Bulk Index Checking After Massive Content Pruning

When you excise 40% of a site's pages in a single weekend—think 80,000 old product variants or a blog graveyard from 2012—the immediate shriek isn't about 301 mapping. It's the void of not knowing what Googlebot actually kept. **Bulk index checking after massive content pruning** is the only way to prove the corpses are gone. Without that proof, you leak crawl budget on 410s, soft 404s, and ghost canonical chains that mess with your remaining index quality. A gut-check via manual URL Inspection in Search Console is fantasy at this volume. You need a mechanical, auditable, repeatable pipeline that answers one question across six figures of URLs: "Is it indexed, yes or no?"

I've done this dance on three e-commerce clean-ups where the product team deleted entire categories and expected SEO to "just handle it." In one case, 72,000 URLs were still flagged as indexed two weeks after a 410 rollout because we trusted a developer's spreadsheet and not an actual mass verification script. The correct approach is to automate the index check with rate limits, retries, and a clear exit condition—otherwise you're treading water in a sea of ambiguity.

The workflow that follows is opinionated and heavily skewed toward code because the Search Console interface is a toy for anything beyond 5,000 URLs. We'll rip through real bash loops, Python scripts hitting Google's URL Inspection API, and a third-party bulk checker that eats 100k URLs for breakfast. Edge cases, quota traps, and the mandatory art of splaying requests across time will all get the unvarnished treatment.

Why Manual Index Checks Collapse Under a Content Purge

The URL Inspection tool inside Google Search Console is a scalpel. You need a combine harvester. Your brain will short-circuit if you try to paste 500 URLs one by one, wait for the spinning loader, and log the result in Excel. The API is the only way out. Google's Inspect URL endpoint gives a programmatic yes/no on index status, but it's bound by a per-project quota of 2,000 calls per day—a figure that remains unchanged for most users. So a 100k URL audit would take 50 days if you only rely on that endpoint. That's a scheduling nightmare.

Most large-scale audits collapse because the operator doesn't factor in throttling, authorization tokens that expire after an hour, and the simple fact that the API may return 429 with zero human-readable warning. The 429 doesn't just slow you down. It poisons your run if you haven't saved state. Recovering from a failed batch without an accumulated checkpoint means starting over. That's why we build in idempotent logging and progressive checkpoints from the first line of code.

:::warning The URL Inspection API does not accept a bulk POST of multiple URLs. Each call is one URL. That architectural constraint forces you to either chain HTTP calls sequentially (slow but safe) or spread them across multiple service accounts (fast but risky if you flub the quota math). :::

Picking a Method for 20,000–200,000 URL Audits

Four methodologies exist, and they aren't equal. The file-based check with curl and a simple loop works for up to 5,000 URLs if you have a weekend. A Python script with asyncio and OAuth2 service accounts can push 2,000 URLs per day per project, and you might spin up multiple projects if you're truly desperate. Then there are purpose-built SaaS index checkers that abstract the quota juggling for you—SpeedyIndex's bulk endpoint and similar tools claim to process batches of 10,000 URLs in one POST, returning index statuses in seconds. The fourth option is the nuclear one: combine the Google Search Console API's searchanalytics:query data with click filters to infer index presence indirectly, but that's janky and unreliable for pruned pages that got zero impressions.

For full control, you marry a home-rolled Python script with a third-party bulk checker as a backup. The former gives you raw, auditable API responses directly from Google. The latter does the heavy lifting when you need to re-run the entire list in 20 minutes, not 50 days. The cost of the SaaS route is usually trivial compared to the engineering hours saved.

> Rule of thumb: If you pruned more than 5,000 pages, you cannot manually check their index status in the Search Console UI. You will burn out. Automate from hour zero.

Automated Bulk Checking with Google’s URL Inspection API (Python & Bash)

Let’s write working scripts you can copy-paste and run after setting up an API key and verifying site ownership. These are not conceptual; they are the exact loops we used to audit 72,000 URLs for a fashion retailer that had removed half its SKU landing pages.

First, the Python approach. It reads a plain text file of absolute URLs, sends each one to the urlInspection.index:inspect endpoint, and logs the indexStatusResult.coverageState field. The sleep between calls prevents the 429 hammer.

```python import requests import time import json API_KEY = "YOUR_API_KEY" SITE_URL = "sc-domain:example.com" # must match Search Console property INPUT_FILE = "pruned_urls.txt" OUTPUT_FILE = "audit_results.jsonl" with open(INPUT_FILE) as f: urls = [line.strip() for line in f if line.strip()] results = [] for i, url in enumerate(urls): endpoint = f"https://searchconsole.googleapis.com/v1/urlInspection/index:inspect?key={API_KEY}" payload = { "inspectionUrl": url, "siteUrl": SITE_URL } try: resp = requests.post(endpoint, json=payload) if resp.status_code == 200: data = resp.json() indexed = data.get("inspectionResult", {}).get("indexStatusResult", {}).get("coverageState", "UNKNOWN") results.append({"url": url, "indexed": indexed}) else: results.append({"url": url, "indexed": f"ERROR_{resp.status_code}"}) except Exception as e: results.append({"url": url, "indexed": f"EXCEPTION_{str(e)}"}) if (i + 1) % 100 == 0: print(f"Checked {i+1}/{len(urls)}") time.sleep(5) # gentle backoff after every 100 time.sleep(1) # 1 req/sec to stay well inside quota with open(OUTPUT_FILE, "w") as out: for r in results: out.write(json.dumps(r) + "\n") ```

The coverageState returns strings like "Submitted and indexed" or "URL is not on Google". If you get 429, the script stops. You must implement checkpointing—saving the last processed URL index—so you don’t start over. That’s the real-world gotcha that turns a two-night job into a week-long death march.

For quick-and-dirty checks under 2,000 URLs, a bash loop works. It’s primitive but transparent.

```bash while read url; do curl -s -X POST

"https://searchconsole.googleapis.com/v1/urlInspection/index:inspect?key=$API_KEY" \ -H "Content-Type: application/json" \ -d "{\"inspectionUrl\": \"$url\", \"siteUrl\": \"$SITE\"}" \ | jq '.inspectionResult.indexStatusResult.coverageState' >> index_log.txt sleep 1.5 done Using a Bulk Checker for Immediate 100k+ Audits

Google’s rate limits make the raw API impractical for massive volumes without parallelizing across multiple projects—a violation of their terms if you’re not careful. Third-party index checkers that aggregate data from multiple upstream sources or maintain their own index caches can return a verdict on 80,000 URLs in under five minutes. The SpeedyIndex bulk index checker is one such creature: you POST an array of URLs, and it answers with a per-URL status (indexed, not_indexed, unknown) almost instantly.

```bash curl -X POST "https://api.speedyindex.com/v1/check/bulk" \ -H "Authorization: Bearer YOUR_TOKEN" \ -d '{"urls":["https://example.com/purged-page-1/","https://example.com/purged-page-2/","https://example.com/purged-page-3/"]}' ```

A Python wrapper makes it easy to process a file of URLs in chunks of 10,000. The response JSON maps each URL to its index flag, and you dump it to a CSV for your reporting. This approach doesn’t hit Google directly; the service uses its own optimized infrastructure, so you bypass the 2,000/day quota entirely. That’s the commercial trade-off: you pay a subscription, but you get the job done by lunch.

In practice, we ran the SpeedyIndex check alongside our own Python script on a 72k set. The third-party tool identified 1,200 URLs still indexed that the Google API had missed because of 403 errors on non-www variants we’d forgotten to add as properties. That kind of cross-comparison saves your neck.

## Pitfalls That Will Wreck a Bulk Index Check Mid-Flow

- **Incomplete URL normalization:** You feed https://example.com/page but your property sc-domain:example.com expects the trailing slash. Google’s API may treat them as different entities. Pre-normalize every URL with a script that adds trailing slashes according to your canonical pattern before hitting the endpoint.
- **Expired OAuth2 tokens:** If you use OAuth instead of an API key, the token lives 1 hour. Mid-run failure without a refresh mechanism kills the process. Use service
```

account key with long-lived credentials and a token refresh wrapper.

- **Interpreting “INDEXED” as final:** A coverageState of “Submitted and indexed” doesn’t mean the page still appears in SERPs. It could be indexed but omitted via low-quality classification. For pruned pages, always cross-check with a site:example.com/purgedpage query (not scalable for bulk, but spot-check).
- **Forgetting HTTP authentication on staging checks:** If the pruned pages were from a staging subdomain protected by .htpasswd, Google can’t access them, and the index status will be meaningless. Drop basic auth before running checks.
- **Silent 403 errors:** The API can return a 403 with a body like "The caller does not have permission". This doesn’t mean the URL isn’t indexed—it means your API key isn’t authorized for that property. Log all non-200 status codes as a separate error bucket, don’t conflate with “not indexed.”

Decision Flow for an Index Audit After a Cleanup

Before running any loop, you need to understand where each URL lands. The following chart outlines the typical state machine. You start with the entire list, split into chunks, feed them into the check engine, and then act on the result. A “still indexed” flag doesn’t always mean a disaster—sometimes the page is a legitimate redirect target that should stay. But you must know.

```mermaid flowchart LR A[Collect pruned URLs + sitemap leftovers] --> B[Split into batches of 500] B --> C{Check engine} C -->|Indexed| D[Flag for review: maybe 301/410 missed] C -->|Not indexed| E[Log as clean] C -->|Error 403/429| F[Pause, retry after token refresh] D --> G[Cross-check with sitemap/301 map] E --> H[Cumulative deindexed count] ```

Real-World Case: 72,000 Pruned Blog Posts in 48 Hours

One e-commerce brand nuked a ten-year blog archive with a hand-rolled 410 header for each URL. They assumed that Google would naturally drop the pages after a few days. Two weeks later, the index status report from their developer showed 14,000 pages still indexed. They were using a Google Sheets add-on that could handle only 1,000 URLs before crashing. We rebuilt the pipeline with the Python script above plus the
```

SpeedyIndex bulk endpoint as a verification run. The Google API run (spread across three GCP projects, each with its own API key) covered the entire list in 14 hours because we batched 100 URLs per minute with precise sleep. The bulk checker completed in 8 minutes and flagged an additional 340 URLs that the API had missed due to property mismatches. The uncovered mess—old staging subdomain URLs, leftover AMP versions, and non-canonicalized pagination trails—forced them to tighten their 410 rules at the edge layer.

That real mess became a permanent cleanup script they now run quarterly to catch rogue indexed URLs. The lesson: never trust the assumption “I pruned them, so Google will drop them.” Verify with cold, hard, auditable log files. Otherwise you’re six months into an algorithmic hit and you don’t know why.

Common Headaches After the Initial Audit

Q: Why are 410-returning pages still marked as indexed?
Google doesn’t instantly deindex pages that serve 410. It may take weeks of recrawls. The index check only confirms the current state, not the future. You need to keep re-checking the same list weekly until the count drops to zero.

Q: Can I use the Indexing API to request removal instead of checking?
The Indexing API is for notifying Google of new or updated pages, not for bulk deindexation. It can speed up crawling of your 410/301 pages, but a URL_UPDATED call doesn’t guarantee removal. Combine API calls with proper HTTP status codes.

Q: What’s a safe request interval to avoid 429 from Google’s API?
Empirically, 1.5 seconds between calls (1 req/1.5s) with a short burst of 10 calls then a 60-second pause keeps you under the rate limit radar. For large lists, you may still get a 429 at the 2,000th call of the day. Implement exponential backoff and checkpointing.

Q: Do third-party bulk index checkers compromise security?

Read the privacy policy. Most send only the URL string, not credentials. If your URLs contain sensitive parameters, strip them before sending. For internal staging URLs behind a VPN, these tools won’t be able to verify anyway.

Automate the Confirmation or Accept Permanent Index Pollution

Every hour you spend manually poking around Search Console after a massive content cull is an hour you’re not fixing the actual crawl waste. Build the script once, wire it into your CI/CD for post-deployment audits, and tie the output to a Slack alert when more than X% of pruned URLs remain indexed past the 30-day mark. The alternative is a silent, creeping crawl budget hemorrhage that no amount of hreflang fixes or Core Web Vitals polish will offset. The code snippets above form a starting point; the real win is in treating index verification as an automated gate in your release process, not a fire drill.

Further Reading

1. Google Search Central. "How Google Search Works." developers.google.com
2. Google Search Central. "Sitemaps Overview." developers.google.com
3. Google Search Central. "Crawling and Indexing." developers.google.com
4. Bing Webmaster. "Submit Sitemaps." bing.com/webmasters