one Company per client, fully isolated, with per-Company stats and CAC.
- **Founders going to market** — early days, narrow ICPs, every conversation matters, the funnel and CAC stats inform pricing and positioning decisions.

It's less of a fit for large sales orgs with established CRM workflows, mass-outreach operations that send identical templates to thousands of leads, or anyone looking for an off-the-shelf contact database to query.

What LeadHunter isn't

A few deliberate non-features so you know what to expect:

- **Not a sending engine.** You keep sending through email / LinkedIn / Instagram / WhatsApp / phone yourself; LeadHunter logs the exchange. Sending well across every channel is a different product.
- **Not a contact database.** LeadHunter doesn't ship with a pre-loaded index of millions of companies. You bring accounts (CSV / Google Maps discovery / one-off lookup) or capture them inbound.
- **Not a marketing-automation platform.** No drip sequences, no scheduled blasts, no automated A/B tests. The point is that every account gets human judgment.
- **Not a CRM replacement** if you need invoicing, contract management, support tickets, or anything past the "from prospect to first conversation" loop. LeadHunter complements those tools — it doesn't try to replace them.

What you'll do today

- **If you want the bigger picture first**, read How LeadHunter works — the end-to-end flow from empty workspace to operational outbound program, with what to expect at each phase and the rhythm you'll settle into after the first cycle.
- **If you're ready to dive in**, follow the Quick start. Fifteen minutes from sign-in to scored accounts.
- **If you want the data model in detail**, the Concepts section explains the status lifecycle, dedupe stack, and merge semantics that keep your database free of duplicates.
- **If you'd rather read offline**, the full manual is available in three formats — cover, table of contents, every concept and how-to page in one file, regenerated when the docs change:
  - **EPUB** (~140 KB) — best for Kindle, Kobo, Boox, Apple Books, and any reflowable e-reader. Send-to-Kindle accepts this file directly; Amazon converts it to KFX on their end.

- **Pocket-edition PDF** (~2.6 MB, 5.5"×8") — fixed-layout PDF in mass-market paperback dimensions; sized for 6-7" e-ink readers (Kindle, Kobo, Boox) so text is readable without zooming.
- **PDF for A4 print** (~2.5 MB, A4) — the standard letter-paper variant if you're printing or reading on a larger tablet / laptop.

Zero duplicates, on purpose

The most important thing to know about LeadHunter: **it refuses to create duplicates**. Every import path, every research pass, every API post runs through the same five-level dedupe stack — Google Place ID, name + city (exact, post-normalisation), phone, website domain, fuzzy name within the same city. The first four auto-merge into the existing row; the fifth surfaces for human review. Re-importing the same list twice is safe.

When duplicates *do* get merged, every unique field from every duplicate is preserved on the survivor — LeadHunter never silently overwrites. See Merge duplicates for the full story.

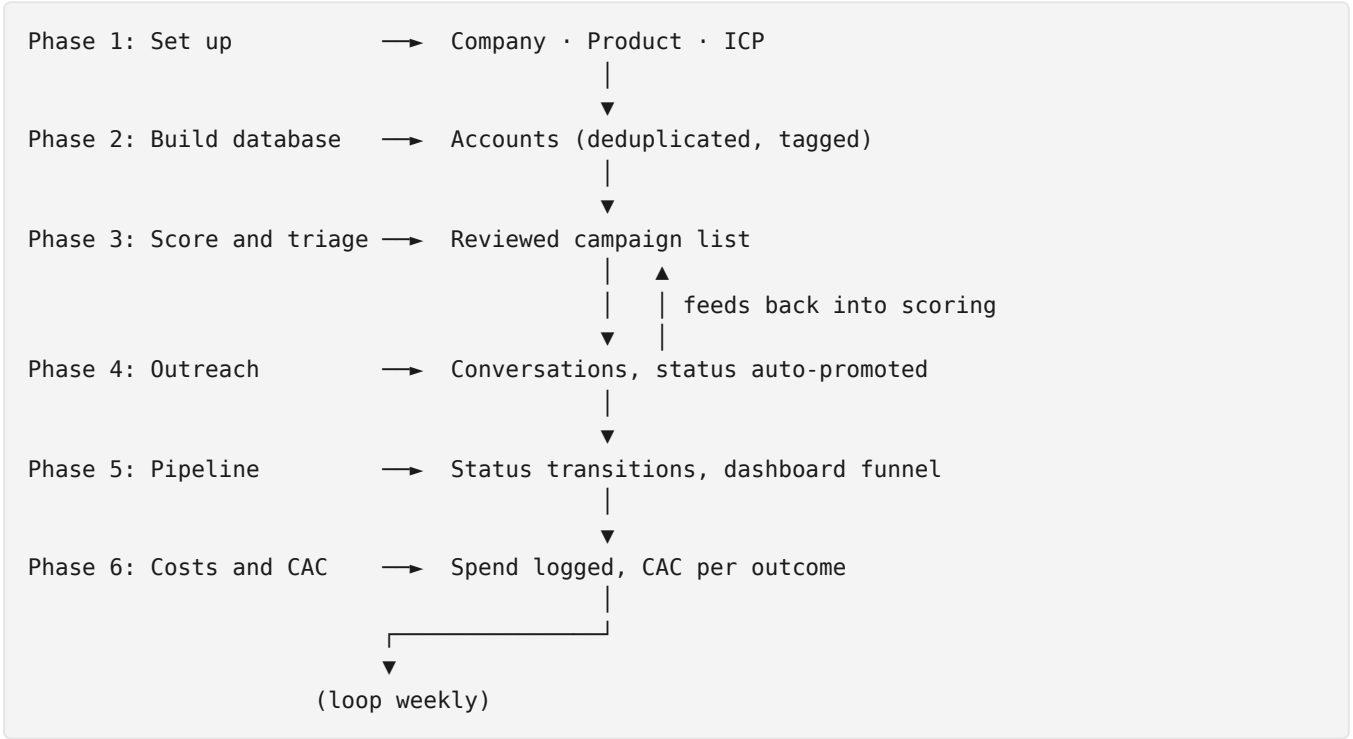
START HERE · CHAPTER 2

How LeadHunter works

LeadHunter is opinionated about *how* you use it. This page lays out the intended flow end-to-end so you can see what each phase looks like, what to expect at the end of it, and the rhythm you'll settle into after the first cycle.

The Quick start walks you through the first fifteen minutes mechanically. This page is the bigger picture around it — read it before, after, or in parallel.

The big picture



The first three phases are mostly one-time setup work. Phases 4–6 are where you'll live day-to-day, with periodic returns to phases 2 and 3 as you add new accounts and refine your targeting.

Skip cases — when phases compress

Not every starting point is the same. A few common patterns:

- **You already have a CSV from your CRM.** Phase 2 collapses to a single import. The dedupe stack handles the heavy lifting.
- **You're an agency running a client brief.** Phase 1's ICP is half-written for you — paste the brief into the Product description and Generate ICP from there. Often a single approve-or-revise pass instead of multiple iterations.
- **You're picking back up a Product from a previous campaign.** Phases 1 and 3's calibration are already done — the existing scores reuse for any new campaign of that Product (see ICP and scoring → Score caching). You drop straight into Phase 2 to add new accounts and Phase 4 to start reaching out.

The phases are a complete framework, not a mandatory sequence — skip and revisit as your situation warrants.

Phase 1 — Set up your workspace

Goal: A Company, a Product, and an ICP good enough to start scoring.

Steps:

1. Create a Company. One per business; agencies run one per client.
2. Add a Product — name, description, website URL, default communication language.
3. Add 2–3 **example good URLs** and 2–3 **example bad URLs** to the Product. This is the single biggest lever on scoring quality — don't skip it.
4. Generate the ICP from the Product. Review the draft, edit anything that's off, save.
5. (Optional) Define custom fields for whatever your business tracks beyond the standard fields.
6. (Optional) Invite teammates.

Expected outcome: An empty Accounts list, but everything downstream is calibrated. Scoring works as soon as accounts arrive.

Common pitfalls:

- *Vague Product description.* The AI can't write a sharp ICP from one-line positioning. Two paragraphs is the minimum.
- *Skipping example URLs.* Scoring quality plateaus without them.
- *Trying to cover two audiences with one Product* (e.g. enterprise + SMB). Use two Products with two ICPs instead — see Company and Product.

Phase 2 — Build your account database

Goal: A starter set of accounts to work with. Anywhere from 50 to a few thousand, depending on how broad your market is.

Inputs:

- Existing CSV/XLSX from your CRM, Apollo, ZoomInfo, etc.
- Google Maps text searches (*"bike shops in Berlin"*) for geographic discovery.
- Individual URLs or domains for known leads via the Account lookup.

Use any of them — or all three. See Import accounts for the flows in detail.

Expected outcome: A populated Accounts list, deduplicated against itself.

Common pitfalls:

- *Worrying about duplicates.* Don't. Every import path deduplicates silently before writing. Re-importing the same list twice is safe — see Merge duplicates.

- *Forgetting to tag existing customers.* Mark them `customer` (lifecycle status) and `client` (relationship type) before running campaigns, so the outreach guardrails can protect them. See Account → Relationship types.
- *Importing every contact in your CRM.* Quality over quantity. Start narrower than you think.

Phase 3 — Score and triage

Goal: A ranked, reviewed list of strong-fit accounts ready for outreach.

Steps:

1. Create a Campaign pointing at your Product.
2. Bulk-add accounts — from a saved filter, an import, or by hand.
3. Wait for scoring. Each account gets a 0–10 fit score plus a label (`excellent` ≥8, `moderate` 5–7, `mismatch` <5) and 2–4 typed reasons (positive / negative / neutral).
4. Sort by score. Open the top accounts and read the AI's reasons.
5. **Approve** strong fits. **Reject** ones the AI missed on. Set an **override score** where it's clearly off.

Expected outcome: A campaign with 10–50 reviewed accounts ready to reach out to.

:::tip[Review isn't busywork — it sharpens scoring] Once you've approved or rejected about ten accounts on a Product, your verdicts start feeding back into future scoring as concrete *"this is what good looks like for us"* examples. The model gets sharper for your specific business with every review. Below ten the calibration is held back (small samples are noisier than no sample), so the early reviews are exactly when this work pays off most. :::

The campaign page also shows a **10-bucket score-distribution histogram** (`0-1` through `9-10`) once scoring runs — a healthy distribution is bimodal (excellent + mismatch piles with a modest moderate middle); a flat or all-moderate shape usually means the ICP or example URLs need work. See Campaign → Score distribution.

If scoring fails partway through. The campaign's `scoring_status` flips to `failed` and stops processing further accounts. Scores from accounts that completed before the failure are kept — re-running scoring picks up where it left off and only scores accounts that don't yet have one. If you see fewer scored accounts than expected, check `scoring_status` on the campaign before assuming the worst.

Common pitfalls:

- *Trusting the number, not the reasons.* The score is a starting point; the reasons are the substance. A 9/10 for the wrong reasons is still wrong.
- *Reaching out to unreviewed accounts.* At least skim the top of the list first — you'll spot patterns the AI missed.

Phase 4 — Outreach and conversations

Goal: First messages out, replies pasted in, history kept consistent.

Steps:

1. Open an approved campaign-account.
2. Pick a contact (or add one).
3. **AI Draft** → review → edit → send through your normal channel (email, LinkedIn, IG, WhatsApp, phone).
4. **Log Sent** in the conversation panel so LeadHunter knows it went out.
5. When the account replies, **Inbound Paste** the reply text. The AI translates it into your language if needed.
6. Repeat with **AI Continue** for follow-ups.

LeadHunter is a **communication log**, not a sending platform — you send through your own channels and paste the exchange in so scoring, history, and team visibility all stay in sync. See Messaging for the five message modes in detail.

The conversation thread lives in two places: the **campaign-account view** (just this campaign's messages with the account) and the **account detail page** (every conversation across every campaign that account has been in). When an account moves to a new campaign, its history follows them.

Expected outcome: Each reached-out account has a conversation thread. The account's status auto-promotes from `prospect` to `contacted` on the first outbound; `responded` flips on when a reply lands.

:::tip[Let LeadHunter pick the language] The fallback chain runs `Account.language → Campaign → Product → User → English`. Set a language on an individual account if you know one, otherwise leave it blank and let the campaign/product default win. If the account has a language set, that *always* wins — including across campaigns. See Campaign → Language. :::

Common pitfalls:

- *Forgetting to Log Sent.* If LeadHunter doesn't know you reached out, outreach status doesn't move and the dashboard funnel doesn't see the activity.
- *Drafting in a language the account doesn't speak.* Let the system pick.
- *Cold-pitching a `do_not_contact` account.* Hard-blocked with a clear error — there's no force flag. Resolve the status mismatch instead.

Phase 5 — Pipeline and ongoing operations

Goal: Track relationships as they evolve.

Steps:

- Move status as deals progress: `contacted` → `in_negotiation` → `customer` (or `lost`).
- Add relationship types as they become true — a prospect that converts becomes a `client`; the same account might also be a `partner` or `press` contact later.
- Mark opt-outs as `do_not_contact`. This is sticky and survives merges so opt-outs stay opt-outs.
- Watch the **dashboard outreach funnel** — contacts initiated, clicked, responded, closed — daily, weekly, monthly. Treat it as your activity gauge. (*Clicked* comes from tracked links: contacts who clicked something you sent but haven't replied yet — your warmest silent prospects.)

Expected outcome: Your account database stays current with reality. The funnel tells you whether outreach effort and conversion are trending right.

:::tip[Two corrections you'll want to know about] The dashboard's **Closed (won)** count is stricter than *"count every customer transition"*: it requires the account's current status to still be `customer` or `lost`, so an *"oops, wrong account"* revert drops back out. And when you load existing customers (a CRM import, a long-standing renewal), tick **"Mark as pre-existing customer"** when changing the status — that flags the transition so it doesn't fake-inflate this month's wins. See Account → Recording a pre-existing customer. :::

Common pitfalls:

- *Letting status drift.* If your accounts say `prospect` when half of them are customers, every guardrail and every report drifts with them.
- *Skipping the monthly dedupe pass.* New imports introduce new duplicate candidates; resolve them before they accumulate.
- *Importing an existing customer book without the pre-existing flag.* The dashboard's win count balloons, then you have no clean baseline to measure new campaign wins against.

Phase 6 — Costs and CAC

Goal: Pair every outreach outcome with what it cost so the numbers you read are honest.

Steps:

- Log spend as it accrues — Adwords invoices, agency fees, internal hours, software subscriptions, content production, event costs. Each entry goes against a specific campaign with an amount + currency + date + vendor.
- The campaign's **Costs & expenses** panel surfaces total spend (per currency), cost per outreached account, and cost per responded account in real time.
- The **Stats by product and campaign** page rolls up cost per closed-won customer across the last 30 days, both per campaign and per product.

Auto-tracked API costs (LLM tokens, Google Maps calls, Tavily searches) are surfaced **alongside** user-entered expenses on every campaign — same panel, same totals — so a campaign's full cost is one number, not two surfaces you have to add together. The Account detail page also shows a per-account API-cost block, so you can answer *"how much did processing this single account actually cost?"* directly on the row.

Expected outcome: Every campaign carries a real cost-of-acquisition number you can compare against other campaigns and against the value of a customer.

Common pitfalls:

- *Forgetting the costs side.* A response rate without a cost is half a story. Logging spend weekly is the minimum cadence to keep the numbers honest.
- *Mixing currencies in a single campaign.* LeadHunter shows per-currency totals but doesn't compute CAC across currencies (no FX layer). Either normalise to one currency or run cost analysis externally.

See Track campaign costs and CAC for the full mechanics.

The recurring rhythm

After the first cycle, you'll settle into something like this:

- **Daily** — send today's outreach; paste yesterday's replies; update status on anything that moved.
- **Weekly** — review fresh scores, approve/reject five to fifteen accounts to keep feeding the model; refresh saved filters as your segmentation evolves; add this week's new imports; log this week's ad spend / agency invoices / hours against the right campaign.
- **Monthly** — run Find Duplicates; revisit the ICP and example URLs based on what closed-won; archive completed campaigns; check the dashboard's 30-day trend lines; read the per-campaign CAC on the Stats page and pick the most efficient campaigns to do more of.

The product is built to reward this rhythm. The dashboard funnel reads better with consistent activity; the scoring model calibrates better with continuous review; the database stays clean if duplicates are caught monthly rather than yearly.

Where to go next

- **First-time setup** — Quick start.
- **Data model** — Account, Campaign, Company and Product, ICP and scoring.
- **Operations** — Run your first campaign, Log a conversation, Track campaign costs and CAC.
- **Hitting a snag?** — FAQ covers the most common first-week questions.
- **Recent changes** — What's new.

Quick start

A mechanical fifteen-minute walkthrough that takes you from sign-in to scored accounts ready for review. This page gets you operational; for the why-behind-each-step framing and the rhythm you'll settle into after the first run, read How LeadHunter works — before, after, or in parallel.

Before you start

You'll get the most out of the next fifteen minutes if you have these ready:

- **A LeadHunter account** at leadhunter.humans2agents.com. Sign up there first if you haven't.
- **A product you're selling**, with a public website you can paste in and a one-or-two-paragraph description in your head.
- **2–3 example accounts** (their websites) of customers or prospects you'd love to win. These calibrate scoring more than anything else.
- **A starter list of accounts** — could be a small CSV from your CRM, a list of bookmarks, or just *"I want to find every bike shop in Berlin."*

If you only have the first two, you can still get to scored accounts in this session — you'll discover the rest via Google Maps in step 4.

1. Create your first Company *(2 min)*

A **Company** groups everything: products, accounts, campaigns, custom fields, saved filters, team members. Most users run one. If you're an agency, run one per client.

1. Sign in. You'll land on the dashboard.
2. Open the **Company switcher** in the top bar.
3. Choose **New company**, give it a name (e.g. *"Acme Outbound"*), save.

Everything you do from now on is scoped to whichever company is active in the switcher — accounts, campaigns, statistics, custom fields, the lot.

2. Add a Product *(3 min)*

A **Product** is what you're selling. Its description, website, and example URLs feed the AI's scoring model.

1. Go to **Products** → **New product**.
2. Fill in:
   - **Name** — e.g. *"FleetMate"*.
   - **Website URL** — your product's public landing page.
   - **Description** — one or two paragraphs. The clearer the better; the AI uses this to draft the ICP and to score accounts. *"AI-powered fleet management"* won't fly. *"FleetMate is a maintenance scheduling*

tool for small commercial fleets (5–50 vehicles); we focus on independent garages and last-mile delivery operations in the EU." is closer.
    - **Communication language** — default language the AI drafts outreach in. Campaigns can override.
3. Save.

3. Generate and approve the ICP *(4 min)*

The **Ideal Customer Profile** describes who you want to sell to: sector, geography, firmographics, buying signals, anti-patterns.

1. Open your new Product.
2. Click **Generate ICP**. LeadHunter visits your website + reads your description and drafts a structured ICP — usually in 10–20 seconds.
3. The draft opens in **Pending approval**. Review every field. Edit anything that's off.
4. Click **Approve**. The Product is now ready for scoring.

The single biggest lever on scoring quality after the ICP: add 2–3 **example good URLs** (websites of accounts you'd love to win) and 2–3 **example bad URLs** (accounts that look like fits but aren't) to the Product. The model anchors heavily on these. Skip them and scores read fuzzy.

4. Add some accounts *(2 min)*

Three paths — pick whatever fits your starting data:

- **Paste a URL or domain** under **Accounts** → **Lookup**. Works for a Google Maps URL, a website, or a bare domain like `acme.com`. Pick `Quick` mode for fast extraction; `Deep` mode (slower, more tokens) also extracts decision-maker contacts from team / about pages.
- **Upload a CSV or XLSX** under **Accounts** → **Import**. The wizard previews the columns, the AI suggests a mapping (*"`org_name` → name"*), you confirm, and LeadHunter runs deduplication against your existing database. Dry-run mode is recommended for anything over ~50 rows.
- **Search on Google Maps** with a query like *"bike shops in Berlin"*. Each result becomes an Account, or merges into an existing one via the dedupe stack.

For this first run, aim small — 5–20 accounts is enough to see the loop working. The point of the first session is to validate the scoring + outreach flow on something you can actually skim. Once that loop feels right, scale up: a sharp scoring pass on a few hundred well-targeted accounts beats a noisy pass on thousands.

If any of these came in through Adwords forms, Instagram DMs, referrals, or other inbound channels, set the **acquisition channel** on each — they'll auto-start at `contacted` instead of `prospect`. See Track inbound leads.

5. Run your first campaign *(3 min)*

1. **Campaigns** → **New campaign**, pick the Product you just made. Name the *audience*, not the product (*"Berlin bike shops Q2"*, not *"Spring campaign"*).

2. Add some accounts — pick a saved filter for bulk-add (the reach estimator tells you how many will land before you commit), or add them one by one from the campaign's accounts panel.
3. **Wait for scoring**. LeadHunter scores each account against the Product's ICP and writes typed reasons. With 5–20 accounts on a first run it's a couple of minutes; a few hundred takes 5–15. You can keep working while it runs. Each account ends with:
   - An `ai_score` — 0–10.
   - A `score_label` — `excellent` (≥8), `moderate` (5–7), or `mismatch` (<5).
   - **2–4 reasons**, each tagged **positive** (concrete fit), **negative** (concrete gap), or **neutral** (context).
4. Sort by score descending. Open the top accounts. **Read the reasons** — a 9/10 for the wrong reasons is still wrong.
5. Approve the strong ones, reject the misses, set an override score where the AI was clearly off.

Every approve and reject you make is a calibration signal. Once a Product crosses ~10 reviewed accounts, those verdicts start feeding back into the scoring prompt — every subsequent batch gets sharper. **The first campaign is when this matters most** — be thorough.

6. Log your first outreach *(1 min)*

Open an approved campaign-account, click the **Conversation** tab.

1. Pick a contact (or add one).
2. Mode **AI draft**, channel **email** (or whatever surface you'll actually send from — channel matters because the dashboard funnel attributes responses against it later).
3. The AI writes a fresh first-touch message in the account's language. Edit until it reads right.
4. Send through your own email client / LinkedIn / IG / WhatsApp.
5. Click **Log sent** in LeadHunter — the message is recorded, `outreach_status` flips to `sent`, the account's lifecycle status advances `prospect → contacted`, and the dashboard funnel counts the account as **initiated**.

When a reply comes back, paste it via **Inbound paste**. The funnel counts the account as **responded**, cohort-attributed back to the original initiation date.

What if something doesn't work

A few common first-run hiccups:

- **The Generate ICP button finishes but the ICP reads vague.** Most often the Product description is too thin. Two paragraphs of specific positioning — *what you sell, who it's for, who it isn't for* — beats one line every time. Regenerate after editing.
- **The first batch of scores all cluster around 5–6.** Same root cause as above, plus missing example URLs. Add 2–3 good and 2–3 bad URLs on the Product and re-run scoring.
- **Accounts I added by hand show as duplicates.** That's the dedupe stack working — if the same Google Place ID, phone, or website already exists, the new data is merged into the existing row. Look in *Accounts → Duplicates* if you want to see what merged where.

- **My CSV import skipped a column.** The wizard's column-mapping step is where you tell LeadHunter what each column means. Re-run with `dry_run=true` first to see what would land before committing.

What's next

You've now got an end-to-end loop running. From here:

- **Log spend as it accrues** — Adwords invoices, agency fees, internal labor hours. LeadHunter computes cost-of-acquisition per outreached / responded / closed customer. See Track campaign costs and CAC.
- **Read the funnel** — the dashboard home shows initiated / responded / closed across today / 7d / 30d. Track week-on-week, not the latest day (cohort attribution makes the right edge low by design).
- **Settle into the rhythm** — daily outreach + reply pasting; weekly fresh-score reviews; monthly duplicate cleanup + ICP refinement. See How LeadHunter works → The recurring rhythm.

Where to read next

- How LeadHunter works — the end-to-end flow with the why behind each phase.
- Run your first campaign — same path, longer-form walkthrough with edge cases.
- Account — the row that everything else hangs off.
- ICP and scoring — what the model sees, how it calibrates, when to rescore.

Glossary

A working glossary of LeadHunter terminology. Skim it once when you're new; refer back when a term feels overloaded or you can't remember which field carries what.

Grouped by concept area, not alphabetised — related terms read better next to each other.

Top-level containers

- **Company** — Top-level tenant. Sometimes called a *project* internally (you may see it in older URLs). One company per business, or one per client if you're an agency. Everything else hangs off a Company; data never crosses between companies. See Company and Product.
- **Product** — What you're selling. Carries the description, ICP, default communication language, and example URLs that calibrate scoring. A Company can have multiple Products.
- **User** — A person with a login. One user can belong to many Companies via team memberships.

Accounts and contacts

- **Account** — An organisation in your database (a company, a venue, a podcast — anything you'd put on an org-level row). Can be a customer, prospect, partner, press contact, supplier — *the same row* serves all roles. See Account.
- **Contact** — A person inside an Account: name, email, phone, job title, seniority, LinkedIn URL. Distinct from Account; one Account can have many Contacts (or zero).
- **Contact role kind** — Tags how to interpret a Contact's `seniority` / `department` / `is_decision_maker` fields. The same `seniority='c_level'` value reads as *"C-Suite executive"* for a `buyer`, *"Editor-in-chief"* for a `journalist`, and *"Senior career level"* for a `candidate`. Seven kinds: `buyer` (default), `journalist`, `candidate`, `investor`, `partner`, `influencer`, `other`. Default `buyer` preserves the sales-shaped reading for existing contacts.
- **Lead** — Historical term, still used in URLs and verb phrases (*"lead generation"*, *"hunting leads"*). The data row itself is now called *Account*; the discovery flow is still called *lead hunting*. Same thing.
- **Contract end date (`contract_end_at`)** — Optional renewal-cycle date on Account. Drives the renewal goal: campaigns at `goal='renewal'` filter for customers approaching this date. NULL for accounts not on a renewal cycle (most prospects, all press / candidates / investors).

Status, types, and channel — the three orthogonal fields

- **Status** (lifecycle) — Single-valued pipeline state on an Account: `prospect`, `contacted`, `in_negotiation`, `customer`, `lost`, `do_not_contact`. Answers *"where in the pipeline?"*
- **Relationship types** — Multi-select labels on an Account: `client`, `prospect`, `reseller`, `affiliate`, `supplier`, `partner`, `investor`, `press`, `influencer`, `analyst`, `competitor`, `candidate`, `personal_network`. Answers *"what is this account to me?"*
- **Acquisition channel** — Single-valued marketing-channel attribution: `outbound` (default), `adwords`, `meta_ads`, `linkedin_ads`, `organic_search`, `organic_social` (Instagram DM, X, …), `referral`,

`event`, `partner`, `cold_inbound`, `other`. Answers *"how did this lead first reach me?"*

- **Acquisition metadata** — Free-form JSON bag attached to an account for channel-specific data (UTM parameters, ad campaign id, Instagram thread URL, referrer email, …).
- **Status source** — Audit-trail field recording *how* a status transition happened: `manual`, `campaign_outreach`, `import_exact`, `import_fuzzy`, `import_llm_verified`, `inbound_acquisition`, `pre_existing` (operator-marked relationship that predates LeadHunter).
- **Status history** — Append-only audit trail on each Account; every status change writes an entry with from/to/at/by/source/reason.
- **Pre-existing customer** — A Customer-status transition flagged as a relationship that predates LeadHunter (a renewal client, a CRM import). Still on the audit trail, but the dashboard's *Closed (won)* metric skips it — so importing your existing book of business doesn't fake a win.
- **Auto-promotion** — Lifecycle-status changes LeadHunter makes for you without explicit operator action. Two cases: (1) creating an account with an inbound acquisition channel starts it at `contacted` instead of `prospect`; (2) logging the first outbound on a `prospect` advances it to `contacted`. Forward-only; never demotes.

Guardrails

- **DNC (`do_not_contact`)** — Blanket GDPR/CAN-SPAM opt-out. Sticky — survives merges, hard-blocks outbound on every campaign goal. No override flag.
- **Per-purpose DNC (`do_not_contact_purposes`)** — Newer, finer layer: a list of campaign goals the account opted out of. `["sales"]` means "block sales campaigns from contacting me, but research interview asks / press pitches / event invites are still OK". Sticky like DNC, but goal-scoped. See Account → Per-purpose opt-outs.
- **DNC history (`dnc_history`)** — Audit trail of per-purpose opt-out / opt-in events: `{purpose, action, at, by, source, reason}`. Mirrors `status_history` shape.
- **Hard block** — Outreach refused, no override available. Applies to `do_not_contact` status, to `personal_network` relationship type universally, and to `competitor` for most goals (admitted in press / event / research where competitor accounts are legitimate targets).
- **Soft block** — Outreach refused unless an explicit `include_<type>=true` flag is set. Applies to `customer` status (in goals that block customers) and to `supplier` / `investor` / `press` / `analyst` / `candidate` relationship types (per goal — a press campaign won't block press accounts, etc.).
- **Block explanations** — When bulk-add hits a guardrail, the response carries a goal-aware reason per blocked bucket: a Sales campaign tells you *"this Sales campaign soft-blocks press accounts; pass `include_press=true` or switch the goal"*. The frontend renders these directly under each blocked count.

Lifecycle of a row

- **Archive** — Soft-hide a Product, Campaign, or Account from default lists while keeping every reference intact (campaign rows, scores, conversations, expenses, audit trail). Accounts also carry a free-text **archive reason**. Reversible at any time.
- **Delete** — Permanent removal, cascades to every related row. Campaigns and Accounts require typing the exact name to confirm. Products are protected if any campaign references them — archive instead.

Scoring

- **Target persona** (a.k.a. **ICP** / Ideal Customer Profile) — Structured description of who a campaign targets: sector, geography, firmographics, buying signals, anti-patterns. Keyed by **(Product, Goal)** — a sales campaign of a product carries a buyer persona, a press campaign of the same product carries a publications-and-journalists persona, etc. Each independent, each with its own approval state. AI-generated, operator-approved. The "ICP" label remains the alias in sales-shaped flows. See Target persona and scoring.
- **Persona confidence** — How confident the AI is in the persona draft: `low` / `medium` / `high` plus a 0–100 score. Low-confidence drafts are usually a signal that the Product description is too thin and worth sharpening before approval.
- **Revision feedback** — Free-text you provide when asking the AI to redraft the persona (*"focus more on independent garages, drop the enterprise framing"*). Feeds into the regeneration prompt; bumps `revision_count`.
- **Fit score** — A 0–10 rating for how well an Account matches the goal's target persona, with a categorical label and 2–4 typed reasons. Keyed by **(Product, Account, Goal)** — a sales score and a press score for the same account on the same product are independent rows.
- **Score label** — Three values: `excellent` (≥8), `moderate` (5–7), `mismatch` (<5).
- **Score reasons** — 2–4 short, typed justifications: **positive** (green check, concrete persona match), **negative** (red X, concrete persona gap), **neutral** (amber, context that doesn't push either way).
- **Score distribution** — The 10-bucket histogram (`0-1` , `1-2` , …, `9-10`) on the campaign detail page. A healthy persona usually produces a bimodal distribution; flat or all-moderate distributions signal an under-anchored persona.
- **Stale-score banner** — Orange banner on the campaign detail page when accounts have scores from a *different* goal than the campaign's current goal. Surfaces after a goal switch (the cached scores rated against the previous rubric); one-click Rescore clears it.
- **Override score** — A campaign-specific manual score that supersedes the AI score for that campaign only. The (Product, Account, Goal) score row doesn't change.
- **Calibration feedback loop** — Once a (Product, Goal) pair crosses ~10 reviewed accounts (approved + rejected), your recent verdicts feed back into the scoring prompt as concrete examples. Goal-scoped — approvals on the press persona don't affect the sales persona scoring of the same product.

Campaigns and outreach

- **Campaign** — One outbound effort against a list of Accounts, scoped to a single Product. See Campaign.
- **Campaign goal** — The *why* of outreach. Eleven values: `sales` (default), `partnership` , `press` , `influencer` , `investor` , `recruiting` , `customer_expansion` , `win_back` , `event` , `research` , `renewal` . Reshapes the target persona, scoring rubric, message intent, bulk-add guardrails, and CAC label per campaign. See Outreach reasons and targets.
- **Goal history (`goal_history`)** — Audit trail of goal transitions on a Campaign: `{from, to, at, by, source, reason}` . Auto-appended whenever the goal changes; mirrors `Account.status_history` . The change endpoint accepts an optional `goal_change_reason` body field that gets stamped into the entry.

- **Close metric label** — Goal-aware noun phrase for "what counts as a close" — drives the third CAC card on the Costs panel. Sales reads *"Cost per closed-won customer"*, press reads *"Cost per press reply"*, recruiting reads *"Cost per candidate engaged"*, renewal reads *"Cost per renewal"*.
- **CampaignAccount** — Join row between a campaign and an account. Carries per-campaign workflow: review status, outreach status, chosen contact, override score, tags, notes, and the **role-in-campaign override**.
- **Role in campaign (`role_in_campaign`)** — Optional per-campaign lens override on a CampaignAccount. The same Account can be a paying customer AND a press contact AND a research participant; different campaigns view it through different lenses. When set, the outreach drafter receives a *"Role for this campaign: treat this account as `<role>` regardless of its other tags"* directive so the message respects the lens.
- **Review status** — Per-campaign-per-account: `pending`, `approved`, `rejected`. Drives the calibration feedback loop.
- **Outreach status** — Per-campaign-per-account workflow: `not_started`, `draft`, `scheduled`, `sent`, `bounced`, `responded`.
- **Scoring status** — Per-campaign state for the background scoring job: `idle`, `running`, `completed`, `failed`.
- **Bulk-add from filter** — Action that pulls every account matching a saved filter into a campaign in one pass, honouring all guardrails.
- **Reach estimator** — Pre-flight count of how many accounts will land when you bulk-add — accounts for every guardrail so the number is what will actually be added.
- **Outreach inbox** — The campaign's *Outreach* tab. Shows every conversation thread in the campaign sorted by what needs attention (recent inbound first, then drafts that haven't been sent).

Conversations and messages

LeadHunter is a **communication log**, not a sending engine. You send through your own channel and paste the exchange in.

- **Channel** — Per-message: `email`, `linkedin`, `instagram`, `twitter_x`, `whatsapp`, `phone_call`, `sms`, `other`. What the cohort funnel attributes against later.
- **Direction** — `outbound` (you sent) or `inbound` (they sent).
- **Mode** — How an outbound message was produced: `ai_draft` (first outbound), `ai_continue` (continuation), `type_translate` (you wrote in your language, LeadHunter translated), `log_sent` (you sent it elsewhere; paste to record). Inbound is always `inbound_paste`.
- **`original_text`** — The authoritative version of a message — what the account actually received (outbound) or sent (inbound), in their language.
- **`translated_text`** — A reader-friendly view of the same content in another language. Populated for `type_translate` outbound (the user's draft) and `inbound_paste` (the translation into your reading language).
- **Reading language** — Your User-level communication language. Inbound text is translated *into* it. Independent of UI language.

Costs and CAC

- **Campaign expense** — A user-entered cost line item against a campaign: Adwords spend, agency fees, labor hours, software, content, events. Ten kinds. Carries amount + currency + date + vendor.
- **Expense kind** — One of `adwords`, `meta_ads`, `linkedin_ads`, `other_ads`, `agency`, `labor`, `software`, `content`, `event`, `other`.
- **Quick-add expense** — The 💸 button on every row of the campaigns list. Opens a modal pre-filled for that campaign so you can log an expense without opening the campaign first.
- **Primary currency** — The currency with the largest total on a campaign. Drives the headline number and the CAC math.
- **CAC** (Cost of Acquisition) — Total spend ÷ outcome count. LeadHunter computes three flavours: per outreached account, per responded account, per closed-won customer. Only computed when user-entered expenses share one currency.
- **API costs** — Auto-tracked per-call costs for LLM tokens, Google Maps lookups, Tavily searches. Recorded automatically (you don't enter them) and surfaced alongside user-entered expenses on every campaign's Costs panel. Always in USD. CAC is computed from the user-entered side only — see Usage and costs.

Discovery and research

- **Research** — A tracked discovery / enrichment activity. Imports, Google Maps searches, website scrapes, AI deep-research all create a Research record. See Research.
- **Lookup** — One-off discovery from a URL, domain, Maps URL, or text query. Three modes: `none` (no AI), `quick` (basic extraction), `deep` (multi-page crawl + contact extraction).
- **Auto-enrichment** — Background job that runs after every new account creation, discovering and scraping the website automatically.

Deduplication and merging

- **Dedupe stack** — The five-level matching pipeline run on every account creation: Google Place ID (100%) → name + city exact (95%) → phone (90%) → website domain (85%) → fuzzy name in same city (≥85%). Levels 1–4 auto-merge; level 5 surfaces for review.
- **Golden record** — The survivor row after a merge. Carries every unique field from every duplicate.
- **`merge_history`** — Audit trail on the survivor recording absorbed-row snapshots, decisions, and overrides. Readable for forensic audit; not a one-click undo.
- **AI duplicate review** — LLM pass that classifies each duplicate group as `same` / `different` / `uncertain` to speed up triage of large lists.
- **Bulk merge** — Multi-group merge action for resolving many duplicate groups in one request. Useful after a noisy import surfaces dozens of fuzzy candidates.

Filters and segmentation

- **Custom field** — Per-Company schema for data not in the standard model: store size, contract expiry, tier, etc. Eight types. Referenced in filters as `cf.<key>`.

- **Saved filter** — A named, reusable query over your Account database. Can be **private** (yours only) or **shared** across the Company. Powers smart lists, campaign bulk-add, and API queries.
- **Filter DSL** — The structured query format: `{match: "all"|"any", conditions: [{field, op, value}]}`.
- **Match mode** — Per-filter: `all` (AND) or `any` (OR). No nested mixing.
- **Virtual field** — `has_email` / `has_phone` / `has_website`. Resolves to *"is this column non-empty?"* Only `is_true` / `is_false` apply.

Dashboard

- **Outreach funnel** — The three numbers on the dashboard home: contacts initiated, responded, closed (won). Each counts distinct accounts, not messages. **Closed (won)** only counts transitions to `customer` that stuck (current status is still `customer` or `lost`) and weren't tagged as a pre-existing relationship — see Account → Recording a pre-existing customer.
- **Cohort attribution** — Responded and closed are attributed back to the *initiation* date, not the response date. So the rate at the latest day is "low by design" until accounts have had time to respond.
- **Stats by product and campaign** — Per-product and per-campaign breakdown of the funnel, with Cost column (total spend + most-meaningful CAC available).

See also

- How LeadHunter works — the end-to-end flow that ties all of this together.
- Quick start — fifteen-minute mechanical walkthrough.
- Concepts — every concept above has a dedicated page.

Company and Product

LeadHunter has two top-level entities: **Company** and **Product**. Everything else hangs off them.

Company — your tenant

A **Company** is the highest-level container. It carries:

Bucket	Holds
Identity	`name`, `slug` (auto-generated from name, unique site-wide), `description`, `is_active`
Members	Team users and their roles (`owner`, `admin`, `member`)
Owned data	Accounts, Products, Campaigns, Conversations, Research records, Custom-field definitions, Saved filters, Campaign expenses
Provenance	`created_at`, `updated_at`

One user can belong to many Companies. Each Company is fully isolated — no data ever crosses between them. If you run an outbound agency, you'd typically have one Company per client. See Team and companies for the multi-tenant story in detail.

The active Company is selected in the **Company switcher** in the dashboard top bar. Everything you see on every page — accounts, campaigns, statistics, costs — is scoped to that selection. The selection persists across sessions via local storage.

> **A quick word on terminology.** Internally this entity is called a *project*. In the UI we call it a *Company* because that's how users actually think about it. The two terms mean the same thing — you may occasionally still see *project* in a URL or an older screen.

Product — what you're selling

A **Product** lives inside a Company and represents something you want to sell. A Company can have multiple Products — each with its own positioning, ICP, and campaigns.

Bucket	Holds
Identity	`name`, `slug` (auto-generated, suffixed on collision within the project)
Positioning	`description` — long-form, the AI's primary anchor for ICP generation and scoring
Discoverability	`website_url` — the AI visits this during ICP generation
Defaults	`communication_language` — drives outbound when neither Account nor Campaign override
Scoring calibration	`example_good_urls`, `example_bad_urls` — 2–3 of each, the single biggest lever on scoring quality after the ICP
Lifecycle	`archived_at`, `created_at`, `updated_at`, `created_by`
Related	One ICP (one-to-one); many Campaigns; one fit score per Account it's been scored against

The **`description`** field is the single most important field on the product. The AI uses it to write your ICP and to score accounts; vague descriptions produce vague ICPs and noisy scores. Two paragraphs minimum, specific about who the product is for and who it isn't.

ICP lives on the Product

The Ideal Customer Profile is attached to the Product, not the Campaign. Two campaigns of the same product share the same ICP and the same scoring calibration. Rescoring once benefits every campaign that uses the product — see ICP and scoring → Score caching.

If you need a *fundamentally different audience* for the same product (e.g. *enterprise* vs *SMB*, *EU* vs *US*, *agencies* vs *direct sellers*), the right move is usually to create a **second Product** with its own ICP — not two campaigns of the same product. See the heuristic below.

When should I make a second product vs another campaign?

Signal	What it means	Action
Same product, different geography or segment, **same fit criteria**	One audience, you want to slice by market or vertical.	One Product, multiple Campaigns.
Same offering but the **ICP changes meaningfully** (size band, sector, buying signals)	The model would score the audiences differently.	Two Products, each with its own ICP.
Different positioning of the same software	The *what* is identical, the *for whom* differs.	Two Products.
Truly separate offerings (different SKU, billing, sales motion)	These are distinct products in any business sense.	Two Products.

Rule of thumb: **if you would draft the ICP differently, it's a different Product.** Splitting on ICP keeps each product's scoring sharp; splitting on language or geography alone wastes the calibration work.

Why this matters for scoring

Scoring is per-(Product, Account). The same account showing up in three campaigns of the same product gets scored **once**. That keeps AI cost down and scores consistent: an account doesn't suddenly drop from

`excellent` to `moderate` because you started a new campaign.

The **calibration feedback loop** is also per-product — every approve / reject decision you make in a campaign's review queue feeds back into scoring for *that product*'s future scores, not the Company's. So a noisy review pass on Product A doesn't leak into Product B's scoring quality. See ICP and scoring → The feedback loop.

If you swap a campaign's product (allowed, as long as the new product belongs to the same Company), the campaign's accounts will need to be rescored against the new ICP. LeadHunter handles that with a single click in the UI.

Archive vs delete

Both Product and Campaign support **archiving** — a soft-hide that keeps every related row in the database (campaigns, conversations, scores, expenses, audit trail) but hides the item from default list views. Toggle `Show archived` to see them again.

Action	Effect
Archive a Product	Hidden from the products list. All campaigns of that product also disappear from default lists unless un-archived separately. Re-activating is one click.
Delete a Product	Refused if any Campaign references it — archive instead. Once safe, hard-delete removes the row permanently.
Archive a Campaign	Hidden from default lists; every conversation, score, expense, and audit-trail entry is preserved.
Delete a Campaign	Name-confirmed (you type the exact campaign name). Cascades to the campaign's CampaignAccount rows, conversations, ICP, and expenses. Underlying Accounts are never affected.

Bias toward archiving. Deletes are permanent and the data they take with them is often what makes the next campaign sharper.

Cross-tenant boundaries

Two reminders about what *can* and *can't* cross the Company boundary:

- **What doesn't cross**: accounts, scores, campaigns, expenses, custom fields, saved filters, conversation history, audit trails. Everything operational.
- **What does cross**: your User profile, your communication language, your dashboard UI language, your *Recent* entities list. These follow you across every Company you belong to.

The dedupe stack is also Company-scoped, which means the same business legitimately existing in two Companies (e.g. an agency's two clients both reach out to *Acme Corp*) will be two unmerged rows. That's the right answer — they're operationally independent.

Read next

- Account — the organisations that flow through your products.

- Campaign — running a specific outbound effort against a list of accounts.
- ICP and scoring — how the per-product fit score is computed and calibrated.
- Team and companies — the multi-tenant + roles story in detail.

Account

An **Account** is an organisation stored in LeadHunter. It might be a prospect, a customer, a partner, a reseller, a competitor, a journalist, an investor, a candidate, or part of your personal network — *the same row* serves all of those roles, with multi-select **relationship types** to label what each account is to you.

This is intentional. Most CRMs split "leads" and "customers" and "partners" into separate tables and then quietly accumulate duplicates across them. LeadHunter keeps one row per organisation and uses lifecycle + relationship type to describe context.

What an account carries

The standard fields, grouped by what they describe:

Bucket	Fields
Identity	`name`, `business_type`, `specialization`, `language`, `logo_url`, `rating`
Company profile	`year_founded`, `employee_count_estimate`
Location	`address`, `city`, `state_province_region`, `postal_code`, `country`, `latitude`, `longitude`, `google_place_id`
Contact	`phone`, `email`, `website` *(plus a separate **Contacts** list of people inside the account — see below)*
Social profiles	`linkedin_url`, `instagram_url`, `x_url`, `facebook_url`, `youtube_url`, `tiktok_url`
Lifecycle	`status`, `status_changed_at`, `status_changed_by`, `status_source`, `status_history`, `external_contact_history`
Classification	`relationship_types`, `acquisition_channel`, `acquisition_metadata`, `acquired_at`
Provenance	`source` (how the row entered the system), `is_verified`, `notes`, `merge_history`
Cached content	`website_content` *(scraped text used by the AI)*, `website_scraped_at`, `website_discovery_attempted`
Custom	`custom_fields` — per-company schema; see Custom fields.

Most fields have a sensible default and are optional. The only truly required field is `name`. Everything else can be populated incrementally — auto-enrichment fills website + content + language for free shortly after the row is created.

What happens after you add an account

When an account lands in the database — by import, lookup, Google Maps discovery, or one-off entry — LeadHunter queues a background job that:

1. **Discovers the website** when only a name + location was provided. Uses Gemini grounding plus a domain-guessing fallback.
2. **Scrapes the discovered URL** and stashes the text content on the account (`website_content`). The cached content is what the scoring model reads later — it doesn't re-fetch on every score.
3. **Extracts structured fields** from the scraped content — the company's logo, phone, email, address, the per-platform social URLs, founding year, employee-count bucket, primary language, and the business name when the row arrived without one.

Auto-enrichment is **idempotent** at three levels — so manual edits and partial fills always survive a re-enrich:

- **Row level**: a row counts as already enriched when it has cached website content and the scrape is recent (within the cache window). Stale rows re-enter the queue automatically so the data doesn't go cold; fresh rows are skipped entirely so a re-enrich doesn't redo work.
- **Field level**: every structured field is **only written when the current value is blank**. If you manually corrected the LinkedIn URL or fixed a phone number, that value wins forever — re-enrichment can fill *other* blanks but never overwrites yours. The same rule applies to merge survivors: every field already populated from one of the absorbed rows stays put.
- **Discovery level**: when website discovery runs once and finds nothing, the row is flagged "discovery attempted" so subsequent passes don't loop on the same dead-ends. You can still set a website manually at any time; the next enrich picks it up from there.

Wrong-kind URLs are auto-rediscovered. If the `website` value you imported turns out not to be a website at all — an audio stream URL from a radio export, a PDF link from a press pack, a download URL — enrichment now detects that on the first byte (no 15-second timeout, no wasted bandwidth) and tries to find the company's real website automatically. When the stream carries a hint (Shoutcast/Icecast servers often include a homepage URL in their headers, for free), that hint wins. Otherwise the discovery agent runs and proposes a candidate URL, which is then verified to actually serve a webpage before the swap happens. The original URL is never lost — it lands in the account's `notes` field with a small audit block explaining what kind of resource it was and why it got replaced, so you can always go back.

You'll see an account grow website content, a language tag, socials, founding year, and a headcount bucket within minutes. Each row carries its own auto-tracked enrichment cost (USD, per model) so you can answer *"how much did this account cost to fill in?"* — exposed on the account detail. See Usage and costs.

You can also re-enrich on demand from the **Enrich** button on the Accounts page — three modes (only non-enriched / first 50 for a smoke test / currently filtered). The progress bar on the **Tasks** page reflects real work done, and large runs survive a redeploy without losing the chunks that already landed.

Company profile — year founded, employees, socials

Three structured signals beyond the free-form `business_type` make accounts easier to segment and easier to pitch.

- **`year_founded`** — the year the business started, when known. Filterable in saved filters with the usual numeric operators (*"founded since 2020"*, *"established 10+ years"*). The auto-enrichment agent fills it when the website states it explicitly (*"since 1985"*, *"founded in 2007"*); otherwise it stays blank and you can enter it manually.

- **`employee_count_estimate`** — a coarse headcount bucket: `1-10`, `11-50`, `51-200`, `201-1000`, or `1000+`. The strongest single fit signal for B2B targeting, and the only headcount shape that survives across data sources (LinkedIn ranges, manual entry, scraped about-pages all map onto these five buckets). The scoring agent reads it when computing fit, so a 5-person shop and a 5,000-person enterprise won't get scored against the same rubric.
- **Per-platform social URLs** — `linkedin_url`, `instagram_url`, `x_url`, `facebook_url`, `youtube_url`, `tiktok_url`. One URL per platform, filled by enrichment from the website's `<a href>` links (the first canonical match wins; share-intent links like *"share on Facebook"* buttons are skipped). They render as clickable chips on the account detail page so jumping into a DM thread is one click away. Manual edits are preserved on re-enrich.

All seven are optional and skipped silently when the website doesn't mention them. They're available as targets in the CSV import column-mapper too — common headers (`LinkedIn`, `IG`, `Twitter`, `Founded`, `Employees`, `Company Size`, …) are auto-recognised.

These fields are different from the **Contact**-level social fields. A Contact's `linkedin_url` and `twitter_handle` point at a specific person inside the account (the marketing director, the CEO); the Account-level URLs point at the company's own profiles. Both layers can be populated at once and they describe different things.

Status — where in the pipeline

Each account has a single **status** (lifecycle state):

Status	Meaning
`prospect`	Default. Not yet contacted.
`contacted`	You've reached out — at least one outbound message has been logged.
`in_negotiation`	Active deal conversation.
`customer`	They bought.
`lost`	Deal fell through.
`do_not_contact`	Hard opt-out. **GDPR / CAN-SPAM stickiness — survives merges.**

Status transitions are recorded with their **source** — `manual`, `pre_existing` (see below), `campaign_outreach` (auto-promoted when you log a first outbound message), `import_exact` / `import_fuzzy` / `import_llm_verified` (set during import dedupe), or `inbound_acquisition` (auto-promoted when the account is created with an inbound acquisition channel). Every change appends to `status_history`, so you have a permanent audit trail of who set what when, and why.

Auto-promotion in one place

Status moves forward without operator action in two cases:

- **Inbound channel on create** — when an account is logged with an `acquisition_channel` that's an inbound type (Adwords, Instagram DM, referral, …), it skips `prospect` and starts at `contacted`. The account has already reached out to you.

- **First outbound on a `prospect`** — when you log a first outbound message in any campaign, the account advances from `prospect` to `contacted`.

Recording a pre-existing customer

Sometimes the account is already a customer when you add them to LeadHunter — a relationship that predates your funnel, an existing client you're loading from a spreadsheet, or a long-standing renewal. You want them in the system as `customer`, but you don't want the dashboard to claim them as a new win you just closed.

On the account page, when you change status to **Customer**, tick **"Mark as pre-existing customer"**. The transition is still saved with the full audit trail (you can see when and by whom), but the dashboard's **Closed (won)** funnel skips it. Cost-per-customer math (CAC) skips it too — these accounts weren't paid for by the campaigns you're running today.

What counts as a win

The dashboard's **Closed (won)** metric counts a status transition only when both are true:

- The account's *current* status is `customer` or `lost`. A transition that was later flipped back to `prospect` or `contacted` is treated as a correction, not a real win, and drops out of the count.
- The transition wasn't tagged `pre_existing`.

Click **Closed (won)** on the dashboard to jump straight to the matching accounts list, filtered to `customer` + `lost`.

Relationship types — what they are to you

Orthogonal to status, an account can carry one or more **relationship types** (multi-select — one row can be both `client` and `partner`, or `reseller` and `press`).

The thirteen values, with their campaign-outreach guardrail policy:

Type	Use when…	On bulk-add
`client`	They pay you.	Allowed (`customer` lifecycle status separately gates outbound — see below).
`prospect`	Sales target — the default classification on most rows.	Allowed.
`reseller`	Sells your product downstream.	Allowed.
`affiliate`	Sends you referrals on commission.	Allowed.
`partner`	Strategic, co-marketing, integration.	Allowed.
`influencer`	Has audience reach you want to borrow.	Allowed.
`supplier`	You buy from them.	**Soft block** — flip `include_supplier=true` to override.
`investor`	Current or potential investor.	**Soft block** — flip `include_investor=true`.
`press`	Journalist, publication, podcaster.	**Soft block** — flip `include_press=true`.
`analyst`	Industry analyst (Gartner-style).	**Soft block** — flip `include_analyst=true`.
`candidate`	Hiring pipeline.	**Soft block** — flip `include_candidate=true`.
`competitor`	Direct competition.	**Hard block** — never overridable.
`personal_network`	Friends, family, ex-colleagues.	**Hard block** — never overridable.

Two separate guardrails apply at bulk-add time too:

- **`do_not_contact`** (lifecycle status) — *blanket* opt-out. Blocks every campaign regardless of goal. Hard, never overridable. Survives merges (GDPR / CAN-SPAM sticky).
- **`customer`** (lifecycle status) — soft block for sales-shaped goals. Flip `include_customers=true` to re-engage existing customers in a campaign. (Customer-expansion / renewal / event / research / partnership goals admit customers by default — see Outreach reasons and targets.)

These rules only apply at *campaign bulk-add*. Logging an individual message to an account already in a campaign doesn't re-run these checks — opt-outs are the exception: both the blanket `do_not_contact` status *and* a per-purpose opt-out matching the campaign's goal are enforced at message-send time too, so an opt-out recorded *after* the account joined a campaign still blocks that campaign's composer.

Per-purpose opt-outs

Real-world consent is purpose-scoped. An account that opted out of sales emails may still be fine receiving research interview requests; a press contact who asked to be removed from your media list may still attend events. LeadHunter exposes this via `do_not_contact_purposes` — a list of campaign goals the account has opted out of, separate from the blanket `do_not_contact` status.

A customer who wrote in *"please stop selling to me but we'd love to participate in your customer research"* gets `do_not_contact_purposes=['sales']`. Sales campaigns are blocked from contacting them; research, customer-expansion, renewal, and event campaigns can still reach them.

Recording an opt-out is a click on the **Per-purpose opt-outs panel** in the Account detail page right column (between Relationship Status and Relationship Type). The panel renders current opt-outs as removable chips, has a goal-picker for new opt-outs (with `source` and `reason` fields), and shows the most recent audit entries inline. Behind it: `POST /api/accounts/{id}/record-dnc/` writes the purpose to `do_not_contact_purposes` *and* appends an entry to `dnc_history` — same shape as `status_history`. The audit trail covers `opt_out` and `opt_in` events, with `source` (`manual` / `inbound_request` / `import`) and an optional reason. This is the GDPR / CASL paper-trail surface — keep it accurate.

The two layers compose: an account is blocked from a campaign if *either* `status='do_not_contact'` (blanket) *or* the campaign's goal is in `do_not_contact_purposes` (purpose-specific). Neither layer is overridable from the bulk-add include flags.

Zero duplicates

Accounts are deduplicated at the moment of creation. LeadHunter checks:

1. **Google Place ID** (exact match → auto-merge, 100% confidence).
2. **Name + city** (exact after normalisation — strips legal suffixes, punctuation, filler words → auto-merge, 95%).
3. **Phone** (normalised, exact match → auto-merge, 90%).
4. **Website domain** (normalised, exact match → auto-merge, 85%).
5. **Fuzzy name within the same city** (≥85% similarity → *suggested* merge for human review).

Auto-merges at levels 1–4 happen silently. Level-5 fuzzy candidates surface under **Accounts → Duplicates** for human review.

When a merge runs, the survivor becomes the **golden record**: every unique field from every duplicate is preserved (non-generic email beats `info@`, the company's own domain beats an aggregator URL, the longer scrape wins, custom-field arrays union, and so on). The survivor's `merge_history` JSONB grows an entry recording the absorbed rows' full snapshots — visible on the account detail page, so you can always read what was on each side before the merge.

Lifecycle status escalates and never demotes. `do_not_contact` survives every merge. The AI picks a canonical name when the duplicates disagree (*"Joe's Pizza"* beats *"Joe's Pizza Restaurant LLC Holdings 2014"*).

Merges aren't reversible — the absorbed accounts are deleted at the end. The `merge_history` snapshot is an audit trail, not an undo button. See Merge duplicates for the full operator workflow.

Archive and delete

Two different ways to remove an account from your day-to-day view — pick by intent.

Archive soft-hides the account from default lists while keeping every reference intact: campaign rows, scores, conversations, status history, expenses, audit trail. Use it when the row still belongs in your history but you don't want to see it any more — the business closed, merged into a competitor, pivoted out of your market, or simply stopped being relevant. You can attach a free-text **reason** when archiving (recommended — *"business closed Q1 2026"*, *"merged into X"*, *"out of geography"*), which is visible on the account header and in the audit trail. Archived accounts can be unarchived at any time, with a single click, no data loss.

A common case for archive: the lifecycle status fields (`lost`, `do_not_contact`) describe *intent toward the account* — we tried to sell and didn't close, or they opted out. **Archive** describes the *account's own state* — the organisation no longer exists or is no longer relevant. Both can be true at once (you can archive a `do_not_contact` account if the business also closed), and the two flags are independent.

Delete permanently removes the row and cascades to every campaign row, score, conversation, contact, and expense reference. There is no undo. The Delete action on the account detail page requires you to **type the exact account name** to confirm — the typo gate is intentional, since the operation can't be reversed. Use this only when you're sure the row should never have existed (a test row, a manual data-entry mistake that the merge flow can't fix). For everything else — even *"definitely not coming back"* — archive is the right answer.

Both actions live in the **Danger Zone** card at the bottom of the account detail page.

Acquisition channel — where the lead came from

Each account also carries an **acquisition channel** — the marketing channel that brought them in: `outbound` (default — you found them), `adwords`, `meta_ads`, `linkedin_ads`, `organic_search`, `organic_social` (Instagram DM, X, …), `referral`, `event`, `partner`, `cold_inbound`, or `other`.

Inbound channels (everything except `outbound` and `other`) **automatically start the account at `contacted`** instead of `prospect`, because they've already reached out to you. The change is recorded in the audit trail with the channel as the reason, so it's clear why the lifecycle skipped a step.

Channel-specific details — UTM parameters, ad campaign id, Instagram thread URL, the referrer's email — go in **acquisition metadata**, a free-form key/value bag attached to the account.

See Track inbound leads for examples on logging Adwords clicks, Instagram DMs, and referrals.

Custom fields and segmentation

Your business probably tracks things that don't fit standard fields — store size, vehicle count, license tier, contract expiry, whatever. Define them once as custom fields on your company; they then appear on every account and can be used in saved filters and campaign targeting.

Contacts live inside accounts

An **Account** is the organisation. A **Contact** is a person sitting inside it — first / last name, email, phone, mobile, job title, department, seniority (`c_level`, `vp`, `director`, `manager`, `senior`, `entry`), LinkedIn URL, twitter handle, plus flags for *primary contact* and *decision-maker*. One account can have many contacts (or zero if you've only logged the org so far).

The two are different things — don't confuse them. Filters, scoring, and campaigns all operate on Accounts. Outbound messages target a chosen Contact within the Account (via the campaign-account's `contact_*` fields), so the *person* you're emailing is per-campaign even though the *organisation* is global.

One Contact per account can be flagged **primary**. The primary contact is the default target for the conversation log when no other choice has been made for a campaign.

Notes are operator-only

The `notes` field on an account is **operator-only** — the AI doesn't read it during scoring or message drafting. Use it for things you want your team to see but not the model: pricing context, internal politics, *"Maria says don't email after 6pm"*.

If you want a piece of context to influence scoring, put it in a custom field or in the account's `business_type` / `specialization` — those *are* fed into the scoring prompt. If you want it to influence outbound drafting, put it in the conversation history (every prior message is read when AI Continue runs).

Status, types, and channel — what each one answers

Three orthogonal fields describe an account's place in your world, and it's worth keeping them straight:

Field	Answers	Single or multi?
`status`	Where in the sales pipeline?	Single
`relationship_types`	What kinds of relationship does this account have *to me*?	Multi
`acquisition_channel`	How did this lead first reach me?	Single

A `customer` (status) tagged as both `client` and `reseller` (relationship_types) that arrived via `adwords` (acquisition_channel) is a perfectly coherent row. Each field answers a different question; none of them subsumes the others.

Which campaigns this account is on

The account detail page has a **Campaigns** card listing every campaign this account is part of. Each row shows the campaign name, its goal, and where the account stands in that campaign (review status and outreach status), with archived campaigns sorted to the bottom.

There are two links per row:

- The **campaign name** opens the campaign.
- The **Outreach** button jumps straight into that campaign's outreach inbox *with this contact already open* — so you go from "looking at the account" to "writing the next message" in one click, instead of opening the campaign, finding the right tab, and scrolling to the row. The deep link opens the contact even when it's filtered out of, or sits past, the inbox's current page.

If the account isn't in any campaign yet, the card says so and points you toward adding it to one. See Campaign for how accounts move into outreach, and Log a conversation for working a thread.

Read next

- Campaign — how accounts move from "in the database" to "in active outreach"
- ICP and scoring — how the per-product fit score is computed
- Merge duplicates — the operator workflow when LeadHunter flags candidates

Campaign

A **Campaign** is one outbound initiative against a list of accounts. Every campaign points to exactly one Product, which means it inherits that product's ICP, default language, and scoring calibration. The campaign is the unit you operate against — bulk-add accounts to it, score them, review the scores, log conversations against them, and measure spend against the outcomes it produced.

Every campaign also carries a **goal** — *why* you're reaching out. The default is sales prospecting; ten others cover partnership recruitment, press / visibility, investor outreach, recruiting, customer expansion, win-back, and more. The goal reshapes the ICP, the scoring rubric, the message intent, and the guardrails — same product, different intent, different outcome. See Outreach reasons and targets.

Switching a campaign's goal mid-flight is allowed but the existing scores stay as they were (they were rated against the previous goal's rubric). When that happens, the campaign detail page shows an orange "rescore to refresh" banner that tells you how many accounts have scores under a different goal and links straight to the scoring page — one click and you're back in sync.

What a campaign carries

Bucket	Fields
Identity	`name`, `slug` (auto-generated from name, collisions suffixed within the project), `project`, `product`
Intent	`goal` (sales / partnership / press / investor / recruiting / customer expansion / win-back / event / research / influencer / renewal — see Outreach reasons and targets)
Scope	`audience_config` (saved-filter / manual selection)
Language	`communication_language` (inherits from Product when blank)
Workflow	`status` (operator-facing stage), `scoring_status` (background-job state), `archived_at`
Cached statistics	`total_accounts`, `scored_accounts`, `approved_accounts`, `outreached_accounts`, `responded_accounts`, `response_rate`
AI surface	`icp` (inherited from product, read-only on the campaign), `score_distribution` (histogram of fit scores, see below)
Provenance	`created_by`, `created_at`, `updated_at`

The cached statistics are maintained as you operate — outbound logs bump `outreached_accounts`, inbound pastes bump `responded_accounts`, reviews bump `approved_accounts`. The numbers you see on the campaigns list and on the campaign detail header come from these columns, so they're fast even on large projects.

Language

Each campaign carries its own **communication language**. It drives the language of every piece of AI-generated content the campaign produces — the ICP summary, the reasons attached to each fit score, and the outbound message drafts. Leave it blank to inherit the language set on the Product; set it explicitly to override the product default for this campaign only. This lets you run one campaign of the same product in Spanish for the Spain market and another in English for the UK without duplicating the product.

When sending a message, language resolution checks the **account** first, then walks the chain Account → Campaign → Product → User → English. So an account whose own language is set to Catalan always reads Catalan, regardless of which campaign reaches them.

Workflow

A campaign moves through a workflow `status`:

1. **`draft`** — Created, no accounts yet.
2. **`icp_definition`** — Verify the inherited ICP, adjust as needed, approve.
3. **`database_config`** — Decide where accounts come from (saved filter, manual list, import).
4. **`scoring`** — Each account in scope gets a per-product fit score.
5. **`outreach`** — You're actively reaching out.
6. **`active`** — Ongoing campaign with mixed reach-out + reply work.
7. **`paused`** — Temporarily on hold; everything is preserved, you've just stepped away.
8. **`completed`** — Campaign is wrapped up.

The status is an **operator hint**, not a hard state machine — LeadHunter doesn't refuse to score a campaign because its status says "outreach". You can move freely between stages, and the name is shown to your team for context, not enforced.

Archive is orthogonal: any campaign can be soft-hidden via the archive action (sets `archived_at`). Archived campaigns disappear from the default list but stay in the database with every conversation, score, expense, and audit-trail entry intact. Use archive for *"campaign is done but I want the history"*; use **delete** (name-confirmed) only when you're sure.

Renaming a campaign is inline on the detail page — click the title, type, save. No extra modal.

Exclude from global stats

A campaign can be flagged **"Exclude from global stats"** from the detail page's *Aggregate stats* row. When the box is ticked, the campaign's outreach, replies, and inbound-close events stop contributing to:

- the **Dashboard** funnel charts (initiated / responded / closed totals and daily series),
- the **cohort response rate** and **cohort close rate**,
- the **High-ICP reached** tile and its 30-day rate (any account whose only campaign membership is the excluded campaign drops out),
- the **Stats by product and campaign** *by product* and *by channel* roll-ups,
- and the per-product CAC ratios.

The campaign's own page is unaffected — its score distribution, accounts panel, costs panel, and cost summary all still show real numbers. On the *Stats by product and campaign* page, the campaign row is still listed with its own totals, but tagged with an amber **Excluded** chip so you can tell at a glance that it isn't being counted toward the project-wide funnel.

The High-ICP rule is account-level rather than event-level: an account stays counted as long as it has at least one membership in a *non-excluded* campaign (or no campaign membership at all, e.g. a pure-inbound lead). So enrolling the same account in a second, real campaign restores it to the High-ICP count automatically.

Use the flag for campaigns whose numbers would mislead the funnel rather than describe it:

- **Experiments and calibration runs** — a one-off test campaign you used to dial in the ICP, not real outreach.
- **Bulk imports of pre-existing customer relationships** — accounts that became customers before LeadHunter ever sent a message, with a single retroactive "log sent" / "status → customer" event each. Counting those as a 100% close-rate cohort would tilt the dashboard for weeks.
- **Internal sanity checks** — a campaign you reach out to yourself with from another seat, just to verify formatting.

One subtlety: status → customer transitions live on the **Account**, not on the campaign, so flipping this checkbox doesn't retroactively unwin a customer. If a particular conversion shouldn't count, flip the account back to `prospect` first, or stamp the `status_history` entry as `source=pre_existing` — the funnel already excludes that source.

Scoring status

Separate from the workflow `status` above, every campaign carries a **`scoring_status`** that reflects the background scoring job:

- **`idle`** — nothing in progress.
- **`running`** — the score-leads task is processing this campaign's accounts. The campaign page polls and updates as rows come back.
- **`completed`** — the most recent scoring batch finished successfully.
- **`failed`** — the most recent batch hit an error. The error is recorded and the operator can retry.

Scores reuse across campaigns of the same product — see ICP and scoring → Score caching.

Score distribution

Once scoring has run, the campaign page shows a **10-bucket histogram** of fit scores (`0–1` , `1–2` , …, `9–10`). It's the fastest way to read whether the ICP is well-calibrated — a healthy distribution skews bimodal (excellent + mismatch piles, modest moderate middle); a flat or all-moderate distribution usually means the ICP isn't sharp enough yet. See ICP and scoring → Why are my first batch of scores reading vague.

Campaign accounts

The join row between a campaign and an account is a **CampaignAccount**. It carries per-campaign workflow data — the fields you'll touch most as an operator:

Field	What it holds	Where it moves
`review_status`	Have you, the operator, vetted this AI-scored account yet?	`pending` → `approved` / `rejected` on the campaign's review queue.
`outreach_status`	What's happening with outbound?	`not_started` → `draft` → `scheduled` → `sent` → `bounced` / `responded`. Forward-only.
`override_score`	Optional manual score that supersedes the AI score **for this campaign only**. The Product-level AI score doesn't move.	Set from the **Override score** link on each row in the campaign's Score review page. The dialog accepts a value 0–10 and a Clear button reverts to the AI score.
`contact_name` / `contact_title` / `contact_email` / `contact_linkedin`	Which specific person the campaign's outreach is aimed at, when the underlying account has multiple contacts.	Picked from the account's Contact list when you start a thread.
`tags`	Free-form per-campaign labels — *"Q2 priority"*, *"warm intro"*, etc.	Edited on the row.
`notes`	Per-campaign notes that don't belong on the account itself.	Edited on the row.

The underlying AI fit score lives on the **Product**, not on the CampaignAccount, so adding an account to a second campaign of the same product doesn't trigger a re-score.

The operator pipeline

The day-to-day loop runs through these CampaignAccount fields:

1. **Scoring fills `ai_score`** for every row when the campaign first scores.
2. You **review** — sort by score, read the AI's reasons, mark each as `approved` or `rejected`. Or set `override_score` if the AI was clearly off.
3. For approved rows you **kick off outreach** — open the Conversation tab, pick a contact, draft and send. Each `Log sent` flips `outreach_status` to `sent` and counts the account as *initiated* in the dashboard funnel.
4. When the lead replies, **paste the inbound** — `outreach_status` flips to `responded` and the dashboard's response cohort gets credit.

The campaign detail page surfaces all four stages — review queue, accounts panel, conversation log, costs panel — so the loop stays inside one screen.

Approvals and rejections feed back into scoring

Every approve or reject you mark is a calibration signal. Once a product crosses ~10 reviewed accounts across all its campaigns, the most recent approvals + rejections start flowing into the scoring prompt as concrete examples. See ICP and scoring → The feedback loop.

This makes the first campaign on a new product the most important one to be thorough on the review step — every decision shapes the next several hundred scores.

Pruning the low-fit tail after scoring

Once scoring finishes, the bottom of the list is usually noise — accounts the model flagged as **mismatch** (score < 5.0) that aren't worth your outreach time. The campaign-accounts panel has a **Remove low-fit** button next to the *In this campaign* count that clears them in one pass.

The flow is two-step: clicking the button runs a **dry-run preview** and shows the count it would remove, then a confirm dialog. Nothing is deleted until you confirm.

What it targets: only the **mismatch** band (the lowest tier). **Moderate** and **excellent** accounts are never touched. Unscored accounts are also left alone — they haven't been judged yet, so the model has no opinion to act on.

What it keeps by default, even when they're scored as mismatch:

- Accounts you've **manually approved** (`review_status='approved'`). Your judgment beats the model's — if you've already vetted the account, the AI's "mismatch" call is the wrong one to honour. The preview dialog tells you how many were kept on this axis.
- Accounts where **outreach has already started** — any `outreach_status` other than `not_started` (`draft` , `scheduled` , `sent` , `bounced` , `responded`). Removing them would orphan the conversation thread. The preview surfaces this count separately so you know exactly what's being skipped and why.

Both defaults can be turned off if you really want a hard prune, but the safe defaults are the right call almost always.

The removal is scoped to **this campaign only**. The same account in a second campaign of the same product is untouched — and the underlying Product-level score isn't deleted either, so if you later add the account to a new campaign with a tightened ICP and rescore, the row gets a fresh chance.

This is a destructive operation (no soft-delete, no archive — the `CampaignAccount` rows are gone). The account itself, its contacts, and any conversation messages on **other** campaigns are unaffected. If you removed a row by accident, re-add it from the *Available to add* column.

Downloading approved accounts as a CSV

The campaign-accounts panel has a **Download CSV** button next to **Add new account**. By default it pulls the rows you've marked **Approved** with four columns — **Name**, **Website**, **Location** (City and country combined), and **AI Score**. One click, file lands in Downloads.

The same modal lets you widen the row filter (Approved + Pending, or every status including rejected) and tick on any of ~40 optional columns grouped by topic:

- **Contact info** — Phone, email, the per-campaign selected contact (name, title, email, LinkedIn)
- **Address parts** — Street, city, state/region, postal code, country, latitude/longitude as separate columns (handy when importing into another CRM that wants them split)
- **Account socials** — LinkedIn, Instagram, X, Facebook, YouTube, TikTok

- **Business profile** — Business type, specialization, year founded, employee count bucket, language, rating
- **Pipeline and tags** — Account status, relationship types, acquisition channel, captured value, campaign tags
- **Workflow and audit** — Review status, outreach status, override score, score reasons, internal notes, source, imported ID, date added
- **Custom fields** — Every per-company custom field shows up automatically in its own section, so columns like "Annual revenue" or "Industry segment" are one click away

A typo in the column name returns an error instead of silently emitting a blank column, so you can't end up with a file that looks correct but is missing something.

Guardrails

Bulk-adding accounts to a campaign honours every protection LeadHunter knows about, in one pass. The full matrix depends on the campaign's **goal** — a Press campaign doesn't block press accounts, a Recruiting campaign doesn't block candidate accounts, a Customer expansion campaign treats customers as the target audience. See Outreach reasons and targets for the per-goal table.

Two invariants survive every goal:

- `do_not_contact` **accounts are always blocked.** GDPR / opt-out is sticky regardless of campaign intent. To re-engage someone, change their status manually first.
- `competitor` **and** `personal_network` **accounts are always hard-blocked.** Never overridable, in any goal.

Everything else flexes per goal: customer status is admissible in expansion / renewal / event / research / partnership goals and blocked elsewhere (overridable with `include_customers`); the supplier / investor / press / analyst / candidate soft-blocks each turn off for the goal that *targets* that type.

When the UI shows you blocked accounts after a bulk-add, it groups them by reason and tells you exactly which flag flips them in — phrased in terms of *this* campaign's goal. A bulk-add into a Sales campaign that hit press accounts will tell you "this Sales campaign soft-blocks press accounts; pass `include_press=true` to override, or switch the goal so press is the target audience". A Press campaign would not produce that message because press accounts aren't blocked there.

Guardrails apply at the moment of bulk-add. Once an account is in a campaign, logging individual outbound messages to it doesn't re-run these checks (DNC is the exception — DNC is enforced at message-send time too).

Costs and CAC

Every campaign carries a ledger of **expenses** — Adwords spend, agency fees, internal labor hours, software, content, events. Add entries from the campaign detail page or via the quick-add button on the campaigns list. The campaign then computes cost-of-acquisition:

- **Cost per outreached account** — total spend ÷ accounts you've reached out to.

- **Cost per responded account** — total spend ÷ accounts that have replied.
- **Cost per closed-won customer** — shown on the *Stats by product and campaign* page; cohort-attributed using the last-30-day window.

Mixed currencies show per-currency totals without auto-conversion — LeadHunter doesn't carry an FX layer. CAC is only computed when expenses share one currency; otherwise the UI surfaces a per-currency breakdown and explains the missing ratio.

Auto-tracked API costs (LLM tokens, Google Maps calls, Tavily searches) sit in their own card on the same panel: USD total + a per-model breakdown of which model spent what. It's kept out of the CAC math because it's in USD and your user-entered expenses may not be, but the two cards together answer the *"what did this campaign actually burn?"* question. Per-account enrichment cost is exposed on the account detail.

See Track campaign costs and CAC.

Read next

- Outreach reasons and targets — what each goal does and how to pick one.
- ICP and scoring — how the score column on every CampaignAccount gets populated.
- Run your first campaign — the end-to-end operator walkthrough.
- Build a saved filter — populate a new campaign in one bulk-add.
- Track campaign costs and CAC — log spend, read efficiency.

Acquisition channels

A **channel** in LeadHunter is one inbound source of new accounts: Google Ads, Meta Ads, LinkedIn Ads, organic search, organic social, referrals, events, partners. Channels live next to Campaigns but answer a different question.

- **Campaigns are outbound.** You pick the targets, score them against an ICP, and reach out. Campaign costs are the labor + agency + content you spent reaching that specific cohort.
- **Channels are inbound.** Leads come to *you* through them. The cost is the monthly ad spend (Adwords, Meta) or event budget — entered as time-bucketed line items, not per-account.

Both surfaces feed the same outreach workflow once an account is in the system. The split is purely about **where the spend lives** and **how the ROI math works**.

The shape of a Channel

Field	What it does
Kind	One of `adwords`, `meta_ads`, `linkedin_ads`, `organic_search`, `organic_social`, `referral`, `event`, `partner`, `cold_inbound`, `other`. Same enum as the `acquisition_channel` field on individual accounts.
Label	Optional sub-label so one project can have multiple channels of the same kind — e.g. an Adwords "Search EU" channel and an Adwords "Display retargeting" channel. Blank label = the default channel for that kind.
Status	`active` / `paused` / `ended`. Paused channels stop collecting new spend but keep their history.
Started / ended on	Optional date range — when the channel went live, when it stopped.
External reference	Free-text link to the platform — your Google Ads customer ID, the Meta ad account, an event slug.

Channels are **scoped per company** (the same way Campaigns and Products are). Each company in LeadHunter has its own set of channels.

Spending: monthly buckets

Channel spend is logged as **time-bucketed line items**. The typical entry is one row per month: "Adwords May 2026 = €1,200". For one-off costs (a conference sponsorship, a single-day burst), set the period's start and end to the same date.

Each line item carries:

- **Amount + currency** (ISO 4217 — EUR / USD / GBP / …). No FX conversion happens; the rollup groups by currency.
- **Period start / end** (closed range, both required).

- **Vendor** — "Google Ads", "Meta", "EventCo".
- **External reference** — invoice number, ad-platform campaign ID.
- **Description + metadata** — anything you want in the audit log.

You can hand-enter spend monthly, or import a CSV of historical line items. There's no ad-platform sync yet — see the roadmap below.

What you get back

Every channel exposes three rollup blocks on its detail page:

Cost summary

Total spend per currency, primary currency (the one with the largest total), and cost-per-acquired / cost-per-responded / cost-per-closed-account ratios when expenses share one currency. Mixed currencies skip the CAC math and explain why — adding an FX layer is on the roadmap.

Value summary

If you've set lifetime values on Products, Campaigns, or individual Accounts (see Account value), the channel sums the captured value from every customer it acquired:

- **Total value by currency**.
- **Customers with value** vs **customers total** — the difference is "how many customer accounts are missing an LTV entry that an operator should fill in".
- **By source** — how the captured value was set: `manual` (operator), `auto_campaign` (snapshotted from a campaign override), `auto_product` (snapshotted from the product default).
- **Average account value** in the primary currency.

ROI

Value ÷ spend, expressed as a multiplier (e.g. `25.00` = €25 in customer value per €1 of channel spend). Only computed when both sides share a single currency *and* those currencies match; otherwise the reason explains why (mixed currencies, no expenses, no customers with value).

How leads attach to a channel

Two paths:

1. **Manually**: edit an Account and pick the channel from the dropdown.
2. **Auto-attach**: setting `acquisition_channel` on an Account (via the dropdown, CSV import, or API) auto-creates the project's default channel of that kind if one doesn't already exist, and points the Account at it. This means the legacy "acquisition channel" string on every account stays in lockstep with the structured channel for cost rollups, with no extra work from the operator.

Outbound and "other" acquisition kinds **don't** create channel rows — outbound is owned by the Campaign surface, and "other" is too ambiguous to auto-route.

Inbound accounts always start as `contacted`

When you create an account on an inbound channel (anything except `outbound` and `other`), LeadHunter automatically promotes its status from `prospect` to `contacted` — the account has already reached out, so the default "untouched prospect" status would be wrong.

The promotion records `source: inbound_acquisition` in the status history, so the audit trail is clear about *why* the account opened in `contacted` state. You don't need to change anything else.

Dashboard breakdown

The stats dashboard now shows three columns: **by product**, **by campaign**, and **by channel**. Each channel row carries today / 7d / 30d acquisition counts plus its cost / value / ROI scoped to the last 30 days.

This is where the channel ROI story becomes operational: a glance tells you which channels are pulling their weight this month and which are bleeding cash.

How channels and campaigns relate

An inbound account can also be added to a campaign — the channel records *where it came from*, the campaign records *how you're engaging it*. A pure-inbound account that you don't follow up on outbound only has a channel; an outbound prospect only has a campaign; an inbound lead you outbound-nurture has both.

Costs **don't double-count**: ad spend lives on channels, outreach labor lives on campaigns. The two roll up separately and tell complementary stories.

What's coming

Today's channel surface is the foundation. The roadmap:

- **Webhook ingestion** — public endpoints per channel that accept form submissions / ad-platform lead notifications / IG DM bridges, so accounts get created automatically when a form is filled in or an ad delivers a lead. UTM parameters and referrer data land in the account metadata.
- **Ad-platform sync** — daily OAuth pulls from Google Ads, Meta Marketing API, LinkedIn Campaign Manager to populate spend automatically instead of monthly hand-entry.
- **Per-click attribution** — when the ad platform delivers CPA data, individual accounts can carry their own acquisition cost (rather than just sharing the monthly bucket).
- **Email gateway** — a per-channel forwarding address so inbound emails create accounts and threaded conversation messages automatically.
- **Multi-touch attribution** — split a customer's value across the channels that touched them, weighted by first-touch / last-touch / linear / time-decay.

See What's new for what just shipped, and the roadmap section in the value-tracking guide for the LTV side.

Outreach reasons and targets

Cold outreach isn't a single activity. A sales prospecting note to a CFO and a press pitch to a journalist look superficially similar — short, personalised, with an ask — but everything that actually drives results is different. The target audience, the definition of a "good fit", the message angle, and what counts as success all differ.

LeadHunter makes that explicit with a **goal** on every campaign. Pick one when you create the campaign and the rest of the platform adapts: the ICP framing, the fit-scoring rubric, the message-draft tone, and the bulk-add guardrails all shift to match what you're actually trying to accomplish.

The eleven goals

The eleven goals cover the reasons businesses actually do outbound. Pick the one closest to your intent; the default is **Sales prospecting**, which is what most outbound is.

Goal	What it's for	Who you're targeting
Sales prospecting	Find companies likely to buy.	Decision-makers at companies matching your buyer persona.
Partnership / reseller	Recruit resellers, integrators, affiliates.	Companies whose customers also need what you sell.
Press / visibility	Get coverage.	Journalists, publications, podcasts, newsletters in your space.
Influencer / community	Reach amplifiers.	Creators and community leaders with audience overlap.
Investor outreach	Raise capital.	VC funds, angel investors, family offices whose thesis fits your stage.
Recruiting	Source candidates.	People matching the role's skills and seniority.
Customer expansion	Upsell or cross-sell.	Existing customers who'd benefit from more product.
Win-back / reactivation	Re-engage lost accounts.	Former customers worth a second pass.
Event invitation	Drive attendance.	People the event is relevant to (no constraints on relationship type).
Research interviews	Recruit participants.	People whose perspective is valuable to your research.
Renewal	Lock in renewals.	Current customers approaching the end of a term.

Setting a goal isn't a one-way door — you can change it on the campaign detail page if the angle shifts. Switching the goal re-applies the new guardrails on subsequent bulk-adds; existing accounts stay in the campaign and their fit scores stay as last computed (rescore explicitly if you want fresh scores under the new framing).

What changes when you pick a non-default goal

Five things adapt to the goal you choose:

1. The Ideal Customer Profile is reframed

When you generate or revise the ICP, the AI is told what *kind* of target to model. For a **Press** campaign that means "publications, journalists, podcasts that cover our space and reach the audience we want — editorial slant matters more than company size". For a **Recruiting** campaign it's "candidates with the skills, experience, and seniority we're hiring for". For **Partnership** it's "companies that would be strong resellers, integration partners, or affiliates".

You still get the same ICP card with industries / target roles / pain points — but the values reflect the goal. A press ICP will list editorial domains as industries and roles like "Senior Editor" or "Reporter" as job titles. A recruiting ICP will list adjacent companies and the role's level. The structure is consistent so you can review and revise it the same way; only the content shifts.

See ICP and scoring for how the ICP is generated and refined.

2. The fit-scoring rubric shifts

Fit scoring still produces a 0-10 number with a typed list of reasons (positives, negatives, neutral context). What changes is **what is being scored against**.

- For **Sales**, the score answers *"how well does this match our buyer profile?"* — industry alignment, size, signals in the website that suggest the pain points.
- For **Partnership**, the score answers *"how strong a partner would this be?"* — shared audience, product complementarity, credibility as a reseller or integrator.
- For **Press**, the score answers *"is this the right outlet for our story?"* — editorial coverage of the space, audience size and engagement, editorial angle compatibility.
- For **Recruiting**, the score answers *"how well does this person fit the role?"* — skills alignment, level, recent stack, signs of openness.

The numbers mean the same thing — 8–10 is excellent, 5–7 is moderate, 0–4 is mismatch — but the *axis* the score is rated against changes with the goal. A score of 9 in a Press campaign means "highly relevant outlet, will probably take the pitch"; a 9 in Partnership means "strong shared-audience partner".

3. The outbound message intent matches the goal

When you draft a message, the AI starts with what *kind* of message it's writing. A **Sales** draft is a short cold opener that introduces the product and asks for a call. A **Press** draft is a pitch with a newsworthy angle up front, no sales language, and a concrete offer (data, an interview, an exclusive). A **Partnership** draft proposes collaboration, not a sale. A **Recruiting** draft is a sourcing note specific to the role with a low-friction next step.

You'll see the difference even with identical product context and the same target account. The tone, the call-to-action, and the opening hook all shift to fit the goal. You can still translate, edit, or rewrite — the AI is giving you a starting draft that matches the goal, not a final message.

See Messaging for the full conversation log.

4. The add-to-campaign guardrails flip

LeadHunter's guardrails — the rules that decide *who's allowed into this campaign* — depend on the goal. A few invariants survive everything:

- `do_not_contact` **accounts are always blocked.** This is a hard GDPR / opt-out rule. No goal, no flag, no flow can override it. To re-engage someone on DNC, change their status manually first.
- `personal_network` **accounts are always hard-blocked.** Bulk-blasting your warm personal contacts through a campaign is always the wrong move — that's what a one-off intro is for. No goal flips this.
- `competitor` **accounts are hard-blocked for most goals, but admitted for press / event / research.** A trade publication that covers your competitors is still a valid press pitch target; a roundtable invites peers; market research routinely interviews competitor employees. For the other goals (sales, partnership, investor, recruiting, expansion, win-back, renewal, influencer), competitors stay hard-blocked.

Everything else flexes per goal:

- A **Sales** campaign soft-blocks supplier / investor / press / analyst / candidate accounts — they're not buyers, you can override per type if you really want.
- A **Press** campaign *admits* press and influencer accounts by default — they're the target. It still soft-blocks suppliers and candidates.
- A **Recruiting** campaign admits candidate accounts and blocks the others.
- A **Customer expansion** or **Renewal** campaign treats customers as the target audience — they're not blocked, they're who the campaign is for.
- An **Event** or **Research** campaign has no soft-blocks at all — anyone might attend or be interviewed.

When you bulk-add accounts to a campaign and the guardrails block some of them, the UI tells you exactly which accounts were skipped and why — DNC, customer status, competitor, supplier, and so on — with a "flip this flag to include them" message where overrides apply.

5. The "what's success" semantics adjust

For a **Sales** campaign, success is *the account became a customer*. For **Press** or **Influencer** or **Investor**, success is *they replied* — coverage and meetings come from there. For **Partnership**, success is *they became a partner*. For **Recruiting**, *they engaged about the role*. For **Win-back**, success is again *became a customer* (after having been lost). For **Renewal**, it's *they renewed* (stayed a customer through the renewal date).

The dashboard funnel routes the **closed** event per goal:

- **Sales, customer expansion, win-back, renewal** close when the account transitions to `status=customer` — the legacy meaning, unchanged.
- **Partnership, press, influencer, investor, recruiting, event, research** close when the account first replies. For these goals "they responded" *is* the success metric — a journalist replying is a press hit; a candidate replying is a recruiting win.

So a press campaign that gets one outbound + one reply shows `responded=1` and `closed=1` (they are the same event for press). A sales campaign with the same shape shows `responded=1` and `closed=0` until the

account also transitions to customer. The `initiated` and `responded` columns are universal across goals; only `closed` adapts.

Partnership and customer-expansion currently use the inbound-response signal as a proxy — the cleaner signals (partner relationship-type acquired, customer usage expanded) aren't tracked in audit-log form yet.

Picking a goal

A few rules of thumb:

- **If it's sales, pick Sales.** It's the default for a reason. Most outbound is sales.
- **If it's not sales but you want to reach a *company* (not a person at a specific role), pick the closest match.** Partnership for resellers/integrators; Press for media; Investor for funds; Event for invites; Research for interviews.
- **If it's not sales but you want to reach a *person* in a specific role, pick Recruiting.** Influencer / Press also reach people, but they're targeting *publications* or *creators with platforms* — Recruiting is for the candidate-as-individual case.
- **For existing customers**, pick Customer expansion (upsell) or Renewal (lock-in). They differ in the message intent — expansion leads with new value, renewal leads with a soft "still on track to renew?".
- **For lost customers**, pick Win-back. It's tonally different from a fresh sales pitch — leads with what's changed since you parted ways.

If your intent doesn't fit cleanly, **start with the closest goal and switch later if needed**. The cost of switching is one click; the cost of using "Sales" for everything is a long tail of messages and scores that don't match what you're actually doing.

A worked example: three campaigns on one product

Same product, same company, three different goals:

- **Campaign A — Sales prospecting.** ICP: directors of marketing at B2B SaaS companies with 50-200 employees. Scoring rubric: rate fit as a buyer. Message intent: cold opener proposing a 15-minute call. Blocked: customers, suppliers, press, investors, analysts, candidates.
- **Campaign B — Press / visibility.** ICP: tech publications and podcasts covering go-to-market / sales tooling. Scoring rubric: rate fit as a press outlet. Message intent: pitch a newsworthy angle (new launch, surprising data, fresh perspective). Blocked: competitors, personal network. Press accounts are the target, so they pass.
- **Campaign C — Customer expansion.** ICP: existing customers using the basic plan with growing team size. Scoring rubric: rate expansion potential. Message intent: friendly account-management nudge mentioning a specific upgrade fit. Blocked: nobody who's not already a customer matters here.

Each campaign uses the same Product (same description, same example URLs) but produces completely different ICPs, score reasons, message drafts, and audience lists — because the goal sets all of that.

Read next

- Campaign — the unit of outreach work.

- ICP and scoring — how fit scores get assigned.
- Messaging — drafting and logging the outbound itself.
- Run your first campaign — end-to-end walkthrough.

Target persona and scoring

The **target persona** describes who a campaign is reaching, and lives on the **(Product, Goal)** pair: a sales campaign of a product carries a buyer persona, a press campaign of the same product carries a publications-and-journalists persona, a recruiting campaign carries a candidate persona — each independent, each with its own approval state. We historically called this the "Ideal Customer Profile" (ICP) and the term is still used as an alias in sales-shaped flows, but the underlying object is goal-agnostic. Every account that enters a campaign gets a **fit score** for that goal: a number 0–10, a categorical label, and a short list of typed justifications.

Scoring is the bridge between your raw account database and an actionable campaign list. Done well, it turns *"here are 2,000 leads"* into *"here are the 80 to focus on this week."*

Structure of a target persona

LeadHunter's target persona is **structured** — not just a paragraph. The AI generates it from your Product's description, website, and example URLs, then you review and approve.

Field	What it holds
`name`	Short label for the persona (e.g. *"The Tech-Forward CFO"* for sales, *"Senior B2B SaaS Reporter"* for press, *"Senior Backend Engineer, Series A Berlin"* for recruiting). Auto-generated; editable.
`summary`	Plain-language overview of who you're targeting. Read first by the model.
`industries`	List of sectors / editorial domains / specialties relevant to the goal.
`job_titles`	List of relevant decision-maker / journalist / candidate / partner roles.
`pain_points`	What this persona cares about — what would motivate them to engage. For sales it's product-relevant pain; for press it's editorial angles; for candidates it's career drivers.
`company_size_min` / `company_size_max`	Size band — employees for company targets, audience size for publications, team size for partners. Either may be omitted.
`confidence_level` / `confidence_score`	How confident the AI is in this draft (`low` / `medium` / `high` , plus a 0–100 score). A *low*-confidence first draft is the signal to spend more time editing before approval.
`source_urls`	URLs the AI cited while drafting. Useful when reviewing — you can see what the model was reading.
`is_approved`	True once you click **Approve**. Scoring still runs against a draft, but approval is the signal the document is calibrated.
`revision_count`	Number of times the persona has been re-generated. Bumped on every regenerate.
`revision_feedback`	Free-text you provide when asking the AI to redraft (*"focus more on tier-1 trade press, less on general business pubs"*) — fed into the regeneration prompt.

You can edit any field after generation; the model uses the whole structured persona plus the Product description on every score.

ICP review and approval

A fresh ICP starts in **Pending approval**. You see the AI's draft, edit it, click **Approve** to mark it ready for scoring. Until approval, scoring runs against the draft anyway — but the *"this ICP is approved"* flag is the signal that the document is calibrated and stable.

If the first draft is off, you have two options:

1. **Edit in place** — change individual fields and re-approve. Best for small refinements.
2. **Regenerate with feedback** — fill in `revision_feedback` (*"focus on independent garages, drop the enterprise framing"*) and ask the AI to re-draft. The feedback flows into the regeneration prompt; `revision_count` increments. Best for course corrections.

You can revise the ICP at any time. Pay attention to `confidence_level` on the draft — a `low`-confidence draft from a sparse Product description usually means the model couldn't anchor; sharpening the Product description before regenerating is more effective than editing the ICP by hand.

How scoring works

Goal-aware framing

Scoring asks *"how well does this account fit the target?"* — but the answer to "what target?" depends on the **campaign's goal**. For a Sales campaign it's "how well do they fit our buyer profile". For a Press campaign it's "how well does this outlet fit our story". For Recruiting it's "how well does this person fit the role". For Partnership it's "how strong a partner would this be".

The 0-10 scale and the typed reasons (positive / negative / neutral) stay the same regardless of goal, so a 9 always means *very strong fit* and a 3 always means *clear mismatch*. What shifts is the **axis** being rated. The model is told the goal up front so the score reasons it returns are framed correctly — *"covers our space editorially"* for Press, *"complementary product stack"* for Partnership, *"recent stack matches the role"* for Recruiting.

See Outreach reasons and targets for the full goal list.

What the model sees — the full ordered list

Every input that contributes to scoring, in the order it appears in the prompt. If a section is blank or empty, it's skipped entirely — no empty headers reach the model.

Rubric block — built once per scoring run:

1. **Language** — what language to write reasons in. Resolved from Campaign → Product → your account default → English.
2. **Campaign goal** — sales / partnership / press / investor / recruiting / customer expansion / win-back / event / research / renewal / influencer. Decides which axis the score is rated against (see Goal-aware framing above).

3. **Target persona (ICP)** — the structured persona: name, summary, industries, job titles, pain points, company size range. Generated once from the Product description + website + example URLs; revise or regenerate on the campaign page.
4. **Product name** — the Product label.
5. **Value proposition** — the Product's free-form description. Capped at **2000 characters**. Edit it on Settings → Products; changes take effect on the next rescore.
6. **Campaign scoring brief** — optional free-text guidance specific to this campaign — *"accounts must have a physical retail location"*, *"down-weight chains; prioritise owner-operated stores"*, *"weight community-event capability heavily"*. Capped at **2000 characters**. Edit it inline on the campaign page or through the "Give feedback & rescore" modal on the scoring review page. Blank by default.
7. **Good-fit example URLs** + scraped content — 2–3 URLs on the Product, fetched in parallel. Capped at **2000 characters per URL**. **Single most powerful lever after the persona itself** — short of a sharp persona, sharp example URLs do more for scoring quality than anything else.
8. **Bad-fit example URLs** + scraped content — same shape.
9. **Calibration examples** — your most recent approvals + rejections, once the feedback loop crosses the ~10-account threshold for the (product, goal) pair. Up to 3 of each side. Below threshold the section is omitted — small samples are noisier than no sample.

Per-account block — repeated once per account in the batch:

10. **Account ID** — load-bearing identifier the AI echoes back so we can map scores to accounts.
11. **Name** — the account's name.
12. **Location** — city, region, country, joined with commas.
13. **Business type** — free-form category, set during enrichment.
14. **Specialization** — sub-category if any.
15. **Website** — URL.
16. **Rating** — only included in scoring (not outreach), out of 5.
17. **Language** — the account's own language. The model uses it as a signal, not as a writing instruction.
18. **Email, Phone** — contact details if known.
19. **Notes** — free-form operator notes. Capped at **500 characters**. Treat this field as operator-shared AI context, not private scratch space — it goes to the model.
20. **Custom field values** — every non-empty key/value pair from the custom fields you've defined for the Company under Settings → Custom fields.
21. **Website content** — the cached scraped text from the account's website. Capped at **3000 characters**. Wrapped in delimiters that tell the AI to treat it as untrusted evidence (a prompt-injection guardrail).

What it produces

Field	Type	Meaning
`ai_score`	0–10	Numeric fit score.
`score_label`	enum	`excellent` (≥8), `moderate` (5–7), `mismatch` (<5).
`score_reasons`	array	2–4 short, typed justifications — see below.

Score reasons are typed for at-a-glance reading

Each reason carries a type so the UI can show you what's pulling the score up vs. down without you having to read prose:

Type	Icon	What it means	Example
Positive	green check	Concrete ICP match	*"FM rock station — matches ICP industry."*
Negative	red X	Concrete ICP gap	*"Single-location operator — below the size range."*
Neutral	amber warning	Relevant context that doesn't clearly help or hurt	*"Spanish-language content."*

`excellent` scores lead with positives, `mismatch` scores lead with negatives, and when both signals are present the AI includes at least one of each so the trade-off is visible.

Cost and latency

Scoring runs on **Gemini Flash** by default — fast and cheap. Per-account cost is a fraction of a cent (typically <€0.001); a batch of a few hundred accounts finishes in 5–15 minutes. Cost is tracked per account and per campaign, with a per-model breakdown on both surfaces — see Usage and costs.

When scoring fails

The campaign's `scoring_status` flips to `failed` when the background job hits an error (provider quota exhausted, malformed account data the LLM rejects, etc.). Scores from batches that completed before the failure are kept — re-running scoring will pick up where it left off and only score the accounts that don't yet have a score. Today the campaign page doesn't surface the failure prominently, so if a rescore finishes faster than expected and you see fewer scored accounts than you launched with, check `scoring_status` on the campaign.

What scoring does *not* see

- **Conversation history with the account** — past outbound and inbound aren't part of the scoring prompt. (They are part of the AI-continue drafting prompt — different surface.)
- **Activity from other companies in your tenant** — scoring is strictly scoped to the active Company's Product and Account.
- **Paid data sources unless you've integrated them** — LeadHunter's default scoring doesn't subscribe to ZoomInfo, Apollo, etc. If you've enriched accounts with data from those services and stored it in custom fields, *that* data goes in; the integrations themselves don't.

The `notes` **field on the account** *is* sent to the scoring prompt (truncated to 500 characters). Notes are a useful place to drop short context the model should weigh — *"Referred by an existing customer"*, *"Recently raised Series B"*, *"Operates in two countries"* — and they'll influence the score and the reasons. Treat the field like operator-shared AI context, not private scratch space.

Score caching: one per (Product, Account, Goal)

The fit score lives on the **(Product, Account, Goal)** triplet — one score row per campaign goal. That means:

- The same account in three sales campaigns of the same product is scored **once** — adding it to a second sales campaign reuses the existing score (no Gemini cost, no waiting).
- The same account in a sales campaign **and** a press campaign of the same product gets **two** scores — they're rated against different criteria (buyer fit vs editorial fit) so reuse would be wrong. Each goal owns its own score.
- Swap a campaign to point at a different product, and all its accounts will need to be rescored. The dashboard offers a single-click bulk rescore for this case.
- Switch a campaign's **goal** and existing scores stay as they were (they're for the previous goal); rescore to get fresh scores under the new framing.

Per-campaign **overrides** are scoped to a single account and don't touch the underlying (Product, Account, Goal) score, so a manual adjustment in one campaign doesn't bleed into the others. The API for setting an override exists today; the in-app UI for it is on the roadmap, so for now overrides are set programmatically rather than from the scoring page.

Sibling campaigns share scores — the rescore caveat

Because the score is per (product, goal), two campaigns of the same product **at the same goal** literally read from the same score row — different brief, different audience filters, same cached score. When you use the "Give feedback & rescore" flow on Campaign A, the new scores **also become what Campaign B sees** if B targets the same product and goal.

What survives the rescore on those siblings: every approval / rejection you've made, every per-account override you've set, every outreach interaction. Only the AI's score and the reasons it wrote change.

The rescore modal warns you explicitly when sibling campaigns exist, so you can decide whether to proceed. If you need genuinely independent rubrics for two campaigns, either (a) put them on different goals (e.g. one Sales + one Customer Expansion) so each gets its own score row, or (b) clone the Product so the two campaigns each have their own product context — the trade-off is that the Product's example URLs and value proposition must be maintained in two places.

Give feedback & rescore — the iterative loop

The scoring review page (`Score review` in the dashboard) has a **Give feedback & rescore** button next to the "Score remaining" action. Click it whenever you notice the AI is overlooking something across many scores — *"the persona should include a physical retail location"*, *"we're scoring chains too high, prioritise owner-operated stores"*, *"weight community-event capability heavily for this campaign"*. The modal:

1. Pre-fills the textarea with the campaign's current **scoring brief** so you can revise, append, or rewrite freely.
2. Saves whatever you type to `Campaign.scoring_brief` immediately — even if the rescore itself fails, your feedback isn't lost.
3. Re-evaluates **every** account in the campaign under the new brief — both already-scored and unscored — overwriting the cached AI score and reasons in place.
4. Polls until scoring completes, then shows the new distribution.

Approvals, rejections, and overrides are preserved across the rescore — only the AI's verdict changes. If you've already calibrated the campaign by approving/rejecting 10+ accounts, those validations still feed the scoring few-shot, so the brief stacks on top of your prior calibration rather than replacing it.

The brief stays on the campaign indefinitely. Subsequent rescores (manual or automatic) keep using it until you change or clear it.

The feedback loop — your reviews calibrate the model

The biggest lever for improving scoring quality isn't tweaking the ICP — it's marking accounts as **Approved** or **Rejected** in the campaign review queue. Once a (product, goal) pair crosses about **ten** reviewed accounts across all its campaigns, LeadHunter starts feeding your most recent approvals and rejections to the scoring prompt as concrete examples of *"what good looks like here"* and *"what to skip even when the surface details look fine."*

Below that threshold, the calibration few-shot is held back — small samples are noisier than no sample, so the original example URLs on the Product remain the only anchor.

The signal is **goal-scoped within a product**: approvals on a sales campaign of Product X don't leak into a press campaign of the same product, and vice versa. The two are rated against different criteria — mixing them would teach the model the wrong lesson. The same Account approved in two sales campaigns of the same product counts once; the same Account being rescored never gets fed its own past verdict (avoids a self-reinforcing loop).

This is why the first campaign on each new (product, goal) pair is the most important one to be thorough on the review step — every approve/reject you make there is shaping the next several hundred scores for that pair.

Reasons are written in the campaign's language

ICP summary text, score reasons, and outbound message drafts are all produced in the same language. The chain is **Campaign.communication_language → Product.communication_language → your account default → English**. Set the campaign's language to run parallel campaigns of the same product in different markets without duplicating the product.

The Account's own `language` field affects outbound message drafting (the message is written in the account's language) but not scoring — scoring renders reasons in the campaign's language regardless of where the account is located.

What if the account has no website?

Some accounts arrive with a name, a city, and nothing else — phone-only entries, scanned business cards, old CRM exports. Scoring can still run, but with no website content the model is leaning on the structured fields alone (business type, custom fields, location). Scores tend to cluster in the moderate-mismatch range and the reasons are vaguer.

Two ways to improve them:

1. Let **auto-enrichment** discover and scrape the website (the default for newly-created accounts).
2. Trigger **Deep research** from the account detail page (multi-page crawl + contact extraction).

Once content lands, rescoring picks up the new signal.

After scoring: prune the mismatch tail

Once a scoring pass completes, the bottom of the list — the **mismatch** band — is usually not worth your outreach time. The campaign-accounts panel has a one-click **Remove low-fit** action that clears these rows, with safe defaults that preserve anything you've already approved or started outreach on. See Campaign → Pruning the low-fit tail after scoring for what it removes, what it keeps, and the per-campaign scope.

When to rescore

Re-run scoring when:

- **You edit the ICP** — the largest possible scoring shift.
- **You change the example URLs** on the Product — second-largest shift.
- **You enrich accounts with new data** — website discovery completed for accounts that had no URL; custom fields backfilled from an import.
- **You've passed the calibration threshold** — if you've just reviewed your tenth account, the next batch of *fresh* scores benefits from the calibration examples; existing scores don't update until you rescore them.

The same account isn't auto-rescored across campaigns of the same product — the score lives on the (Product, Account) pair and is sticky until you explicitly rescore.

Single-account vs bulk rescore

Two entry points for re-running scoring:

Mode	Where	Use when
Quick-score on one account	The account detail page → **Score** button.	You've just enriched one account and want its score refreshed without re-running the whole campaign.
Bulk-rescore the whole campaign	The campaign page → **Re-score** action.	You edited the ICP, changed example URLs, or just crossed the calibration threshold and want the existing accounts updated.

A rescore **overwrites** the previous score in place — there's no per-account score history kept. If you want to see what the previous score was for a specific account before re-running, screenshot it or take a note; the

new score will replace it.

What about the score distribution?

The campaign detail page shows a **10-bucket histogram** of fit scores after scoring runs (`0-1`, `1-2`, … `9-10`). It's the fastest way to read whether the ICP is well-calibrated. See Campaign → Score distribution.

Read next

- Run your first campaign — the operator-side workflow, including the review-and-calibrate pass.
- Company and Product — why ICP lives on the product and not the campaign, and when to spin up a second product instead of a second ICP.
- Custom fields — define the structured business context the scoring prompt reads.
- Account — what scoring actually reads from, and the operator-only `notes` field it deliberately doesn't.

Custom fields

LeadHunter ships with the obvious fields — name, website, phone, address, status — but every business tracks attributes specific to its market. **Custom fields** let you define those once, attach them to every account in your company, and reference them everywhere you can reference a built-in field: account forms, saved filters, campaign bulk-add, the scoring prompt.

Custom fields are scoped to a **Company** — a bike-shop CRM's `fleet_size` is invisible to a podcast directory in another Company; neither company's schema pollutes the other's data.

Field types

Type	What it stores	Example
`text`	Short single-line string	License number
`textarea`	Long multi-line string	Notes about the location
`number`	Integer or decimal	Number of franchises
`date`	YYYY-MM-DD	Contract renewal date
`boolean`	true / false	Has loyalty program?
`select`	One value from a list	Tier: Gold / Silver / Bronze
`multiselect`	Many values from a list	Languages spoken
`url`	Validated URL	Reviews page

Picking the right type

A quick decision guide for which type to reach for:

You want to capture…	Use
One label from a finite list (Gold / Silver / Bronze)	`select`
Many labels from a finite list (multi-tag)	`multiselect`
A yes/no flag	`boolean`
A count you'll want to filter or sort by	`number`
A specific date	`date`
A clickable URL	`url`
A short opaque string (license number, account ref)	`text`
A long free-form blob	`textarea` (but consider: would `notes` on the Account itself do?)

Picking the most-restrictive type that fits makes the field usable in saved filters with the right operators. `text` is the catch-all, but a `select` would let you filter cleanly while `text` only supports substring matching.

Anatomy of a definition

From **Settings** → **Custom fields** → **Add field**, each definition carries:

Property	What it does
Key	Machine name. Lowercase, snake_case, starts with a letter (regex `^[a-z][a-z0-9_]{0,79}$`). Shows up in filter expressions as `cf.<key>`.
Label	Human-readable name shown on account forms (e.g. *"Fleet size"*).
Type	One of the eight above.
Product scope	Optional. Leave it on *Company-wide* and the field shows on every account; pick a product and the field is grouped under that product on the account screens. Handy when a field only makes sense for one product line.
Options	Required for `select` / `multiselect`. The exact strings counted as valid values.
Deep-link template	`url` fields only — turns the field into a one-click link into an external system. See Deep links into external systems.
Help text	One-line hint shown next to the input on the account form.
Required	When on, account create / edit forms reject submissions that don't fill this field. **Partial updates skip the check**, so a re-import that omits the field doesn't fail.
Order	Display position on account forms. Lower numbers appear first.

The schema is **per-Company**, not global. A field defined under *Bike Shop CRM* doesn't show up under *Radio Stations* — each tenant's fields are isolated. Within a company you can narrow a field further to a single **product** with *Product scope* — the value still lives on the account, the product is just how it's grouped in the UI.

How values land on accounts

Two paths an operator can use:

1. **The account form** — every defined field shows up in the *Custom fields* section of the new-account and edit-account screens, in the order you set. Validates on save against the declared type.
2. **CSV / XLSX import** — map a column to a custom field key in the import wizard's column-mapping step. The **static-values** trick from Import accounts also works here: set a batch-wide custom-field value (e.g. *every row in this file is* `cf.source_list = "Trade show 2026"`) without needing a column for it.

> Auto-enrichment doesn't fill custom fields. The discover-website + scrape + language-detect pipeline only populates the standard fields (`website`, `language`). If you need a custom field populated automatically, fill it on import or on the account form.

Deep links into external systems

A `url` field can be more than a place to paste a link. Give it a **deep-link template** and LeadHunter builds a one-click link into your CRM, billing system, admin panel — anywhere an account also lives — without anyone pasting a URL per row.

You write the template once, with `{placeholders}` LeadHunter fills in per account:

```
https://crm.example.com/contact/{imported_id}
https://your-admin.example.com/client/{value}
```

Three kinds of placeholder:

Placeholder	Filled with
`{value}`	A value the operator types into this field on the account (use this once per template).
`{cf.<key>}`	The value of another custom field on the same account.
Account fields	The account's own data — `{imported_id}`, `{id}`, `{website}`, `{email}`, `{phone}`, `{name}`, `{city}`, `{country}`, `{google_place_id}`.

The standout is **`{imported_id}`**. When you import a list, LeadHunter remembers each row's original ID from the source system. A template like `https://crm.example.com/contact/{imported_id}` then deep-links *every imported account* straight back into that system — no per-account typing at all. That's the quickest way to make LeadHunter a launchpad into the tools your team already uses.

Set it up in **Settings → Custom fields → Add field**:

1. Choose type **URL**.
2. Fill **Deep-link template** with your URL and placeholders. A live preview shows what the link will look like.
3. Optionally set a **Link label** (e.g. *"Open in CRM"*, *"Enacast admin"*) — that's the text shown instead of a raw URL. It falls back to the field's label.
4. Optionally set **Product scope** if the link only applies to one product.

On each account, the field renders as a labelled, clickable link that opens in a new tab. If the template needs a `{value}`, you'll see an input to type it; if it's built entirely from account data, there's nothing to fill — the link just appears. When a referenced piece of data is missing on an account (e.g. a row with no imported ID), no broken link is shown for that account.

A few practical notes:

- **Safe by design** — templates must point at an `http(s)` web address with a fixed host (e.g. `https://crm.example.com/…`); placeholders go in the path or query string, not the host. Values you type are encoded into the link so odd characters can't break it.
- **Plain URL fields still work** — leave the template blank and the field behaves like before: paste a whole URL and it renders as a link.

- **Exports include the built link** — when you download a campaign's accounts as CSV, a deep-link column carries the finished URL, ready to hand to another tool.
- **One manual value per field** — if you need two different external IDs on the same account, make two fields.

> **The other direction shipped: webhooks.** Deep links take *you* into the other system; outbound webhooks send data the other way — notifying your external system the moment an account is created, changes status, or replies. Pulling state *back* (full two-way sync) is the remaining roadmap step.

Custom fields feed scoring

Every value you put in a custom field is part of the prompt the scoring model reads when ranking the account against the Product's ICP. So a thoughtfully-defined custom field is one of the easiest ways to **inject business context into scoring** without re-writing the ICP.

A bike-shop CRM with `cf.fleet_size`, `cf.brands_carried`, and `cf.service_offered` on every row gives the model concrete numerical and categorical signals it can use directly. The same accounts without those custom fields force the model to guess from the website alone.

This is the difference between custom fields and the `notes` field on the Account, which is operator-only and *isn't* read by the scoring prompt. See Account → Notes are operator-only.

Filtering and segmentation

Reference custom fields in saved filters with the `cf.` prefix:

```
match: all
conditions:
  country = Spain
  cf.tier in [Gold, Platinum]
  cf.contract_renewal_date lt 2026-06-30
```

Custom fields support the same operators as built-ins, modulo the field type — `gt` / `lt` apply only to numbers and dates; `is_true` / `is_false` only to booleans; `contains` differs between `text` (substring) and `multiselect` (array membership). See Build a saved filter for the full operator matrix and worked examples.

How custom fields interact with merges

When two accounts merge, custom fields are unioned:

- `text` / `textarea` / `number` / `date` / `url` / `select` / `boolean` — if only one row has a value, that value sticks; if both have values, the **survivor's wins** but the absorbed row's value is recorded in `merge_history` so you can read what was overridden.
- `multiselect` — **union of both arrays.** No loss.

Nothing is silently dropped, even on a conflict.

Adding, renaming, deleting

Action	Effect
Add a field	Instant. Existing accounts immediately gain the field with a null value; new accounts pick it up on the next form load.
Rename label or help text	Instant. Filters and forms keep working — they reference the key, not the label.
Rename a key	**Not safe.** References in saved filters and import mappings break silently. Prefer creating a new field with the new key, backfilling values, then deleting the old one.
Delete a field	Removes it from the schema (stops showing in forms; rejected by the import wizard). The **values stay on existing accounts** in the underlying JSON blob — they just become invisible. Cheap to re-add; expensive to recover deleted history. If you're sure you want the values gone too, ask support for a one-off cleanup.

What's *not* a good fit for custom fields

- **Personal data on individual people** — names, emails, phone numbers of specific contacts. Those belong on the **Contact** model, not on the Account's custom fields.
- **High-cardinality free text** you never plan to filter on. A 10KB description of every account belongs in `notes`, not in a `textarea` custom field — custom fields are meant to be queryable, and JSON-indexed long text doesn't query efficiently.
- **Anything that's the same for every account** — that's a company-level setting, not a field on each row.
- **Anything you'll want to aggregate across the whole database** — define a custom field that filters cleanly (number, date, select) and use saved filters for the aggregation; the saved-filter reach estimator is the closest thing to a count-by query in LeadHunter today.

Worked example 1 — a bike-shop CRM

For a Company running outbound to bike shops, useful custom fields might be:

Field	Type	Why
`fleet_size`	`number`	Target by store size. *"Shops with ≥20 bikes in stock"* is a meaningful audience.
`brands_carried`	`multiselect`, options = the brands you care about	*"Shops that carry Trek but not Specialized"* is a meaningful slice.
`service_offered`	`boolean`	Some products only fit shops with a workshop.
`contract_renewal_date`	`date`	For existing customers — filter for *"renewing in the next 90 days"*.
`loyalty_program_url`	`url`	Useful for partnership campaigns; one click from the row.

A saved filter combining `country = Spain` + `cf.fleet_size gte 20` + `cf.brands_carried contains Trek` + `status = prospect` gives you the audience for next quarter's premium-Trek-dealer campaign.

Worked example 2 — a podcast / radio-station directory

A Company running outbound to media outlets might define:

Field	Type	Why
`format`	`select`, options = `news`, `music`, `talk`, `mixed`	The Product description and ICP are different per format.
`weekly_audience`	`number`	Reach the model can compare directly against the ICP's audience floor.
`genres`	`multiselect`, options = the music genres you care about	*"Rock and electronic, but not classical"* is a real targeting move.
`accepts_sponsorship`	`boolean`	Hard prerequisite — set it after the first conversation.
`frequency_mhz`	`number`	Geographic + technical filter for FM-only campaigns.

The product description on a *"podcast advertising platform for music shows"* product references `cf.genres` and `cf.weekly_audience` directly, so the model has structured signals to score against — not just whatever it can extract from the station's website.

Read next

- Build a saved filter — every saved filter speaks the same `cf.<key>` syntax.
- Import accounts — map CSV columns to custom fields, or set batch-wide static values.
- Account — the row your custom fields hang off.
- ICP and scoring — how custom field values become part of the scoring prompt.

Account value (LTV)

When an Account converts to **customer**, LeadHunter records its **expected lifetime value** alongside the close event. That single number powers the ROI rollups on channels and campaigns — without it, you'd see how much you *spent* but not how much you *captured*.

This page explains the three layers that feed the value, when each one wins, and how to set up sensible defaults so most accounts auto-fill at conversion time.

The three-layer chain

The same Account can have its value set at three levels, with this precedence:

Layer	When it wins	Where to set it
Account	Always when set — sticky.	Account detail page → **Captured value** field.
Campaign	Falls back to the campaign that produced the customer when the Account doesn't have its own value.	Campaign settings → **Account value override**.
Product	Final fallback when neither Account nor Campaign carries a value.	Product detail page → **Default account value**.

Operator-set values are sticky. If you type a value into Account → Captured value, future auto-snapshots never overwrite it. This is the right default for B2B deals where the actual contract value matters more than any rule of thumb.

How the snapshot happens

When you change an account's status to `customer` (manually, via API, via a CSV import that sets the status field), LeadHunter:

1. Checks if the Account already has a captured value set. If yes → done; nothing changes.
2. Looks for the most recent campaign that account participated in, and checks for a campaign-level value override. If found → snapshot that value onto the Account, with source = **Auto: Campaign override**.
3. Otherwise looks for the account's product (via any campaign membership) and uses the product default. If found → snapshot, with source = **Auto: Product default**.
4. If none of the above resolve, the Account is left without a captured value — you can fill it in by hand at any time.

The snapshot is **frozen at conversion time**. Changing the product default next month does not retroactively update historical customer values — the same way you wouldn't restate last quarter's revenue when your price list changes. Each Account remembers what its value was when *it* closed.

Re-affirming a customer doesn't re-snapshot

If you change status to `customer` on an account that's already a customer (a CSV re-import, a data sync that touches the row again), the snapshot is a no-op. Existing values stay put. The status-change audit trail entry is still appended, so you can see when the import touched the row.

The four sources

Every customer Account carries a `captured_value_source` tag explaining how the value got set:

Source	What it means
Manually entered	An operator typed the value (or it came in via API / CSV with the field set). Sticky — won't be overwritten by future auto-snapshots.
Auto: Campaign override	The campaign that produced the customer had a value override set, and LeadHunter snapshotted it.
Auto: Product default	No campaign override; the product's default value was used.
(blank)	No value captured. Either the account is a customer without product/campaign context (pure-inbound), or the product/campaign didn't have defaults configured when the close happened.

The dashboard breakdown shows the **By source** distribution per channel and per campaign, so a glance tells you whether you're mostly auto-snapshotting from defaults or manually entering deal values.

Currencies

Each layer carries its own currency:

- **Product**: `default_account_value_currency` (ISO 4217 — EUR / USD / GBP / …).
- **Campaign**: `account_value_currency` — falls back to the product's currency when blank.
- **Account**: `captured_value_currency` — snapshotted along with the amount.

No FX conversion happens. Rollups group by currency, and the ROI calculation only runs when the channel's / campaign's spend currency matches the captured-value currency for a given account. Mixed currencies surface a `roi_unavailable_reason` explaining which currencies are present.

If your product is priced in USD but you've logged Adwords spend in EUR, the channel's ROI will explicitly tell you the currencies differ and skip the math rather than producing a misleading number.

How value rolls up

On a Channel

Channel detail pages show the **value summary** — total captured value across every customer Account that was attributed to that channel, with currency breakdown and the by-source distribution. Pair it with the cost summary on the same page and you get **ROI = value / spend** in the same primary currency.

On a Campaign

Campaign detail pages carry the same shape — total customer value attributed to that campaign, vs. the campaign's logged expenses (labor, agency, content, event, etc.). The ROI block computes campaign-level return when spend and value share a currency.

Attribution: per-channel vs. per-campaign

- **A channel** has at most one Account attached via the `acquisition_channel` FK, so a customer's value counts once per channel.
- **A campaign** can share an Account with another campaign (the same lead in two outreach efforts). That customer's value will appear in both campaigns' rollups — each campaign gets credit for the full deal. Useful for ROI math at the campaign level; not the right shape for a company-wide rollup where you'd want first-touch / last-touch / linear attribution. Multi-touch attribution is on the roadmap.

When to set values vs. leave defaults

A few practical patterns:

SMB self-serve product with predictable pricing. Set `Product.default_account_value` to your average customer LTV. Every customer auto-snapshots at that number. Operators only override when a specific deal is unusually large or small.

Mixed self-serve + enterprise. Set the product default for SMB. Create an enterprise campaign and set `Campaign.account_value_override` to the enterprise tier's expected LTV. Customers closed via the enterprise campaign auto-snapshot at the higher number; everything else falls back to the SMB default.

Long-cycle B2B with hand-negotiated deals. Leave the defaults blank (or set conservative placeholders) and have the operator enter the actual deal value at close. The auto-snapshot does nothing because the value is already manually set. You get accurate per-deal numbers at the cost of operator effort.

Pure-inbound (no campaign context). Pure-inbound customers (Adwords lead that converted without ever being added to an outbound campaign) won't have a product to fall back to — there's no campaign membership to read the product from. Operators need to enter the value at close. The status change still triggers the snapshot, but it resolves to "no value" and silently skips.

What the field does NOT track

- **Churn or contract end** — `captured_value` is a single number at conversion. If a customer churns or expands, the value doesn't update automatically. Operators can edit it any time; a structured value-history journal is on the roadmap.
- **MRR vs. ACV vs. contracted value** — the field is just a number. Decide which definition you're using and stick with it across all customers of a product so the rollups are comparable.
- **Discounting** — if the SMB self-serve product is on a discount this quarter, the product default doesn't know. Update the default when you change prices, or override at the campaign level for the discount-period campaign.

What's coming

- **Value history journal** — when a customer churns or expands, record the change with a timestamp so the rollup can reflect net captured value over time, not just the conversion snapshot.
- **Per-acquisition cost** delivered by ad-platform CPA data — pairs with channel webhook ingestion. Today's ROI rolls up channel-level spend; per-account cost would let you see ROI per deal.
- **Cohort ROI** — *"of customers acquired in May, what's the cumulative captured value vs. May spend over the following 6 months?"* The current rollup is point-in-time; cohort ROI needs a join on the conversion date.
- **Multi-touch attribution** — split a customer's value across the channels and campaigns that touched them, under configurable models (first-touch / last-touch / linear / time-decay).
- **Currency conversion** — daily FX snapshots so cross-currency rollups can produce a single ROI number instead of refusing the math.

Research

A **Research** record is LeadHunter's audit trail for *how an account got into your database*. Every Google Maps sweep, CSV import, URL lookup, AI enrichment pass, and operator-logged manual research session has its own row — so months later you can answer *"why do we have this account, who put it there, what method was used, and what did it cost?"*

Research is the audit-trail equivalent of `status_history` on the account itself — but for discovery, not lifecycle.

Where to find it

The dashboard's **Research** page (sidebar → Research) is the home for the log. It shows:

- **Stats** — total / in progress / completed / failed.
- **A filterable list** of every research record in the active Company, newest first, with status, method, result counts, and the user who triggered it.
- A **New research** action for logging research you did yourself — useful when discovery happened outside LeadHunter (a directory you bought, a trade-show booth scan, a vendor list) and you want the record for accounting and team visibility.

What kicks off a research record

Method	Trigger	UI surface
Google Maps search	Text query like *"bike shops in Berlin"*.	Accounts → Discover
Google Maps URL	Paste a `maps.google.com/...` URL.	Accounts → Lookup
Website / domain	Paste `acme.com` or a full URL — LeadHunter scrapes it.	Accounts → Lookup
CSV / XLSX import	Upload a spreadsheet through the import wizard. One research record per file.	Accounts → Import
AI deep research	Optional follow-up pass that fills missing fields and writes salesperson notes.	Accounts → Lookup, `Deep` mode
Manual research log	You did the discovery yourself; you're recording it.	Research → New research

The `method` field on the record captures which path produced it: `manual`, `google_maps_api`, `tavily_research`, `apollo_enrichment`, `web_scraping`, `csv_import`, `api_integration`, `other`.

What a research record carries

Bucket	Fields
Identity	`name`, `description`, `method`, `status`
Execution	`started_at`, `completed_at`, `duration_seconds`, `executed_by`
Tooling *(automated runs only)*	`script_name`, `script_version`, `api_key_used` (the env var name, not the key value)
Search parameters	`countries`, `cities`, `search_terms`, `search_radius_meters`, `instructions`, `configuration`
Results	`total_items_found`, `new_items_added`, `items_updated`, `duplicates_found`, `errors_count`
Cost	`api_calls_made`, `estimated_cost` (USD)
Provenance	`source_files`, `notes`, `error_log`, `created_by`, `created_at`, `updated_at`

Search-parameter fields (`countries`, `cities`, `search_terms`) make it easy to **re-run a discovery** later: a Google Maps sweep done six months ago records exactly what was searched, so a fresh sweep with the same terms is one click away to catch new businesses.

Lifecycle

A research record moves through five states:

1. **`planned`** — created but not started yet. Multi-step wizards (CSV import column-mapping, complex Maps queries) hold the research here while the operator configures it.
2. **`in_progress`** — the discovery / scrape / enrichment is actively running. Long imports stay here for minutes.
3. **`completed`** — finished successfully. Result counts and cost are populated.
4. **`partial`** — finished with errors but produced some results. The `error_log` records what went wrong; the results that came through *are* on accounts.
5. **`failed`** — couldn't produce useful results (Google Maps quota exhausted before the first hit, the CSV was malformed, the website refused to scrape). The error is recorded so you can read it and decide whether to retry.

Forward-only: a research record doesn't go back to `planned` once it's started running.

How research connects to accounts

When research completes, every account it produced has gone through the same five-level dedupe stack. Each result is either:

- **Created fresh** — the dedupe stack didn't find a match.
- **Merged into an existing account** — the dedupe stack matched at confidence 85+ and absorbed the new data into the survivor.
- **Surfaced as a fuzzy candidate** — match at the 5th level (fuzzy name within the same city, ≥85% similarity); awaiting human review under Accounts → Duplicates.

The research record's counts reflect that split:

Field	What it counts
`total_items_found`	Everything the discovery produced (pre-dedupe).
`new_items_added`	Rows where the dedupe stack didn't match — new accounts created.
`items_updated`	Rows that were merged into existing accounts at levels 1–4 (auto-merge).
`duplicates_found`	Rows that surfaced as fuzzy candidates awaiting review.
`errors_count`	Items that hit a per-row error (malformed input, scrape failed, …) — see `error_log`.

So a quick scan of *"discovered 142 places, 87 new, 51 merged into existing, 4 awaiting duplicate review, 0 errors"* tells you what to look at next.

Costs and quotas

External calls cost money — Google Maps charges per place lookup, Tavily charges per search, every LLM call burns tokens. Each cost is auto-attributed to the research record that triggered it, and the **Tasks page** (sidebar → Tasks) shows a per-row **Cost** column with the USD total and total tokens used:

```
$0.0234 · 12.4k tok
```

Hover the cell for the per-model breakdown:

```
Total: $0.0234 · 18 calls
Tokens: 11.2k in / 1.2k out

google-gla:gemini-3.1-flash-lite: $0.0180 · 12 calls · 11.2k tok
google_maps: $0.0054 · 6 calls
```

Use the column to spot a run that cost more than it should have (a website that needed the LLM fallback for every account, an enrichment job that scraped a few thousand pages), and the model breakdown to see *which* part of the work was expensive — Google Maps Places lookups, the cheap Gemini lite model, or a costlier model swapped in for a specific run.

For the cross-research breakdown (provider totals, daily trends, per-research-record-comparison views), see Usage and costs — the polished Usage page that surfaces all of this in one screen is on the roadmap.

Manually-entered campaign expenses (Adwords, agency fees, labor hours) **don't** flow into research records — those attach to campaigns instead, since they're paying for a specific outbound effort, not for one batch of discovery. See Track campaign costs and CAC for that side.

When to look at the research log

- **A weird account showed up in a campaign** — open the account, check the `source` field, and trace back to the research record. Useful when you don't remember which import dropped it in.

- **You ran a Google Maps sweep months ago** and want to re-run the same query to catch new businesses. The `search_terms`, `countries`, and `cities` fields on the old research record carry exactly what you searched.
- **You're auditing AI cost spend** and want per-discovery attribution — sort by `estimated_cost` desc.
- **An import partially failed** (`status='partial'`). Open the record, read `error_log`, decide whether the errors are retryable.

Read next

- Import accounts — the three discovery / enrichment paths in detail.
- Account — the row that research produces.
- Merge duplicates — what happens to fuzzy candidates a research record surfaces.
- Usage and costs — the cost surface research records contribute to.

Messaging

LeadHunter is a **communication log**, not a sending engine. You talk to the account through the channel you already use — Instagram, LinkedIn, X, email, WhatsApp, phone, SMS — then paste the exchange into LeadHunter so the AI, the scoring, and the team-wide conversation history all have full context.

This is intentional. Sending is hard to do well across every channel an outbound team actually uses, and LeadHunter would compete badly with the dedicated tool for each. Keeping LeadHunter out of the sending path means it's compatible with every channel you'll ever care about, including the ones that don't exist yet.

The conversation thread

Each **campaign-account** (the join row between a campaign and an account — see Campaign → Campaign accounts) has its own conversation thread. Open any account in a campaign and you see the per-campaign timeline; open the **account detail page** instead and you see conversations aggregated **across every campaign that account has been in**, so the history follows the organisation even when the campaign changes.

The campaign-wide **outreach inbox** is a third view: open a campaign and switch to the Outreach tab to see every conversation across every account in that campaign, sorted by what needs your attention (recent inbound first, then drafts you haven't sent yet).

Opening a conversation updates the page URL, so you can **refresh without losing your place** and **share a link to a specific conversation** with a teammate — they'll land straight on that account's thread.

Inbox filters

The **search box** matches across the account name, contact name and email — and also the **internal note** and **AI draft directions** on each row. So if you stashed an external reference like an order number or CRM id in a contact's note, searching that number jumps straight to them.

The four tabs (**To contact**, **Awaiting reply**, **Responded**, **All**) split the inbox by where each account sits in the funnel. The **Filters** panel below the search box narrows further — it opens by default with **Approved** + **Excellent** pre-selected so you land on the "work this now" queue every time you load the page; clear them to widen the view:

- **Review status** — only accounts you've approved, only ones still pending review, only rejected ones. Useful for working through the approved queue end-to-end before touching anything questionable.
- **Match label** — keep only `excellent`, `moderate`, or `mismatch` AI matches.
- **Score range** — minimum and maximum on the AI score (0.0–10.0). Set `min` to `7` and ignore everything else until you've worked the best fits.
- **Channel** — show only accounts you've already touched on Instagram, LinkedIn, email, WhatsApp, phone, SMS, or X. The channel filter looks at every message in the thread, not just the latest one — useful when you've drifted between channels with the same account.

Tab counts stay constant when filters are on, so the badges always tell you the true campaign size. Only the visible list (and the `count` at the bottom) reflects the filters. Pick any combination — they all `AND` together. Hit **Clear all** to start over.

Internal notes per campaign-account

Each row in the inbox carries a free-text **Internal note** that only you and your team see. It's per-campaign-account, not per-account — the same lead in two campaigns (e.g. a sales campaign and a press campaign) has two independent notes, because the prep context usually differs.

Typical uses:

- *"demo live since Mon — wait a few days before pinging"* (you built a custom demo and want it to bake before reaching out)
- *"warm intro coming from Ana next Thu — don't outbound until then"*
- *"called once, no answer; try after 4pm Madrid time"*
- *"prefers LinkedIn over email"*

When set, the note shows as a sticky-note snippet on the row preview (left sidebar) so you can scan the queue and spot which leads have context attached. Click the row and the editor lives in the conversation pane (just below the **AI draft directions** box); edits save automatically when you click away from the textarea. Saving the note doesn't move the account through the outreach funnel — it's purely a reminder field. **The note is never sent to the AI** — if you want the drafter to act on something, put it in the AI draft directions instead (see Goal-aware drafting).

If you need a note that follows the lead across every campaign (e.g. "this is Ana's old boss"), use the account-level notes on the account detail page instead. The two fields are independent.

What a message carries

Field	What it holds
`direction`	`outbound` (you sent) or `inbound` (they sent).
`channel`	Which surface the message lived on. Eight choices: `email`, `linkedin`, `instagram`, `twitter_x`, `whatsapp`, `phone_call`, `sms`, `other`. The per-message channel is what the cohort funnel attributes against later, so picking the right one matters.
`generation_mode`	How the text was produced — one of five (see below).
`status`	`draft` (scratch — doesn't move the funnel) or `sent` (logged as out / in).
`original_text`	**Authoritative** version: what the lead actually sent or received, in their language.
`original_language`	ISO 639-1 code of `original_text`.
`translated_text`	Optional reader-friendly view of the same content in another language.
`translated_to_language`	ISO code of `translated_text`.
`sent_at`	When the message was sent. For modes other than `Log sent`, defaults to "now". For `Log sent`, you set it explicitly to when it really went out.
`created_by`	The user who logged the message.
Audit	`created_at`, `updated_at`.

There's no separate "subject" or "thread id" — threading is implicit (every message attached to the same campaign-account belongs to the same thread).

The five drafting modes

Mode	Direction	Use when
AI draft	outbound	First outbound message in this thread. AI writes **three drafts, each a different approach**.
AI continue	outbound	Continuing a thread that already has history. AI reads the prior messages and proposes **three** next moves.
Type & translate	outbound	You wrote the message yourself in your language. LeadHunter shows the translation into the account's language and sends the translation.
Log sent	outbound	You already sent it elsewhere (your own email client, on LinkedIn directly, …). Paste the text to record it.
Inbound paste	inbound	The account replied. Paste their reply; LeadHunter shows you the translation into your reading language.

Three approaches, pick one

When you hit **Draft with AI**, you don't get one take — you get **three**, each with a short label (*Direct & concise*, *Warm & personal*, *Outcome-led*, …) and a one-line "why". The angles genuinely differ — a punchy one-liner, a relationship-first opener, a value/outcome pitch — so you can choose the register that fits the

account instead of nudging a single draft toward what you wanted. Each one shows its approach, the message, and the reasoning.

Two buttons on every draft:

- **Copy** — puts that exact text on your clipboard so you can paste it into Instagram, LinkedIn, your email client, or wherever the conversation actually happens. It flips to *Copied* for a moment so you know it took. Since LeadHunter logs rather than sends, copy-then-paste is the normal last step.
- **Use this** — loads that draft into the editable box below so you can tweak it and **Log message** to record it here. The first approach is loaded automatically, so you can edit-and-log immediately or switch to another with one click.

Hit **Redraft** for three fresh approaches if none land. The Copy button also lives under the editable box, so whichever draft you settle on is one click from your clipboard.

How modes populate the two text fields

Knowing which mode wrote which field helps when re-reading old threads:

Mode	`original_text`	`translated_text`
AI draft / AI continue / Log sent	The sent message (account's language)	(empty)
Type & translate	The translated/sent message (account's language)	Your original draft (your language)
Inbound paste	The account's reply (their language)	Translation into your reading language

The asymmetry on **Type & translate** is the one users miss most: `original_text` is the message that *actually went out*, not your scratchpad draft. If you go back six months and read a translated outbound, the version that left your hands is the canonical one.

AI drafting and translation cost

Every **AI draft**, **AI continue**, **Type & translate**, and **Inbound paste** message triggers an LLM call (translation in/out, prior-thread summarisation when there's history). The cost is small per message — a fraction of a cent on Gemini Flash by default — but attributes per (Campaign, Account) under API costs. **Log sent** doesn't call the LLM (you provide the final text directly), so it's free.

If translation fails (rare; usually a transient provider error), the message is still saved with `original_text` only — `translated_text` stays empty and you can retry from the message panel.

Goal-aware drafting

The drafter writes a different *kind* of message depending on the campaign's goal. Same product, same account — but a sales campaign opens with a cold pitch, a press campaign opens with a newsworthy angle, a recruiting campaign asks the candidate for 15 minutes about a specific role, a partnership pitch proposes a collab shape, an investor approach leads with a one-line traction signal. The 0–10 scoring scale and the typed reasons stay consistent across goals; what shifts is the message's voice and call-to-action.

Several things compose into the draft prompt, in order:

1. **Message intent** — derived from `Campaign.goal`. Reads as *"write a cold sales opener — introduce the product, name one specific thing about the lead, end with a soft ask for a short call"* for sales, *"a short press pitch — newsworthy angle first, why-now hook, no sales language"* for press, and so on for the eleven goals.
2. **Target persona + product context** — the (Product, Goal) ICP is rendered into the prompt so the model has the persona definition the campaign rates against.
3. **Account context + thread history** — the account itself, plus the `role_in_campaign` override if set (see below).
4. **Your directions** — the campaign brief and per-account directions you set yourself (see below). These come last so the model weighs them most heavily.

If the campaign's goal changes mid-flight, the next draft you generate uses the new intent immediately. Existing already-drafted messages don't auto-rewrite.

Steering the draft: campaign brief + per-account directions

The goal sets the *kind* of message; you set the *specifics*. Two free-text fields let you hand the drafter directions in your own words — tone, length, angle, what to mention, what to avoid. Both are optional, blank by default, and take effect on the very next draft (nothing to regenerate).

- **AI outreach brief** — campaign-wide. Open the **AI outreach brief** panel at the top of the campaign's outreach inbox (left sidebar) and type directions that apply to *every* draft in the campaign: *"keep it under 60 words, friendly, no emojis; lead with the integration angle"*, *"always reference that we're a local Barcelona company"*, *"formal register, no first names"*. A small dot on the panel marks a campaign that has a brief set. It saves when you click away.
- **AI draft directions** (per account) — open any conversation and you'll see an **AI draft directions** box at the top, right above the internal note. Type directions that apply to *this one account*: *"they demoed last week — reference it"*, *"warm intro from Ana is pending, mention it lightly"*, *"do not bring up pricing yet"*. These are layered **after** the campaign brief and win on any conflict, because they're the most specific. Accounts with directions set show an ✨ *directions* marker on their inbox row.

The two stack: the campaign brief sets the house style, the per-account directions handle the one-off context. **Neither is the same as the internal note** — the note is private prep that the AI never reads; these two fields are *written for* the AI.

If the same account is in two campaigns, its directions are per-campaign (they live on the campaign-account, like the internal note), so a sales angle and a press angle stay independent.

Per-campaign role override

Sometimes the same Account plays different roles in different campaigns. A SaaS company is your customer (sales), but their CTO is also a regular podcast guest (press), and the head of engineering recently spoke at your conference (event). Each campaign needs the message to read with the right voice.

Setting **`role_in_campaign`** on a CampaignAccount tells the drafter: *"for this campaign, treat this account as `<role>` regardless of whatever primary tags it carries elsewhere."* The drafter respects the lens — a press

campaign with `role_in_campaign='press'` produces a press pitch even when the account's primary `relationship_types` say `client`. Blank means *"use the account's tags as-is."*

Setting it is a click in the campaign's accounts panel: each row's chip cluster shows a faint dashed `+ role` affordance when no override is set, or a colored chip with the override label when one is set. Clicking opens a popover with all 13 relationship-type options — pick one to apply, or pick *"Use the account's tags (no override)"* to clear.

This is the right escape hatch when one account legitimately wears multiple hats and you don't want to blur them across campaigns.

Tracked links

LeadHunter logs messages, it doesn't send them — so between "sent" and "responded" the funnel used to be blind. **Tracked links** close that gap: a short link tagged with the campaign and contact that records every click.

In the conversation pane, open the **Tracked links** panel:

1. Paste the destination (the field pre-fills with the account's website) and an optional label, then click **Create**.
2. Copy the short link and paste it into your message wherever you'd have pasted the long URL.
3. When the contact clicks it, they're redirected instantly to the destination — and the click lands on the conversation: the panel shows the click count and the last-click time, and the inbox row gets a **clicks** chip so you can spot warm contacts at a glance.

Good to know:

- **Bot clicks don't count.** Link previews (WhatsApp, Slack, LinkedIn unfurlers) and crawlers hit your links constantly; LeadHunter logs them but keeps them out of the click count, so "2 clicks" means a human clicked twice.
- **No personal data is stored** about the clicker — no IP addresses, just the time, the browser family, and where the click came from.
- **Destinations can't be edited** after creation. A link you already sent must keep meaning what it meant when you sent it; create a new link instead.
- Deleting a link makes it stop redirecting — and deletes its click history.

Clicks also feed the **dashboard funnel**: a *Clicked* stage now sits between *Initiated* and *Responded* (with its own cohort click-through-rate chart and a column on the per-product / per-campaign stats tables), so you can see how much of your outreach gets attention even before anyone replies.

DNC and per-purpose opt-outs

Language resolution

Outbound messages render in the **account's language** if you've set one. Otherwise LeadHunter walks a fallback chain until it finds a setting:

```
Account.language
   → Campaign.communication_language     (per-campaign override)
   → Product.communication_language       (product default)
   → User.communication_language          (your personal default)
   → English
```

The campaign-level override means you can run one campaign of the same product in Spanish (for the Spain market) and another in English (for the UK) without duplicating the product. See Campaign → Language.

Your **reading language** — what LeadHunter translates inbound text *into* — is your User-level communication language. It's independent of the UI language, so you can run the dashboard in English while writing to and reading from accounts in Spanish.

Override the languages for one conversation

The resolved direction shows as an editable **From** → **To** at the top of the composer. The two fields are pre-filled from the chain above, but you can change either one — type any language or pick a common one from the dropdown, and use the swap button to flip the direction. The override drives both the AI drafts and the type-&-translate output for **this conversation only**; it isn't saved. Switch to another conversation (or refresh) and you're back to the resolved languages, and a **reset** link restores them at any time. Use it for a one-off — say this lead's account language is detected as Spanish but you happen to know they prefer English. To make the change stick for every future message, edit the account's language (or the campaign / product / your own default) instead.

Status auto-promotion

Logging a message **as sent** advances the pipeline automatically:

- **Outbound sent** → campaign-account `outreach_status` becomes `sent` (or `bounced` / `responded`), and the dashboard funnel counts the account as **initiated**.
- **First outbound on a `prospect` account** → account-level `status` advances to `contacted` with `source: campaign_outreach` in the audit trail.
- **Inbound paste** → campaign-account `outreach_status` becomes `responded`; the dashboard funnel counts the account as a **response**, cohort-attributed back to the original initiation date.

All forward-only. Already-customer accounts don't regress when you log a follow-up; a second outbound doesn't re-fire the prospect → contacted promotion.

Drafts (`status='draft'`) don't fire any of this. They're scratch space — useful while you iterate, but they don't move the funnel until you flip them to `sent`.

When the lead reaches out first

The auto-promotion model assumes you initiate, then they reply — the common outbound case. For **inbound-acquired** accounts (Adwords form-fill, Instagram DM, referral), the account is created with an `acquisition_channel` and starts at `status: contacted` automatically before any message is logged. See Track inbound leads.

When the *first* message on such an account is your reply rather than their initial reach-out, log their initial message as an **Inbound paste** first (with the right `sent_at`), then your reply as **AI continue** or **Type & translate**. The funnel attribution works correctly either way — initiation is your first outbound, response is their first inbound; their *original* outreach pre-dating LeadHunter doesn't count as a response.

Two layers of opt-out apply to outbound message logging:

- **Blanket DNC** — outbound to a `status='do_not_contact'` account is blocked on every campaign goal. The UI shows a clear error and points at the account's status history so the operator can see when and why the flag was set. No override.
- **Per-purpose DNC** — outbound is blocked when the campaign's `goal` is in the account's `do_not_contact_purposes` list. An account that opted out of `sales` can still receive `research` interview asks or `press` pitches; an account that opted out of `press` can still receive sales messages. The list is the GDPR / CASL-shaped consent surface, distinct from the blanket flag. Recorded via `POST /api/accounts/{id}/record-dnc/` (or whichever UI control surfaces it). See Account → Per-purpose opt-outs.

Other relationship-type guardrails (`competitor`, `personal_network`, soft blocks like `supplier` / `investor` / `press` / `analyst` / `candidate`) only apply to campaign **bulk-add** — not to logging individual messages on accounts already in a campaign. The rationale: if it's in the campaign, it cleared the bulk-add filters.

Read next

- Log a conversation — the operator-side walkthrough with a worked outbound → reply → continue example.
- Campaign — the language-resolution chain in full, plus the outreach guardrails.
- Account — the row whose lifecycle status the message log moves.

Import accounts

There are three ways to get accounts into LeadHunter. All three run the same dedupe checks against your existing database, so you can't accidentally create a second row for an account you already have — and the merge keeps every unique field from every duplicate rather than dropping data on the floor.

Pick the path that matches the source:

Source	Use	Where
One-off entry	Quick add by name, URL, or search query	**Accounts → Lookup**
Existing list (CSV / XLSX)	Bring over a sheet you already maintain	**Accounts → Import**
New market discovery	Sweep a geo/category in one pass	**Accounts → Discover (Google Maps)**

1. Single-account lookup

Best for ad-hoc additions — *"there's this one bike shop I want to track, can you find it for me?"*

From **Accounts → Lookup**, paste any of:

- A website URL (`https://acme.com`) — scraped + enriched.
- A bare domain (`acme.com`) — LeadHunter prepends `https://`.
- A Google Maps URL — pulls address, phone, hours, rating, place id.
- A plain-text query (*"bike shops in Berlin"*) — Google Maps text search; the top match is used. Requires Google Maps to be configured.

Three research modes

Each lookup runs in one of three modes — the dropdown next to the URL field:

Mode	What runs	When to pick
None	Plain scrape; no LLM.	You're importing a vetted URL and don't want to spend Gemini quota.
Quick (default)	AI extracts business name, sector, language, key fields you didn't provide.	Most of the time.
Deep	Quick + multi-page crawl (about / team / staff / contact pages) + decision-maker contact extraction.	When you want named contacts pre-filled, not just the company.

Deep mode is heavier (more page fetches, more tokens) but produces an account that's already populated with one or two contacts with names, titles, and (when discoverable) emails. Use it for high-value accounts where you'd otherwise hand-research the buying group.

2. CSV / XLSX bulk import

Best for bringing over a list you already maintain — your CRM export, a trade-show booth scan, an industry directory you bought, etc. **Accounts → Import** runs a three-step wizard.

Step 1: Upload

Drop a `.csv`, `.xlsx`, or `.xls` file. LeadHunter parses the headers and shows you a sample of the first few rows so you can confirm parsing worked (Excel files with ragged rows — rows that have fewer cells than the header — are handled correctly).

Step 2: Map columns

This is the step where you tell LeadHunter what each column means. The AI proposes a mapping based on the column names and the sample rows — e.g. `org_name → name`, `tel → phone`, `country_iso → country`. Review and adjust.

Standard target fields:

`name`, `address`, `city`, `state_province_region`, `postal_code`, `country`, `phone`, `email`, `website`, `business_type`, `specialization`, `rating`, `language`, `status`, `notes`, `acquisition_channel`, `latitude`, `longitude`, `google_place_id`, `imported_id`.

Aliases — `state`, `province`, and `region` all resolve to `state_province_region`, so you don't have to rename your column.

Skip — set the target to `skip` for columns you don't want imported. Skipped columns can still be preserved as JSON (see *Save extras* below).

`imported_id` — map your source's stable row id here. If your file has a column like `Station UUID`, `CRM Record ID`, `Customer ID`, or any other identifier that uniquely identifies a row in the source dataset, map it to `imported_id`. Re-importing the same file (or a new export from the same source) will then resolve every match via that id at the highest confidence — even when name + city don't agree, or are missing entirely. Without an `imported_id`, dedupe falls back to name + city + phone + website + fuzzy name, which can miss matches when the file is sparse (radio-station directories, lead-magnet form dumps, etc.). Common header names that the AI auto-suggests: `ID`, `External ID`, `OriginalID`, `Imported ID`.

`name` **is mandatory** — the import refuses to start without at least one column mapped to it.

Already imported this file? You'll see a warning.

If the bytes of the file you just uploaded exactly match a previous import in this company, the mapping step shows a yellow banner at the top: *"You've already imported this exact file"* — with the prior import date, who ran it, how many rows landed, and how many of those accounts still exist. Cancel if you uploaded the wrong file; proceed if the warning is just informational (you re-uploaded on purpose). The dedupe stack will skip the duplicates regardless.

Step 3: Run (and the options worth knowing)

A few options control how the import behaves:

- **Static values** — constants applied to every row. Useful when the file doesn't carry a column for it but the value is the same for the whole batch. Examples: `country=Spain` for a Spanish trade-show export,

`language=Catalan` for a regional directory, `acquisition_channel=event` for a booth scan. Anything you set here overrides values from the file.

- **Save unmapped columns as `extra_data`** — when on, every column you mapped to `skip` (or didn't map at all) is preserved as a JSON dict on each account. Use this when the source has fields LeadHunter doesn't model but you don't want to lose them.
- **Enrich accounts after import** — when on, every newly-created account is queued for **website discovery + scrape** (logo, phone, email, address, the per-platform social URLs, year founded, employee-count bucket, business language, and the business name when the row arrived without one) once the rows land. The enrich runs as a separate background job that shows on the **Tasks** page. Off by default — it costs Gemini calls and time. Tick it for a small trusted list; leave it off for a 50k-row directory unless you actually want every row enriched. You can always enrich later from **Accounts → Enrich** (see After the import: enrichment).
- **Source label** — a free-text tag stamped on each imported row's `source` field. **Defaults to the filename** (lower-case, with separators tidied — `radio_stations_radiobrowser.xlsx` becomes `radio stations radiobrowser`). Edit it to whatever provenance marker reads best to you and your teammates: `"LinkedIn export 2026-Q2"`, `"Trade show 2026"`, `"Stripe customers"`, etc. Beyond the label, every imported row also carries the file's **MD5 fingerprint** invisibly, so you can later answer "show me every account that came from this specific upload" exactly even if multiple imports share the same label.
- **Dry-run** — runs every dedupe check and shows you what *would* happen (created / merged / fuzzy candidates) without writing. Recommended for any import larger than a handful of rows.

Big files run in the background

Files above about **2,000 rows** are handed to a background worker so the browser doesn't sit on the request. You'll see a **live progress bar** with rows-processed / total, animating as the import advances:

- You can **close the dialog** at any time — the import keeps running on the server, and the accounts show up on **Accounts** as soon as it finishes.
- The worker imports rows in **chunks of 2,500**, each in its own transaction. If a single row in chunk N gets rejected by the database (e.g. a column type that doesn't fit), only that chunk rolls back — the chunks that already committed stay in place. You can fix the offending rows in the source file and re-import just them, instead of losing 50,000 good rows because of 100 bad ones.
- The job also shows on the **Tasks** page as a row of method **CSV Import**. Click it for full details (timing, totals, error log).

Smaller files (and any dry-run) finish synchronously — same modal, no progress step. Smaller imports also run as a single transaction (no chunking), since one rejected row in a 200-row file is small enough to retry whole.

What the result panel shows

When the real import finishes, the summary tells you: rows created, rows merged into existing accounts, fuzzy-match candidates that need your review, and rows skipped (with reasons). Fuzzy candidates land in **Accounts → Duplicates** for confirmation.

For chunked imports, a small footnote lists how the file split — e.g. *"Imported in 21 chunks of 2,500 rows. 19 committed, 2 rolled back."* If everything went well, all chunks committed and the import is **complete**. If some

chunks fail (bad rows in only part of the file), the import is **partially complete** — the good chunks landed, the bad ones didn't, and the result panel shows which row numbers tripped the database so you can fix them.

If LeadHunter had to **truncate** any over-long values to fit the column limits (`Name` is capped at 255 characters, `Website` at 500, `Language` at 50, etc.), an amber summary box lists the count per field — e.g. `language: 126 · name: 17 · website: 2`. The rows still imported; you just lost the tail of those long strings. Fix the source file if the truncated values matter.

If **every** chunk rolled back (the source file is broken end-to-end), the panel turns red, says "Import failed — nothing saved", and lists the underlying database errors. Nothing lands; fix the source and retry.

Undo this import

Picked the wrong file by accident? On the result panel, the **Undo this import** button deletes every account that landed from this file in one click — plus any campaign memberships, conversation messages, and scores that got attached to those accounts in the meantime. The import history record stays so you have an audit trail, but the rows are gone.

It uses the file's MD5 fingerprint (stamped invisibly on every imported row) to find exactly what to delete, so it never reaches into accounts that came from a different upload or were entered manually.

The undo is **destructive and cannot be reversed**. If you imported a 50k-row CRM export, ran outreach campaigns against some of those rows, and then click Undo, those campaign rows and the messages you sent are gone too. Most operators want Undo right after the import lands ("oops, wrong file"); after that point, fixing duplicates one-by-one in **Accounts** → **Duplicates** is usually safer than wholesale revert.

3. Google Maps discovery

Best for entering a new market — *"who are all the bike shops in Berlin?"* Run a query from **Accounts** → **Discover** and LeadHunter pages through Google Maps results, fetches the place details for each, and adds them as accounts.

Each result runs through the same dedupe stack, so re-running the same query a month later **doesn't double-count** — it merges fresh place details (new phone, updated hours, current rating) into the row you already have. The `google_place_id` constraint makes the match exact.

Requires Google Maps API to be configured on your environment; ad-hoc lookups by Maps URL or domain don't depend on it.

After the import: enrichment

LeadHunter can run an **enrichment pass** on every freshly-imported account that:

1. Tries to **discover the website** via Gemini grounding (and a domain-guessing fallback when the model has nothing).
2. **Scrapes** the discovered URL and caches the text content on the account.
3. **Extracts structured fields** from the page: logo, phone, email, address, the per-platform social profile URLs (LinkedIn / Instagram / X / Facebook / YouTube / TikTok), `year_founded`, the coarse

`employee_count_estimate` bucket, primary business language, and the business name when the row arrived without one.

Three ways this gets triggered:

- **Single-account lookup** (Quick / Deep modes) runs the enrich inline — it's part of the lookup itself.
- **Bulk import (CSV / XLSX)** — tick **Enrich accounts after import** during step 3. The enrich runs as a separate background job once the rows land. Off by default so you can decide; a 50k-row directory imported without the box ticked stays cheap and fast, and you can enrich a subset later from the **Enrich** button on the Accounts page.
- **Google Maps discovery** rows arrive already-populated from the Maps result (website + phone + address), so the enrich is usually a no-op and is skipped.

Enrich already-imported accounts

The **Enrich** button on the Accounts page (next to **Import**, **Export**, **Find Duplicates**) opens a small dialog with three modes:

- **Only non-enriched accounts** *(default)* — skips every account that already has scraped website content. Recommended for project-wide enrichment.
- **First 50 (test run)** — caps the run at the 50 oldest non-enriched accounts. A cheap smoke test before kicking off a multi-hour pass on a large project.
- **Currently filtered accounts** — applies the page's current filter set (search, status, country, business type, advanced filters …) and enriches only those rows. Useful when you want to enrich a specific country slice or every "prospect" before a campaign launch.

Whichever mode you pick, the run lands on the **Tasks** page with a real **live progress percentage** — *not* the placeholder it used to park at. Large jobs (>200 accounts) are sliced into chunks server-side so the percentage advances steadily instead of jumping from 0% to 100% at the end, and so a redeploy mid-enrich doesn't lose the work that already landed (it picks up where it left off automatically — see below).

How "already enriched" is decided

LeadHunter tracks enrichment state at three independent levels, so manual edits and partial fills always survive a re-enrich:

1. **Row level — should we scrape at all?** A row counts as already enriched when it has cached website content *and* the scrape is recent. Stale rows (older than the scrape cache window) re-enter the queue automatically.
2. **Field level — should we overwrite?** Every structured field (phone, email, the social URLs, `year_founded`, `employee_count_estimate`, …) is **only written when the current value is blank**. So if you manually corrected the LinkedIn URL on one account, a re-enrich never overwrites it.
3. **Discovery level — did we already try?** When website discovery fires (Gemini grounding + domain guessing) and finds nothing, the row is flagged "discovery attempted". Subsequent runs skip it so we don't loop on the same dead-ends.

You'll see freshly-imported accounts grow website content, a language tag, socials, founding year, and a headcount bucket over the following minutes. The activity is tracked under API costs and visible on the Tasks

page.

Deploy-safety: large enrich jobs survive a redeploy

Enrichment runs that take longer than a worker's lifetime (long imports, big project-wide enriches) used to risk losing in-flight work if the platform redeployed mid-run. The current pipeline:

- **Slices large jobs into chunks** that each finish well within a worker's time limit.
- **Persists the chunk plan** so a worker that picks up after a restart knows exactly which slices already landed and which still need to run.
- **Auto-resumes orphaned runs** when a fresh worker comes online and once an hour on a steady cadence. A job that was interrupted continues from where it left off; you don't need to click Retry.

If a run can't be auto-resumed (a CSV import whose upload-cache expired, for example), it's flagged **failed** on the Tasks page with a one-line explanation and a Retry button.

Deduplication, in order

Every path runs the same dedupe stack. The first level that matches wins:

1. **Google Place ID** — exact match → auto-merge (100% confidence). The hardest constraint; a Google place is unambiguous.
2. **Imported ID** — exact match on `imported_id` within the same company → auto-merge (100% confidence). When you mapped the source's stable row id (Station UUID, CRM Record ID, etc.) during the import, a re-import resolves here first — even when name + city don't agree or are blank. This is the level that makes monthly CRM re-exports clean: every row that already exists is found here.
3. **Name + city** — exact match *after normalisation* (legal suffixes, punctuation, filler words stripped). Auto-merge at 95%.
4. **Phone** — normalised exact match (strips formatting, country codes, spaces). Auto-merge at 90%.
5. **Website domain** — normalised exact match (`www.`, scheme, trailing slash stripped). Auto-merge at 85%.
6. **Fuzzy name within the same city** — ≥85% similarity. *Suggested* merge — surfaces in **Accounts** → **Duplicates** for you to confirm.

When a candidate is **auto-merged** (levels 1–5), every unique field from every duplicate is preserved on the survivor (the "golden record"), and the audit trail records what was merged. When a candidate is **suggested** (level 6), it shows up in **Accounts** → **Duplicates** for you to confirm or reject — LeadHunter never auto-merges fuzzy candidates.

For the full merge workflow + field-winner rules, see Merge duplicates.

Common pitfalls

- **Importing without dry-run.** For anything over ~50 rows, run dry-run first. It costs nothing and tells you exactly what the dedupe stack will do — including which rows will merge into existing accounts.
- **Skipping the column mapping review.** The AI suggestion is usually right, but it's worth a scan — a `notes` column that lands on `description` and vice versa is annoying to clean up later.

- **Forgetting to set `acquisition_channel` for inbound imports.** If you're loading a Stripe-customer export or an Adwords-leads CSV, set `acquisition_channel` as a static value (`adwords`, `cold_inbound`, etc.) so the rows land at `status='contacted'` and slot into the attribution dashboard instead of looking like outbound prospects.
- **Re-importing a file you've already imported.** LeadHunter detects this — the mapping step shows a yellow warning with the prior import's date, who ran it, and how many rows are still around. The dedupe stack then skips every match, so re-imports of unchanged files are safe no-ops. The trap is when the *file* changed (new export from the same CRM) but you didn't map a stable id: without `imported_id`, dedupe falls back to name + city + phone + website, which can miss matches on sparse files and create duplicates. Map the source's stable id to `imported_id` for any dataset you'll re-import regularly.

Read next

- Merge duplicates — confirm and resolve fuzzy candidates LeadHunter flagged.
- Track inbound leads — attribute imports from Ads / forms / referrals correctly.
- Build a saved filter — once accounts are in, slice them into campaign-ready lists.

Track inbound leads (Adwords, Instagram, referrals)

LeadHunter was built around **outbound** prospecting: you search, you find, you reach out. But once a company gets going, leads start showing up from other places too — someone clicks a Google Ads form, replies to a story on Instagram, gets introduced by a customer.

This guide shows how to log those inbound leads with proper channel attribution, so you can answer questions like *"how many customers came from Adwords last quarter?"* and so the conversation history starts in the right state from day one. For the broader **Acquisition channels** model — channel-level monthly spend, CAC, customer-value rollups, and ROI per channel — see Acquisition channels and the companion guide on value tracking.

The acquisition channel field

Every Account carries an **acquisition channel** — the marketing channel that produced the lead. The default is `outbound` (you found them). The other values describe how the lead reached you:

Channel	When to use
`outbound`	**Default.** You cold-prospected the account.
`adwords`	Click on a Google Ads result → form fill.
`meta_ads`	Facebook or Instagram paid ad → lead form.
`linkedin_ads`	LinkedIn Ads → form fill.
`organic_search`	Found you in Google search → form fill on your site.
`organic_social`	DM on Instagram, reply on X, organic LinkedIn message.
`referral`	Existing customer or partner introduced them.
`event`	Trade show, conference, meetup — exchanged details in person.
`partner`	Co-marketing partner sent them over.
`cold_inbound`	Someone emailed you cold, or filled a generic contact form.
`other`	Doesn't fit any of the above.

The channel feeds into:

- **Initial status.** Inbound channels (everything except `outbound` and `other`) auto-promote the account from `prospect` to `contacted` on creation. The account has already reached out — it would be misleading to call them a "prospect" still untouched.
- **Filtering and segmentation.** You can build saved filters and slice the dashboard by channel.

- **The audit trail.** The status change carries `source: inbound_acquisition` so it's clear *why* the account opened in `contacted` state.

Setting the channel from the dashboard

The most common path — log a lead by hand and pick the channel from a dropdown.

When creating a new account

From **Accounts** → **New account**, fill the standard fields (name, email, phone, website, …) and pick the channel from the **Acquisition channel** dropdown. Each option shows its icon plus a short hint of when to use it, and inbound channels are tagged with a green **inbound** badge.

When you pick an inbound channel, a green note appears: *"Inbound channel — new accounts start at status 'Contacted' instead of 'Prospect'."* That's a heads-up that your save will skip the usual `prospect` step. Save, and the account opens on its detail page already at **Contacted** with the audit-trail entry visible in the status history.

Changing the channel on an existing account

Open the account detail page. The right column has an **Acquisition** card under Relationship Type with the same dropdown — change the channel and the value saves immediately. Below the dropdown you'll see the **acquired-at** date and a collapsible **Metadata** block listing whatever channel-specific data the row carries (UTM parameters from the original submit, the Instagram thread URL, the referrer's email, etc.).

Changing the channel on an existing account **doesn't** retroactively re-promote the status — the audit trail is forward-only.

Filtering and searching by channel

On the accounts list, the **Channel** column shows a compact chip for any inbound channel (`outbound` is hidden so the column stays quiet for the cold-prospecting majority).

To slice by channel for reporting, build a **saved filter** (Accounts → Filter builder → field "Acquisition channel"). Combine with status, country, custom fields, and date ranges to build views like *"Adwords leads from last quarter that haven't responded yet."*

Storing channel-specific data

Each channel comes with its own bag of metadata — UTM parameters from an ads click, the Instagram thread URL, the referrer's name, a form id. There's a free-form **acquisition metadata** field for that.

The dashboard shows the metadata as a collapsible block on the account detail page. It's most useful as a permanent attribution record — you fill it once when the lead comes in, then refer to it later (for filtering, reporting, threading to the original ad).

The shape is up to you. Useful conventions:

- Adwords → `utm_source`, `utm_medium`, `utm_campaign`, `gclid`, `ad_id`, `form_id`.
- Meta Ads → `utm_source`, `utm_campaign`, `fbclid`, `ad_id`, `form_id`.

- Instagram DM → `network`, `handle`, `thread_url`.
- Referral → `referrer_account_id`, `referrer_email`, `referrer_note`.
- Event → `event_name`, `booth`, `date`.

Anything you store is available afterwards for filtering, reporting, and threading through to the conversation history.

Worked example 1 — A lead from a Google Ads form

Someone clicks one of your Google Ads, fills the form on the landing page, and lands in your inbox as a notification. Here's the LeadHunter-side flow.

1. **Accounts** → **New account**. Paste their company website or the email's domain into the lookup box; pick **Quick** mode. LeadHunter auto-extracts the business name, sector, location.
2. Set **Acquisition channel** to **Google Ads**. The form shows the green "Inbound — starts at Contacted" hint.
3. In **Acquisition metadata**, paste whatever the form submission gave you — UTM parameters, the `gclid`, the ad campaign id, the form id. Conventions are above; the field is free-form, so the keys are whatever fits.
4. (Optional) Set **Acquired at** to the submission timestamp from your form-builder. Otherwise the default of "now" is fine.
5. Save.

The account opens at `status: contacted`, with an `inbound_acquisition` entry in its status history naming the channel. The funnel will count this account as **initiated** the moment your first outbound message goes out via the conversation log.

Worked example 2 — An Instagram DM

Someone slides into your DMs asking about your product. You want to track the account properly and keep the conversation log threaded.

1. **Accounts** → **New account**. Type the company name or the IG handle into the form (LeadHunter can't auto-discover from an Instagram URL the way it can from a website — you'll fill the fields by hand).
2. Set **Acquisition channel** to **Organic social**.
3. Fill **Acquisition metadata** with the IG handle, the thread URL, and a short note about what they asked.
4. Save.
5. Open the account and click into its **Conversation** tab. Paste their first message as an **Inbound paste** to start the thread.

The account is now at `status: contacted` with the IG thread URL one click away in the account's right-column metadata, and the conversation log carries the back-and-forth — auto-translated into your reading language if they wrote in a different one.

Worked example 3 — A referral from a customer

A current customer sends a contact your way. You want to track both the new account *and* the referrer.

1. **Accounts** → **New account**. Fill the new account's details (or paste the company website into the lookup).
2. Set **Acquisition channel** to **Referral**.
3. In **Acquisition metadata**, record who sent them: the referring customer's name, email, and a short note about what they're after.
4. Save.

The account starts at `contacted`, the referrer trail lives on the account permanently, and a quarter from now you can build a saved filter for *"every account acquired via referral that became a customer"* to see which existing customers send you the most business.

Filtering by channel

Once you start tagging accounts with their channel, slice your database however you want — the filter-builder on the Accounts list has **Acquisition channel** in its field picker. Combine with status, date, and custom fields to build views like *"Adwords leads from last quarter that haven't responded yet."* Save the filter to re-use it across campaigns.

See Build a saved filter for the field/operator surface in detail.

What about leads that came from multiple places?

You can only set **one** acquisition channel per account — pick the one that first introduced you. If the same account later showed up at an event, log that as a regular conversation event rather than overwriting the original channel. The channel is *"where they came from"*, not *"every place we've crossed paths."*

If LeadHunter detects the new account is a duplicate of one you already had, the dedupe stack kicks in and the channel of the original (older) account wins. See Merge duplicates.

Read next

- Account — the broader concept this field lives on.
- Log a conversation — once the inbound account exists, capture the back-and-forth.
- Build a saved filter — segment by channel for reporting and campaign targeting.

Build a saved filter

A **saved filter** is a stored query over your accounts. You define the conditions once and re-use the filter everywhere LeadHunter accepts one — the accounts list, campaign bulk-add, the API. Think of them as smart lists that stay in sync as your data changes: an account that newly matches the filter shows up automatically, and one that drifts out disappears the same way.

Saved filters are the answer to *"how do I target the same audience across three campaigns without re-picking accounts each time?"*

Where you use a saved filter

Surface	What it does
Accounts list (sidebar shortcut)	Live view of every account that matches the filter.
Campaign bulk-add (`Bulk-add from filter`)	Pulls every matching account into the campaign in one click, honouring all guardrails. The reach estimator tells you exactly how many will land before you commit.
API (`/api/accounts/?saved_filter=<id>`)	Programmatic access — the same filter, same matches.
Inline filter (the filter-builder in the Accounts list)	Build ad-hoc, optionally save.

Building a filter from the UI

From **Accounts** → **Filter** (the funnel icon above the table), the filter-builder appears as a panel with a list of condition rows.

Step 1: Pick the match mode

A filter has one **match mode** at the top:

- **All** (AND, default) — every condition must be true. *"Spanish customers"* = `country = Spain` AND `status = customer`.
- **Any** (OR) — at least one condition must be true. *"customers or lost deals"* = `status = customer` OR `status = lost`.

There's no nested mixing of AND and OR. When you need *"(A and B) or (C and D)"*, save two filters and combine them at the campaign level — or use one of the multi-value operators below (`in`, `not_in`) which already act like OR within a single field.

Step 2: Add conditions

Click **+ Add condition** and pick the **field**, then the **operator**, then the **value**. The form adapts to what you pick — a date field gets a date picker, a multiselect gets a chip selector, a boolean gets a toggle.

Step 3: Save it

Click **Save filter**. Give it a name (e.g. *"Spanish fintech prospects"*), optionally a description, and pick **Private** or **Shared**:

- **Private** — only you see and can use it. The default.
- **Shared** — visible and usable by every member of your company. Useful for "the canonical X list" everyone references.

You can flip a filter between private and shared at any time if you own it. Shared filters protect against drift: when one person edits it, everyone sees the update.

What fields can you filter on?

Three categories:

Standard account fields

The columns LeadHunter knows about on every account:

`name`, `country`, `city`, `state_province_region`, `postal_code`, `email`, `phone`, `website`, `business_type`, `specialization`, `language`, `rating`, `source`, `is_verified`, `status`, `relationship_types`, `acquisition_channel`, `acquired_at`, `created_at`, `updated_at`.

A few worth highlighting:

- **`status`** — the lifecycle (`prospect`, `contacted`, `in_negotiation`, `customer`, `lost`, `do_not_contact`). Combine with `in` / `not_in` to slice the pipeline (*"prospects and contacted, not lost"*).
- **`relationship_types`** — multi-tag (one account can be both `client` and `partner`). Use `contains` to match *any* account that carries a given type: `{relationship_types contains partner}` finds every partner account, whether it's *also* a client or not.
- **`acquisition_channel`** — the marketing channel that produced the lead (`outbound`, `adwords`, `meta_ads`, `referral`, …). Combine with `in` to scope to paid inbound: `{acquisition_channel in [adwords, meta_ads, linkedin_ads]}`.

Custom fields (`cf.<key>`)

Anything your company defines under **Settings → Custom fields**. Reference them with the `cf.` prefix — e.g. `cf.tier`, `cf.contract_renewal_date`, `cf.fleet_size`. The operators available depend on the field's type (number / date / select / multiselect / text / boolean / url).

See Custom fields for how to define them.

Virtual "has X" booleans

Three convenience flags that resolve to *"is this column non-empty?"*:

- **`has_email`** — account has an email on the org row.
- **`has_phone`** — account has a phone.
- **`has_website`** — account has a website.

Only `is_true` / `is_false` apply. Useful for filtering out accounts that aren't reachable: *"prospects with a phone and a website, missing email is OK"* = `{has_phone is_true}` AND `{has_website is_true}`.

Operators

Operator	Works on	Notes
`eq` / `neq`	any	Exact match.
`in` / `not_in`	any	Value is a list. Acts as OR within the field.
`contains`	text, url, `relationship_types`	For text fields → substring. For `relationship_types` → array contains the value.
`starts_with`	text, url	Prefix match.
`gt` / `gte` / `lt` / `lte`	number, date, datetime	Use ISO strings for dates (`2026-01-01`).
`exists` / `not_exists`	any	True when the field has *any* value (non-empty, non-null).
`is_true` / `is_false`	boolean, virtual `has_*`	

Two worked examples

"Spanish fintech prospects that have an email"

```
match: all
conditions:
  country = Spain
  status = prospect
  cf.industry_segment = fintech
  has_email is_true
```

Spanish prospects in your fintech segment that you can actually email. Save this and use it to power weekly outreach campaigns — every new fintech-tagged prospect you add appears in the next campaign automatically.

"Paid-inbound accounts from this quarter, sorted by recency"

```
match: all
conditions:
  acquisition_channel in [adwords, meta_ads, linkedin_ads]
  acquired_at gte 2026-01-01
```

Pulls every Adwords / Meta / LinkedIn Ads lead from Q1, regardless of status. Combine with **Stats by product and campaign** to see how the paid funnel converted. Re-running the same filter next quarter only needs the date updated.

Using a filter in a campaign

From the campaign's accounts panel, click **Bulk-add from filter** and pick your saved filter. The **reach estimator** tells you exactly how many accounts will land before you commit — it accounts for every guardrail (DNC, customers, competitors, personal network, soft-blocked relationship types) so the number you see is what will actually be added.

After confirmation, every matching account is added in one pass. The response tells you what was created vs. blocked vs. already-in-the-campaign, with per-block-reason ID lists so you know exactly what to flip (`include_customers`, `include_press`, etc.) if you want to override.

The filter is evaluated *at the moment you click bulk-add* — it's a one-shot snapshot. Later additions to the filter's matches don't auto-join the campaign. If you want the campaign to keep growing as new matches arrive, re-run bulk-add periodically.

Editing and sharing

From **Accounts** → **Saved filters** (sidebar), every filter you own or that's shared with the project shows up. Open one to edit conditions, rename, flip private↔shared, or delete. Deletes are immediate (filters are cheap to recreate); they don't affect the underlying accounts.

If you flip a shared filter to private, it disappears for everyone except you. The reverse — private to shared — makes it visible to the whole project immediately.

Common pitfalls

- **Building mixed AND/OR in one filter.** The match mode is per-filter, not per-condition. When you genuinely need *"(A and B) or (C and D)"*, save two filters and call bulk-add twice, or restructure to use `in` for the OR side and `all` for the AND side.
- **Forgetting that `contains` on `relationship_types` is array-membership, not substring.** `{relationship_types contains partner}` matches accounts that have `partner` in the array, *not* accounts whose tags contain the word "partner" as a substring. The two coincide for short tag names but the semantics matter when you start using `not_in` and overlaps.
- **Sharing a private-by-default filter without renaming it.** *"My experiment"* is fine while it's yours; not great when your whole company starts using it. Rename before sharing.
- **Expecting bulk-add to be subscriber-style.** The filter is evaluated once at the moment of bulk-add. It's not a saved subscription — re-run if you want the campaign to absorb later additions.

Read next

- Custom fields — define `cf.*` keys your filters will reference.
- Run your first campaign — use the filter to populate the campaign in one click.
- Track inbound leads — set `acquisition_channel` so the filter can slice by source.
- Merge duplicates — clean up the database so the filter's matches are accurate.

Run your first campaign

A complete walkthrough from an empty campaign to your first reviewed scores and reaching out. Plan for about 30 minutes of hands-on work spread across a couple of sessions while scoring runs in the background.

Before you start

You'll need three things in place. If any are missing, finish the Quick start first.

- A **Company** (the project you're working in).
- A **Product** with a description, website, and at least an example or two of good-fit accounts. The product is what the campaign promotes.
- A handful of **accounts** in your database — at minimum a few dozen, ideally a few hundred. Import them (Import accounts), build them from Google Maps lookups, or let inbound channels accumulate.

The product needs an approved **ICP** before scoring can run. If you haven't generated one, do that first from the product's detail page — LeadHunter drafts it from your website + example URLs, you review and approve. See ICP and scoring for the field-by-field breakdown.

1. Create the campaign

Campaigns → New campaign:

- **Campaign goal** — the first thing you pick, because everything else adapts to it. Defaults to *Sales prospecting*; pick a different goal (partnership, press, recruiting, customer expansion, win-back, event, research, influencer, investor, renewal) when the intent isn't a sales pitch. The page hero, the ICP framing, scoring rubric, message drafts, and guardrails all flex per goal. See Outreach reasons and targets.
- **Product** — pick the one this campaign promotes. A campaign points to exactly one product, and switching is allowed within the same company.
- **Name** — name the *audience*, not the product. *"Berlin bike shops Q2"* tells you something; *"Spring campaign"* doesn't.
- **Communication language** — leave blank to inherit the product default. Set it explicitly when you're running parallel campaigns of the same product in different markets (one in Spanish for Spain, one in English for the UK).

Save. The campaign opens in `draft` status with an empty accounts panel.

2. Populate it with accounts

Three ways to fill the campaign — combine them freely:

- **From a saved filter** — best for repeatable plays. Open the campaign's accounts panel, click **Bulk-add from filter**, pick a saved filter. The reach estimator tells you how many accounts will land before you

commit.

- **By hand** — open the panel's accounts side, search for an account by name or website, click to add. Use the **+ Add new account** shortcut to slot in one specific company you don't have yet.
- **From an import** — accounts created during an import wizard can be routed straight into a campaign.

Bulk-add honours every guardrail in one pass. Two rules are universal across every campaign goal:

- `do_not_contact` accounts are hard-blocked (never overridable). GDPR / opt-out sticky.
- **Competitors** and **personal network** accounts are hard-blocked (never overridable).

Everything else depends on the campaign **goal**:

- **Customers** are blocked in sales / press / recruiting / investor / influencer / win-back campaigns unless you flip `include_customers`. They're admissible by default in customer expansion / renewal / event / research / partnership campaigns — those are the goals where reaching customers is the point.
- **Suppliers, investors, press, analysts, candidates** are soft-blocked in goals that don't target them, with per-type override flags. A press campaign won't block press accounts; a recruiting campaign won't block candidate accounts; etc.

The UI tells you exactly which accounts were blocked, why, and which override (if any) unlocks them. See Outreach reasons and targets for the full per-goal matrix.

For attribution on inbound campaigns, also set the relevant **acquisition channel** on accounts before adding them — see Track inbound leads. It doesn't change the campaign mechanics, but it lets you slice "Adwords accounts that responded" later.

3. Wait for scoring

Scoring runs in the background. The campaign's **scoring_status** advances `idle → running → completed`, and you can keep working — the page polls the status and updates the score column as rows come back.

What to expect:

- **Per account: a few seconds** for the LLM call, plus website discovery + scraping for accounts that arrived without a URL. Most batches of a few hundred finish in 5–15 minutes.
- **Cost**: a small fraction of a cent per account on the default Gemini model. Auto-tracked under API costs — you don't enter it.
- **Reuse**: if the account already has a score against this product (from a previous campaign of the same product), the score is reused — no re-scoring cost.

Each account ends with three things attached:

- `ai_score` — a 0–10 number.
- `score_label` — `excellent` (≥8), `moderate` (5–7), `mismatch` (<5).
- `score_reasons` — 2–4 short, typed justifications (positive / negative / neutral). Read them — they tell you *why* the AI picked that score, and reading a handful is the fastest way to spot a mis-calibrated ICP.

If the reasons feel off across many accounts, edit the **ICP** on the product (sector, firmographics, anti-patterns) and rescore. The ICP is what the model anchors on.

4. Review the scored list

If the campaign is large, the bottom of the list is usually noise — accounts scored as **mismatch** that aren't worth your review time. Click **Remove low-fit** on the campaign-accounts panel to clear them in one pass; approved and already-contacted rows are preserved by default. Details: Campaign → Pruning the low-fit tail after scoring.

Then sort by score descending. Skim the top, and decide row-by-row:

- **Approve** the account if the AI got it right and you want to reach out.
- **Reject** if the AI got it wrong — it's a mismatch or the timing's bad.
- **Override** the AI score if you want to bump or knock a specific account up or down without rejecting it.

Approvals and rejections aren't just for *you*. Once a product has roughly ten reviewed accounts across all its campaigns, LeadHunter starts feeding your most recent approvals and rejections to the scoring model as concrete examples of *"what good looks like here"* — and the next batch of scores gets sharper. Below that threshold the example URLs on the product remain the only anchor. So treat the review pass as calibration, not just triage.

The cohort dashboard on the home page will eventually attribute responses and customer-close events to the campaign's accounts; the more accurate your reviewed list, the more honest those numbers are.

5. Reach out

Open an approved account, click into the **Conversation** tab. Five drafting modes are available:

Mode	When
AI draft	First outbound message. Generates in the account's language.
AI continue	Continuation. The AI reads the prior history.
Type & translate	Write in your language, see the translated version, send the translation.
Log sent	You already sent it elsewhere — paste the text to record it.
Inbound paste	The account replied. Paste; AI translates into your reading language.

For each account: pick a contact (or add one if you don't have it yet), draft, edit until it's right, send through your normal channel (email, IG, LinkedIn, WhatsApp, phone), and click **Log sent**. LeadHunter records the message and:

- Bumps the campaign-account `outreach_status` to `sent`.
- Promotes the account's lifecycle `status` from `prospect` to `contacted` (forward-only — already-customer accounts don't regress).
- Counts the account as **initiated** in the dashboard funnel.

When replies come in, paste them via **Inbound paste**. The campaign-account flips to `responded`; the dashboard funnel counts the account as a response.

6. Log spend as it accrues

A campaign is only half-tracked without the cost side. As you run paid ads, pay agencies, or spend internal hours on this campaign, log each expense to the **Costs & expenses** panel on the campaign detail — or use the 💸 **quick-add** button on the campaigns list. LeadHunter then surfaces **cost per outreached account**, **cost per response**, and (after a few customer closes) **cost per customer** on the *Stats by product and campaign* page.

The full mechanics: Track campaign costs and CAC.

7. Watch the funnel

Open the **Dashboard** (sidebar → home). Three numbers tell you whether the campaign is working:

- **Initiated** — outreach started, distinct accounts.
- **Responded** — accounts that replied (cohort-attributed, by initiation date).
- **Closed (won)** — accounts that converted to `customer` (cohort-attributed).

Cohort attribution means the rate at the latest day is "low by design" — accounts initiated yesterday haven't had time to reply yet. Compare *windows* to each other (last 30d this month vs. last 30d last month) instead of reading the right edge of the chart as gospel.

The **Stats** page (sidebar → *Stats*) splits the same numbers per product and per campaign — handy when you're running several campaigns of the same product and want to see which one is the most efficient.

8. Close out

When the campaign is done:

- **Archive** — soft-hides the campaign from the default list while keeping every conversation, score, audit-trail entry, and account link intact. Use this for "completed but still want the history."
- **Delete** — name-confirmed (you type the campaign name to confirm). Cascades to every campaign-account, conversation, and ICP for this campaign. Use only when you're sure.

`do_not_contact` opt-outs and `customer` lifecycle states on accounts stick around — those live on the account, not the campaign.

Common pitfalls

- **Skipping the ICP review.** A draft ICP that's never approved will still score, but the reasons will read fuzzy. Spend ten minutes editing the ICP before running scoring on a few hundred accounts.
- **Bulk-adding everything you have.** The reach estimator exists so you can stop before importing 4,000 accounts and burning Gemini quota on a single test run. Start with a few dozen of the strongest candidates, calibrate, then go wide.

- **Treating reviews as one-and-done.** Approvals and rejections feed back into scoring once you've passed the ten-reviewed-account threshold for the product. Your first campaign on a new product is the best one to be thorough on.
- **Forgetting the costs side.** A campaign's success isn't response rate alone — it's response rate at what cost. Logging spend as you go (or once a week) means CAC is honest when you go to compare campaigns.

Read next

- ICP and scoring — how the AI actually picks a score.
- Build a saved filter — make populating the next campaign one click.
- Log a conversation — full reference on the five drafting modes and the language fallback chain.
- Track campaign costs and CAC — close the loop with cost-of-acquisition stats.

Track campaign costs and CAC

A campaign isn't just outcomes — it also costs something. LeadHunter lets you log every expense against a campaign (agency, hours, software, content, events) so the numbers you read aren't *"we got 12 customers"* but *"we got 12 customers for €4,200, that's €350 each."*

Note (May 2026): ad spend (Adwords / Meta Ads / LinkedIn Ads / other ads) **moved to the channel surface** — see Track inbound leads and Acquisition channels. The thinking: ads bring leads *to you* (inbound), so the spend lives on the channel that captures the lead. Campaigns retained the outbound-effort expenses below. Historical ad-spend rows were migrated automatically — you'll find them under the matching channel.

What you can log

Each expense entry is one line item against one campaign:

Kind	When to use
Agency	External agency or consulting fees.
Labor	Internal hours — copywriting, design, ops, video editing.
Software	SaaS / tooling subscription that supports this campaign.
Content	Video / photo / copywriting production.
Event	Trade show, conference, meetup costs.
Other	Anything that doesn't fit.

Each entry carries: **amount + currency**, **date incurred**, **vendor** (who got paid), and an optional **description** + **external reference** (invoice number, ad campaign id, …).

Adding expenses from the dashboard

From the campaign detail page

Open the campaign and scroll down to **Costs & expenses**. Click **Add expense** — an inline form appears with all the fields. Pick a kind from the dropdown (each option shows its icon and a short hint), type the amount, set the currency and date, optionally fill the vendor and description, and click **Save expense**.

The form pre-fills the currency from the campaign's existing expenses, so once you've logged your first EUR entry you don't re-type EUR on every subsequent add.

Quick-add from the campaigns list

Don't want to open the campaign first? From `/dashboard/campaigns/`, every row has a 💸 button. Click it and a modal opens pre-filled for that specific campaign — pick a kind, type the amount, save, done. Useful when you just paid an Adwords invoice and want to register it across three campaigns in 30 seconds.

The **Cost** column on the campaigns list shows each campaign's running total in its primary currency, so you can see at a glance which campaigns are accumulating spend.

Editing or removing an entry

Every row in the Costs & expenses table has an edit pencil and a delete trash icon. Click the pencil — the row collapses into the same form, populated with the current values. Save the changes (or cancel) and the row updates in place. Delete asks for a confirmation and removes the entry; cost summary refreshes immediately.

Reading the numbers

Every campaign gains a **cost summary** with four cards at the top of the Costs & expenses panel:

- **Total** — broken down per currency. A campaign that mixes EUR and USD shows both rather than pretending they're comparable.
- **Cost per outreached account** — total spend divided by the number of accounts you've reached out to. The cheapest gauge of efficiency.
- **Cost per responded account** — total spend divided by accounts that have *replied*. Tighter than just "outreached."
- **Cost per "closed"** — total spend divided by the campaign's goal-routed close count. The label adapts: a sales campaign reads *"Cost per closed-won customer"*, a press campaign reads *"Cost per press reply"*, a recruiting campaign reads *"Cost per candidate engaged"*, a renewal campaign reads *"Cost per renewal"*. For sales / customer-expansion / win-back / renewal goals the count is accounts that transitioned to `customer` status; for the other seven goals the count is first inbound replies (a journalist taking the pitch is a press hit, an investor replying is an investor reply, etc.).

For an all-time view across campaigns and products, go to **Stats by product and campaign** (sidebar → *Stats*). The Cost column on both the per-product and per-campaign tables shows total spend plus the most meaningful CAC available for that row — cost per closed if there are closed-won customers, else per responded, else per initiated. Cost numerator and outcome denominator both use the last 30 days, so the ratio is comparable across rows.

Mixed currencies

Cost-of-acquisition (CAC) is only computed when **all expenses on a campaign share one currency**. Mixing EUR and USD without an FX layer would give misleading averages, so LeadHunter surfaces a per-currency total and explains the missing CAC instead. Either normalise to one currency or compute the conversion externally.

The Cost column on the list and Stats page shows a small `+` next to the primary-currency total when mixed currencies are present, so you know at a glance the headline number isn't the full picture.

Worked example — running an Adwords campaign

You're running a Spring Bikes campaign with €2,400 in spend, plus 12 hours of internal time at €60/hr.

1. Open the campaign detail page → **Costs & expenses** → **Add expense**.

2. **Kind:** Google Ads. **Amount:** 2400. **Currency:** EUR. **Date:** the month-end you paid. **Vendor:** Google Ads. **Description:** "April CPC".
3. Click **Save expense**. The summary cards update immediately.
4. Click **Add expense** again. **Kind:** Labor. **Amount:** 720. **Description:** "12h × €60 — campaign management".
5. Save.

Total is now **€3,120**. If your campaign has reached 240 accounts and 18 responded, you'll see:

- Cost per outreached account: **€13.00**.
- Cost per responded account: **€173.33**.

Two weeks later, three of those responses convert to customers. The **Stats by product and campaign** page shows **€1,040 per customer** for this campaign in the last 30 days — and rolled up to the *product* across all its campaigns.

What about automated API costs?

LeadHunter automatically tracks the per-call cost of every LLM, Google Maps, and Tavily call attributed to a campaign. That number lives separately under **API costs** — it's auto-derived, you don't enter it manually. To get the *full* cost of running a campaign, combine the two: manually logged spend from this guide plus auto-tracked API usage.

Read next

- Account — the rows you're paying to reach.
- Run your first campaign — get a campaign going so there's something to attribute costs to.
- Track inbound leads — for accounts that came in through Ads, the channel attribution + costs work together to show ROI per channel.

Track customer value and ROI per channel

LeadHunter tracks **how much you spent** on each acquisition channel and each campaign (see Track campaign costs and Acquisition channels). To turn that spend into **return**, you also need to tell it how much each captured customer is worth. This guide walks through the three-layer setup that gets most accounts auto-valued, and how to handle the cases that need manual entry.

The model is explained in detail in Account value (LTV). This guide is the operator playbook.

Step 1 — Set a default value on the Product

This is the workhorse. Most customer accounts will snapshot at this number unless you override.

1. Go to **Products** → **[your product]** → **Edit**.
2. Set **Default account value** to your typical customer's lifetime value, and **Default account value currency** to the matching currency (EUR / USD / etc.).
3. Save.

Pick a number that reflects your actual definition — annual contract value, total expected lifetime margin, gross revenue — and use the *same* definition across every customer so the rollups are comparable. A common starting point: average closed-deal value over the last 12 months.

Effect: every Account that converts to `customer` and doesn't have its own captured value (or a campaign override on the campaign that produced it) auto-snapshots at this number, tagged with source = **Auto: Product default**.

Step 2 — Optionally override on specific Campaigns

When a campaign targets a higher- or lower-value segment than the product default, set an override on the campaign itself.

1. Go to **Campaigns** → **[your campaign]** → **Edit**.
2. Set **Account value override** + **Account value currency** (the currency falls back to the product's if blank).
3. Save.

Effect: customers produced by this campaign auto-snapshot at the override value (Auto: Campaign override) instead of the product default.

Examples:

- **Enterprise vs. SMB campaigns of the same product.** Product default = €2k SMB; enterprise campaign override = €15k.
- **A win-back campaign** where you expect lower-than-typical deal sizes.

- **A press / partnership campaign** with no direct customer revenue — set the override to 0 so those accounts don't inflate ROI on the channel that produced them.

Step 3 — Operators enter deal values at close (when needed)

For long-cycle B2B deals where every contract is hand-negotiated, the actual number matters more than any default. Operators enter the value at close:

1. Open the Account detail page.
2. Scroll to **Captured value** (next to the relationship status).
3. Type the deal value + currency.
4. Save.

LeadHunter records the source as **Manually entered** — and from then on, even if someone changes the product default or a campaign override, this Account keeps its hand-entered number.

You can do this *before* the status change to `customer` (the snapshot will be a no-op because the value is already set) or after — entering it later overwrites whatever auto-snapshot happened at close.

Step 4 — Check the rollups

After a few weeks of customer conversions, the rollups start to surface real numbers.

On a channel

Go to **Acquisition channels → [your channel]**. The page shows:

- **Value summary**: total captured value by currency, breakdown by source (Manual / Auto: Campaign / Auto: Product), customer count + how many have captured values set.
- **ROI**: `value ÷ spend` when both share a currency. A `25.00` reading means €25 of captured customer value per €1 of channel spend.

If `customers_with_value` is lower than `customers_total`, you've got customers without LTV entries. Either set product / campaign defaults to cover them automatically, or click into each Account and enter a value.

On a campaign

Go to **Campaigns → [your campaign]**. The Cost section shows the same triple (cost / value / ROI), this time scoped to customers attributed to that campaign.

On the dashboard

The stats dashboard breakdown renders the three rollups side-by-side (by product, by campaign, by channel), with the 30-day window most operators care about.

Common patterns

Self-serve SaaS with a single price plan. Step 1 only. Product default = your average customer LTV. Every customer auto-snapshots; the operator only intervenes when a specific deal is unusually large.

Self-serve + sales-led tiers. Step 1 for self-serve. Step 2 with campaign overrides for each enterprise tier campaign. Optionally step 3 when an enterprise customer signs at a non-standard ACV.

Hand-negotiated B2B. Skip step 1 or set conservative placeholders. Step 3 is where the real numbers come from. Operators learn to update Captured value as part of the close ritual.

Press / partnership / recruiting campaigns. Step 2 with override = 0 (or leave blank). These aren't direct-revenue campaigns and you don't want them inflating ROI on the channels they ran on.

Troubleshooting

A customer is missing from the value rollup. Check three things: (1) the account's status is `customer` (not `in_negotiation` or `contacted`); (2) `captured_value` is set on the account — the rollup excludes customers without a value; (3) the account is attributed to the channel/campaign you're looking at (acquisition_channel FK for channels; campaign membership for campaigns).

The ROI block says "currencies differ". Spend and value need to be in the same currency for the math to run. Either change one to match, or wait for the currency-conversion feature on the roadmap.

The status changed to customer but no value was captured. The product / campaign defaults aren't set, or the account has no campaign membership (a pure-inbound that you didn't enroll in a campaign). Set the value by hand, or back-fill the product default and re-open + re-save the account to re-trigger the snapshot (you can also just type a value directly).

A customer's value is wrong (outdated default). The snapshot is frozen at conversion time. Edit the Account's Captured value field directly — operator-set values become sticky and won't be overwritten by future changes.

What this does NOT do

- It does not track **churn or contract changes** — once a customer has a captured value, the number stays. If a customer churns or you expand the contract, edit the field. A structured value-history journal is on the roadmap.
- It does not pull deal values from your CRM. There's no Salesforce / HubSpot sync today. CSV import / API write is how to bulk-update.
- It does not enforce a particular definition (ACV vs. ARR vs. lifetime margin). Pick one definition per product and stick with it.

See Account value (LTV) for the full data model, Acquisition channels for how spend rolls up, and What's new for the latest changes.

Log a conversation

LeadHunter is a **communication log**. You keep talking to accounts on the channels you already use — email, LinkedIn DMs, Instagram, WhatsApp, phone — then paste the messages into LeadHunter so the per-account history, funnel stats, and scoring stay current across the team.

The conversation log isn't a sending engine. LeadHunter doesn't connect to your inbox or your phone — that's intentional. You stay in control of *how* and *where* you send; LeadHunter helps you draft, translate, remember, and measure.

Where the thread lives

Two entry points show the same conversation:

- **Campaign view** — open a campaign, click an approved account, hit the **Conversation** tab. You see every message exchanged *in this campaign*.
- **Account detail** — open the account directly (`/dashboard/accounts/{id}`). The conversation log aggregates across **every** campaign that account has been in, so you have one timeline whether the account is now in a different campaign than the one that first contacted them.

Pick a channel

Every message carries a **channel**, which is part of the record so your future self knows whether *"hey, did you see my note?"* was an email or an Instagram DM. The eight values: **email**, **linkedin**, **instagram**, **twitter_x**, **whatsapp**, **phone_call**, **sms**, **other**.

`email` is the default. Switch the channel on each message — the per-message channel is what the cohort funnel attributes responses to later, not the campaign default.

The five drafting modes

Each message is logged in one of five **modes**. The right mode depends on who wrote it and whether translation was involved.

Mode	Direction	Use when
AI draft	outbound	First outbound message in this thread. AI writes a fresh draft.
AI continue	outbound	Continuing an existing thread. AI reads the prior history and drafts the next move.
Type & translate	outbound	You wrote the message yourself in your language. LeadHunter shows the translation into the account's language; you send the translation.
Log sent	outbound	You already sent it elsewhere (in your own email client, on LinkedIn directly, …). Paste the text just to record it.
Inbound paste	inbound	The account replied. Paste their reply; LeadHunter shows the translation into your reading language.

What gets stored in each mode

Every message has two text fields. Knowing which is which helps when you go back and read old threads:

- `original_text` — the **authoritative** version. For outbound, what the account *actually* received (in their language). For inbound, what they *actually* sent.
- `translated_text` — a reader-friendly view of the same content in another language.

The four modes populate them differently:

Mode	`original_text`	`translated_text`
AI draft / AI continue / Log sent	The sent message (in the account's language)	(empty)
Type & translate	The translated/sent message (account's language)	Your original draft (your language)
Inbound paste	The account's reply (their language)	Translation into your reading language

That asymmetry matters: when you re-read a `type_translate` outbound, the *original_text* is the version that left your hands, not your scratchpad draft.

Languages: the resolution chain

Two languages are decided per message — what *you* read, and what the *account* reads.

Source (what you write in):

> your `User.communication_language` → `'en'`

Target (what the account reads):

> `Account.language` → `Campaign.communication_language` → `Product.communication_language` → `User.communication_language` → `'en'`

A real example: the product is set up in English, the campaign in Spanish, the lead is Catalan-only. The Catalan setting on the *account* wins — LeadHunter translates outbound into Catalan regardless of what the campaign says. The campaign-level language only kicks in for accounts that don't have their own `Account.language` set.

You can also override the language manually on any single draft — useful when an English-speaking contact at a Catalan organization writes back to you in English and you want to keep the rest of the thread in English without changing the account.

Per-language `communication_language` on the campaign exists so you can run *two campaigns of the same product in different markets* without duplicating the product. See Campaign → Language.

What auto-fires after logging

Logging a message **as sent** isn't just record-keeping — it advances the pipeline:

- **Outbound** with `status='sent'` (any of `ai_draft`, `ai_continue`, `type_translate`, `log_sent`) → `CampaignAccount.outreach_status` becomes `sent` if it isn't already. The dashboard funnel counts this account as **initiated**.
- **First** outbound on a `prospect`-status account → account `status` advances to `contacted`. The status-history entry records `source: campaign_outreach` and references the campaign id.
- **Inbound paste** → `outreach_status` becomes `responded`. The dashboard funnel counts the account as a **response**, cohort-attributed back to the original initiation date.

All promotions are **forward-only**. Already-customer accounts don't regress when you log a follow-up. Already-contacted accounts don't re-fire the promotion when the second outbound goes out.

Drafts (`status='draft'`) don't fire any of this — they're scratch space. Switch to `sent` to log the message as truly out.

Guardrails: DNC

Outbound to a `do_not_contact` account is blocked at the API. The UI shows a clear error pointing at the account's status history, so the operator can see when and why it was flagged. There's no "include DNC anyway" override — the GDPR/CAN-SPAM opt-out is sticky on purpose. If you genuinely need to contact a DNC account (a single follow-up to confirm the opt-out, for example), change the account's status manually first.

Other relationship guardrails (`competitor`, `personal_network`, customer-class soft blocks, etc.) only apply to **campaign bulk-add** — they don't apply to logging individual messages on accounts that are already in a campaign. The rationale: if it's already in the campaign, it cleared the bulk-add filters.

Worked example — outbound, reply, continue

1. **First outbound**. Open the conversation, pick channel **email**, mode **AI draft**. The AI generates a first-touch message in the account's language. Edit until it reads right. Click **Send** (LeadHunter records it; you copy-paste into your actual email client and hit send there).
2. The account's `status` flips `prospect → contacted`, the campaign-account's `outreach_status` flips to `sent`, the dashboard's "initiated" counter ticks up.
3. **Two days later, they reply**. Copy their email body, click **Inbound paste**, channel **email**. The reply is stored as `original_text` (their language); the translation into your reading language appears as `translated_text`.
4. The campaign-account's `outreach_status` flips to `responded`. The dashboard's "responded" counter ticks up, attributed back to step 1's initiation date.
5. **Continue the thread**. Switch to **AI continue**. The AI reads the prior two messages — *your* draft from step 1 and *their* reply — and proposes a continuation in the account's language. Edit, send via your email client, log.

The thread on the account detail page now shows three messages in the order they happened, with channel chips, mode chips, language flags, and the auto-promoted status transitions visible in the account's status history sidebar.

Common pitfalls

- **Picking the wrong mode for translated outbound.** When you draft something in your own language and translate it before sending, that's **Type & translate** — not `Log sent`. The mode matters because `type_translate` stores both versions; `log_sent` only stores one and your future self loses the source.
- **Forgetting to switch channel on a non-email exchange.** `email` is the default and most users leave it. If the thread is on LinkedIn or Instagram, switching to the right channel before sending is what makes the cohort funnel correctly attribute *"responses on LinkedIn"* a quarter from now.
- **Keeping drafts forever.** Drafts don't advance any status. If you draft a message and never send it, the account stays at `prospect`, the campaign stays at `initiated=0`, and nothing in the funnel records the work. Either flip the draft to `sent` once you've sent it, or delete the draft.
- **Re-pasting an inbound reply you already pasted.** Inbound paste counts toward responses; duplicating the same reply double-counts. The conversation history is your source of truth — check before pasting.

Read next

- Run your first campaign — get to the point where there's something to log.
- Account — what each lifecycle status means and which transitions are forward-only.
- Campaign — the language-resolution chain in full, plus the outreach guardrails.
- Track campaign costs and CAC — log the spend that produced these conversations.

Merge duplicates

Duplicates are the **#1 threat to data quality**. Every duplicate erodes trust the moment you reach out — sending the same message twice to the same account is the fastest way to look like a spammer. The whole product is built around **zero duplicates**: prevent at the moment of creation, detect on demand, **resolve by merge**.

LeadHunter never auto-deletes a duplicate. Merge is always the operation, because deleting loses information you might still want.

The five detection levels

Every account creation — manual entry, CSV import, Google Maps lookup, API post — runs through the same five-level stack. The first level that matches wins and the action depends on the confidence:

Level	Match	Confidence	Action
1	**Google Place ID** — same place	100%	Auto-merge
2	**Name + city** — exact, post-normalisation	95%	Auto-merge
3	**Phone** — normalised, exact (strips formatting, country codes)	90%	Auto-merge
4	**Website domain** — normalised, exact (`www.` , scheme stripped)	85%	Auto-merge
5	**Fuzzy name within the same city** — ≥85% similarity	85+	**Suggest** — surfaces in *Find duplicates*

Name normalisation strips legal suffixes (`Inc` , `LLC` , `GmbH` , `SA` , `BV` , …), filler words (`the` , `a` , `and` , …), punctuation, and common substitutions (`&` → `and`). So *"Joe's Pizza & Co."* and *"Joe's Pizza and Co LLC"* are exact-matched at level 2, not fuzzy.

Levels 1–4 are deterministic, exact matches — LeadHunter is confident enough to merge without asking. Level 5 is the one that surfaces for human review.

Finding duplicates on demand

LeadHunter surfaces the same scanner from two places:

- **Accounts → Find duplicates** — scans the whole project. Use this for periodic clean-up after large imports.
- **Campaigns → (campaign) → Scoring → Find duplicates** — scans the same way but only returns groups that contain at least one account in *this* campaign. Use this when you're reviewing scored accounts and notice two rows that look like the same business. Cross-campaign twins still show up in the same group so you see the full picture before merging.

Both return groups of likely duplicates, sorted by confidence descending, and use the exact same merge flow. Each group shows:

- Every account in the group with its name, location, phone, website, and key identifiers side-by-side.
- The proposed **survivor** (the "golden record" — the row that will absorb the others).
- A field-by-field diff so you can see exactly what gets merged where.

AI-assisted review for large lists

When *Find duplicates* returns dozens of groups, an **AI review** button runs each group through an LLM pass that classifies them as `same` / `different` / `uncertain` with a one-line rationale. Use it to filter out obvious false positives before merging — the AI doesn't decide, it just helps you skim faster.

Bulk merge

For groups the AI marked confidently as `same` (and that you've spot-checked), the **Bulk merge** action runs every selected group through the same merge logic in one request. Useful after a noisy import where dozens of obvious duplicates landed in one go.

What a merge actually does

When a merge runs (auto on levels 1–4, on-confirm for level 5):

1. **Every unique field from every duplicate is preserved on the survivor.** If one duplicate has a phone and the other has a website, the survivor ends up with both.
2. **Smart picks on conflicts** when both rows have a value:
   - **Email** — prefers a non-generic address (`maria@acme.com` wins over `info@acme.com`).
   - **Website** — prefers the company's own domain over an aggregator URL (e.g. `acme.com` over a Yelp listing).
   - **Notes** — concatenated, de-duplicated, with attribution.
   - **Latitude + longitude** — picked as a pair from the same source row (never half from one, half from the other).
   - **Website content** — the longer, fresher scrape wins.
   - **Numeric / rating fields** — the higher value wins.
   - **Custom fields** — deep-merged dict; multi-select arrays union.
3. **The survivor's `merge_history` JSONB grows an entry** recording: which accounts were absorbed, their full snapshot (so you can read what was on each row before the merge), who ran the merge, when, and which fields were overridden.
4. **Everything pointing at the absorbed rows is re-pointed at the survivor**: campaign-account rows, conversation messages, research records, scores. No history is lost.
5. **Lifecycle status escalates, never demotes.** The internal order is `prospect < contacted < in_negotiation < lost < customer < do_not_contact`. So:
   - `do_not_contact` **always wins** (GDPR / CAN-SPAM stickiness — an opt-out survives every merge).
   - `customer` beats `lost` — a recorded customer relationship outranks a never-sold lost deal even if the customer is older.

    - The audit trail records the previous status of every absorbed row.

6. **Acquisition channel sticks to the survivor's value** by default. Use the field-override controls on the merge dialog if the absorbed row's channel was actually correct (e.g. you imported the outbound-discovered row first, but the inbound Adwords row was the real origin).
7. **AI picks the canonical name** when the duplicates have different names — *"Joe's Pizza"* over *"Joe's Pizza Restaurant LLC Holdings 2014"*. You can override it manually in the merge dialog before confirming.
8. **Absorbed rows are deleted** at the end. Their data lives on in the survivor's fields and in `merge_history`, but the `Account` rows themselves are gone.

Preview before you commit

Every merge in the UI runs through a **preview** step first — same logic as the real merge, but no writes. The preview shows you the field-winner picks (with the source row that won), the LLM-selected name, the merge_history entry that *would* be written, and any overrides you've set. Confirm to commit, or back out to adjust.

For programmatic merges, hit the preview endpoint before the merge endpoint — same shape.

Why we merge instead of delete

Deleting loses information. Even a "clearly bad" duplicate often has one field the survivor doesn't — a phone number, a typo correction in the address, a different decision-maker contact, a different acquisition channel. Merging keeps every signal *and* the audit trail.

The cost of merging conservatively (keep everything) is one bigger row. The cost of deleting aggressively is permanent data loss. We optimise for the first.

Merges aren't reversible — but the history is preserved

When you confirm a merge, the absorbed accounts are deleted. There's no "unmerge" button — the survivor row is now the truth and re-creating a clean split is hard to do automatically (which campaign-account row belonged to which side? which conversations?). The `merge_history` JSONB *records* the absorbed rows' full snapshots so you can always read what was on each side, but restoring them is a manual rebuild.

In practice this rarely bites — most merges are obvious by the time they happen — but it's worth knowing before you bulk-merge 50 fuzzy candidates without spot-checking.

When in doubt

The trade-off:

- **Merging two accounts that turn out to be different organisations**: rare, but messy when it happens. The survivor row carries fields from both, conversations mix, the merge_history is the record. Manual rebuild required.

- **Not merging two accounts that are actually the same**: common, low cost initially. Eventually, you reach out to the same place twice and look like a spammer.

Bias toward merging when the AI's confidence is high and you've spot-checked the fields. Bias toward leaving them when names look similar but the addresses are clearly different (two unrelated *Acme Bikes* in two countries).

Common pitfalls

- **Bulk-merging without spot-checking the AI review.** AI review is a triage helper, not a decision. Skim the `uncertain` group at minimum before clicking bulk-merge.
- **Forgetting that levels 1–4 already auto-merged on import.** *Find duplicates* shows you what slipped through levels 1–4 — usually accounts that share a fuzzy name but no place id, phone, or website overlap. If you're seeing what looks like an obvious phone duplicate, it usually means the phone wasn't on one of the rows at import time.
- **Manually merging accounts the system flagged as `different`.** The AI is wrong sometimes, but when it confidently says two accounts are different, it's worth re-reading the field diff before overriding.
- **Treating `merge_history` as a backup.** It's an audit trail. Restoring an absorbed row from the snapshot is a manual exercise — possible, not push-button.

Read next

- Account — the row model the merge produces.
- Import accounts — where most duplicates originate, and how to prevent them.
- Track inbound leads — inbound channels create duplicate-prone overlap with outbound rows; tagging acquisition_channel correctly helps disambiguate.

Push events to your tools — webhooks

Webhooks push LeadHunter events to any system that can receive an HTTPS POST — your CRM, Zapier, n8n, Make, a Slack relay, your own backend. Deep-link custom fields let you jump *into* your other tools; webhooks are the other direction: LeadHunter telling your tools what just happened.

What gets pushed

Event	Fires when
`account.created`	A new account lands in the company (manual, import, lookup, inbound)
`account.status_changed`	An account actually moves in the pipeline — `prospect → contacted`, `→ customer`, `→ do_not_contact`, …
`message.inbound`	An inbound reply is logged on a conversation

Every payload carries the account's core fields **including** `imported_id` — so if the account came from your CRM, the event arrives already labelled with *your* record id and matching it on the receiving side is a lookup, not a fuzzy search. Custom fields ride along too.

Set one up

1. Open **Settings** → **Webhooks** (per company).
2. Add the destination URL (must be `https://`), pick the events, save.
3. **Copy the secret immediately** — it's shown once, at creation. If you lose it, rotate it (rotation invalidates the old one).
4. Hit **Test** — LeadHunter sends a synthetic `webhook.test` event through the real delivery pipeline so you can verify your receiver end-to-end before real traffic flows.

Verifying the signature

Every delivery is signed so your receiver can prove it came from LeadHunter and wasn't tampered with. The `X-LeadHunter-Signature` header carries `sha256=<hex>` — the HMAC-SHA256 of the **raw request body** with your webhook's secret as the key. Recompute it on your side and compare:

```
import hashlib, hmac

expected = "sha256=" + hmac.new(
    secret.encode(), raw_body, hashlib.sha256
).hexdigest()
ok = hmac.compare_digest(expected, request.headers["X-LeadHunter-Signature"])
```

Two more headers help with bookkeeping: `X-LeadHunter-Event` (the event type) and `X-LeadHunter-Delivery` (the delivery id).

Delivery semantics — what your receiver should expect

- **At-least-once.** Failures are retried with backoff (a 500, a timeout, a 429). The payload's top-level `id` is stable across retries and redeliveries of the same event — dedupe on it if your handler isn't naturally idempotent.
- **Respond fast with a 2xx.** Anything else counts as a failure. Do the heavy work async on your side.
- **Permanent rejections aren't retried.** A 404/401/410 marks the delivery failed immediately — retrying wouldn't heal it.
- **Dead endpoints switch themselves off.** After 20 consecutive failures the webhook is auto-disabled (you'll see *"auto-disabled after repeated failures"* on the settings page). Fix the receiver, hit Test, then re-enable — the failure streak resets.

Replaying

Each webhook's **Recent deliveries** panel shows every attempt with its status, response code, and error. Any delivery can be **redelivered** with one click — same payload, fresh attempt — which is the tool for "our endpoint was down for an hour" recoveries. Delivery history is kept for **90 days**; anything older is pruned automatically.

Pushing data back in — upsert by imported id

Webhooks are LeadHunter → your system. The return path is one endpoint:

```
POST /api/accounts/upsert-by-imported-id/
{ "project": "my-company", "imported_id": "crm-4711",
  "status": "customer", "custom_fields": {"plan": "pro"} }
```

LeadHunter finds the account whose `imported_id` matches (the same correlation key the webhook payloads carry) and patches **only the fields you send** — `custom_fields` merge per key, and a `null` deletes a key. No match → the account is created. Status changes go through the normal pipeline, so the audit trail and your *other* webhooks fire exactly as if an operator made the change — and pushing the same status twice is a no-op, so periodic full-state re-syncs don't pollute the history. Authenticate with your API token like any other endpoint.

Together that's the full loop: a webhook tells your CRM an account replied; your CRM closes the deal and pushes `status: customer` back.

What's next

Richer two-way sync (bulk upserts, field-level sync rules) builds on the upsert endpoint above; webhooks + upsert already close the basic loop.

Read next

- Custom fields → Deep links into external systems — the inbound-click half of integrations.
- Import accounts — where `imported_id` comes from.

Stay close to your clients — Client Pulse

Winning the deal is half the job; Client Pulse is the other half. It closes the loop after an account becomes a customer, with three pieces that work together:

1. **Usage signals** — your own product's backend sends LeadHunter a tiny ping when a client opens the app, uses a feature, starts a trial, or converts. LeadHunter shows you, per client, when they were last active and which features they actually use.
2. **An "In trial" lifecycle stage** — sits between *In negotiation* and *Customer*, driven automatically by those pings or set by hand.
3. **A contact cadence** — per company you decide "every client hears from us at least every N weeks" (6 by default). Clients that fall behind surface everywhere you look, until someone reaches out.

The combination is the point: heavy product usage with no recent contact is an upsell conversation waiting to happen; silence on both fronts is churn risk you can still act on.

Send usage pings from your product

Any backend that can make an HTTPS POST can feed Client Pulse. Get your **ingest key** from **Settings → Client Pulse** (per company) — like webhook secrets, it's shown once at creation and can be rotated.

```
POST https://leadhunter.api.humans2agents.com/api/pulse/ingest/
X-LeadHunter-Ingest-Key: <your key>
Content-Type: application/json
```

```
{
  "events": [
    {
      "imported_id": "crm-123",
      "kind": "feature_used",
      "feature": "invoice_upload",
      "occurred_at": "2026-07-11T09:30:00Z"
    },
    {
      "imported_id": "crm-124",
      "kind": "app_opened"
    }
  ]
}
```

- **Batch up to 100 events per request.** Send from a background job or queue, never from your product's request path — a lost ping is noise, a slow request is a real cost.
- **Match clients by `imported_id`** — the same correlation key used by webhooks and imports, so an account that came from your CRM is addressable by *your* record id. You can pass the LeadHunter account id instead if you have it.

- **Events that don't match any account don't fail the batch** — the response lists them under `unmatched` so you can spot clients that haven't been imported yet. Never retry an unmatched event in a loop; import the account, then let the next ping land.
- `occurred_at` is optional (defaults to arrival time), `metadata` is an optional small JSON object for anything extra you want on the event.

Event kinds

Kind	What it means	What LeadHunter does
`app_opened`	The client logged in / opened your product	Updates last-seen
`feature_used`	The client used a specific feature (`feature` required)	Updates last-seen + per-feature usage counters
`trial_started`	The client started a trial	Moves the account to **In trial**
`became_customer`	The client converted	Moves the account to **Customer** (and captures account value)
`churned`	The client cancelled	Moves the account to **Lost**
`custom`	Anything else (`feature` names it)	Counted like a feature, triggers nothing

If several of your products feed the same company, namespace the feature slug by product — `accountant.invoice_upload`, `nutrilens.scan` — so the per-client feature list stays readable.

The "In trial" stage

`trial_started` moves an account to **In trial** from any earlier pipeline stage — including **Lost**: a returning trial reopens the relationship, and the status history records the reopening so nothing is silently un-lost. It never demotes an existing customer, and *Do not contact* is never touched.

Trials are protected from cold outreach the same way customers are: campaign goals that block customers also block trials, with an explicit *include trials* override when a campaign genuinely targets them. Status changes made by pings appear in the account's history with their own source, so you can always tell automation from a human edit.

Configure the contact cadence

In **Settings** → **Client Pulse** (per company):

- **Contact every N days** — default 42 (6 weeks).
- **Tracked stages** — by default *Customer* and *In trial*; add *In negotiation* if you want the nudge pre-close too.
- **Weekly digest** — on by default; pick the weekday. Every company member gets it.

Per client, on the account page, you can override the cadence ("this one is high-touch — every 2 weeks") or **snooze** the reminders until a date ("they asked for space until September").

What counts as contact: logging an outbound message on any of the account's conversations (see logging conversations) or recording a contact event on the account. Either resets the clock. Inbound replies don't — *them* writing *you* is not you keeping in touch.

Where overdue clients surface

- **The Clients page** (`Dashboard → Clients`) — every tracked client with last contact, last seen in your product, top features, and a days-overdue badge; sorted most-overdue first. Log a touch or snooze directly from the row.
- **The weekly digest email** — the overdue list, each row pairing *last contact* with *last seen*, so churn risks and upsell openings read differently at a glance.
- **The dashboard home** — an "Overdue clients" card linking to the filtered list.
- **A webhook** — subscribe to the `client.contact_due` event (see webhooks) to route nudges into Slack or your own tools. It fires once per overdue episode, not once per day — it re-arms only after someone actually makes contact.

Reading a client's pulse

Each account's detail page carries a **Pulse** card: first and last seen, activity over the last 90 days, and which features the client uses most. Use it before every check-in call — "I saw you've been using the CSV export a lot" is a much better opener than "just checking in".

Install LeadHunter as an app

LeadHunter is an installable web app (PWA). Installing it gives you a dedicated window with its own icon — no browser tabs to lose — and faster loads, because the app shell is cached on your device. Your data still comes from the server, so you need a connection to work with accounts and campaigns.

Install on desktop (Chrome, Edge, Brave)

1. Open your LeadHunter dashboard in the browser.
2. Look for the **install icon** in the address bar (a monitor with a down arrow), or open the browser menu and choose **Install LeadHunter…** / **Save and share → Install page as app**.
3. Confirm. LeadHunter opens in its own window and appears in your launcher / dock / Start menu like a native app.

Install on Android

1. Open the dashboard in Chrome.
2. Tap the ⋮ menu → **Add to Home screen** (or accept the install banner if one appears).
3. Confirm. The LeadHunter icon lands on your home screen and opens full-screen, without browser chrome.

Install on iPhone / iPad

Safari doesn't show an install prompt, but the manual route works:

1. Open the dashboard in **Safari**.
2. Tap the **Share** button → **Add to Home Screen**.
3. Confirm the name and tap **Add**.

Updates

You don't manage versions. When a new release ships, the app downloads it in the background and shows a **"New version available"** prompt — click **Reload** and you're on the latest version. If you ever suspect you're on a stale version, a normal page reload fetches the newest one.

Good to know

- The installed app covers the **dashboard** (everything under `/dashboard`). Public pages and these docs stay in your browser.
- Installation is per device and per browser profile. Installing on your laptop doesn't affect your phone.
- If the install option doesn't appear, your browser may not support PWAs (Firefox desktop, for instance) — the dashboard works exactly the same in a regular tab.

Team and companies

LeadHunter is multi-tenant by design. One user can belong to multiple **Companies**, and each company is fully isolated — no data ever crosses between them. A Berlin agency running outbound for ten clients runs ten companies; an internal team running its own outbound runs one.

The Company is the unit of *tenancy*: accounts, products, campaigns, custom fields, saved filters, conversations, expenses, dashboards — everything you operate on is scoped to the active company.

The Company switcher

The **Company switcher** in the dashboard top bar selects the active company. Every page you see — accounts, campaigns, statistics, stats by product — is filtered to that choice. The selection **persists across sessions** via local storage, so reloading the page or coming back tomorrow keeps you on the same company.

Three places in the dashboard surface multi-company context:

- The **switcher itself** — quick toggle between companies you belong to.
- **All companies overview** (`/dashboard/companies`) — cross-company dashboard funnel; reads accounts and outcomes from every company you belong to. Picking a company tile sets the active company and routes you to its single-company dashboard.
- **Companies list** (`/dashboard/projects/`) — full list with creation, archiving, member counts.

Creating a new company

From **Company switcher** → **New company**, or via **Companies** → **New**:

- Give it a name (e.g. *"Acme Outbound"*, *"Berlin Clients — BikeCo"*).
- The slug is generated automatically from the name; collisions get suffixed.
- The user creating the company becomes its first **owner**.

There's no soft limit on how many companies a user can belong to or create. There's no hard cross-company quota — each company has its own usage tracking (LLM tokens, Maps calls, Tavily searches, user-entered expenses), and you read the cost per company under Usage and costs.

Roles

Three roles, ordered by capability:

Role	Data read/write	Add/remove teammates	Change roles	Delete the company
Member	✓	✗	✗	✗
Admin	✓	✓	✓ (members + admins)	✗
Owner	✓	✓	✓ (everyone)	✓

Everyone in the company sees and edits the same data — there's no per-account or per-campaign permission. Roles control **team management**, not data access.

A guard prevents the **last owner** from being removed or demoted — the company would otherwise become un-administrable. Add a second owner before transferring or leaving the role.

Adding a teammate

Team management is self-serve: open **Settings → Team** (per company).

- **Invite by email.** Enter the teammate's email and a role. If they already have a LeadHunter account, they're added to the company immediately; otherwise they receive an invitation email pointing at signup, and the moment they sign up **with that email** the company attaches automatically — nothing else to click.
- Invitations expire after **14 days** and can be revoked from the pending-invites list. Re-inviting the same email replaces the earlier pending invitation.
- **Roles and removal.** Owners change roles and remove members from the same page; anyone can leave a company themselves. The last-owner guard applies everywhere — add a second owner before stepping back.

Only **owners** can invite, change roles, or remove others.

Revoking access

Removing a teammate (from Settings → Team or via the API):

- **Does not** delete the user's account or their data on any *other* company they belong to.
- **Does not** delete any accounts, campaigns, or messages *this* company stores — those belong to the company, not the user.
- **Does** revoke the user's access to this company immediately. Their Company switcher loses the entry on their next page load.

There's no offline grace period or token revocation delay.

Data isolation — what crosses, what doesn't

Surface	Scope
Accounts, contacts, status history	**Per-company.** Never crosses.
Campaigns, conversations, expenses	Per-company.
Custom fields and their definitions	Per-company.
Saved filters	Per-company (+ optionally private per-user — see below).
Research records, API usage attribution	Per-company.
Dedupe stack	Per-company. The same business can exist in two companies as two separate rows; LeadHunter doesn't merge them.
Score reuse	Per-(Product, Account). Since both live in one company, scores never cross.

If you're running an agency, each client's accounts are isolated from every other client's. You can't accidentally pull a competitor's contact list into another client's campaign.

Per-user vs per-company

A few things live on the **User**, not the Company — so they follow you across every company you belong to:

- **Communication language** — the default language you write in. Read by the message-language fallback chain when no other override applies. Independent of the dashboard UI language.
- **Dashboard UI language** — what the dashboard chrome is in.
- **Recent-entities list** — the *Recent* section in the command palette. Tracks what *you* opened, not what others did.

Everything else — saved filters, custom field definitions, accounts, campaigns, conversations — lives on the Company.

Saved filters: shared vs private

Saved filters are an exception worth knowing about. Each filter is either:

- **Shared** (`owner` null) — visible and usable by every member of the company. Use these for canonical lists everyone references.
- **Private** (`owner` set) — visible only to the user who created it. Useful for scratch filters or experiments before sharing.

You can flip a filter between private and shared at any time if you own it. The same name can exist once as a shared filter and once as your own private filter — they don't collide on `(project, owner, name)`.

Common patterns

Pattern	Setup
Solo operator	One company, you as owner.
Internal team	One company, you as owner, teammates as members. Add a second owner before going on holiday.
Agency	One company per client. Each client's company has you (the agency) as a member or admin; the client themselves may or may not have a login. Sharing accounts between clients is impossible by design — that's the point.
Mixed	Some teammates belong to multiple companies; e.g. you have your own outbound *and* run a couple of clients. The switcher and the *All companies overview* are built for this case.

Common pitfalls

- **Expecting an email-invitation flow.** Today, team management is API-only and requires the teammate to be a registered LeadHunter user already. Plan around that.
- **Forgetting which company is active.** Every page is scoped to the switcher. If your campaigns list is empty when you expected dozens, glance at the top bar before assuming the data's gone.
- **Sharing private saved filters by accident.** Filters default to private; flipping to shared is intentional. The reverse — shared → private — disappears the filter for everyone except you, which is occasionally what you want and occasionally a surprise.
- **Trying to remove the last owner.** The API refuses with a `400`. Add a second owner first, then demote or remove.

Read next

- Company and Product — what lives inside a company and why ICP attaches to products rather than companies.
- Usage and costs — per-company usage attribution and the auto-tracked vs user-entered cost split.
- Build a saved filter — including the shared vs private distinction.

Usage and costs

A LeadHunter campaign costs money in two completely different ways, and the docs split them along that line:

- **Auto-tracked API costs** — every LLM call, Google Maps lookup, Tavily search, etc. that the platform fires on your behalf. LeadHunter records these for you; you don't enter them.
- **User-entered campaign expenses** — Adwords spend, agency fees, internal labor hours, software subscriptions, content production, event costs. *You* log these against the campaign that's paying for them.

Both attribute back to the campaign that triggered them, so the full picture is *auto-tracked API costs + user-entered expenses = real campaign cost*. Both surfaces show on the campaign detail page side-by-side. Cost-of-acquisition (CAC) is computed from the user-entered side only — auto-tracked API spend is reported alongside but kept separate (it's in USD, your expenses may not be, and we don't FX-convert).

Auto-tracked API costs

Every external call the platform makes is logged as a usage row with an attributed cost. Ten providers are tracked out of the box:

Provider	Category	What it's used for
Google Maps Places API	Discovery	Text searches and place-detail lookups
Tavily AI Search	Research	Web search during AI deep research
Apollo.io	Enrichment	Contact discovery (when integrated)
OpenAI GPT-4o	LLM	ICP generation, scoring, drafting
OpenAI GPT-4o Mini	LLM	Cheaper drafting and translation
Google Gemini Flash	LLM	Default scoring model — fast + cheap
Google Gemini Pro	LLM	Higher-quality reasoning on demand
Anthropic Claude Opus	LLM	Highest-tier reasoning
Anthropic Claude Sonnet	LLM	Mid-tier reasoning
Anthropic Claude Haiku	LLM	Fastest Anthropic tier

The provider list lives in seeded data and grows over time. New providers added to the platform get their own rows; the API exposes them at `GET /api/api-providers/`.

What gets recorded per call

Each usage row carries:

- **Provider** — which service the call went to.

- **Model name** — for LLM calls, the exact model (e.g. `gemini-3.1-flash-lite`, `gpt-4o-mini`) so a campaign's spend can be split across the models that ran it.
- **Operation type** — what kind of call it was (e.g. `completion`, `places_search`, `details`).
- **Project, research, campaign, account** — four-way attribution. Per-account is the lever for *"how much did this account cost to enrich?"*; per-campaign is the lever for *"what's this campaign actually burning?"*.
- **API calls + input/output tokens** — the volume that produced the cost.
- **Cost** in USD, computed from the provider's pricing model (per-call or per-token).
- **Langfuse trace id** — when LLM observability is on, the trace id links the cost row back to the prompt/response in Langfuse.

Costs are also rolled up nightly for fast trend queries — that's what powers the daily-trend chart on the cost endpoints.

Where to read the API-cost numbers

- **Per campaign** — the **Costs & expenses** panel on every campaign detail page shows the auto-tracked LLM total in USD with a per-model breakdown, side-by-side with your user-entered expenses. The top models by spend appear with their call count.
- **Per account** — the auto-tracked LLM cost for one account (its website discovery, AI enrichment, single-lead rescores) is exposed on the account detail payload and visible on the account drawer.
- **Per research run** — the **Tasks** page (sidebar → Tasks) has a **Cost** column showing the USD spend and total tokens used by each run (`$0.0234 · 12.4k tok`), with a per-model breakdown in the cell tooltip. Sort by Cost desc to spot the run that cost ten times the rest. This is the same data the **Cost** field on the Research detail payload exposes via API.
- **Whole company, one page** — the **Usage** page (sidebar → Usage) is the unified view: this-month / last-30-days / all-time totals, a 90-day daily cost trend, 30-day breakdowns by model and by provider, the top campaigns and tasks by AI spend (each row links through), and your user-entered campaign + channel expense totals per currency side by side with the auto-tracked spend.

User-entered campaign expenses

Distinct from auto-tracked API costs. These are the things LeadHunter has no way of knowing about — ad spend on platforms you pay directly, agency invoices, the hours your team puts in. You log them as line items against a specific campaign.

Ten kinds: `adwords`, `meta_ads`, `linkedin_ads`, `other_ads`, `agency`, `labor`, `software`, `content`, `event`, `other`. Multi-currency aware (no FX conversion); CAC is only computed when a campaign's expenses share one currency.

The mechanics — the inline panel on the campaign detail page, the 💸 quick-add on the campaigns list, the Cost column on the stats page — live in their own guide: see Track campaign costs and CAC.

Four-way attribution

Both surfaces share an attribution model so the underlying data can always answer *"what did this cost?"* — per campaign, per account, per research record, per provider. The campaign detail page covers the

campaign-level slice; the account detail covers the per-account enrichment slice; deeper breakdowns (per-provider, per-research, cross-campaign) ship with the Usage page.

Budget alerts

Set a **monthly API budget** per company on the Usage page: a progress bar tracks month-to-date spend against it, and LeadHunter emails every team member when spend crosses your **alert threshold** (default 80%) and again at **100%** — once per level per month, re-arming automatically on the 1st. The budget is an alert, not a cut-off: nothing stops running when you cross it (quota *enforcement* remains on the roadmap below).

Plan limits and billing

Today, **LeadHunter doesn't block operations when usage hits a threshold** — there's no built-in quota enforcement layer. Heavy usage just produces a bigger bill from the underlying providers. If you're running LeadHunter on your own provider credentials (Gemini key, Maps key, Tavily key, …), you'll see the actual usage on those provider dashboards too.

Plan / quota / billing-period enforcement at the LeadHunter level is on the roadmap. Until it lands, the per-provider dashboards on Google Cloud / OpenAI / Anthropic / Tavily remain the source of truth for *"am I about to overshoot my own budget?"*.

Keeping costs down

Five concrete patterns:

1. **Score reuse across campaigns.** The fit score lives on the **(Product, Account)** pair, not on the CampaignAccount. Adding the same account to a second campaign of the same product **reuses the existing score** — zero Gemini cost, instant. Don't create per-campaign products unless you genuinely need different ICPs.
2. **Auto-enrichment short-circuits.** New accounts get website-discovery + scraping for free *only when they arrive without those fields*. Imports that ship a populated `website` + `website_content` skip the enrichment job — the auto-enrichment signal checks for this. Pre-populating the import file saves quota.
3. **Dry-run before bulk imports.** Set `dry_run=true` on `/import_mapped/`. It runs every dedupe check and shows you the projected merge / create / fuzzy-candidate counts without spending a Gemini call on extraction. Stops the *"I just discovered the wrong column mapping after spending €40 on enrichment"* problem.
4. **Use saved filters over fresh Google Maps sweeps.** Once your database has the bike shops in Berlin, building a saved filter that finds them is *free*. Re-running the original Google Maps query a month later only catches *new* shops — but it also re-pays the place-details API cost for the ones it merges into existing rows. If you don't expect a lot of new places, prefer the filter.
5. **Specific Product descriptions.** A sharp Product description + good example URLs means the first batch of scoring is well-calibrated. A vague description means a lot of moderate-confidence scores, more manual overrides, more rescoring as you tighten the ICP — every rescore is fresh Gemini cost.

Common pitfalls

- **Confusing the two cost surfaces.** CAC is computed from the *user-entered* side only — the auto-tracked side is reported in USD next to it, not folded in. A campaign with €0 user-entered expenses but $40 of auto-tracked Gemini scoring will read "no CAC" but still show the $40 LLM total. For full-cost analysis, eyeball both cards together.
- **Forgetting to log user-entered spend.** Auto-tracked costs accumulate by themselves. Adwords invoices and agency fees only show up in the CAC math if *you* log them. Make logging spend a weekly habit.
- **Re-running the same Google Maps search.** Each re-run hits the place-details API for every result, even ones that merge into existing rows. Slightly cheaper than the first run (since the dedupe stack short-circuits the *creation* path) but not free.
- **Expecting plan limits to stop you.** They won't, today. Watch the per-provider dashboards on Google Cloud / OpenAI / Anthropic.

Read next

- Track campaign costs and CAC — the user-entered side, in detail, with worked examples.
- Campaign → Costs and CAC — how the two surfaces combine into per-campaign cost.
- Research — the audit-trail records that auto-tracked API cost attaches to.
- ICP and scoring — score caching and when to rescore (the biggest LLM-cost lever).

FAQ

Grouped by topic. If your question isn't here, check How LeadHunter works for the bigger picture, or reach out to support.

About the product

What is LeadHunter for?

It's an outbound research + outreach platform for small teams running their own go-to-market. You bring a product and a rough idea of who you want to sell to; LeadHunter turns that into a scored database of fit accounts, runs the AI scoring against your ICP, helps you draft and translate outbound messages, and keeps the per-account conversation log so the team stays in sync. Read Welcome for the long version.

Does LeadHunter send messages for me?

No. LeadHunter is a **communication log**, not a sending platform. You keep talking to accounts through your normal channels — email, LinkedIn, Instagram, WhatsApp, phone — and paste the exchanges back into LeadHunter so the AI, the scoring, the dashboard funnel, and the team have full context. The product is intentionally out of the sending path. See Messaging.

How does this differ from Apollo / ZoomInfo / Clay?

Apollo and ZoomInfo are primarily **contact databases** — you query an existing index of millions of companies and people. LeadHunter is the layer on top of that work: you bring (or discover) your own accounts, and LeadHunter scores them against *your specific* product's ICP, drives the daily outreach loop, and measures CAC. Clay is closer in shape but skews heavily towards programmatic data pipelines; LeadHunter skews towards small teams running outbound by hand with strong AI assistance. The three are complementary more than competitive.

Why is the data called *Account* but the product is called *LeadHunter*?

The product is named for the **verb** — *hunting leads*, *finding leads*, *lead generation* is the discovery activity. The **noun** that gets stored is an *Account* (one row per organisation, neutral about whether it's a prospect, customer, partner, press, supplier, etc.). Calling the row "Lead" baked a sales-only assumption into a model that needs to span every kind of relationship. The product kept its name because the verb-phrase is still accurate.

Is LeadHunter hosted or self-hosted?

Both flavours exist. The **hosted product** at `leadhunter.humans2agents.com` is managed for you — provider keys (Gemini, Google Maps, Tavily) are configured by the platform, costs flow through your plan, infrastructure is handled. **Self-hosted** deployments run the same Django + Next.js + Postgres stack on your own infrastructure, with your own provider API keys (set via environment variables). Self-hosted users own their own data and quota; hosted users get zero-config plus support.

Accounts and data

What happens when I try to create an account that already exists?

LeadHunter merges instead of duplicating. The five-level dedupe stack runs on every account creation; if any level matches, the new data is merged into the existing row (and surfaced for review only for the lowest-confidence fuzzy matches). Nothing is silently overwritten — every unique field from every duplicate is preserved on the survivor, and `merge_history` records the audit trail. See Merge duplicates.

Can I undo a merge?

Not automatically. The absorbed accounts are deleted at the end of a merge; `merge_history` records snapshots of them, so you can read *what* was on each side, but restoring them is a manual rebuild. Bias toward spot-checking AI-suggested merges before bulk-approving large lists.

Can I move data between companies?

Not directly. Companies are isolated tenant boundaries by design. If you really need to move data, export from the source company (CSV) and re-import into the target — the dedupe machinery handles the rest. Accounts, custom fields, saved filters, campaigns, expenses all need to come over separately.

Can I bulk-export my accounts?

Yes. From the Accounts list, you can export the visible filter as CSV. Filtering first lets you scope the export to whatever subset you need; the full database is one click away when no filter is applied.

What happens to my data if I cancel?

Cancelling closes the account but doesn't immediately purge the data. Contact support to request a final export and a confirmed deletion timeline; both are on the operational roadmap as self-serve actions but aren't first-class UI buttons yet.

What happens to a deleted custom field's values?

The schema is removed (the field stops showing on account forms and the import wizard refuses new mappings for it), but **existing values stay on the underlying account rows**. Re-creating a field with the same key brings the values back into view. The trade-off is deliberate: cheap to re-add a field, expensive to recover deleted history. If you genuinely want the values gone, ask support for a one-off cleanup. See Custom fields → Adding, renaming, deleting.

ICP and scoring

How do I rescore everything against an updated ICP?

Open the Product, edit the ICP, click Save. Then use **Score campaigns** to re-run scoring on every account in every campaign of that product. The score lives per-(Product, Account), so one rescore propagates to every campaign sharing that product. See ICP and scoring → When to rescore.

Can I score remaining accounts on all my campaigns at once?

Yes. The **Campaigns** list page (`/dashboard/campaigns`) has a **Score remaining (N)** button in the top-right action bar — `N` is the number of active campaigns that have unscored accounts and a target persona set. Clicking it dispatches AI scoring on every eligible campaign in one shot.

The button is intentionally **conservative**:

- Only **non-archived** campaigns are touched.
- Campaigns without a target persona yet are skipped (you have to define the persona before scoring can run).
- Campaigns whose accounts are *all* already scored are skipped.
- Campaigns mid-scoring (one is already running) are skipped.
- Archived campaigns are completely excluded.

A confirmation dialog lists how many campaigns will be scored, then a summary alert reports what actually happened — including the reason for every skip ("no target persona yet", "all accounts already scored", "already scoring", etc.).

Already-scored accounts are **never re-scored** by this button (the AI score is cached per product + goal). To re-score everything after editing a persona or brief, use the per-campaign **Give feedback & rescore** action on the Score review page instead.

How long does scoring take?

Scoring runs on Gemini Flash by default — fast and cheap. A batch of a few hundred accounts typically finishes in **5–15 minutes**. The campaign page polls the scoring status and updates the score column as rows come back, so you can keep working while it runs. Per-account cost is a fraction of a cent.

My scoring run looks stuck — how do I unwedge it?

The scoring review page hides its **Score N remaining** and **Give feedback & rescore** buttons while a run is in progress so you can't double-spend AI quota. If a run wedges (rare — a deploy interrupted mid-batch, the model API stalled, a worker recycled), you'll see the "AI scoring in progress…" banner stay frozen.

The scoring page now carries a **Cancel run** button in the header while a run is active. Clicking it confirms and then flips the run back to a clean state — the **Score N remaining** button reappears immediately so you can retry. Already-scored accounts aren't re-charged (the AI score is cached per `(product, goal)` so retry only picks up the unscored tail).

Cancel is not instant — the worker stops at the next safe checkpoint, never mid-LLM-call. Typical latency:

- **Scoring a batch**: ~10–30 seconds (one batch of ~10 accounts finishes).
- **Scraping account websites** (the "Preparing accounts" phase): ~15 seconds (one in-flight HTTP request finishes).
- **AI-filling missing business fields** (rare deep-fill phase): up to ~3 minutes worst case (one stuck Gemini call's timeout).

Whatever scores or scraped content the worker had already produced land normally — they're cached, so a retry skips them.

If a scoring run wedges in a way the cancel button can't reach (an underlying worker crash that prevented any state from being written), an hourly safety-net sweep on the server flips wedged runs back to "failed" automatically. Two detectors run together:

- **Fast detector** — a run stuck "in progress" for more than 30 minutes with **zero accounts processed** is unambiguously wedged in the enrichment phase. Flipped at the next hourly tick.
- **Hard backstop** — any run stuck "in progress" for more than 2 hours regardless of progress, in case a slow-grind wedge IS writing progress but won't ever finish.

The sweep also runs immediately on every server restart, so a deploy clears wedged runs without delay. After it flips the row, the **Score N remaining** button reappears for retry.

Same action is also available via API for scripted recovery:

```
POST /api/campaigns/{id}/cancel-scoring/
```

Idempotent: running it on a campaign that isn't actively scoring is a safe no-op.

What does *Deep research* actually do?

Deep research goes beyond the standard *Quick* mode (which scrapes the home page and extracts the business name + sector + language). It additionally **crawls the about / team / staff / contact pages** of the website and runs an LLM extraction to pull out 1–3 decision-maker contacts (name, title, sometimes email). Use it for high-value accounts where you'd otherwise hand-research the buying group; the cost per run is meaningfully higher than Quick so it's not the default.

Why are my first batch of scores reading vague?

Almost always one of three causes:

1. **The Product description is too thin.** One line of positioning isn't enough to anchor a sharp ICP. Two paragraphs is the minimum.
2. **No example URLs.** The 2–3 *good* and *bad* example URLs on the Product are the single biggest scoring lever after the ICP itself. Skipping them produces moderate-confidence scores with weak reasons.
3. **No website on the account.** Phone-only or name-only accounts can be scored, but the model is leaning on structured fields alone — scores cluster in the moderate range, reasons stay vague. Let auto-enrichment discover the website, or trigger **Deep research** from the account detail page.

Will scoring improve as I use LeadHunter more?

Yes — every approve or reject you mark in a campaign's review queue is a calibration signal. Once a Product crosses ~10 reviewed accounts, your verdicts start feeding back into the scoring prompt as concrete *"what good looks like here"* examples, and subsequent batches get sharper. The signal is product-scoped — approvals on Product A don't leak into Product B's scoring. See ICP and scoring → The feedback loop.

What languages can the AI write in?

Any of the major languages the underlying LLM supports — English, Spanish, Catalan, French, German, Italian, Portuguese, Dutch, Polish, and many more. Set the account's `language` field to the language they

read (or let the campaign default win); the AI drafts outbound and translates inbound into your reading language. See Messaging → Language resolution.

Why are the labels `excellent` / `moderate` / `mismatch` instead of `great` / `good` / `bad`?

The labels reflect *fit confidence*, not desirability. `mismatch` means *"the AI thinks this account doesn't match this product's ICP"* — which is useful information even if the account is a great business. The category labels stay neutral on quality so the score is about predicting outcome, not making a value judgement.

Outreach and inbound

How do I track an account that came in through Google Ads?

Set the **Acquisition channel** to `adwords` when you create the account (or after the fact, from the account detail page). Channel-specific data — UTM parameters, the `gclid`, the ad campaign id — goes in the **Acquisition metadata** block. Inbound channels auto-promote the account from `prospect` to `contacted` because the lead has already reached out to you. See Track inbound leads.

Can a single account be in multiple campaigns?

Yes. Same account in three campaigns of the same product gets **scored once** (the score lives on the Product–Account pair); per-campaign workflow (review status, outreach status, override score, chosen contact) lives on the CampaignAccount join row. Same account in campaigns of *different* products gets scored once per (Product, Account) pair.

Can one campaign target multiple products?

No — each campaign points at exactly one Product. If you need to push two products to the same audience, run two campaigns. The constraint exists because the score is per-(Product, Account), and a campaign that mixed products wouldn't have a coherent score column.

What does the dashboard funnel actually count?

Distinct **accounts**, not messages. A campaign that sends three follow-ups to the same account counts as one *initiated*, not three. Responded counts the account once on its first inbound; closed reads the account's status-history for any transition to `customer`. Cohort-attributed back to the initiation date — see How LeadHunter works → Phase 5.

Costs and CAC

What's the difference between *API costs* and *Campaign expenses*?

Two completely different cost surfaces:

- **API costs** are **auto-tracked** — every LLM call, Google Maps lookup, Tavily search the platform fires on your behalf. You don't enter these; LeadHunter records them.
- **Campaign expenses** are **user-entered** — Adwords spend, agency fees, labor hours, software, content, events. *You* log them against the campaign they're paying for.

Together they give the full cost of running a campaign. CAC on the *Stats by product and campaign* page is computed from the user-entered side. See Usage and costs and Track campaign costs and CAC.

Will LeadHunter stop running operations if I hit a usage limit?

No — there's no built-in quota enforcement layer today. Heavy usage just produces a bigger bill from the underlying providers. Plan-level enforcement is on the roadmap; until it ships, watch your provider dashboards (Google Cloud, OpenAI, etc.) for the actual quota surface.

Can I run LeadHunter against my own API keys?

For self-hosted deployments, yes — the platform reads provider keys (Gemini, Google Maps, Tavily, etc.) from environment variables. For the hosted product, keys are managed by LeadHunter and costs flow through your plan. Contact support if you're considering BYO keys for a hosted setup.

Why is the Cost column on my campaigns list empty?

Two possibilities. Either no **expenses have been logged** for that campaign yet (the Cost column reflects user-entered campaign expenses — Adwords, agency, hours, etc.), or expenses *have* been logged but in **mixed currencies**. The headline column shows the primary currency total; a small `+` next to it indicates other currencies are present. To log expenses, open the campaign and use the *Costs & expenses* panel, or click the 💸 icon directly on the row in the campaigns list. See Track campaign costs and CAC.

Why is my campaign's CAC showing as null?

Two common cases. Either the campaign has **expenses in multiple currencies** (LeadHunter doesn't carry an FX layer, so CAC isn't computed across currencies — you'll see a per-currency total and a *cac_unavailable_reason* explanation), or no outreach has happened yet (CAC = total ÷ count; the denominator is zero). Normalise the campaign to one currency and log at least one outbound message; the CAC math will appear.

Operational

What's the difference between *archive* and *delete*?

Archive soft-hides an item (Product, Campaign, or Account) from default lists but keeps every conversation, score, expense, and audit-trail entry intact. Use it for *"completed but I want the history"* — or, on an Account, *"this business closed / merged / pivoted out, but I want the relationship history."* Accounts can also carry a free-text **archive reason** that shows up on the account header and audit trail.

Delete is permanent — Products are protected if any campaign references them (archive instead), Campaigns require typing the exact name to confirm, and Accounts require typing the exact account name to confirm and cascade to every related campaign row, score, and conversation. Underlying Accounts are never affected by *campaign* delete.

How do team and roles work?

Three roles — `owner`, `admin`, `member` — all with the **same data access** but different team-management capabilities. Owners can delete the Company and remove other owners (with a last-owner guard); admins can add/remove teammates and change roles for members and admins; members read and write all the

Company's data but can't touch team config. Adding teammates is currently **support-driven** (no self-serve invite UI yet). See Team and companies.

Where do I see who did what?

Several audit-trail surfaces:

- **Account status history** — every status change (manual or auto-promoted) records who triggered it, when, with what reason and source. Visible on the account detail page.
- **Conversation log** — every message records `created_by` and timestamp. Threads are visible per-campaign and on the account detail page.
- **Merge history** — the survivor of any merge carries `merge_history` JSON with snapshots of the absorbed rows, who ran the merge, when, and the field-winner decisions.
- **Research records** — `created_by` + `executed_by` per record; the dashboard's Research page is filterable by user.

No single unified "team activity" view today — different audit trails live on the entities they describe.

Can two teammates edit the same account at the same time?

Yes — there's no row-level locking, and the last save wins. In practice this is rarely a problem because the dashboard typically shows different teammates working on different campaigns or different review queues. If you do hit a conflict (e.g. one person changes status while another edits notes), the second save overwrites — refresh and merge manually.

Where do I report bugs or request features?

GitHub Issues or your support contact. Bugs with clear reproduction steps and feature requests with a concrete use-case get the fastest response.