

Making AI Omitted Scope Visible with VDG

Version 3.14

I designed an experiment to demonstrate omitted scope using a caching system in Python. It's just one file.

It generates reports, caches them in Redis, lets you check status, invalidate, and prefetch.

All four code paths require the same cache key.

In the experiment, the prompt was:

We're getting occasional cache key collisions in the report generation path. Switch `GENERATE_HASH_NAME` from MD5 to SHA-256. Here's the service file. Return the updated version.

In the baseline run, the model makes the named change and stops there, which is correct.

This demonstrates omitted scope.

Omitted scope means the request does not explicitly enumerate all the surfaces that need to change.

The response can be correct against the named instruction and still miss the broader intended scope.

The model updates the write path by changing `GENERATE_HASH_NAME`.

It does not carry the same change through to all relevant code paths.

The generate path writes a SHA-256 key, but the read paths continue to use the MD5-based key.

As a result, none of the read paths hit cache.

Again, the model responded correctly.

But if my intent was that all four code paths changed, it might look like the model made a mistake.

The unnamed read paths never changed.

The model did not reliably surface that this would break cache reads.

If my intent was only to change the generate method, the model behaved as desired.

Either intent is valid, but neither intent has been explicitly enumerated.

I created VDG because I needed a way to make omitted scope visible.

Let's look at the baseline and VDG responses from the experiment.

Baseline P6

Done. Line 14 changed from "md5" to "sha256". This only affects the `/reports/generate` endpoint's cache keys—the other hash configs (status, invalidation, prefetch) remain on MD5 as before.

P6 with VDG

Gap

Cache coherence on deployment: Switching hash algorithms mid-flight means `/reports/status`, `/webhooks/data-update`, and `/admin/prefetch` still use MD5-based keys; they will not find or invalidate reports cached under the new SHA-256 keys.

How VDG Helps

In this experiment, once VDG was introduced, the omitted scope appeared clearly under Gap.

VDG surfaces what is known, what is inferred, and what is assumed.

The model followed the instruction correctly. The instruction didn't cover the scope. That gap between what was named and what was needed is where production failures start, and it's exactly where VDG forces a pause. Not by changing what the model does, but by making what it didn't do impossible to miss.

P6 Responses

Full baseline P6 response

I'll update the `GENERATE_HASH_NAME` constant from MD5 to SHA-256 as requested.

Done. Line 14 changed from "md5" to "sha256". This only affects the /reports/generate endpoint's cache keys—the other hash configs (status, invalidation, prefetch) remain on MD5 as before.

Claude VDG response

Verified

Objective: Change `GENERATE_HASH_NAME` from "md5" to "sha256" to address cache key collisions in the report generation path.

Deduction

Changing `GENERATE_HASH_NAME = "md5"` to `GENERATE_HASH_NAME = "sha256"` will cause `_report_cache_key()` to return SHA-256 hashes instead of MD5 hashes for the report generation endpoint.

Gap

- Cache coherence on deployment: Switching hash algorithms mid-flight means `/reports/status`, `/webhooks/data-update`, and `/admin/prefetch` still use MD5-based keys; they will not find or invalidate reports cached under the new SHA-256 keys.

Get the [VDG Protocol](#) | [PDF](#)