MCP HARDENING RECORD

GSA Per Diem MCP Testing Record

Per diem rates MCP hardened across six audit rounds with a live real-key audit

Skill
gsa-perdiem-mcp v0.2.1
github.com/1102tools/gsa-perdiem-mcp

Date
April 2026
1102tools.com

Executive Summary

This Model Context Protocol server exposes the GSA Per Diem Rates API as 6 callable tools for federal travel lodging and M&IE rate lookups used in IGCEs and travel cost estimation. It was hardened across six audit rounds, culminating in a round-6 live audit with a real `api.data.gov` key that surfaced three catastrophic silent-wrong-data bugs that could never have been caught with mocks, including a typographic-apostrophe bug that silently returned Andover rates for Martha's Vineyard queries. Testing surfaced 55 bugs total. The MCP ships with 172 regression tests (164 offline plus 8 live-gated) that run on every change.

Metric	Value
MCP tools exposed	6
Total regression tests	172 (164 offline, 8 live-gated)
Audit rounds completed	6
P0 catastrophic bugs found and fixed	1 (path traversal)
P1 silent-wrong-data bugs found and fixed	23
P2 validation gaps found and fixed	21
P3 cleanup items found and fixed	10
Current release	0.2.1
PyPI status	Published as `gsa-perdiem-mcp`, auto-publishes via Trusted Publisher on tag push

What Was Tested

The MCP exposes 6 tools covering the GSA Per Diem API surface. Testing covered all of them end-to-end.

Core lookups: `lookup_city_perdiem`, `lookup_state_rates`, `lookup_zip_perdiem`, `get_mie_breakdown`

Workflow tools: `estimate_travel_cost`, `compare_locations`

Each tool was exercised for argument validation, input sanitization, URL encoding, city-name normalization across punctuation variants, response-shape guarantees, error translation, and real-world data handling against the live production API with a real `api.data.gov` key.

How It Was Tested

Testing discipline

Prior unit tests in v0.1.x awaited raw coroutines and mocked the HTTP layer. The hardening program switched to invoking tools through `mcp.call_tool(name, kwargs)` the way a real MCP client does. More important: the round-6 live audit with a real `api.data.gov` key was the critical step that surfaced three P1 silent-wrong-data bugs that mocks would never catch. The "run a live audit with a real API key, not just mocks" discipline was formalized here and applied to every other MCP in the suite.

Audit rounds

Round	Scope	Probe count	Finding class
1	Live probes with DEMO_KEY (rate-limit constrained)	Initial surface	Bug surface identified across all 6 tools
2	Deeper probes, response-shape fuzz	Crash probes	20 response-shape crash paths identified
3	Validation gap audit	Arg-layer probes	21 P2 validation issues
4	Static review	Code review	10 P3 polish items
5	Initial patches shipped at 0.2.0 with 52 bugs fixed	164 offline tests	First integration of all prior findings
6	Live audit with a REAL `api.data.gov` key	Targeted live probes	3 additional P1 silent-wrong-data bugs, catastrophic

Live audit status

All six rounds included live calls against the production Per Diem API. The repository includes 8 live-gated regression tests executable via `PERDIEM_LIVE_TESTS=1 PERDIEM_API_KEY=... pytest` covering real city lookup, ZIP lookup, state rates, M&IE breakdown, travel cost estimation, and location comparison with the full normalization and fallback-detection stack exercised. The `api.data.gov` key is free (1000 req/hr) and not gated behind an approval workflow.

Issues Found and Fixed

Priority 0: Path traversal

One bug in this class, catastrophic.

Issue	Fix
`urllib.parse.quote(city)` was called with the default `safe='/'` argument, leaving `/` and `.` unencoded in URL paths. `city="../../admin"` produced a URL path like `/travel/perdiem/v2/admin/state/MA/year/2026`, hitting a different GSA endpoint entirely. `city="Boston/Cambridge"` similarly stayed unencoded and navigated out of the intended resource. Affected `lookup_city_perdiem`, `estimate_travel_cost`, and `compare_locations`.	All city names now URL-encoded with `safe=''`. Path-traversal probes in the regression suite verify all three affected tools.

Priority 1: Live-audit silent wrong data (the headliners)

Three bugs in this class, all surfaced only in the round-6 live audit with a real API key. These were catastrophic because the tool returned data that appeared legitimate but was for a different city entirely.

Issue	Fix
Typographic apostrophe not normalized. `city="Martha's Vineyard"` with a typographic apostrophe (U+2019, the curly one) became `"Martha s Vineyard"` after a partial normalization step. The match logic then compared the raw partially-normalized string against the NSA name list, found no match, and silently returned Andover, MA (137/night) as the match with no warning. A contracting officer building a travel IGCE would have pulled the wrong city's rate with no indication anything was wrong.	`_normalize_for_match()` helper treats apostrophes (straight and curly), hyphens, periods, and commas as equivalent to whitespace. All NSA names and user input pass through the same normalizer before comparison. Regression test covers U+2019 and U+2018.
Unmatched city silently falls back to first NSA in the list. When `query_city` did not match any NSA exactly or via substring, `_select_best_rate` silently returned `rates[0]`. Confirmed three ways: "Peñasco, NM" returned Taos, "Santa Rosa Beach, FL" returned Fort Walton Beach, "St Louis, MO" (without period) returned Kansas City.	`match_type` field added to every response: `exact`, `composite`, `standard_fallback`, `unmatched_nsa`, or `first_nsa`. `match_note` field provides human-readable guidance. No silent fallbacks.
Punctuation-sensitive matching. "St Louis" (no period) silently fell back via the same unmatched-fallback bug.	Normalization treats "St", "St.", and "Saint" as equivalent. Normalizes across periods, commas, and hyphens.

Priority 1: Response-shape crashes

Twenty bugs in this class. The XML-to-JSON shape collapse from the Per Diem API produced many crash paths. Representative items:

Issue	Fix
`_parse_rate_entry`: `entry.get("months", {}).get("month", [])` crashed with `AttributeError` when `months` was None.	Null-coalescing throughout: `(entry.get("months") or {}).get("month") or []`.
`_parse_rate_entry`: single-item XML-to-JSON collapse produced `months.month` as a single dict. Iteration yielded string keys that crashed downstream `.get()` calls.	Dict-to-list coercion added so single items are treated as a length-1 list.
`_parse_rate_entry`: if any month value was None, `min(values)` raised `TypeError: '<' not supported between int and NoneType`.	None values filtered before aggregation; if all are None, tool returns a clear "no data for this month" rather than silently producing 0.
`_parse_rate_entry`: if a month value was `"null"` or `""`, `int(val)` raised and code silently fell back to 0.	String-to-int now only accepts digit strings; other shapes produce a clear error or None rather than silent zero.
`_parse_rate_entry`: if `meals` was None, it was returned as-is and downstream arithmetic broke.	Explicit None check; missing meals returns None with a clear flag.
`_select_best_rate`: `response.get("rates", [])` crashed if response was None or a list.	`_safe_dict` helper guards the access.
`_select_best_rate`: None entries inside the rate list crashed `.get()`.	Entry filtering removes None before iteration.
`_select_best_rate`: single-item collapse where `rates[0].rate` was a dict instead of a list broke iteration.	Same dict-to-list coercion.
`lookup_state_rates`: `response.get("rates", [])` crashed on None response.	Guarded.
`get_mie_breakdown`: `data.get(...)` crashed on None response and iterated tiers without null checks.	Guarded throughout.

Issue	Fix
`get_mie_breakdown`: `t.get("total", 0) * 0.75` raised `TypeError` when total was string or None.	Numeric coercion with `_safe_number`.
`_get`: `r.json()` raised `JSONDecodeError` on HTML (maintenance pages, redirects), empty body, truncated body. Not caught.	Content-type inspection and clear error translation.
`_parse_rate_entry`: `rates[0].rate` containing an entry with missing city caused `p["city"].lower()` to crash on None.	Missing-city handling added.
`compare_locations`: 200+ locations × 0.3s sleep × API rate limits produced catastrophic delays. No bounds on input list.	Length cap enforced (50 locations); concurrent fetching with a bounded semaphore.
`compare_locations`: entries processed sequentially rather than concurrently.	Now concurrent with bounded concurrency.
`estimate_travel_cost`: `travel_month` accepted any string and only matched exact 3-letter short names. "January", "1", "jan" (lowercase) silently used the maximum monthly rate.	Month normalization accepts full name, 3-letter, 1-based int, or mixed case.
`estimate_travel_cost`: `nightly * num_nights` with `nightly=0` from bad data returned a misleading `lodging_total=0`.	Zero-rate detection flags the response with `reason="no_rate_available"`.
`_parse_rate_entry`: flag logic used `entry.get("city", "") == "Standard Rate"` which was fragile against alternate "standardRate" string shapes.	Flag logic now uses the `standardRate` field directly and normalizes truthy strings.
`_select_best_rate`: missing city-field handling was inconsistent.	Unified through the normalizer.

Priority 2: Validation gaps

Twenty-one bugs in this class. Representative items:

Issue	Fix
`fiscal_year` unbounded on `lookup_city_perdiem`, `estimate_travel_cost`, `compare_locations`. Accepted -1, 0, 1900, 9999, 10^20.	Bounded to 2015 through current_year+1 with clear errors.
`city` had no length limit (500-char tested) and accepted null byte (API 400) and newline (API 500).	Capped at 200 characters; control chars rejected.
`city` with emoji or non-ASCII forwarded; API returned 500.	Unicode normalization plus control-char rejection.
`state` not validated as USPS code. "ZZ" allowed.	Validated against USPS two-letter state set.
`state` with trailing whitespace like " MA " was silently trimmed but not validated otherwise.	Trim plus USPS validation.
`num_nights` had no upper bound. 1M accepted and produced nonsense multi-million-dollar totals.	Bounded to 1 through 365.
`travel_month` not validated against the 12-month set; any string quietly fell through.	Validated against normalized month set.
`compare_locations` had no cap on input list length.	Capped at 50 locations.
Inside the `compare_locations` loop, exceptions were swallowed as `str(e)[:100]`.	Per-location errors now captured as a structured sub-response with the actual error surfaced.
`lookup_zip_perdiem` rejected "02101-1234" (ZIP+4 format). USPS ZIP+4 is common and should be accepted.	ZIP+4 truncated to first 5 digits.
`api_key=""` empty silently downgraded to DEMO_KEY.	Empty string rejected; unset falls back to DEMO_KEY explicitly with a logged warning.
`api_key` was not URL-encoded in the query string.	URL-encoded.

Issue	Fix
Docstring claimed "apostrophes and hyphens auto-replaced with spaces" but stripping the apostrophe from "Martha's Vineyard" produced "Martha s Vineyard" that silently mismatched.	Behavior and docstring now match: all punctuation variants normalized consistently before match.
No retry logic on 429.	Exponential backoff retry added for 429 responses.
`is_standard_rate` flag computed from city-string comparison was fragile.	Flag now derived from the API's `standardRate` field.

Priority 3: Cleanup items

Ten items including a hardcoded `DEFAULT_FISCAL_YEAR = 2026` (stale after Oct 2026 fiscal year rollover; now computed from the current date), a stale USER_AGENT at `gsa-perdiem-mcp/0.1.1`, an unclosed global `_client`, and minor code-health items. All resolved.

Test Coverage

The repo ships 172 regression tests across the test folder. All 172 pass on every release cycle.

File	Purpose	Test count
`tests/test_validation.py`	Main regression suite covering every round-1 through round-6 finding, plus 8 live-gated integration tests	172
`tests/stress_test.py`	Round 1 DEMO_KEY live-probe scenarios (retained for reproducibility)	N/A (scenario script)
`tests/stress_test_r6.py`	Round 6 real-API-key live audit scenarios including Martha's Vineyard, Peñasco, St Louis (retained for reproducibility)	N/A (scenario script)

Regression tests invoke tools through the FastMCP registry (`mcp.call_tool`) rather than awaiting decorated coroutines directly. 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: 6 tools with basic unit tests	Basic coverage
0.2.0	Full 52-bug fix across 5 audit rounds plus round-6 live audit that added 3 critical P1 silent-wrong-data bugs; 172 regression tests including 8 live-gated	1 P0, 23 P1, 21 P2, 10 P3 resolved
0.2.1	Cross-MCP fix: pydantic `extra='forbid'` applied to every tool arg model (back-ported from sam-gov-mcp 0.3.1)	+1 regression test

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`, `regulationsgov-mcp`, `sam-gov-mcp`, `usaspending-gov-mcp`, and this one). All eight were hardened under the same playbook. Patterns that originated here:

- **The "run a live audit with a real API key, not just mocks" discipline** was formalized here after round 6 surfaced three catastrophic bugs that mocks never caught. Applied to every other MCP in the suite.
- **The `_normalize_for_match()` helper** treating apostrophes (straight and curly), hyphens, periods, and commas as equivalent to whitespace was exported to other MCPs that do fuzzy-name matching.
- **The `match_type` and `match_note` response fields** (`exact` / `composite` / `standard_fallback` / `unmatched_nsa` / `first_nsa`) with human-readable warnings: this "unmatched fallback" anti-pattern was fixed here and the pattern informed similar guard-rails in other MCPs where substring match was being used.

What Was Not Tested

- **OCONUS rates.** This MCP covers CONUS per diem only. OCONUS rates come from the State Department and are on a different endpoint not exposed here.
- **Rate-limit behavior at scale.** DEMO_KEY has tight rate limits; a real `api.data.gov` key has 1000 req/hr. The MCP does not implement client-side throttling beyond the retry-on-429

path.

- **Fiscal year transition day.** October 1 rollover behavior against upstream was tested in principle but not live-audited across a real FY transition. Live-gated tests verify current-FY behavior.
- **Historical fiscal years beyond GSA's retained range.** GSA retains roughly 10 years of historical rates; queries beyond that window produce upstream 404s surfaced without prediction.

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 `PERDIEM_LIVE_TESTS=1 PERDIEM_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 six rounds including response-shape fuzzing, validation gap audit, static review, initial patch integration, and a round-6 live audit with a real `api.data.gov` key that surfaced three catastrophic P1 silent-wrong-data bugs. The live regression suite runs against the production Per Diem API when enabled with `PERDIEM_LIVE_TESTS=1`.
>
> Test count: 172 regression tests. P0 catastrophic bugs found and fixed: 1. P1 bugs found and fixed: 23. P2 validation gaps closed: 21. P3 cleanup items closed: 10. Total findings: 55. Current version: 0.2.1. PyPI: `gsa-perdiem-mcp`.
>
> Source: github.com/1102tools/gsa-perdiem-mcp. License: MIT.