EcomLabs

STEP-BY-STEP GUIDE

Google not indexing your new page: the diagnosis in order

Seven steps in the order Google processes a page: URL Inspection, robots.txt, noindex, response and canonical, sitemap and links, a Node script that checks it all at once, and how to request indexing. With a checklist.

FOR	Any site with a Search Console property and new pages that do not show up
TESTED WITH	Node 22.22.3 · a ten-case local test site · public pages read on November 19, 2026
UPDATED	November 19, 2026
DOCS	developers.google.com/search/docs/crawling-indexing
ARTICLE	ecomlabs.dev/g/indexar-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Step 01	Read URL Inspection
Step 02	robots.txt
Step 03	noindex, response and canonical
Step 04	Sitemap and internal links
Step 05	Request indexing once
Appendix	The script
When you finish	Checklist

STEP 01

Read URL Inspection

Only an owner or full user of the property can request indexing. Paste the full URL into the Search Console bar and read the fields in this order; the first one that fails tells you which step to go to:

FIELD	IF YOU SEE	GO TO
Sitemaps · Referring page	Both empty	Step 04
Crawl allowed?	No	Step 02
Page fetch	Error, redirect or 404	Step 03
Indexing allowed?	No, with noindex	Step 03
User-declared vs Google-selected canonical	Different	Step 03

"Discovered - currently not indexed": Google rescheduled the crawl (step 04). "Crawled - currently not indexed": it visited and did not index the page for now; Google says this is common.

STEP 02

robots.txt

If there is a `User-agent: Googlebot` group, Google uses it instead of the `*` group. Inside the group, the longest matching rule wins; on a tie, the least restrictive. Paths are case-sensitive, `*` matches any sequence and `$` marks the end.

```
User-agent: *
Disallow: /admin
Disallow: /*?sort=
Allow: /drafts/public$
Disallow: /drafts/
Sitemap: https://your-domain.com/sitemap.xml
```

Here `/drafts/public` can be crawled (its `Allow` is longer), `/drafts/launch` cannot, and neither can any URL with `?sort=`. After changing the file, request a recrawl in the robots.txt report.

> **CAUTION**
> A page blocked by robots.txt cannot show its noindex to Google. To take it out of Google: unblock it and add noindex.

STEP 03

noindex, response and canonical

```
URL=https://your-domain.com/blog/new
curl -sI "$URL" | grep -i -E "^HTTP|^location|^x-robots-tag"
curl -s "$URL" | grep -o -i -E '<meta[^>]*name="(robots|googlebot)"[^>]*>'
curl -s "$URL" | grep -o -i -E '<link[^>]*rel="canonical"[^>]*>'
```

1. `curl -sI` asks for headers only: you want `200`, no `location` and no `X-Robots-Tag: noindex`.
2. The first `grep` on the HTML pulls out robots or googlebot `meta` tags: none should say `noindex` or `none`.
3. The second pulls out the canonical: it should be the URL itself, absolute, with the same trailing slash.

STEP 04

Sitemap and internal links

Check that the full URL is in the sitemap (the script in the appendix does). Then link the page with an `<a href>` from one that is already indexed: the home page, a category, the blog index or related articles. An `onclick` or a `span` with `href` is not a crawlable link.

STEP 05

Request indexing once

1. Run the script in the appendix, then click **Test live URL**. If it still sees the problem, the fix is not in production.
2. Click **Request indexing** once: there is a daily limit and repeating it does not speed up the crawl. With many pages, submit the sitemap with `lastmod`.
3. If you fixed a whole status in the report, use **Validate fix**: per Google, it typically takes up to about two weeks.

Per Google, indexing "typically takes only a day or so" and "can take up to a week or two", with no guarantee.

APPENDIX

The script

It checks steps 02 to 04 in one go: `node check-indexable.mjs https://your-domain.com/blog/new`. Lines with `-` are failures and exit with code 1; lines with `!` are warnings. It does not run JavaScript or know which canonical Google chose: URL Inspection is still the reference.

```
// Usage: node check-indexable.mjs https://example.com/new-page
// Reads what Google would read for one URL: robots.txt, the HTTP response, the
// robots meta tag, the canonical and the sitemaps. Node 18+, no dependencies.
const url = new URL(process.argv[2]);
const get = (u) => fetch(u, { redirect: "manual", headers: { "user-agent": "check-indexable" } });
const attr = (tag, name) => tag.match(new RegExp(`\\s${name}\\s*=\\s*["']([^"']*)["']`, "i"))?.[1] ?? "";
const fail = [], warn = [];
```

```
// robots.txt: the rules of the most specific group (googlebot, else *).
function rules(txt) {
  const groups = [];
  let open = false;
  for (const raw of txt.split(/\r?\n/)) {
    const m = raw.replace(/#.*/, "").match(/^\s*([a-z-]+)\s*:\s*(.*?)\s*$/i);
    if (!m) continue;
    const key = m[1].toLowerCase();
    if (key === "user-agent") {
      if (!open) groups.push({ agents: [], rules: [] });
      groups.at(-1).agents.push(m[2].toLowerCase());
      open = true;
      continue;
    }
    open = false;
    if ((key === "allow" || key === "disallow") && m[2] && groups.length) {
      groups.at(-1).rules.push({ allow: key === "allow", path: m[2] });
    }
  }
  const agent = groups.some((g) => g.agents.includes("googlebot")) ? "googlebot" : "*";
  return groups.filter((g) => g.agents.includes(agent)).flatMap((g) => g.rules);
}
const matches = (rule, path) => {
  const end = rule.endsWith("$");
  const body = (end ? rule.slice(0, -1) : rule).split("*").map((s) => s.replace(/[.+?^${}()|[\]\\]/g,
"\\$&"));
  return new RegExp(`^${body.join(".*")}${end ? "$" : ""}`).test(path);
};
// Longest matching rule wins; on a tie, allow wins (the least restrictive).
function blocked(txt, path) {
  let best;
  for (const r of rules(txt)) {
    if (!matches(r.path, path)) continue;
    if (!best || r.path.length > best.path.length || (r.path.length === best.path.length && r.allow)) best =
r;
  }
  return best ? !best.allow : false;
}

const robots = await get(new URL("/robots.txt", url));
const robotsTxt = robots.ok ? await robots.text() : "";
if (blocked(robotsTxt, url.pathname + url.search)) fail.push("robots.txt blocks it: Google cannot crawl it, so
it never sees a noindex");

const res = await get(url);
const html = res.status === 200 ? await res.text() : "";
if (res.status >= 300 && res.status < 400) fail.push(`redirects (${res.status}) to
${res.headers.get("location")}: only the target is indexed`);
else if (res.status !== 200) fail.push(`HTTP ${res.status}`);

const header = res.headers.get("x-robots-tag") ?? "";
```

```
if (/\b(noindex|none)\b/i.test(header)) fail.push(`X-Robots-Tag: ${header}`);
const metas = (html.match(/<meta\b[^>]*>/gi) ?? []).filter((t) => /^(robots|googlebot)$/i.test(attr(t,
"name")));
for (const t of metas) if (/\b(noindex|none)\b/i.test(attr(t, "content"))) fail.push(`meta ${attr(t, "name")}:
${attr(t, "content")}`);

const canonicalTag = (html.match(/<link\b[^>]*>/gi) ?? []).find((t) => /(^|\s)canonical(\s|$)/i.test(attr(t,
"rel")));
const canonical = canonicalTag ? new URL(attr(canonicalTag, "href"), url).href : "";
if (res.status === 200 && !canonical) warn.push("no rel=canonical: Google picks one");
if (canonical && canonical !== url.href) fail.push(`canonical points to ${canonical}`);

// Sitemaps from robots.txt, else /sitemap.xml; follows one level of sitemap index.
async function listed(sitemapUrl, depth = 0) {
  const r = await get(sitemapUrl);
  if (!r.ok) return false;
  const xml = await r.text();
  const locs = [...xml.matchAll(/<loc>\s*([^<]+?)\s*<\/loc>/g)].map((m) => m[1].replace(/&amp;/g, "&"));
  if (!/<sitemapindex/i.test(xml)) return locs.includes(url.href);
  if (depth === 0) for (const loc of locs) if (await listed(loc, 1)) return true;
  return false;
}
const sitemaps = [...robotsTxt.matchAll(/^\s*sitemap\s*:\s*(\S+)/gim)].map((m) => m[1]);
if (!sitemaps.length) sitemaps.push(new URL("/sitemap.xml", url).href);
let inSitemap = false;
for (const s of sitemaps) if (!inSitemap) inSitemap = await listed(s);
if (!inSitemap) warn.push("not in the sitemap");

console.log(`${fail.length ? "FAIL" : "OK  "} ${url.href}`);
for (const f of fail) console.log(`     - ${f}`);
for (const w of warn) console.log(`     ! ${w}`);
process.exit(fail.length ? 1 : 0);
```

WHEN YOU FINISH

Checklist

- [ ] URL Inspection shows no crawl or indexing block.
- [ ] No robots.txt rule matches the path, and no noindex in the meta tag or `X-Robots-Tag`.
- [ ] It returns 200 with no redirect, the canonical is the URL itself and the script passes.
- [ ] It is in the sitemap and an indexed page links to it.
- [ ] You requested indexing once and check again in a week.