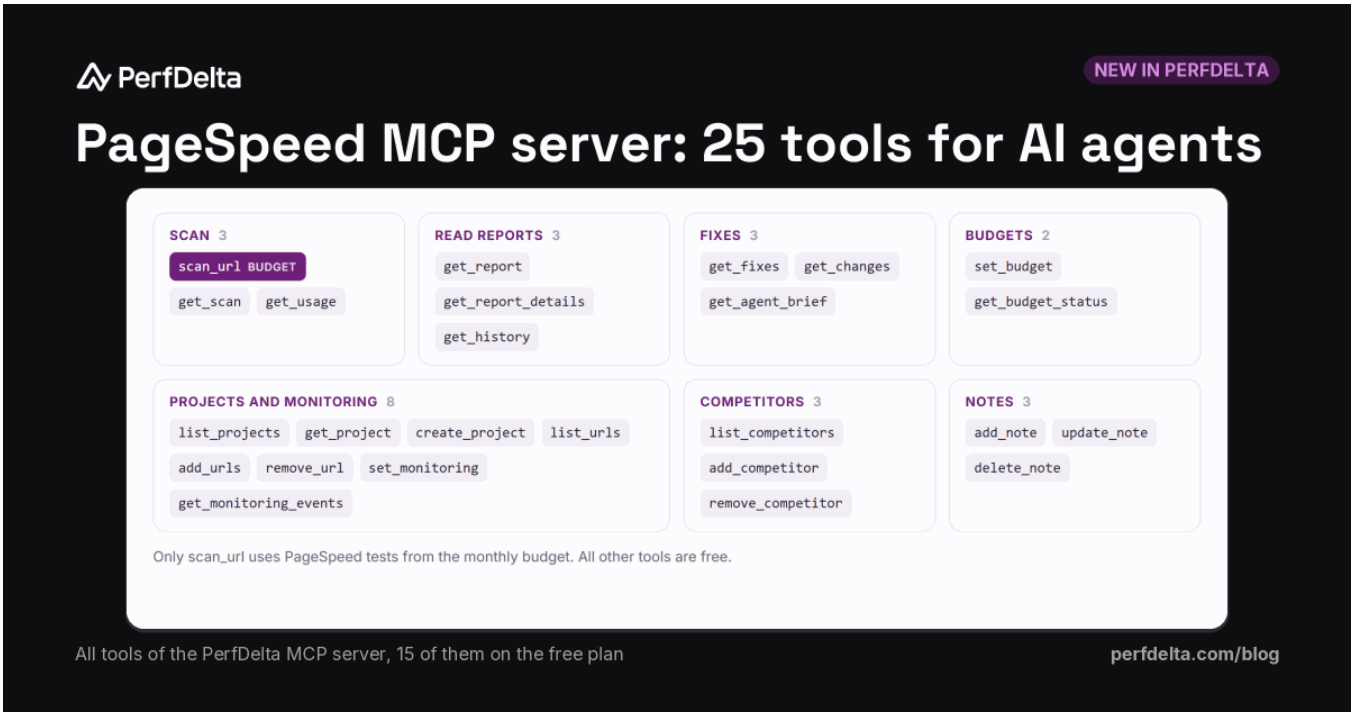
PageSpeed MCP Server: What the 25 PerfDelta Tools Do for AI Agents

Published November 5, 2026 by Christian Schreiber



The PerfDelta MCP server connects AI assistants such as Claude, ChatGPT, Cursor or GitHub Copilot in VS Code directly to your PageSpeed and Core Web Vitals data. Through the Model Context Protocol (MCP), an agent can scan a website, read the full report, fetch fixes, manage projects and monitoring, check performance budgets and attach notes to scans. There are 25 tools in total. Only scanning uses PageSpeed tests from your monthly budget; all other tools are free. The PerfDelta free plan includes 15 of the tools with one scan every 24 hours, all 25 come with the Solo plan.

Set up the PageSpeed MCP server in Claude Code, Cursor and Claude.ai

You need an API key, which you create on the API keys page of your account. In Claude Code, one command is enough:

```
claude mcp add --transport http perfdelta https://perfdelta.com/api/mcp.php \
  --header "Authorization: Bearer YOUR_API_KEY"
```

For Cursor and VS Code, add the same address and header to `.cursor/mcp.json` or `.vscode/mcp.json`. The connector settings of Claude.ai, Claude Desktop and ChatGPT (developer mode) do not accept custom headers. There, use the address with the key as a parameter (`?key=`) as the connector URL. Anyone who knows this URL can use your key, so treat it like a password. All variants are in the API documentation.

Scan and read reports: scan_url, get_scan, get_report_details

- **scan_url** starts a scan of a URL, optionally for mobile or desktop only. With `scope=group` it measures all URLs of a project group, with `project_id` it adds a new URL to the project first.
- **get_scan** shows progress and result. The agent polls every 15 to 30 seconds; a scan usually takes one to three minutes.
- **get_report_details** returns the full report of one of your scans: lab and field data with ratings, page weight, third parties, LCP element, layout shifts, competitor comparison, tech stack, notes and history.
- **get_report** reads the public results of any domain that was scanned before, without using budget.
- **get_usage** shows plan, monthly budget and the scans still possible today. When usage gets high, it warns before scans start to fail.

Causes and fixes: get_fixes, get_changes, get_agent_brief

Three tools are meant for working on the code. **get_fixes** returns the fixes for a domain in the order of the report, with affected files, elements, estimated savings and a text that matches the detected platform. **get_changes** explains a drop: it compares the latest with the previous scan and names new or heavier third parties, changed script bundles, a different LCP element and fixes implemented since then.

get_agent_brief combines everything into a work order for a coding agent, with guardrails, acceptance criteria and a way to verify. The agent then reports back on every fix with a note, and the next brief shows that status. How this works in detail is described in website speed optimization with AI coding agents.

Manage projects, monitoring and competitors

An agent can set up your monitoring without you opening the web app:

- **create_project**, **add_urls** and **remove_url** create projects and maintain their URLs, up to the limit of your plan.
- **set_monitoring** switches regular scans of a URL on or off.
- **add_competitor**, **remove_competitor** and **list_competitors** maintain the competitors of a project. They are compared through field data, which costs no scans.
- **list_projects** lists your projects with the number of their URLs. **get_project** and **list_urls** describe a project with settings, URLs and the latest scan of each.
- **get_history** returns the score history of a URL, **get_monitoring_events** the alerts: regressions, improvements and budget changes of the last 90 days.

Check performance budgets after a deploy: set_budget and get_budget_status

With **set_budget**, an agent sets limits per project or URL, for example for JavaScript weight, lab LCP or field INP, optionally with a goal and automatic tightening. **get_budget_status** checks a finished scan against these limits and answers passed or failed. This lets an agent check a publicly reachable preview or production URL after a deploy. In a CI pipeline without an MCP client, the REST API does the same from the Solo plan. Our performance budget page explains how to choose sensible limits.

Notes on scans: add_note, update_note, delete_note

Notes record what changed on the website, such as "new theme deployed". **add_note** attaches a note of up to 280 characters to one of your scans, **update_note** changes its text and **delete_note** removes it again. Notes appear in the report history and in `get_report_details`. Coding agents also use them to report back on the agent brief in the format `PD-<id> done: …`.

Costs, plans and technical details

- **Costs:** `scan_url` uses one PageSpeed test per URL and device from the monthly budget that the web app, monitoring and MCP share. The other 24 tools are free.
- **Plans:** MCP and the REST API on the free plan with one scan every 24 hours and 15 tools, the full scope from Solo. The budgets per plan are on the pricing page.
- **Protocol:** Streamable HTTP with JSON-RPC 2.0, without sessions. Every tool states through annotations whether it only reads, changes or deletes something. Clients can use this to ask for confirmation before tools such as `remove_url`, `set_budget` or `delete_note`; whether they do is up to the client.
- **How it differs:** the Chrome DevTools MCP server drives a local browser, records traces and runs Lighthouse audits. PerfDelta measures without a local browser and keeps the results: history, monitoring, competitor comparison and budgets across many scans.

Limits of the MCP server

- **Public addresses only:** PageSpeed Insights only measures publicly reachable http(s) URLs. Localhost or a staging environment behind a login cannot be scanned.
- **Daily limit:** manual scans through the web app and MCP are capped per day, at 120 PageSpeed tests on the Solo plan.
- **Monitoring comes first:** manual scans can only use the part of the monthly budget that scheduled monitoring does not need until the end of the month. `get_usage` shows both values.

- **Requests per minute:** each key allows 60 calls per minute; after that, the server answers with a waiting time.
- **One project on the Solo plan:** multiple projects are available from Agency.

Examples of instructions to an agent: "Scan example.com on mobile and explain the worst Core Web Vital." "Why did the score drop since yesterday?" "Create a project with these ten URLs, switch on monitoring and set a JavaScript budget of 500 KB." Without an account, the free page speed test shows what a report contains.