One Database, Many Connectors

An operating system for a small services business: a hub that never becomes a mesh, an agent desk that hands work over instead of parking it, and attribution by identity.

Michael Kaminski 2026-09-23 michael-kaminski.io/papers White paper 02

Summary

A six-person maintenance company runs on one Postgres database, a static web page and twelve serverless functions, and the constraint that no tool ever talks to another tool. The interesting parts are the rules that keep it that way under load, and the agent that works its own queue. This paper describes Transparent Maintenance OS, the internal operating system for Transparent Maintenance, a residential maintenance contractor in metro Atlanta. It covers the hub-not-mesh architecture, the agent desk and its hand-over rule, attribution by identity when lead source is empty, and three production incidents whose fixes are more useful than the features.

Sources of record: the `TransparentOrgManagement` repository, its architecture document (refreshed 2026-09-12) and 30 decision records, and pull requests #24 through #65 merged between 2026-09-12 and 2026-09-18. Hosting cost is $0 on free tiers today; the one paid line item on the roadmap is a $20/month plan because the free one forbids commercial use.

1. The model, in three sentences

A task is done once. A process repeats and spawns tasks. A role owns processes. That is the whole operating model, and nothing is ever deleted. Every other feature of the system is a view onto those three nouns or a mirror of a system that owns some other fact.

The shaping constraint was cost: the system must stay a thin, zero-hosting layer over one database. That constraint produced an architecture that is worth describing on its own, because it is the opposite of what a small company usually ends up with, which is a dozen tools each wired to the next by a Zap.

2. Hub, never mesh

The database is the Brain. It holds the operating model (roles, processes, tasks, people) and a merged copy of what every other system knows, joined in ways no single system can join them. Every other program is one of three things:

Kind	What it is	Examples
Interface	Where people or agents act on the Brain	The web app, Discord, the agent desk
Connected system	A specialist tool with its own value, whose data the Brain mirrors or feeds	The CRM, the field-service system, the calendar, the ad platform
Connector	Stateless code that moves data between the Brain and everything else	Serverless functions on cron

Programs never talk to each other. They talk to the Brain. A client's status is changed in the CRM. A job is changed in the field-service system. The Brain mirrors both and joins them. It is the source of truth only for facts that exist nowhere else: roles, processes, tasks, people, which field-service customers belong to which client, and which CRM stages count as a client.

Four rules follow from that, and each one has paid for itself at least once.

1. **Raw first, typed later.** Inbound data lands exactly as the source returned it, as JSON. Typed views are added only when something reads them. When a sync breaks, the raw row is the evidence.
2. **Connectors are stateless and replaceable.** Everything worth keeping is in the database. A function can be rewritten or moved without losing anything.
3. **Interfaces add value; they do not keep their own copy.** A new interface, whether an agent over an API, a mobile app or a voice bot, reads and writes through the API. It never keeps a private store.
4. **Adding a tool means answering five questions in order:** which facts does it own; does data flow in or out; which key goes in the hosting environment; which migration adds its tables; which decision record explains why.

Who owns which fact is a table in the architecture document, and the table is the design. A client's owner, for instance, follows the CRM's account owner until someone changes it on the internal board, and then the internal board owns it; the sync writes the CRM value only while the owner source still says "CRM". That is one column, and it settles an argument that otherwise happens every week.

3. The twelve-function constraint

The hosting plan allows twelve serverless functions. All twelve are in use. That constraint shows up in the commit history more than any other single fact, and it is worth reading as a design pressure rather than a limitation.

The ad-spend mirror runs as a job parameter on the field-service sync function. The vendor licence panel branches off the people endpoint. The outbound-calling button rides an existing PATCH under a new key. Each of those is a small ugliness, and each is documented at the point of use, with the same sentence: the plan allows twelve functions and all twelve are used. When the plan changes, the sentence tells the next person exactly what to unbundle.

The alternative, paying for the higher plan early, would have cost $20 a month and removed the pressure to keep endpoints few. The decision was to keep the pressure until a paid feature needed the plan anyway, and the roadmap notes that commercial use forbids the free tier, so the upgrade is scheduled rather than avoided.

4. Three incidents and what they taught

The 62 KB query string

On 2026-09-18 a feature added a Call button to the pipeline board. To decide whether to draw the button, it looked up phone numbers by filtering the CRM mirror with an `.in("id", [...])` of every account on the board. The board carries 2,493 accounts. At 24 characters per id that built a query string of roughly 62 KB, and the database gateway answered 400.

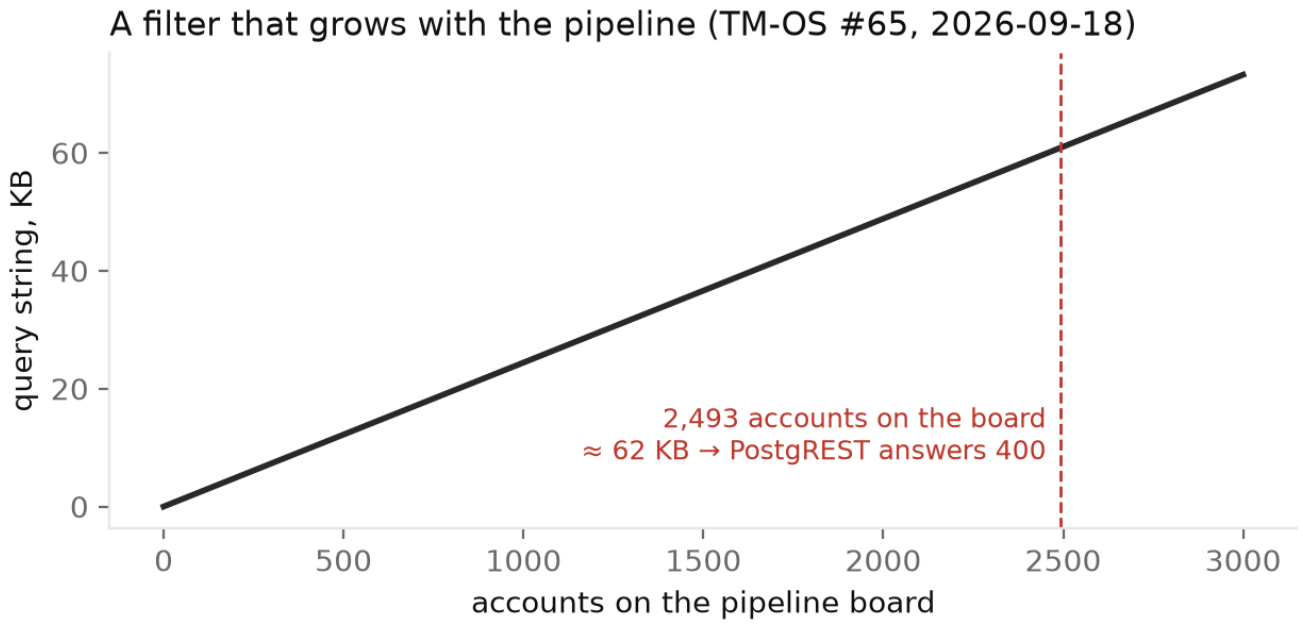

Figure 1. A filter whose length grows with the pipeline. It worked in development against a small board and failed in production against the real one.

The cost was not the Call buttons. The lookup threw, the request failed, and the whole pipeline tab rendered "Bad Request". A tab that worked before the feature existed went down because of an optional decoration on it.

Two fixes, and the second is the one to copy. First, the lookup reads the mirror in pages of 1,000 with no id filter and matches in memory: two small queries for the current 2,498 rows, neither with a query string that grows as the pipeline does. Second, and separately, **the phone lookup can no longer take the board down.** It returns an empty map on any error and logs. A phone number decides whether one optional button is drawn; it is not what the pipeline tab is for, so a failure there costs the buttons and nothing else.

Two tests pin it: a broken client yields an empty map rather than throwing, and the paging keeps only board accounts and only valid numbers.

The gateway timeout

The clients list ran seven queries at once on a free-tier database instance. After a quiet minute, even three parallel reads got a 504 at five seconds while the gateway opened new connections. Three layers changed. The billing-tag rollup, which had been unpacking all 14,719 customers' tags on every load at about 0.8 s warm, became a materialized view refreshed by the sync. The list endpoint runs its queries in two waves, three then four, and then in sequence so a load reuses the one warm connection. Reads that get a 502 or 504 retry once; writes never do; a timeout answers a readable 503 instead of "Gateway Timeout".

The report card that graded A on day one

The weekly report card took clients won since plan start from the scorecard. The scorecard still counted the nine snapshot rows written when stage history began, so the gate read 9 of 21 won and graded A on the first day. The fix is a rule, not a patch: **the first stage-history row per account is a snapshot, not a move, so it never counts as an entry or a hand-off.** A `moves` column carries the rule into the table, and on the same data the scorecard shows 0 clients won since 2026-09-01 instead of nine.

5. Attribution by identity

Money was going out on paid social advertising and the system could not see a cent of it: no reference in the repository, no environment variable, no ad table, no mirror. The mirror was built on 2026-09-14 following the same shape as every other mirror: raw records, one row per ad per day with spend in cents, and lead-form submissions. It is read-only against the ad platform. It never creates, edits, pauses or funds a campaign; campaign creation is out of scope and comes back as its own decision.

The design question was attribution. The obvious join is lead source on the job. Coverage of that field was 0 of 585 completed jobs over twelve months. Waiting for it would mean spending blind. Ad lead forms carry the lead's phone and email, and in the field-service mirror 72% of customers carry a phone against 13% with an email. So leads match on the last ten digits of the phone first, then email.

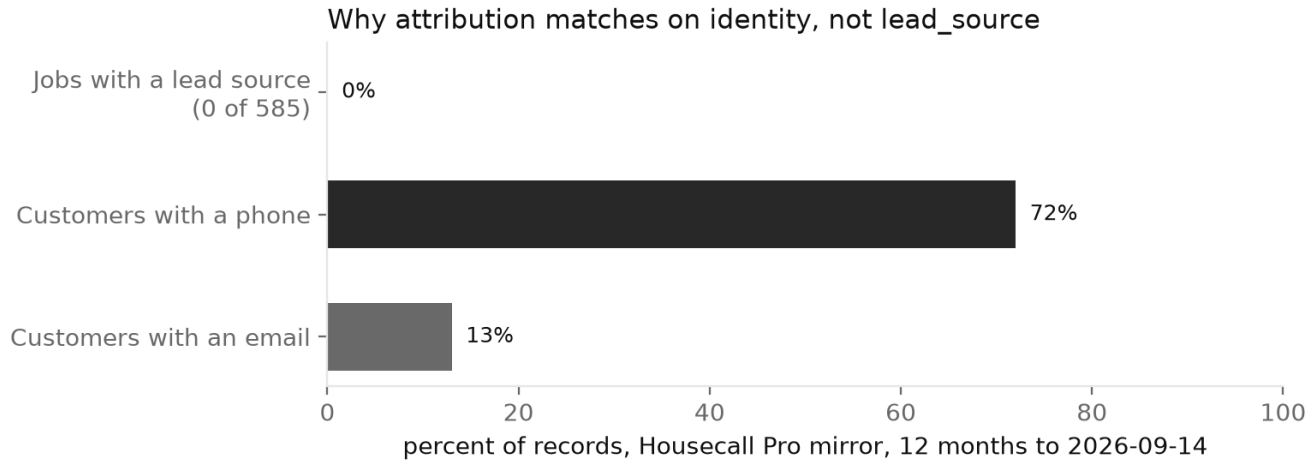

Figure 2. Why the funnel matches on phone. Lead source is empty; phone is the field with coverage.

The output is a funnel, not a cost-per-acquisition number. The ad leads are only in the ad platform, quotes have gone out, none is approved. Cost per lead, acquisition cost and lifetime value are null while their denominator is zero, and the funnel names the rung that is holding. A related fix on the same tab: a paid test nobody has entered a week for has zero leads, and the stop rule kills a channel under four attributed leads, so the tab greeted its owner with a kill verdict before a single number existed. It now returns a distinct "not started" verdict until rows exist and spend is above zero. **Zero data and zero results are different claims, and a dashboard that conflates them will recommend killing things it has never measured.**

6. The agent desk

Assign a task to Claude and a scheduled routine picks it up at 9:00 and 14:00 Eastern, Monday to Saturday. It answers questions into the task's notes, opens a pull request for code (a person merges it), and uses the same API a person does for data changes. Every action shows as "Claude desk" in the audit log, which has recorded every change with before and after values since 2026-09-11.

The desk signs in with a bearer token as its own user, compared in constant time, with no cookie and no origin check, and the middleware attributes its writes to that actor. It never sends email, touches sequences, deletes, or writes to the field-service system, the calendar or the chat platform. Anything it should not do is set to blocked with a reason.

Hand over, don't park

A review of the board on 2026-09-14 found ten open tasks owned by the desk. Nine had no step the desk could take: the last step on each lived in a website builder's settings page, a telephony dashboard, an identity admin portal, the hosting settings, or a CRM field with no API. The desk had followed the old rule, blocked with the manual steps in the notes. But blocked on the agent's queue means nobody sees them.

The rule changed. **When the only step left is outside the desk's systems, the desk reassigns the task to the person who holds that system, sets it to next, and puts the click path in the notes.** "Blocked" goes back to meaning the task waits on another task. The list of human-only systems is written down, and it is short: four settings pages and two CRM tiles. The desk also gained a second connector, the website builder's content API, with an explicit list of what that API does not expose at any plan.

The general principle is one that applies to any agent with a queue: an agent that parks work it cannot finish is a black hole. The queue looks busy and nothing moves. The fix is not a smarter agent. It is a routing rule that puts the work back in front of a person, with the next click written down.

7. The voice gate

The company's licences lived in PDFs, so nothing could answer what expires next. Two tables changed that: company facts that do not expire (address, crews, subcontracted trades, routing contacts) and licences that do, with a renewal window per row. The house rule is 45 days. One certificate carries 30, because its regulator requires renewal filed 30 days before expiry, and where the house rule and the regulator disagree, the regulator wins on its own certificate.

The column that matters for the agent layer is `sayable`. It gates what the outbound calling agent may state on a call, and it defaults to false, because a licence number read to a prospect is a claim the company is making. An expired licence is never stated. Insurance is not seeded, because it is not derivable from the source documents, and the voice agent must be able to say the company is licensed without implying it is insured. Renewal cards spawn one per licence per expiry cycle, owned by the compliance role and flagged as a blocker; the spawn key is the expiry date rather than the run date, so renewing moves the key and the next cycle spawns fresh. Verified against the live database in a rolled-back transaction: one card at the window, zero on a same-day re-run.

8. What the calling agent will not do

The Call button on the pipeline board hands an account to the outbound calling agent, a separate system. What the button does not do is as deliberate as what it does. It writes no consent, because a prospect has given none and manufacturing the evidence the gate exists to check would make the gate meaningless. It says "queued", never "called", because the calling agent runs its own checks (suppression, do-not-call, consent, line type, calling window, daily caps) and decides minutes later. A business landline passes; a mobile without consent is refused, and that refusal is the correct outcome. It reads the number server-side from the CRM mirror rather than trusting the one that reached the browser, because the calling agent would dial what it is sent. Accounts with no sanitised number get no button, which is about a third of the mirror.

9. What is not shown

- Revenue, client names, staff names and licence numbers are in the system and not in this paper.

- The report card's grades and the scorecard's targets are internal. The mechanism (one letter grade per line against a plan-derived requirement, frozen on Mondays, never regraded against last week) is described; the numbers are not.
- Lead volume from paid advertising is a funnel with null cost metrics as of 2026-09-14. When a denominator exists, it will be reported as a number with the date it was read.

10. What to copy

- **One database, and a rule that tools never talk to each other.** The joins live where the data does, and every connector is disposable.
- **Raw first.** Mirror what the source returned, type it when something reads it.
- **An optional feature must not be able to take down the page it decorates.** Return empty, log, and test that path.
- **A snapshot is not a move.** The first row of any history table is a baseline, and counting it as an event inflates every metric built on it.
- **Hand over, don't park.** An agent's queue is not a place to store work a person has to finish.
- **A claim the company makes to a prospect is gated by a column that defaults to false.**
