# From Layers to Answers

A five-step architecture for turning geospatial data into operational intelligence — and a working demonstration that the agent is the easy part.

JEFF FRANZEN · JUNE 2026 · FOR DISCUSSION — INDEPENDENT RESEARCH

The first draft argued a thesis: stop building GIS applications, start building answer infrastructure. Layers in, a documented catalog and a library of analytical recipes in the middle, any interface out. The user asks in plain English and receives a complete, sourced answer.

Two things happened since. The thesis got **built** — a working demonstration now answers live questions against public FEMA data, every number traceable to the query that produced it. And it got **sharper**: a second body of work — Agentic GIS — supplied the missing pieces. The small set of tools that does most of the work. The line that keeps the system trustworthy. And an honest account of where the real labor lives.

## The Old Way Stops at Step One

For two decades the GIS pattern has had three moves: acquire data, publish layers, assemble them into an application — an Experience Builder dashboard, a web map, a story. Then we hand it over and ask the user to figure it out. Which layers answer my question? In what order? With what analysis? That mental work is why every dashboard needs a training session, and why most people ask a GIS person instead of using the tool.

The application was never the product. It was a container for a handful of operations — count these, filter by county, find the nearest, compare two areas — dressed up as a map. The interface is where the real capability goes to hide.

> *The data is excellent. The head attached to it is the bottleneck.*
>
> — THE ORIGINAL THESIS, UNCHANGED

What changed: we no longer have to put a human head on the data to interpret it. A language model can — but only if we stop treating it as the thing that knows the answer, and start treating it as the thing that knows how to find the answer. That distinction is the whole brief.

## The Five Steps

The architecture moves the analyst's thinking out of the user's head and into the pipeline. Two intelligence steps sit between the data and the screen.

1 DATA
**Layers**
Source, clean, publish feature services. Unchanged — RAPT's REST services already do this. Don't migrate or mirror; query live.

2 INVENTORY
**Catalog**
Layers become documented, queryable endpoints, not ingredients dumped into an app: schemas, fields, refresh cadence, and what each layer contains — and doesn't. Documentation as data.

3 INTELLIGENCE · KNOWLEDGE
**Recipes**
The semantic layer. For each question a GIS analyst would answer by hand: which catalog entries it uses, the ordered operations, the output shape, the mandatory caveats. Judgment, written down as machine-readable intent. **Recipes are the product.**

4 INTELLIGENCE · EXECUTION
**Agent**
A language model reads the catalog and recipes, takes a plain-English question, and orchestrates the function calls — geocode, query, count, join — into a complete answer with sources. It never invents a number.

5 OUTPUT
**Any UI**
The head detaches from the body. One backend serves a chat-and-map, a dashboard, a Slack bot, a voice line, a spreadsheet, or a bare API — eight front doors, one engine. **Shown in full on page 5.**

The split between steps 3 and 4 is what makes this five steps, not four. The catalog tells the agent *what exists*; the recipes tell it *what is possible and how*. Without step 3 the agent rediscovers GIS reasoning on every question — and gets it wrong often enough to lose trust. With it, it executes proven recipes. Steps 2 through 5 sit on top of services FEMA already publishes: no migration, no agency permission.

## The Five Primitives

The idea that makes the recipe layer tractable: nearly all read-only operational spatial reasoning reduces to **five primitives**. They use no AI; they take structured inputs and return structured facts. Every recipe is one of these five.

| | |
|---|---|
| **LOCATE** | *Where is it? What is it?* — geocode an address; pull the profile of the county it falls in. |
| **NEAR** | *What's nearby?* — the hospitals, shelters, or facilities within a radius of a point. |
| **COUNT** | *How many?* — the damaged homes, volunteers, or nursing homes inside an area. |
| **SUMMARIZE** | *What's here?* — the vulnerability, capacity, and exposure of a place, compressed. |
| **COMPARE** | *What's different?* — two areas, identical metrics, aligned side by side. |

That is half of operational GIS, in five functions. Point them at wildfires, nursing homes, hospitals, and a risk index, and you answer thousands of questions. The model's only job: read the question, pick the primitive, fill the arguments, translate the result. A router, not an oracle.

> *The LLM should not know the answer. It should know how to find the answer. The tools produce facts; humans own decisions.*
>
> — THE AGENTIC GIS REFRAME

This is the opposite of bolting a chatbot onto a map. A chatbot recalls plausible text and can hallucinate a shelter count; an orchestrator over real tools queries the live layer every time, so every figure traces to a query you can click. Near an operational decision, that is not a nice-to-have — it is the only version safe to deploy.

These five cover read-only, descriptive questions — not routing, optimization, or forecasting. And when the model turns counts into a characterization, that reading is its own, not the data's. Fine for "what kind of place." Dangerous at "which place to prioritize" — the next page.

## The Governance Line

The sharpest point in the brief, and the one that decides whether leadership trusts the tool or shelves it. Three levels, with a bright boundary between them.

| STATEMENT | WHAT IT IS | WHO OWNS IT |
|---|---|---|
| **"1,247 damaged homes in County A, 843 in County B."** | Analytics — a fact | The tool |
| **"County A is under-resourced."** | A judgment | Contested |
| **"Send resources to County A first."** | A decision | The human |

The first is safe; the tool reports it. The third is a command a human must own — it depends on the active hazard, the response objectives, the staffing standard, the current resource posture, none of which live in the data. The middle is where AI projects quietly fail: "under-resourced" *compared to what?* The model has slipped from reporting a number into a judgment it cannot defend.

The rule: **build tools that count and rank; keep the decision human-owned.** Every statement traces to a metric — "Bay County ranks 1st in damaged homes, 7th in volunteers, 2nd in shelter occupancy" — then the human decides. The agent never fabricates and never recommends an allocation.

Not a policy bolted on afterward — a behavior you can demonstrate. Ask the working system to make the call:

> *"Which Florida county should we send disaster resources to first?"*
>
> **IT OPENS** "I can't pick the county for you — that allocation call belongs to your incident command, because it depends on the active hazard, your response objectives, and staffing standards the data doesn't carry."
>
> **THEN** It produces the ranked hurricane-risk table — Miami-Dade, Lee, Palm Beach — every figure from a live query.
>
> **IT CLOSES** "What the data can't decide for you" — naming the live-threat, resource-posture, and objective judgments a human must layer on.

It declined to decide, surfaced the facts that inform the decision, and named the judgment required. That is what makes this deployable next to operations rather than an interesting demo.

# The Detached Head — One Backend, Any Interface

Step five, in full — the payoff of the whole architecture. Because the catalog, recipes, and agent hold the intelligence, the interface is a detachable skin. Build the engine once; surface the same sourced answers wherever responders already work, with **no new analytics**. Eight front doors onto one backend:

### This page
CHAT + LIVE MAP
Ask, watch the queries run, drill into the map. Zero training.

### Operations dashboard
AUTO-REFRESHED TILES
The same recipes feed KPI tiles on a wall display — no analyst clicking.

### Slack / Teams bot
ANSWERS IN THE CHANNEL
A duty officer asks in the ops channel; the sourced answer posts inline.

### Voice / phone
HANDS-FREE IN THE FIELD
Asked aloud, answered aloud — for responders who can't look at a screen.

### Public JSON API
ONE ENDPOINT, ANY CLIENT
Get the sourced answer as JSON — feed Power BI, a sheet, another agency.

### Experience Builder
INSIDE THE TOOL YOU RUN TODAY
Drop the answer into a widget in your existing AGOL app — no rebuild.

### Automated SITREP
SCHEDULED, EMAILED
A nightly job asks the standing questions and emails the brief before the call.

### Spreadsheet function
=RAPT("…")
A cell formula returns the live count — the analyst's surface, new engine.

## Examples — the same engine, any question

A handful of the questions the live demo answers, each tracing to real FEMA queries:

| QUESTION | WHAT COMES BACK |
|---|---|
| **Which US counties have the highest hurricane risk?** | Harris County, TX leads at 100.0; Miami-Dade holds 992 nursing homes; ranked table + map. |
| **Simulate a hurricane at Tampa heading NE.** | Cone over 20 counties: 9.9M residents, 1,504 nursing homes, 82,120 power-dependent. |
| **Hospital capacity within 25 miles of Miami?** | 48 open hospitals, 13,935 beds, 7 trauma centers, 19 helipads. |
| **Compare St. Petersburg and Pensacola.** | Side-by-side risk, vulnerability, and facility counts — no winner declared. |
| **Which nursing homes near Tampa have generators?** | "The data has no backup-power field" — it declines rather than guess. |

## The Hard Part Is the Data — and the Proof It's Worth It

The honest caution: the agent loop is a weekend; the data plumbing is not. The demonstration below was fast to build because the public layers — RAPT, HIFLD, the National Risk Index — arrive already API-shaped and pre-joined. Internal Red Cross data is the opposite: DDA, shelters, DAT, volunteers, NFIRS, all in separate systems that don't speak one language. Joining them is the real, multi-month work. That work is **Project Keystone**: a canonical, FIPS-keyed county table — one row per county, swappable boundary assignments, static indices, dynamic operational columns. Once it exists, the five primitives are trivial and the agent is almost an accessory. Most organizations think they have an AI problem; they have a data-contract problem. The cheap first move isn't code — it's **Phase 0.5**: inventory the Experience Builder widgets that are secretly tools (a DDA filter is `count_damaged_homes`; a shelter map is `find_shelters_near`). Likely 50–75% of the tool catalog already exists, trapped in widgets.

## The Proof — it's running

A working **Answer Engine** implements steps 2 through 5 on public data: a documented catalog of a dozen live layers, the five primitives as recipes, a model orchestrator, a chat-and-map plus a bare JSON API. Ask it to simulate a hurricane at Tampa, and it builds the impact cone and sweeps every layer inside it, live:

| 82,120 | 1,504 | 136 | 9.9M |
|---|---|---|---|
| ELECTRICITY-DEPENDENT (EMPOWER) | NURSING HOMES | OPEN HOSPITALS | PEOPLE · 20 COUNTIES |

Click a county and the analysis re-scopes to it; toggle its census tracts and they shade by social vulnerability. Every figure traces to a query you can open. Asked who to prioritize, it declines. It demonstrates the whole architecture, the five primitives, and the governance line on the most public data available — de-risking the Keystone bet before a dollar is spent on internal plumbing.

## The Ask

An architecture discussion, not a product pitch. Three questions worth debating: Should new GIS products ship as catalog entries plus recipes rather than finished apps? Who curates the recipe layer and owns the governance line? Is Phase 0.5 — the widget inventory that becomes the canonical table — the right place to start on internal data?

**One sentence:** stop building applications, start building answer infrastructure — layers in, a catalog and a library of five-primitive recipes in the middle, any interface out, the tools producing every fact and the human keeping every decision.