

ContextStream MCP Server: README

Contents

Your agent knows code. Give it the decisions behind yours.	2
Start free with MCP	2
Make the next session useful	2
Slipstream agents	2
Proof and data controls	3
npm (Node.js 20+)	3
Hosted MCP without a local process	3
Preview or diagnose setup	3
Scripted and CI setup	4
Package Bouncer (opt-in)	4
Open source and contributing	4
Optional deep project learning	5

ContextStream MCP Server

Persistent memory. Millisecond code search.

Your agent knows code. Give it the decisions behind yours.

Connect your code, docs, and conversations so your agents can find the right files, recall saved decisions, and build on past work across sessions and tools.

Start free with MCP

10,000 monthly credits. No credit card required.

Works with Claude Code, Cursor, Codex, GitHub Copilot, Gemini CLI, Qwen Code, Kimi Code, Muse Code, ZCode, Zed, and more MCP clients, on Claude, GPT, Gemini, Kimi, GLM, Qwen, and other models. Create your account or sign in during MCP onboarding. No separate website signup needed.

macOS and Linux

```
curl -fsSL https://contextstream.io/scripts/mcp.sh | bash
```

Windows PowerShell

```
irm https://contextstream.io/scripts/mcp.ps1 | iex
```

Paste the command into your terminal and follow onboarding to connect your project and supported editor. Restart your editor after setup.

What gets installed and indexed? · Supported clients · [Setup guide](#) · [Prefer to start on the web?](#)

Make the next session useful

Ask your connected agent:

Use ContextStream to find the files relevant to my next change. Cite the sources and retrieve any saved project decisions that should guide the work.

Then ask it to save a real decision and its reason. Start a new session and retrieve that decision without explaining it again. Indexing supplies code context; it cannot recover every undocumented decision.

Slipstream agents

When your workspace has Slipstream, the server adds one tool, `slipstream`, that hands a whole job to a ContextStream specialist agent working with your workspace's context: review a pull request against your team's decisions, write release notes, check docs against the code, tend memory, write a repository guide, read an image, make a PDF, or digest your mail (it never sends). Each agent is priced in credits per run and charged only when the run succeeds; declined, failed, cancelled and timed-out runs are free.

- **Enabling.** A workspace owner or admin turns agents on in the workspace's Slipstream settings. The tool is listed only for a workspace with at least one agent turned on, so everywhere else it costs no prompt. Its description names each agent turned on, its price and the input it needs, so you can just ask: "Which Slipstream agents can I use here?" or "Have Slipstream review PR 42 in acme/api."

- **Workspace.** The local server lists it for the workspace of your session (or of `CONTEXTSTREAM_WORKSPACE_ID`, or of the folder's mapping) and tells clients that follow `notifications/tools/list_changed`, such as Claude Code, when that changes. A hosted connection lists it for the workspace its `X-ContextStream-Workspace-Id` header names.
- **Prices.** A priced agent first answers with its price; your agent shows it to you and runs the job once you accept. To let runs up to a price start without that step, set an auto-accept cap: `CONTEXTSTREAM_SLIPSTREAM_AUTO_ACCEPT_MAX_CREDITS=500` in the local server's environment (your MCP config's env), or the header `X-ContextStream-Slipstream-Auto-Accept-Max-Credits: 500` on a hosted connection. Unset or 0 always asks; the hosted gateway never sets a cap for you.
- **Runs.** A run waits up to 25 seconds for its result; a longer one returns its task id for your agent to follow. A retried call never starts a second run.
- **Terminal.** The same agents run from a shell with `ss run <agent>` (alias `csa`).

Proof and data controls

95 ms median code search in our published benchmark. This is the measured successful-response median for the disclosed configuration, not a latency guarantee. [See results, methodology, and limitations.](#)

Scoped access. Traceable sources. Configurable capture. Indexing sends eligible source contents to ContextStream for hosted search. Transcript saving and local Git metadata capture are on by default and can be turned off. Review data handling and controls before connecting sensitive projects.

Other installation options and troubleshooting

npm (Node.js 20+)

```
npx -y @contextstream/mcp-server@latest setup
```

The npm launcher downloads the native Rust binary for your platform and verifies its SHA-256 checksum. Pin an exact package version for production automation.

Hosted MCP without a local process

For clients supporting Streamable HTTP and OAuth, the hosted endpoint is:

```
https://mcp.contextstream.io/mcp?default_context_mode=fast
```

Supported clients lists one-line add commands for Claude Code, Codex, Gemini CLI, Qwen Code, Copilot CLI, and Droid, and one-click install for VS Code. Use the [MCP documentation](#) for other clients and stdio alternatives. A hosted connection alone does not sync your local checkout.

Preview or diagnose setup

```
contextstream-mcp setup --dry-run
contextstream-mcp doctor --scope=all --only-configured
```

Scripted and CI setup

For automation, wrap the install in `pipefail` so it fails when the download fails, instead of reporting success after installing nothing:

```
bash -o pipefail -c 'curl -fsSL https://contextstream.io/scripts/mcp.sh | bash'
```

`setup --yes` reads the key from `CONTEXTSTREAM_API_KEY` or saved credentials, and needs `--workspace-id` when the account has more than one workspace:

```
export CONTEXTSTREAM_API_KEY=... # or: printf %s "$KEY" | contextstream-mcp configure --api-key-stdin
contextstream-mcp setup --yes --editors=claude --workspace-id=<UUID> --project-path=.
```

Package Bouncer (opt-in)

A hook for Claude Code and Cursor that checks the packages an install command names against their public registry before the command runs. It covers `npm`, `pnpm`, `yarn`, `bun`, `npx`, `pip`, `uv`, `uvx`, `pipx`, `poetry` and `cargo`, including `claude mcp add ... -- npx ...`.

- It **denies** a package the registry does not have, naming the popular package it looks like. An agent may have invented the name, and anyone can register it.
- It **asks** before a barely downloaded lookalike of a popular package.
- It **allows** everything else, including whenever a lookup fails or takes too long.

It skips projects and commands that use a private registry (`.npmrc`, `pip.conf`, `uv` or `poetry` sources, `--registry`, `--index-url`). `CONTEXTSTREAM_PACKAGE_BOUNCER=off` turns it off.

Claude Code: add this entry to `PreToolUse` in `~/ .claude/settings.json`, next to the hooks setup installed. `update-hooks` leaves it alone.

```
{ "matcher": "Bash", "hooks": [{ "type": "command", "command": "contextstream-mcp hook package-bouncer", "timeout": 8 }] }
```

Cursor (`~/ .cursor/hooks.json`):

```
{ "version": 1, "hooks": { "beforeShellExecution": [{ "command": "contextstream-mcp hook package-bouncer" }] } }
```

Open source and contributing

This repository contains the MIT-licensed Rust MCP server, client, editor setup, and release tooling. The hosted backend is separate and is not included here. See the [architecture](#), [release integrity](#), and [latest release](#).

```
cargo build --locked -p mcp-server --bin contextstream-mcp
cargo test --locked --workspace
```

Use the pinned Rust toolchain. For contribution checks and commit sign-off, read `CONTRIBUTING.md`. Report vulnerabilities using `SECURITY.md`. See `LICENSE`, `NOTICE`, and `GOVERNANCE.md` for licensing, trademarks, and project governance.

Let humans be human.

[Docs](#) · [Pricing](#) · [Integrations](#) · [Benchmarks](#)

Optional deep project learning

Deep project learning starts off for new accounts. Plain `contextstream-mcp setup --yes` preserves your current choice. To review consent, run `contextstream-mcp configure --account-learning` on and confirm on the signed-in privacy page. To withdraw, run `contextstream-mcp configure --account-learning off`. Agents cannot consent for you. Doctor and init show status only. See data handling and controls.