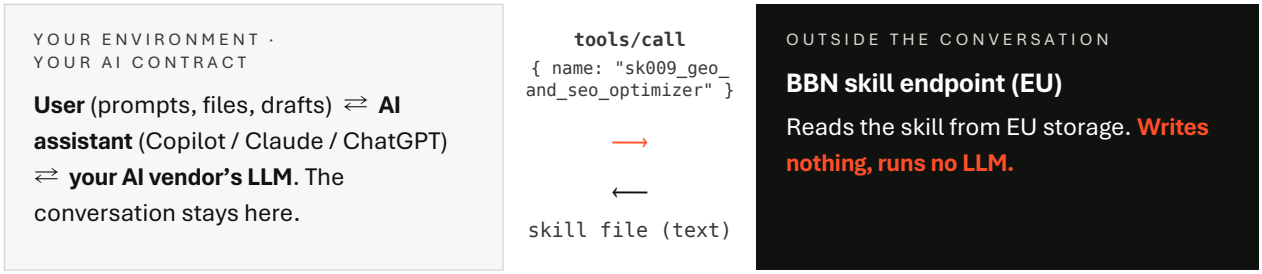
What the endpoint sees.

Your team's prompts never reach us. The BBN AI Studio skill endpoint is a delivery surface. Your AI assistant fetches skill files from it and applies them inside your own AI platform. What your people type, paste, and generate stays where it happens today: Between you and your AI vendor.

This brief shows the mechanism, lists everything our server can technically receive, and gives your IT four ways to check it without taking our word for anything.

01 · HOW A SKILL CALL WORKS

One request goes out. It carries a name, not your data.



The conversation loop on the left never touches the endpoint on the right. The only traffic between them is a skill name out, a text file back.

1 A user works with their AI assistant. The conversation runs between the user, the assistant, and your AI vendor's model. We are not on that path.

2 When the assistant decides one of your calibrated skills fits, it calls our endpoint with the skill's name. This is the complete request:

```
{ "method": "tools/call", "params": { "name": "sk009_geo_and_seo_optimizer", "arguments": {} } }
```

3 The endpoint returns the skill file as text. The assistant applies it to the user's work inside your platform. The result never comes back to us.

There is no field in that exchange for the user's prompt, and none for the output. The Model Context Protocol only sends a tool name plus the arguments the tool declares, and our skill tools declare none.

02 · EVERYTHING OUR SERVER CAN RECEIVE

The full inbound vocabulary, with nothing omitted.

REQUEST	ARGUMENTS	PURPOSE
`initialize`	protocol version, assistant app name + version	handshake
`tools/list, resources/list`	none	show available skills and files
`tools/call on a skill`	none[1]	fetch one skill
`tools/call on the file tools`	one file path or skill slug	list or fetch reference files
`resources/read`	one `skill://` file path	same, via the resources surface

[1] Skills that are steps in a multi-step workflow accept one optional short note (`bypass_reason`) that acknowledges earlier steps were done elsewhere. It toggles a hint in the response and is discarded.

The connection itself shows us what any HTTPS service sees, meaning the caller's IP address, a user-agent string, and the timestamp.

WHAT NEVER ARRIVES

Because no field exists to carry it: **the user's prompt, pasted documents, chat history, the model's output, and the identity of the person typing.**

Requests authenticate with one key per company. There is no personal login toward us.

Three of the parameters above are strings, so in theory an assistant could put other text into them. It gains nothing. A value that matches no known skill or file returns an error, and either way the value is used for the lookup and dropped.

03 · WHAT HAPPENS TO A REQUEST ON OUR SIDE

Look up, read, return. Nothing else.

Look up your endpoint, read the requested skill file, return it. The serving path is three source files, under a thousand lines, and it contains no LLM, no analytics SDK, and no database writes at all. A one-line audit of the MCP code finds zero insert, update, or delete statements. We do not even have usage statistics today.

Every call is stateless. The server issues no session, sets no cookie, and keeps nothing that links one request to the next. Failures land as error entries in short-lived platform logs at our hosting provider; a malformed file path can appear there, chat content cannot, because it never arrives.

04 · WHAT WE STORE ABOUT YOU AT ALL

Two kinds of data. Only one touches our systems.

CALIBRATION MATERIAL · GIVEN DELIBERATELY

Terminology, brand guidelines, personas, reference documents, placeholder values. It is what makes the skills yours. It lives in our database and file storage in the EU (AWS eu-west-1, Ireland), is covered by the DPA, and is deleted at the end of the contract.

USAGE-TIME DATA · THE CEILING

Which skill was fetched, and when. That is the ceiling of what the protocol lets us observe, and today the application records none of it.

No training happens on your prompts. Nothing could: The prompts never reach us, and the skills do not learn from usage.

05 · VERIFY IT YOURSELF

Four checks, ordered by effort. The first two need nothing from us.

1. **Read the contract live.** With your endpoint key, any MCP inspector (for example `npx @modelcontextprotocol/inspector`) or your assistant's connector settings lists every tool with its full input schema. You will find no parameter that could carry conversation content.
2. **Watch the wire.** Your side already logs every request the assistant sends us. Claude Desktop, for example, writes its MCP traffic to local log files on the user's machine, and a corporate TLS proxy shows the same picture. Run a real working session, then read the log. The chat never appears.
3. **Audit the code.** We walk your IT through the three files of the serving path under NDA, including the empty writes-grep from section 3.
4. **Watch our logs live.** A joint session: Your team uses the skills, we share the server log stream. You see fetches and nothing else.

06 · SECURITY POSTURE, QUICK FACTS

The controls around the endpoint.

- One endpoint per company, authenticated with a random 256-bit key. Rotation on request, instantly. Bearer-header auth is supported alongside the key-in-URL form.
- An endpoint can be taken offline with one switch and then answers 404, as does an endpoint with no skills attached.
- Each endpoint serves only that company's calibrated skills. Reads outside the attached set fail by construction, so tenants cannot see each other.
- TLS on every connection. Hosting on Vercel, data at rest with Supabase in AWS eu-west-1 (Ireland).
- The tool surface ships as versioned releases with automated protocol tests and a changelog. Schema changes are announced.

SCOPE

This describes the MCP integration path as shipped on 14 August 2026. It is the most connected of our three delivery paths, and even it carries no content. The Copilot Studio and Import paths run entirely inside your own tenant.

Questions, DPA, subprocessor list?

We answer them with the people who wrote the code.

marco@bbn-ai.studio · www.bbn-ai.studio