NOVA • SELF-SERVE DISCOVERY

How we built self-serve discovery.

The learnings, the architecture, the features, and the load-bearing code behind Nova's conversational discovery and Read — the engine that turns a person's *actual corpus* into brand strategy without a strategist in the room.

PRODUCT	DISCOVERY	AUTHOR	DATE
ArcBase / Nova	v3 · Conversational Moves	Eli Wright	June 2026

ORIENTATION

What this is, and how to read it

This is the working record of how Nova's self-serve discovery actually got built — not the pitch version. It captures what we learned the hard way, the system shape we settled on, and the code that carries the weight. If you're implementing, extending, or debugging any part of the discovery-to-strategy pipeline, start here.

THE ONE THING TO TAKE AWAY

Discovery works when it's built around **conversational moves**, not brand-strategy slot-filling. We stopped asking people for their content pillars and started running the eight moves Courtney used to pull gold out of live calls. Everything downstream — the Read, the generators, the feedback loop — exists to serve that one decision.

CONTENTS

01	The core learning	Why moves beat slot-filling
02	System at a glance	The five-stage pipeline
03	Data model & functions	Tables, edge functions, RPCs
04	The discovery engine	Phases, moves, captures, probes
05	The eleven questions	The full sequence and what each feeds
06	Phase 0 — corpus lift	Pre-filling from connectors
07	Nova's Read	Extraction, fingerprints, citations
08	The shared context loader	One snapshot, every generator
09	The feedback loop	Ratings → aggregates → tuning
10	How to build it	Recommended order of operations
11	Open questions & next	What's unresolved

01 The core learning

The first version asked people brand-strategy questions directly. "What are your content pillars?" "Who's your audience?" "What's your differentiation?" It failed quietly. People gave us the answers they thought we wanted — polished, generic, lifted from their own LinkedIn headline. The output read like everyone else's.

The fix came from re-reading transcripts. Courtney's discovery calls with Steve and Jennifer didn't sound like a strategist filling in a template. They sounded like a good conversation that happened to extract gold. We isolated **eight repeatable moves** she made — the things that consistently produced the verbatim phrases, the coined methods, the tension pairs we ended up building whole brands on. v3 is those moves, codified.

WHAT WE STOPPED DOING

- **Adjective shopping.** "Would you say witty? Formal?" People just agree with the nicest word.
- **Strategist homework.** Asking the client to do the strategist's job — pillars, positioning, avatars.
- **Catch-alls.** "Any other stories?" gets nothing.
- **Label tests.** "Do you identify with X?" invites a yes.
- **Vague provocation.** "Tell me about a time you challenged conventional wisdom."

WHAT WE DO INSTEAD

- **Warm before pressure.** Questions are ordered by temperature. The contrarian layer only opens after three warm-up passes.
- **Capture verbatim.** When someone coins a phrase, we keep it word-for-word and don't follow up. That's the gold.
- **Probe only when thin.** Each question fires at most a few follow-ups, and stops the moment the answer is sufficient.
- **Name what they already do.** If they describe a sequence, we offer to name the method — we don't ask them to invent one.
- **Use both valences.** What grates on them maps to what their audience finds grating too.

> **The design constraint that shaped everything**
>
> A self-serve product has no strategist to read the room. So the room-reading had to be encoded: *when to push, when to shut up, when a phrase is already gold.* That's why the engine carries explicit capture rules, probe rules with stop conditions, and a hard list of anti-patterns it is forbidden to generate. The intelligence isn't in asking more — it's in knowing when to stop.

02 System at a glance

Five stages. The corpus comes in, gets read, gets shaped by the conversation, gets synthesized into one brand snapshot, and every generator downstream pulls from that same snapshot. A feedback loop runs underneath the whole thing.

STAGE 0
Corpus lift
Connectors pull the person's real material — email, LinkedIn, site, calls, drive — into `raw_connector_payloads`.

→

STAGE 1
Pre-fill
Phase 0 engine drafts answers from the corpus so discovery starts warm, not blank.

STAGE 2
Discovery
The 11-question conversational flow. Adaptive follow-ups, verbatim capture, stop conditions.

→

STAGE 3
Nova's Read
Extraction passes synthesize a voice fingerprint + signals into one `brand_dna` record.

STAGE 4
Generators
Content, visual, and strategy generators all read the same context via one shared loader and inject it as a prompt preamble.

←

UNDERNEATH
Feedback loop
Per-item ratings on the Read feed cross-client aggregates that flag which passes need tuning.

Why the snapshot matters

Before the shared loader, each generator built its own view of the client and they drifted. A content post would use one set of tone words, a visual kit another. The fix was a single `loadNovaContext()` that every generator calls. One source of truth, cached per worker, with a confidence band attached so a generator knows how much to trust what it got.

THE TWO HALVES OF DISCOVERY

STAGE	PHASES	WHAT IT GETS AT
`Identity` · Stage 1	Warm Up · What You Do · Contrarian	Who you are and what you believe. The arc, the party line, the POV you'd defend.
`Application` · Stage 2	Live Capture · Voice & Audience · Container & Close	How the work shows up in the world. Live thinking, who needs it, what you'll decline.

03 Data model & functions

Everything lives in Supabase. The shape is deliberately flat: raw material in one table keyed by source type, derived artifacts in their own tables, and audit rows that let us see what the system did and why.

CORE TABLES

TABLE	HOLDS
`raw_connector_payloads`	Every pulled item, tagged with `source_type`, title, metadata, and `pulled_at`. The corpus.
`extracted_signals`	Per-pass output keyed by `pass_type` (voice / theme / audience / framework / goals), with payload, source doc IDs, status, confidence.
`voice_fingerprints`	Tone descriptors, sentence stats, vocabulary signature, openings, closings, and signature quotes. Newest wins.
`brand_dna`	The synthesized snapshot: `profile_data`, source count, last-synthesized timestamp, visual kit status.
`discovery_responses`	The conversation. Each answer's extracted `record` keyed to capture targets.
`discovery_briefs`	Completed discovery output. Presence flips `hasBriefs`.
`nova_read_feedback`	Per-item thumbs, unique on `(client_id, section, item_key)`, optional note.
`connector_audit_log`	What happened: extracted, synthesized, context used, context partial.

EDGE FUNCTIONS

FUNCTION	JOB
`discovery-phase-zero-prefill`	Drafts answers from the corpus before the person starts.
`discovery-adaptive-followup`	Decides whether to probe, which probe, or to stop.
`run-extraction-passes`	Runs the Read. Accepts `force:true` to re-extract.
`extract-signature-quotes`	Backfills signature quotes when the pool is thin.
`reject-signature-quote`	Soft-deletes an admin-rejected quote so all generators inherit the exclusion.

RPCS (THE ADMIN SIGNAL REVIEW)

- **`get_nova_signal_aggregates(window_days)`** — returns cross-client dispute aggregates for a 7 / 30 / 90-day window.
- **`refresh_nova_signal_aggregates()`** — recomputes the daily rollup on demand.

> **NAMING DISCIPLINE**
>
> The discovery library, the adaptive engine, and the pre-fill engine all consume one TypeScript definition so the UI and the AI stay in lockstep. If you change a capture target, change it in `nova-discovery` and the rest follows. Don't fork the shape.

04 The discovery engine

Three rule systems run on every question. **Moves** set the intent. **Capture rules** fire always, harvesting signals without interrupting. **Probe rules** fire only when the answer is thin, and stop the second it's enough.

The eight conversational moves

Each move is a way of getting at something a strategist would chase in a live call. They map one-to-one onto the primary questions.

MOVE	WHAT IT DOES
`arc_opener`	Opens on the chapters that shaped them, not the polished bio.
`sensory_drop`	Drops back into one moment for place, time, sensation.
`third_party_mirror`	Gets how others describe them — in others' words.
`party_framing`	The unpolished version of "what do you do."
`methodology_naming`	Names the method they already use but never named.
`misconception_probe`	The highest-leverage question: what they'd correct.
`phrase_under_pressure`	The personal belief not yet said out loud.
`negative_constellation`	The voice that grates — which reveals the audience.

A ninth move, `corpus_lift`, runs only in Phase 0 pre-fill — it's how the corpus seeds the conversation before a question is asked.

Capture rules — always on

Emit signals in the background. No follow-up fires.

- **GOLD** — a coined phrase or declarative one-liner. Kept verbatim, never probed.
- **TENSION_PAIR** — "X but not Y," "X without being Y."
- **MONIKER** — a self-coined label for what they do.
- **METHODOLOGY** — any described sequence becomes a name candidate.
- **VOICE_FP** — the running voice fingerprint.

Probe rules — conditional

Max three per question. Each has a trigger.

- **SENSORY** — generic answer? Ask where they were.
- **UNPOLISHED** — rehearsed? Ask the off-duty version.
- **EXAMPLE** — vague? Ask who says it wrong.
- **NAME_METHOD** — a sequence? Offer to name it.
- **FLIP_POSITIVE** — after a negative, ask who's getting it right.
- **BOTH_DIRECTIONS / DRAINS_ENERGIZES / WATCHOUT_HANDLING / TESTIMONIAL_PASTE** — situational.

ANTI-PATTERNS — THE ENGINE MUST NEVER GENERATE THESE

`adjective_shopping` `catch_all` `label_test` `strategist_homework` `vague_provocation`. Auto-generated probes are filtered against this list before they're shown.

04 · CONTINUED

Stop conditions & the question shape

A probe that won't stop is worse than no probe. Every question carries a stop condition: a max number of probes, a minimum count of distinct extracted entities, and the keys that must be present before the engine considers the answer sufficient.

```ts
// Each NovaQuestion declares how far the engine may push
interface NovaStopCondition {
  maxProbes: number;        // hard cap on follow-ups
  minEntities?: number;     // distinct extractions to be "enough"
  requires?: string[];      // keys that must exist in the record
  description: string;      // human-readable sufficiency
}
```

LIB/NOVA-DISCOVERY.TS — THE CONTRACT EVERY QUESTION FILLS

The shape of a full question ties it all together — the move, the input mode, what it captures, which rules fire, and which brand-strategy outputs it ultimately feeds:

```ts
// Q1 — the arc opener, abbreviated
{
  id: "Q1_arc",
  phase: "warm_up",
  primaryMove: "arc_opener",
  preferredInput: "voice",
  captureTargets: ["arc_moments[]", "proper_nouns[]"],
  captureRules: ["GOLD", "VOICE_FP"],
  probes: [{
    rule: "SENSORY",
    triggerWhen: "lists titles instead of moments",
    template: "Was there a moment that felt formative?",
  }],
  stopCondition: { maxProbes: 1, minEntities: 3,
                   requires: ["arc_moments"] },
  feeds: ["arc_foundations", "vision"],
  prefillFrom: ["linkedin"],
}
```

THE WHOLE ENGINE IS DATA. THE UI RENDERS IT; THE AI OBEYS IT.

Why this is data, not prompt

Keeping the question definitions as a typed structure — rather than baking them into a prompt — means the front-end progress bar, the adaptive follow-up engine, and the pre-fill engine all read the same source. Change a stop condition once and three systems update. It also means we can lint probes against the anti-pattern list before they ever reach a person.

05 The eleven questions

Ordered by warming temperature. Low-stakes first, the pressure section in the middle once they're talking naturally, a light landing at the end. Q8 onward each pair a primary prompt with a second prompt on the same screen.

#	PHASE / MOVE	THE QUESTION	FEEDS
1	`warm_up` arc_opener	Walk me through the moments that brought you to where you are now.	arc · vision
2	`warm_up` sensory_drop	Pick one of those moments and take me back into it. Where were you?	arc · vision
3	`warm_up` third_party_mirror	How do people who've worked with you closely describe you, in their words?	voice · avatar · traits
4	`what_you_do` party_framing	What do you currently say when someone asks what you do at a party?	moniker · offering
5	`what_you_do` methodology_naming	Walk me through a Tuesday morning, from coffee to dinner.	method · mission · services
6	`contrarian` misconception_probe	What's the misconception in your field you most want to correct?	ownable msg · key msg 1
7	`contrarian` phrase_under_pressure	What do you believe that you haven't said publicly yet?	ownable · key msg 2/3
8	`live_capture` phrase_under_pressure	What's on your mind this week? + 3–5 phrases you say often.	key msgs · voice · mission
9	`voice_audience` negative_constellation	Name a voice in your industry that grates. + Who needs your work now?	voice · avatar · comparison
10	`container_close` negative_constellation	Which format energizes you, which would you decline? + Watch-outs.	offering · services · watch-outs
11	`container_close` third_party_mirror	Five years out, how do they introduce you? + 3 brands like, 1 unlike.	impact · vision · comparison

The hinge question

Q6, the misconception probe, is flagged in the library as the highest-leverage question in the whole discovery. A concrete misconception inverts cleanly into a thesis. Voice memo strongly preferred — we want the rant.

The codified flip

On Q9, `FLIP_POSITIVE` always fires after the negative answer. It's not optional. What grates reveals taste; who's getting it right reveals the aspiration. Both feed the comparison set.

06 Phase 0 — corpus lift

A blank discovery form is a wall. The person stares at "walk me through the moments that shaped you" and freezes. Phase 0 reads the corpus first and drafts a starting point, so the person edits rather than invents.

SOURCE TYPES WE PULL AND HOW WE COUNT THEM

Everything lands in `raw_connector_payloads` tagged by source type. We count per type with parallel head-counts to stay under the 1000-row cap, then sum to a total.

`gmail_sent` `gmail_inbox` `gcal_event` `gdrive_file` `fathom` `linkedin_post` `linkedin_profile` `linkedin_experience` `website_page` `instagram_profile` `instagram_post` `newsletter_post`

WHICH QUESTIONS GET PRE-FILLED FROM WHERE

Q	PRE-FILLS FROM
Q1 · arc	`linkedin`
Q3 · mirror	`testimonials, linkedin`
Q4 · party line	`website_bio, linkedin`
Q6 · misconception	`book, podcasts, newsletter`
Q7 · belief	`podcasts, book`
Q8 · live thinking	`newsletter, podcasts, linkedin`

The principle

Pre-fill seeds, it doesn't decide. A drafted answer from the corpus is a prompt for the person to correct or confirm — correction is itself signal. The contrarian and live-capture layers (Q6–Q8) lean hardest on pre-fill because that's where a person's published work already says more than they'd say cold.

TOP RECIPIENTS — A SMALL BUT USEFUL SIGNAL

From `gmail_sent` metadata we parse the `to` field, strip display names out of `"Name <email>"` form, and rank the five people they email most. It's a cheap proxy for who they actually orient their work around.

07 Nova's Read

The Read is what the person sees: "here's what we learned about your voice, your themes, your audience — and here's exactly where we got it." Trust comes from showing the receipts. Every claim cites its sources by type and count.

Source-type-aware citations

A generic "based on your data" reads like a black box. Instead we build phrases that name the material: *"From 12 sent emails and 3 LinkedIn posts."* Plural sources get counts; profile-type sources stay singular; low-signal inbox volume is suppressed unless it's meaningful.

```
// Plural sources are counted; profiles are named, not counted
gmail_sent      → "12 sent emails"
linkedin_post   → "3 LinkedIn posts"
fathom          → "4 call transcripts"
linkedin_profile→ "your LinkedIn profile"   // singular only
gmail_inbox     → suppressed unless count > 5

// Output, ordered by signal priority:
buildCitation(ids, lookup) → "From 12 sent emails and 3 LinkedIn posts"
```

LIB/NOVA-READ-CITATIONS.TS — RECEIPTS, NOT A BLACK BOX

Confidence bands

Not every Read is equally backed. The context loader assigns a band so the UI — and every generator — knows how much to lean on it.

BAND	THRESHOLD
strong	≥ 1000 sources, ≥ 4 tone descriptors, ≥ 5 signature quotes
moderate	≥ 100 sources, ≥ 2 tone descriptors
thin	has brand DNA but below moderate
none	no brand DNA yet — generators fall back to legacy paths

Signature quotes, and the backfill

Signature quotes carry a lot of the Read's weight, so we guard their supply. If a client has a real LinkedIn pool (five or more posts) but no LinkedIn quotes yet, or fewer than three quotes overall against a real sent-email pool, the Read fires a one-shot, fire-and-forget backfill to `extract-signature-quotes` and patches them in when they land. Admin-rejected quotes are soft-deleted at the source so every wired generator inherits the exclusion automatically.

08 The shared context loader

One function, `loadNovaContext()`, is the single door every Nova-aware generator walks through. It pulls the same `brand_dna` + voice fingerprint + signals snapshot and hands back a partial-but-never-broken context with metadata attached.

FIVE LOCKED DECISIONS

- **D1 — Never throws.** Missing fields come back as null with a structured `_meta` listing what's absent. A thin client still gets usable output.
- **D2 — Cache per worker.** A module-level map keyed by client ID. The worker lifetime is the natural TTL — no stale-cache plumbing.
- **D3 — Log partials once.** When context is partial, one `nova_context_partial` audit row is written. Full loads aren't logged.
- **D4 — Generators log their own use.** Each writes a `nova_context_used` row on success, so we can trace which generator used which keys.
- **D5 — Backwards compatible.** No brand DNA returns `has_brand_dna:false` and the generator falls back to its legacy prompt.

The preamble

Generators don't read the raw context object into their prompts. They call `buildNovaPreamble(ctx, keys)` and ask only for the sections they need — voice, essence, positioning, audience, themes, frameworks, goals, visual, content. It returns a compact block, or an empty string when there's nothing useful, so callers cleanly fall back.

```
// A content generator asks only for what it needs
const preamble = buildNovaPreamble(ctx, [
  "voice", "themes", "goals"
]);

// → injected verbatim ahead of the task prompt:
Brand context (from Nova):
- Voice tones: precise, plain-spoken, self-deprecating
- Signature vocabulary: orientation, drift, constraint
- Themes: complexity, uncertainty, the entropic tax
- Stated goals: thought leadership; paid speaking
```

FUNCTIONS/_SHARED/NOVA-CONTEXT.TS — ONE SNAPSHOT, SELECTIVELY INJECTED

The payoff

This is why a content post, a visual kit, and a strategy doc for the same client now sound like the same person. They're reading the same fingerprint and the same signals. When we reject a bad signature quote once, all wired generators stop using it — no per-generator cleanup.

09 The feedback loop

Self-serve means we can't sit in the room watching faces. The feedback loop is how the product tells us where the Read is wrong — at scale, across every client at once.

HOW IT WORKS

- **Per-item ratings.** Each card in the Read takes a thumbs up or down with an optional note, stored unique on `(client_id, section, item_key)`. Tapping the same rating again clears it.
- **Daily aggregation.** Ratings roll up into cross-client aggregates: distinct negative clients, negative and positive counts, last feedback time, per item.
- **Dispute surfacing.** The admin signal review ranks items by how many *distinct* clients disputed them. Anything disputed by more than one client gets flagged in vermilion.
- **Targeted tuning.** A signal disputed across clients isn't a one-off — it's a pass that needs prompt tuning. The drill-down shows the per-client notes so we know what to fix.

>1	7/30/90	per-pass
DISTINCT CLIENTS = DISPUTE	DAY WINDOWS	TUNING TARGET

The key design choice is counting **distinct clients**, not raw downvotes. One client thumbing down five items is noise — it's taste, or a bad day. Five different clients thumbing down the *same* item is signal: the pass that generated it is systematically off, and that's a prompt-level fix, not a per-client patch.

```
// The dispute test that drives the whole admin view
const isDispute = row.distinct_negative_clients > 1;
// highlighted in #F53B3B — a candidate for per-pass tuning
```

ADMINNOVASIGNALREVIEW — DISTINCT CLIENTS, NOT RAW COUNTS

Closing the loop

Read → rate → aggregate → tune the pass → re-run extraction → better Read. The same negative signal that flags a problem also points at the exact pass to fix. That's the difference between a feedback widget and a feedback *loop*.

10 How to build it

If you were standing this up from scratch, this is the order that works. Each step produces something testable before the next one depends on it.

1 · Connectors and the corpus table

Get `raw_connector_payloads` filling reliably, tagged by source type with clean metadata and `pulled_at`. Everything downstream assumes this is trustworthy. Build the parallel head-count pattern early — you'll hit the row cap otherwise.

2 · The discovery library as data

Write the typed question definitions first — phases, moves, capture targets, probes, stop conditions, feeds. This is the contract. Lint every probe against the anti-pattern list. Don't start on the UI or the AI until the shape is stable.

3 · The adaptive follow-up engine

Implement probe selection and stop conditions against the library. Test the stop logic hardest — a runaway probe loop is the worst failure mode in self-serve. Capture rules run in parallel and never block.

4 · Phase 0 pre-fill

Wire the corpus into drafted answers via `prefillFrom`. Treat corrections as signal. This is what makes discovery feel warm instead of like a form.

5 · Extraction passes and the Read

Build `run-extraction-passes` to write voice fingerprints and signals, then synthesize `brand_dna`. Ship the citation helper alongside — the Read is only trustworthy if it shows sources.

6 · The shared context loader

Stand up `loadNovaContext` and `buildNovaPreamble` before you wire a single generator. Retrofitting generators onto a shared loader is far more painful than building them on it.

7 · Generators, then the feedback loop

Point each generator at the loader. Last, add per-item feedback and the cross-client aggregates. The loop only earns its keep once there's enough Read in the wild to dispute.

11 Open questions & what's next

What's still unresolved, roughly in priority order. None of these block the core flow; all of them would make it better.

QUESTION	WHERE IT BITES
Confidence band thresholds	The 1000 / 100 source cutoffs are reasonable guesses. We should validate them against which Reads clients actually rate well.
Cache invalidation	Per-worker caching is simple but means a fresh extraction won't show until the worker recycles. Fine for now; revisit if it confuses anyone.
Probe ceiling tuning	Max-three probes is a blanket rule. Some questions (Q6) might earn more; some (Q4) fewer. The feedback loop should eventually tell us.
Inbox suppression cutoff	The ">5 inbox conversations" threshold for citation is arbitrary. Low stakes, but worth a look.
Aggregate refresh cadence	Daily rollup plus manual refresh. If dispute review becomes a daily habit, a live view might be worth the cost.

The bet this whole thing makes

That a well-designed conversation, run against a real corpus, can get most of the way to brand strategy without a strategist — and that showing the receipts is what makes people trust it. The feedback loop is the hedge: if the bet is wrong somewhere specific, the disputes will tell us exactly where.

END OF DOCUMENT

Nova · Self-Serve Discovery

ARCBOUND · ARCBASE