MCP HARDENING RECORD

Regulations.gov MCP Testing Record

Rulemaking and docket MCP hardened across three live audit rounds including a 40-probe deep stress

Skill
regulationsgov-mcp v0.2.0
github.com/1102tools/regulationsgov-mcp

Date
April 2026
1102tools.com

Executive Summary

This Model Context Protocol server exposes the Regulations.gov API as 8 callable tools for federal rulemaking dockets, proposed rules, final rules, public comments, and comment-period tracking. It was hardened across three audit rounds (a broad live sweep, response-shape probes, and a deep live stress round). The signature finding was the most dramatic bug of the entire 8-MCP suite: `agency_id=""` (empty string) silently returned all 1,951,938 Regulations.gov documents because the empty string was treated as "no filter." The MCP ships with 51 regression tests (46 offline plus 5 live-gated).

Metric	Value
MCP tools exposed	8
Total regression tests	51 (46 offline, 5 live-gated)
Audit rounds completed	3
P0 catastrophic bugs found and fixed	1 (`extra='ignore'` silent typo drop)
P1 silent-wrong-data bugs found and fixed	10
P2 validation gaps found and fixed	7
P3 cleanup items found and fixed	4
Current release	0.2.0
PyPI status	Published as `regulationsgov-mcp`, auto-publishes via Trusted Publisher on tag push (publisher was reconfigured mid-release after a misconfiguration pointed it at the wrong repo)

What Was Tested

The MCP exposes 8 tools covering the Regulations.gov API surface. Testing covered all of them end-to-end.

Core: `search_documents`, `get_document_detail`, `search_comments`, `get_comment_detail`, `search_dockets`, `get_docket_detail`

Workflow: `open_comment_periods`, `far_case_history`

Each tool was exercised for argument validation, input sanitization (null bytes, script injection, path-traversal IDs), empty-string filter detection, date format checking, pagination edge cases, response-shape guarantees, error translation (WAF 403 vs auth 403 disambiguation), and real-world data handling against the live production Regulations.gov API with a real `api.data.gov` key.

How It Was Tested

Testing discipline

This MCP had never been hardened before the session; 0.1.1 was a first-pass release. The hardening program invoked tools through `mcp.call_tool(name, kwargs)` the way a real MCP client does, paired with a live audit using a real `api.data.gov` API key (same key tier as `gsa-perdiem-mcp`). Live testing was critical: mocks could not have surfaced the 1.95-million-record bug or the WAF-vs-auth 403 error message confusion.

Audit rounds

Round	Scope	Probe count	Finding class
1	Broad live sweep across all 8 tools	Live probes per tool	1 P0, 8 P1, 6 P2, 3 P3
2	Response-shape probes and workflow tools	Targeted shape probes	2 additional P1, 1 P2
3	Deep live stress (40 probes targeting pagination, concurrency, compound filters)	40 probes	No new bugs; 1 UX enhancement (`paged_past_end` flag)

Live audit status

All three rounds included live calls against the production Regulations.gov API. The repository includes 5 live-gated regression tests executable via `REGULATIONS_GOV_LIVE_TESTS=1 REGULATIONS_GOV_API_KEY=... pytest` covering real document search, docket detail, comment detail, open comment periods, and FAR case history. The key is free at `api.data.gov` (1000 req/hr) and is the same key tier as `gsa-perdiem-mcp`.

Issues Found and Fixed

Priority 0: Catastrophic silent wrong data

One bug in this class, the `extra='forbid'` cross-fix:

Issue	Fix
Unknown parameters silently dropped via pydantic's default `extra='ignore'`. Verified live: `search_documents(agency_id="FAR", bogus_typo="x")` succeeded with the typo dropped and returned unfiltered documents.	`extra='forbid'` applied to every tool's pydantic arg model (cross-fix invented in sam-gov-mcp 0.3.1 and back-ported to this MCP in 0.2.0). Typos now raise "Extra inputs are not permitted" before the HTTP call.

Priority 1: Silent wrong data

Ten bugs in this class. The headliner is #1 below, the most dramatic bug of the entire MCP suite.

Issue	Fix
`agency_id=""` returned all 1,951,938 Regulations.gov documents. The empty string was treated as "no filter" and the entire 1.95-million-record corpus was streamed back. Users thinking they were filtering by agency silently got the whole thing. Same failure-mode category later found in `usaspending-gov-mcp` `search_awards()` with no filters.	Empty-string and whitespace-only filter values now raise "agency_id cannot be empty" with pointer to valid agency codes.
Unknown agency IDs ("ZZZ", "FAR DOD") silently returned 0 results with no warning. User could not tell if the agency did not exist, had no documents, or if the query was malformed.	Known-agency list validation added; unknown agencies raise actionable error with suggestion to check `list_agencies`.
Null byte in `search_term` silently accepted and reached the API. Returned 174,000 results with no indication the null byte changed interpretation.	Null bytes and other control characters rejected locally.
`<script>` in `search_term` triggered the WAF and returned HTTP 403, but the server error message said "API key rejected." This was misleading because users saw an auth error when it was a WAF block.	WAF vs auth 403 disambiguation added via header inspection; WAF 403s now surface as "WAF blocked request (not an auth issue)".
Date-swapped range (`posted_date_ge > posted_date_le`) returned 0 silently. No error, just empty results.	Reversed ranges raise actionable error.
`_get` returning None passed through with no response-shape defense; downstream consumers crashed on None.	`_safe_dict` and `_as_list` helpers added throughout.
Bogus sort fields returned API 400 but the MCP did not pre-validate. User got an opaque API error instead of clear guidance.	Sort field validated against the known sort-set with a list of valid options in the error message.
`document_id` with slashes produced HTTP 500 or 301 responses. Path traversal attempt not rejected locally.	ID format validated; slashes and other path characters rejected.
Dates like "2025-01-01T00:00:00Z", "2025/01/01", "not-a-date" reached the API as 400s.	`YYYY-MM-DD` regex enforced at the arg layer.

Issue	Fix
`last_modified_date` format "YYYY-MM-DD HH:MM:SS" was not pre-validated and reached the API as 400.	Format enforced consistently across all date fields.

Priority 2: Validation gaps

Seven bugs in this class:

Issue	Fix
No defensive response parsing (no `_safe_dict` or `_as_list` helpers) existed.	Full defensive-parsing helper set added.
`search_term` was not length-clamped; 2000 characters reached the API.	Capped at 500 characters.
`page_number` was not bounds-checked locally (API returned 400 on 0 or negative).	Bounded to `>= 1` locally.
`open_comment_periods(agency_ids=[])` fell through to a default list silently.	Empty list rejected; at-least-one-agency enforced.
No `_clean_error_body` helper; raw HTML 500 responses leaked into error messages.	Added; HTML bodies now surface as a clean "Regulations.gov returned an HTML error" message.
Document, docket, and comment IDs were not length-clamped or format-validated.	All ID fields validated and clamped.
`agency_ids` list elements were not validated per-element; empty strings were silently kept.	Per-element validation; empty strings rejected.

Priority 3: Cleanup items

Four items including a missing `.github/workflows/publish.yml` (Trusted Publisher was never set up at initial shipping; had to be configured mid-session), a missing `tests/test_validation.py` (only a raw-coroutine `stress_test.py` existed), no `[dependency-groups].dev` in `pyproject.toml`, and a stale USER_AGENT at `0.1.1`. All resolved.

Round 3 UX enhancement: `paged_past_end` flag

Round 3 deep stress surfaced a UX issue that was not a bug: `page_number=20` with `page_size=250` on a 2,152-record collection returned empty data with `total=2152`. The `no_data` flag only triggered when `total` was 0 or None. For "paged past the end" the tool silently showed empty data with a non-zero total. Added a `paged_past_end: true` flag with an explicit "Last page with data is page 9" message. Pattern applied retroactively to `gsa-calc-mcp` pagination.

Response-shape defense

The `_safe_dict`, `_as_list`, and `_clean_error_body` helpers (added as a complete set in 0.2.0) now wrap every Regulations.gov response parsing path. Prior to 0.2.0 there was no defensive parsing at all; any unusual response shape crashed downstream consumers.

Test Coverage

The repo ships 51 regression tests. All 51 pass on every release cycle.

File	Purpose	Test count
`tests/test_validation.py`	Main regression suite covering every round finding, plus 5 live-gated integration tests	51
`tests/stress_test.py`	Round 1 broad live sweep (retained for reproducibility)	N/A (scenario script)
`tests/stress_test_r3.py`	Round 3 deep live stress (40 probes) including the `paged_past_end` scenario (retained for reproducibility)	N/A (scenario script)

Regression tests invoke tools through the FastMCP registry (`mcp.call_tool`). This is the discipline that surfaced the silent typo drop, the empty-string no-filter bug, and the WAF-vs-auth 403 confusion. An autouse fixture resets the shared httpx client between tests.

Release History

Version	Focus	Outcome
0.1.1	Initial release. This MCP had never been hardened before the session.	First-pass release with latent bugs
0.2.0	Full hardening: 22 bugs fixed across 3 rounds. 51 regression tests (46 offline + 5 live-gated) with real `api.data.gov` key. First Trusted Publisher configuration for this project (had to reconfigure mid-release because PyPI Trusted Publisher was pointing at the wrong repo, causing publish workflows to fail with "invalid-publisher" errors until caught and corrected).	1 P0, 10 P1, 7 P2, 4 P3 resolved

Cross-MCP Context

This MCP is one of eight servers in the 1102tools federal-contracting MCP suite (`bls-oews-mcp`, `ecfr-mcp`, `federal-register-mcp`, `gsa-calc-mcp`, `gsa-perdiem-mcp`, `sam-gov-mcp`, `usaspending-gov-mcp`, and this one). All eight were hardened under the same playbook. Patterns reused or established here:

- **The "empty string treated as no filter" bug class** was caught systematically here first. `agency_id=""` returned all 1,951,938 records. Same failure mode later found in `usaspending-gov-mcp` `search_awards()` with no filter arguments returning 25 unfiltered contracts. Same fix shape in both.
- **`extra='forbid'` cross-fix** back-ported from sam-gov-mcp 0.3.1.
- **`paged_past_end` flag pattern** introduced here in 0.2.0 and later applied to `gsa-calc-mcp` pagination.
- **WAF vs auth 403 disambiguation** (inspect headers, do not guess from the error body) was codified here.

What Was Not Tested

- **Rate-limit behavior at scale.** The `api.data.gov` free tier caps at 1000 req/hr. The MCP surfaces 429s but does not implement client-side throttling.

- **Attachments beyond metadata.** `get_document_detail` with attachments returns attachment metadata and URLs. Binary attachment download is not exercised end-to-end.
- **Regulations.gov's `beta` endpoints.** The MCP targets the stable v4 API. Any beta endpoints are out of scope.
- **Bulk comment analysis.** Comments can be voluminous on high-interest rules; the MCP paginates but does not implement bulk-download or streaming.

Verification

All testing artifacts are in the repository. The methodology and fixes are reviewable commit-by-commit in git history. The regression test suite runs via `pytest` in the repo root and can be re-executed by anyone. The live suite runs with `REGULATIONS_GOV_LIVE_TESTS=1 REGULATIONS_GOV_API_KEY=... pytest` using a free `api.data.gov` key.

> **Testing Methodology**
>
> Evaluators: James Jenrette, 1102tools, with Claude Code Opus 4.7 (1M context, max effort, Claude Max 20x subscription) during the hardening playbook execution.
>
> Testing spanned three rounds: a broad live sweep across all 8 tools (most of the findings), response-shape probes and workflow-tool coverage (a few additional findings), and a 40-probe deep live stress round (no new bugs, one UX enhancement). The live regression suite runs against the production Regulations.gov API when enabled with `REGULATIONS_GOV_LIVE_TESTS=1`.
>
> Test count: 51 regression tests. P0 catastrophic bugs found and fixed: 1. P1 silent-wrong-data bugs found and fixed: 10. P2 validation gaps closed: 7. P3 cleanup items closed: 4. Total findings: 22. Current version: 0.2.0. PyPI: `regulationsgov-mcp`.
>
> Source: github.com/1102tools/regulationsgov-mcp. License: MIT.