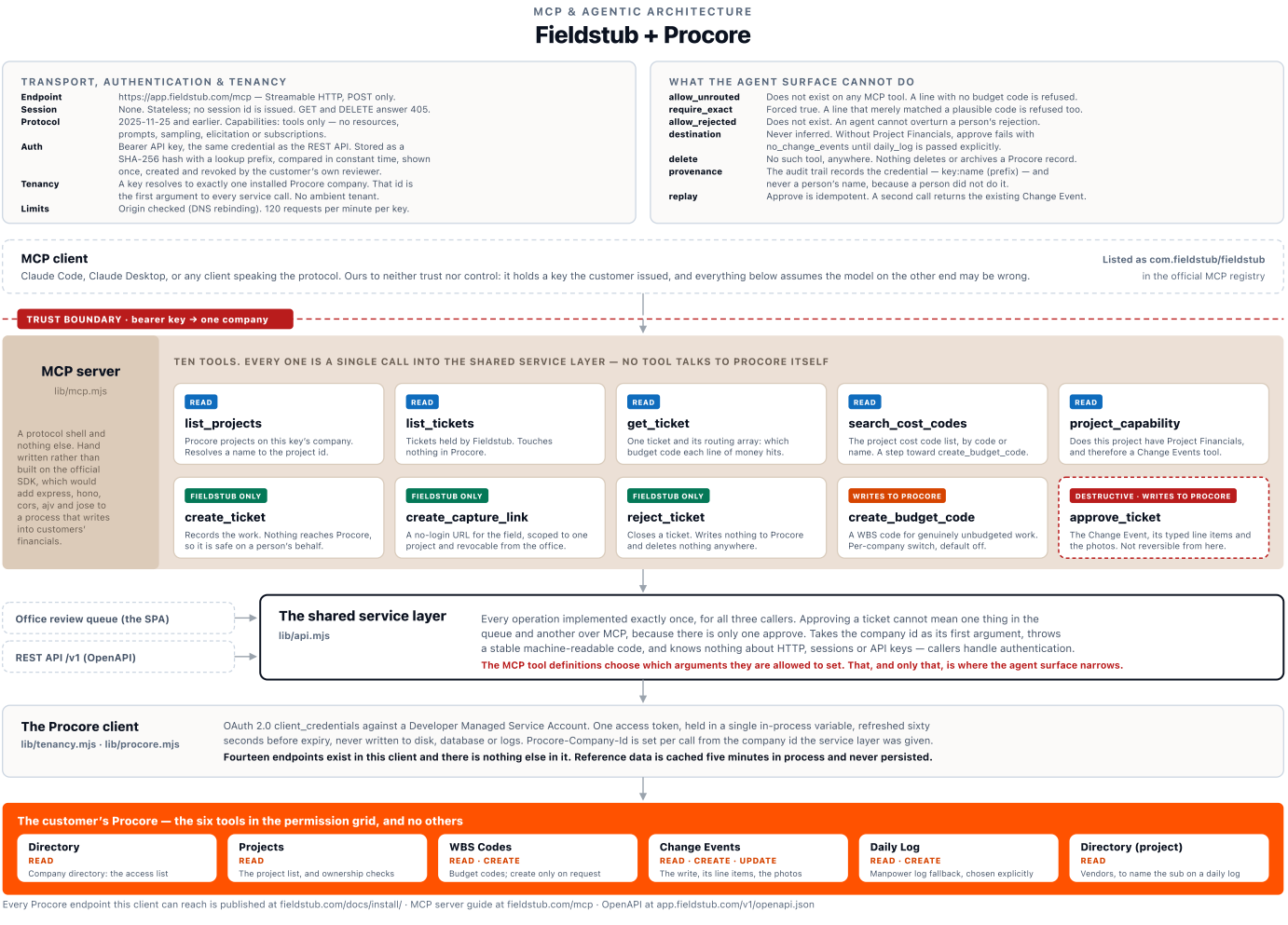
MCP and agentic architecture

How the MCP server is built, what each of its ten tools can reach, and where it meets the service layer the review queue and the REST API also use.

The diagram



Transport, authentication and tenancy

Endpoint	`https://app.fieldstub.com/mcp`. Streamable HTTP, POST only
Session	None. The server is stateless and issues no session id, so there is nothing for a client to carry and nothing for us to expire. GET and DELETE answer 405
Protocol	2025-11-25 and earlier. Capabilities are tools only: no resources, prompts, sampling, elicitation or subscriptions
Authentication	A bearer API key, the same credential as our REST API. Created and revoked by a reviewer inside the customer's own Fieldstub queue, stored as a SHA-256 hash with a short lookup prefix, and shown once at creation
Tenancy	A key resolves to one installed Procore company. That company id is the first argument to every call beneath it, so there is no ambient tenant and no way to name another company
Limits	The Origin header is checked, per the transport spec's DNS-rebinding requirement. A key is capped at 120 requests a minute, so a looping agent surfaces as 429s here rather than as 429s from Procore, where the budget is shared
Discovery	Listed in the official MCP registry as `com.fieldstub/fieldstub`

The server is hand-written rather than built on the official SDK. The SDK pulls express, hono, cors, ajv and jose into a process that writes into customers' financials, and we need none of it: every tool is one request and one response.

The ten tools, and what each one can reach

Three classes. The class says what a tool can reach, not what it is meant to do.

Tool	Class	What it does
`list_projects`	Read only	Procore projects on this key's company
`list_tickets`	Read only	Tickets held by Fieldstub. Touches nothing in Procore
`get_ticket`	Read only	One ticket and its routing array: which budget code each line of money would land on
`search_cost_codes`	Read only	The project cost code list, by code or name
`project_capability`	Read only	Whether a project has Project Financials, and so a Change Events tool
`create_ticket`	Writes to Fieldstub only	Records the work. Nothing reaches Procore, which is what makes it safe to do on a person's behalf
`create_capture_link`	Writes to Fieldstub only	A capture URL for the field that needs no login, scoped to one project and revocable from the office
`reject_ticket`	Writes to Fieldstub only	Closes a ticket. Writes nothing to Procore and deletes nothing anywhere
`create_budget_code`	**Writes to Procore**	A WBS code for unbudgeted work. A per-company switch, off by default
`approve_ticket`	**Writes to Procore, destructive**	The Change Event, its typed line items and the photos. Annotated destructive, and not reversible from here

Interaction with the shared service layer

No tool talks to Procore. Each of the ten is one call into our service layer, and the file implementing the MCP protocol holds nothing but protocol code.

The review queue an office user clicks through, the public REST API and this MCP server are three callers of that layer. There is one implementation of approving a ticket, so approving cannot mean one thing on screen and another over MCP. Everything in the layer takes the company id as its first argument, returns plain JSON, throws a stable error code, and knows nothing about HTTP, sessions or API keys.

Beneath it sits the Procore client. It holds the Developer Managed Service Account access token in a single in-process variable, refreshed sixty seconds before expiry and never written to disk, database or logs, and it sets `Procore-Company-Id` per call from the company id the service layer was given. It contains fourteen endpoints and nothing else, enumerated in the V1 scope note.

> The MCP tool definitions choose which arguments they may set when they call that layer. That is the only place the agent surface narrows.

What the agent surface cannot do

It is narrower than the human one, deliberately. A reviewer looking at a screen can see a flag and weigh it. An agent cannot, so the choices that need a person are not offered to it.

`allow_unrouted`	Does not exist on any MCP tool. A line with no budget code is refused, always. Procore accepts a line item with a blank budget code and returns success, which is how costs go missing, so this is the fence that matters most
`require_exact`	Forced on. A line that merely matched a plausible code is refused too
`allow_rejected`	Does not exist. An agent cannot overturn a person's rejection
`destination`	Never inferred. On a project without Project Financials, approving fails with `no_change_events` until `daily_log` is passed explicitly, because a change event is money on a budget code and a daily log bills nothing
Deletion	There is no delete tool anywhere, and nothing in the integration deletes or archives a Procore record
Replay	Approving is idempotent. A second call returns the existing Change Event rather than writing a second one
Provenance	The audit trail records the credential, as `key:name (prefix)`, and never a person's name, because a person did not do it

---

What we want your read on

`approve_ticket` means an agent holding a customer-issued key can write a Change Event into that customer's Procore without a person clicking approve in our queue.

Four things stand in front of it. The key is issued and revoked by that customer's own reviewer. It is scoped to a single company. The tool refuses any line it could not route exactly. And every write carries our provenance, so the exact set of records touched in any window can be enumerated rather than estimated.

If your platform team wants that path fenced further for V1, held to read and draft only or gated behind a per-company switch like budget code creation, tell us and we will fence it. Better now than at Certification.

Links

MCP architecture diagram, PDF

MCP server guide, written for customers

The integration architecture page

Public REST API and OpenAPI spec

Prepared for the Procore Technology Partner team. Last updated 18 September 2026. Developer app 4d269810-7a51-4e46-88ca-f81eda9bca54.

The other two notes in this set: V1 scope · Daily Log permission

Questions to tim@fieldstub.com. https://fieldstub.com