MCP HARDENING RECORD

GSA CALC+ MCP Testing Record

Labor ceiling rates MCP hardened across four retroactive live audit rounds

Skill
gsa-calc-mcp v0.2.2
github.com/1102tools/gsa-calc-mcp

Date
April 2026
1102tools.com

Executive Summary

This Model Context Protocol server exposes the GSA CALC+ Labor Ceiling Rates API as 8 callable tools for IGCE development, price reasonableness analysis, and federal labor market research. It was hardened across four live audit rounds plus an initial WAF-calibration pass. Testing surfaced 86 bugs total (74 in the initial full audit plus 12 in retroactive deep audits), including the signature `filtered_browse()` bug that returned 265,000 unfiltered records on a zero-argument call. The MCP ships with 117 regression tests (109 offline plus 8 live-gated) that run on every change and can be executed against the real public CALC+ API on demand.

Metric	Value
MCP tools exposed	8
Total regression tests	117 (109 offline, 8 live-gated)
Audit rounds completed	4 retroactive live rounds + initial WAF pass
P1 crashes (shape-shift) found and fixed	19
P1 silent-wrong-data bugs found and fixed	30
P2 validation gaps found and fixed	19
P3 cleanup items found and fixed	6
Retroactive additional findings	12
Current release	0.2.2
PyPI status	Published as `gsa-calc-mcp`, auto-publishes via Trusted Publisher on tag push

What Was Tested

The MCP exposes 8 tools covering the CALC+ Elasticsearch-backed API. Testing covered all of them end-to-end.

Core search: `keyword_search`, `exact_search`, `suggest_contains`, `filtered_browse`

Workflow tools: `igce_benchmark`, `price_reasonableness_check`, `vendor_rate_card`, `sin_analysis`

Each tool was exercised for argument validation, input sanitization, response-shape guarantees, error translation, pagination edge cases, Elasticsearch result-window limits, and real-world data handling against the live production CALC+ API.

How It Was Tested

Testing discipline

Prior unit tests in v0.1.x awaited raw coroutines and mocked the HTTP layer, which bypassed the FastMCP tool pipeline and skipped whole classes of integration bugs. The hardening program switched to invoking tools through `mcp.call_tool(name, kwargs)` the way a real MCP client does, paired with live calls against the production CALC+ API. That change surfaced the `filtered_browse` unfiltered-return bug and the `sin=True` pydantic coercion bug, neither of which was visible from mocked tests.

Audit rounds

Round	Scope	Probe count	Finding class
Initial (pre-0.2.1)	WAF filter calibration with mocked tests	Calibration only	WAF relaxation plus `extra='forbid'` cross-fix
Retro Round 1	Live probing across all 8 tools	Live probes per tool	Control-char slippage, `filtered_browse` unfiltered, `sin` bool trap
Retro Round 2	Compound filters, pagination, NaN/Inf, concurrency	Live stress probes	Response-shape guard gaps, WAF vs exclude parameter, 265K unfiltered fix
Retro Round 3	Length caps, page × page_size overflow	Live probes	`suggest_contains.term` unbounded, ES 10k window overflow
Retro Round 4	Response-shape mock fuzzing	15+ shape probes	Confirmed defensive guards hold; no new findings

Live audit status

All retroactive rounds included live calls against the production CALC+ API. The repository includes 8 live-gated regression tests executable via `CALC_LIVE_TESTS=1 pytest` covering real wildcard search, exact-match lookup, IGCE benchmark stats, price reasonableness, vendor rate card, SIN analysis, and filtered browse with real filters applied. No API key is required; CALC+ is a free, public API behind a WAF.

Issues Found and Fixed

Priority 1: Silent wrong-data bugs

Thirty bugs in this class. Representative and signature items below.

Issue	Fix
`filtered_browse()` with zero arguments silently returned the entire 265,000-record CALC+ dataset. Same failure-mode category as the regulationsgov-mcp `agency_id=""` 1.95M-record bug.	At-least-one-filter guard raises "filtered_browse requires at least one of: labor_category, schedule, sin, business_size, education_level, experience_min, experience_max, price_min, price_max".
Control characters (null byte, newline, carriage return, tab, backspace) slipped through every free-text field: `keyword`, `value`, `term`, `labor_category`, `vendor_name`, `exclude`. URL-encoded and sent silently to the API.	All free-text fields reject control characters up front via a shared `_validate_text` helper.
`sin=True` was silently coerced to `sin=1` by pydantic's implicit bool-to-int conversion before any validator ran. The value `"True"` passed the alphanumeric regex and became `filter=sin:1` to the API, returning zero matches.	`BeforeValidator` added to `Union[str, int, None]` fields to reject `bool` at the type layer. Pattern now reused across the suite.
`proposed_rate=NaN` produced bogus `price_reasonableness_check` output: `vs_median="equal"` and `iqr_position="above P75"` because NaN comparisons all fall to the `else` branch.	Finite-number check enforced on all float inputs. NaN and Inf rejected at the arg layer.
`proposed_rate=Inf` passed pydantic's `float` type (no finite constraint) and leaked into `z_score` and `delta` outputs.	Same finite check applied.
`price_min=NaN` or `price_max=Inf` passed pydantic and hit the API as a 406.	Same finite check.
`exclude` parameter was not WAF-checked, not control-char checked, and not length-capped. `exclude=<script>` was not pre-rejected.	Same `_validate_text` plus the WAF-angle-bracket filter.
Pagination past the end of results silently returned empty. `page=100` of a 2076-record query returned 0 records with no `paged_past_end` flag.	Page-past-end detection added; response now includes a clear flag and guidance.

Issue	Fix
Elasticsearch 10k-result window: `page_size × page > 10000` returned a cryptic 406 "Result window too large".	Pre-clamp added. Combined `page_size × page` is bounded locally before the HTTP call with a clear error. Pattern now reused across ES-backed MCPs in the suite.

Priority 1: Crashes and shape-shift defenses

Nineteen bugs in this class. The `_extract_stats` helper crashed on multiple unusual Elasticsearch aggregation shapes that CALC+ occasionally returns under load:

Issue	Fix
`hits.total` returned as bare int instead of `{"value": N, "relation": "eq"}` object shape. Triggered `AttributeError`.	`_safe_dict` and `_safe_number` helpers normalize both shapes.
`aggregations` returned with null value. `.get()` crashed with `AttributeError`.	Null-coalescing throughout aggregation parsing.
`vendor_rate_card` `hits.hits` returned as null. Triggered `TypeError` on iteration.	`_as_list` helper returns empty list for null or missing.
`_source` field was null in individual hits. `AttributeError` on member access.	Same `_safe_dict` guard.
`suggest_contains` bucket missing the `key` field. Triggered `KeyError`.	Key presence checked before access; missing keys skipped with logged warning.
`price_reasonableness_check` with `avg=0` and `median=None` produced a misleading "above" comparison.	Explicit null and zero checks before comparison; output includes a `reason` field when inputs are degenerate.

Priority 2: Validation gaps

Nineteen bugs in this class. Representative items:

Issue	Fix
`igce_benchmark.labor_category` had no length cap. 600-character strings produced upstream 406s.	Capped at 200 characters.
`suggest_contains.term` had no length cap.	Capped at 200 characters.
`vendor_rate_card.vendor_name` had no length cap.	Capped at 200 characters.
`sin_analysis.sin_code` had no length cap.	Capped at 50 characters.
`experience_max` alone (without `experience_min`) was silently ignored. No half-range filter was built.	Half-range filters now construct correctly in both directions.
Reversed ranges (`price_min > price_max`, `experience_min > experience_max`) were sent raw to the API.	Reversed ranges raise actionable error locally.
Negative `price_min=-50` was accepted and produced `price_range:-50,99999` (pulls everything).	Non-negative constraint enforced.
`price_max=0` produced `price_range:0,0` which matched only $0 rates (useless).	Minimum `price_max=0.01` enforced when `price_min` is not set.
Hardcoded upper bound of 99999 on `price_min`-only queries excluded rates above $99,999/hr.	Upper bound lifted to 999999; documented.
Empty or whitespace-only `keyword` silently returned the full 250K dataset.	Minimum 1-character non-whitespace enforced.
Bogus `education_level="XYZ"` silently accepted and returned 0 records.	Validated against the known education-level set.
`education_level` was case-sensitive at the API; "ba" vs "BA" silently filtered to nothing.	Normalized to uppercase at the arg layer.
`page=0`, `page=-1` were accepted locally and rejected by the API with 406.	Bounded locally to `>= 1`.

Response-shape defense

The `_safe_dict`, `_as_list`, and `_safe_number` helpers now wrap every Elasticsearch aggregation path. CALC+ occasionally returns unusual shapes under load that previously produced type-confusion crashes. All shape variants now normalize cleanly, with a logged warning when an unexpected shape is observed. This defensive-parsing pattern was codified here and reused across the other ES-backed MCPs in the suite.

Test Coverage

The repo ships 117 regression tests across the test folder. All 117 pass on every release cycle.

File	Purpose	Test count
`tests/test_validation.py`	Main regression suite covering every round-1 through round-4 finding, plus 8 live-gated integration tests	117
`tests/stress_test.py`	Retro Round 1 live-probe scenarios (retained for reproducibility)	N/A (scenario script)
`tests/stress_test_r2.py`	Retro Round 2 compound-filter and pagination scenarios	N/A (scenario script)
`tests/stress_test_r3.py`	Retro Round 3 length cap and ES-window scenarios	N/A (scenario script)

Regression tests invoke tools through the FastMCP registry (`mcp.call_tool`) rather than awaiting decorated coroutines directly. This catches bugs in the tool pipeline that raw-coroutine tests miss. An autouse fixture resets `srv._client` between tests so the shared httpx client does not leak across event loops.

Release History

Version	Focus	Outcome
0.1.1	Initial release: 8 tools with basic unit tests	Basic coverage
0.2.1	Minimal cross-fix: WAF filter relaxation (apostrophes, SQL keywords) plus pydantic `extra='forbid'` applied to every tool arg model (back-ported from sam-gov-mcp 0.3.1)	Calibrated WAF, typo'd-param silent drop fix
0.2.2	Full retroactive 4-round audit through the live CALC+ API: 86 bugs fixed (74 initial full audit + 12 retro deep audit); 117 regression tests including 8 live-gated	19 P1 crashes, 30 P1 silent-wrong-data, 19 P2, 6 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-perdiem-mcp`, `regulationsgov-mcp`, `sam-gov-mcp`, `usaspending-gov-mcp`, and this one). All eight were hardened under the same playbook. Patterns that originated or propagated through this MCP:

- **The `_safe_dict`, `_as_list`, `_safe_number` defensive-parsing helpers** were refined here for Elasticsearch-backed APIs and reused across the suite.
- **The `BeforeValidator` bool-rejection pattern** on `Union[str, int, None]` fields was invented here. Pydantic silently coerces `bool` to `int` before any custom validator runs, so the `sin=True` bug required rejecting `bool` at the type layer. Pattern now reused across the suite.
- **The Elasticsearch 10k-result window pre-clamp pattern** was established here for ES-backed APIs. Applied wherever an API is known to be ES-backed.
- **The finite-number check on float params** was codified here because pydantic has no finite constraint on `float` (unlike its `conint` variants).
- **WAF filter relaxation** was calibrated here against real CALC+ behavior: CALC+ WAF-blocks angle brackets and path traversal but accepts apostrophes and SQL keywords. This is different from SAM.gov's WAF, so this MCP's WAF filter is tuned specifically to CALC+.

What Was Not Tested

- **Rate-limit behavior.** CALC+ does not document rate limits publicly. The MCP passes through whatever limits the API enforces but does not implement client-side throttling.
- **WAF drift.** GSA may tighten the CALC+ WAF over time. The MCP's WAF-aware filter was calibrated in April 2026; future WAF changes will be caught only via live-gated tests.
- **Historical rate data.** CALC+ surfaces current awarded rates. Historical award data requires separate queries that are not exposed by this MCP.
- **Payload size limits.** Response sizes on `filtered_browse` or `keyword_search` can be large if the caller accepts the default shape. The MCP bounds per-page results but does not enforce an overall payload ceiling.

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 `CALC_LIVE_TESTS=1 pytest` and requires no API key (CALC+ is a free, public API).

> **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 four retroactive rounds plus an initial WAF-calibration pass. Rounds covered live probing across all 8 tools, compound-filter and pagination edge cases, length caps and ES window overflow, and response-shape mock fuzzing. The live regression suite runs against the production CALC+ API when enabled with `CALC_LIVE_TESTS=1`.
>
> Test count: 117 regression tests. P1 crashes found and fixed: 19. P1 silent-wrong-data bugs found and fixed: 30. P2 validation gaps closed: 19. P3 cleanup items closed: 6. Retroactive additional findings: 12. Current version: 0.2.2. PyPI: `gsa-calc-mcp`.
>
> Source: github.com/1102tools/gsa-calc-mcp. License: MIT.