Engineer

robots.txt

the file telling crawlers where not to go

What it is

Every host on the web can publish, at the fixed path `/robots.txt`, a plain text file of requests addressed to the programs that crawl it. Martijn Koster defined the convention in 1994, and in September 2022 the IETF published it as RFC 9309, the Robots Exclusion Protocol, adding precise language for the rules and instructions for errors and caching.

The file is a list of groups. A group opens with one or more `user-agent` lines, each naming a crawler by its product token, such as `ExampleBot`, or `*` for every crawler. Below them come `allow` and `disallow` lines, each holding a path. A crawler looks for the group that names its own token, matched without regard to case, and obeys that group; when several groups name it, their rules merge into one. Only a crawler with no group of its own falls back to the `*` group, and one that finds neither has no rules to follow.

Within a group, a rule applies when its path matches the start of the address, and the most specific match wins, meaning the one with the most octets. A tie between `allow` and `disallow` goes to `allow`, and an address that no rule matches is allowed. Two characters widen a path: `*` stands for any run of characters and `$` pins the end of the address. A `#` starts a comment, and the file's own path is always allowed.

The RFC also fixes how the file is fetched. It sits at that lowercase path at the top of the service, so every host answers for itself and a file on one subdomain says nothing about another. It is UTF-8 plain text. A crawler should follow at least five redirects to reach it, should not use a cached copy for more than 24 hours, and may cap how much it parses, though never below 500 KiB.

Lines outside the protocol are tolerated. The RFC calls them other records, gives `Sitemap` as its example, and lets a crawler interpret them or not, on one condition: such a line must never end a group.

Use the concept in a decision

Put “What is robots.txt?” to work in a real decision. We can help relate the concept to the system you are building or buying. A site or an app that the AI assistants your customers use can find, read and act on, checked on every release rather than promised once.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Built a multilingual Hugo site in 3 languages — 108 template files serving three languages and four formats

Built a multilingual Hugo site in 3 languages Get in touch →

Why it matters

The RFC states the limit in its introduction: “These rules are not a form of access authorization.” The file is a request. A crawler that honours it stays out, and one that does not reads whatever the server hands it. The security section adds that listing a path in the file makes it public and discoverable, so anything that must stay private needs a real control, such as HTTP authentication.

Google’s documentation narrows the purpose further. It describes the file as “used mainly to avoid overloading your site with requests” and says “it is not a mechanism for keeping a web page out of Google”. A disallowed address can still be indexed when other pages link to it, and then appears in results without a description, because the crawler never read the page. To keep a page out of results, Google points to a `noindex` rule or a password, and a `noindex` rule only works on a page the crawler is allowed to fetch.

What the file does well is the job it was written for: steering crawlers away from parts of a site that would cost the server requests and give a search engine little, such as unimportant or near-identical pages. Its reach is also its risk. One line under `*` applies to every address and every crawler that honours the file, and a change can take up to a day to be seen, so `Disallow: /` typed in the wrong group closes a whole site.

What it looks like in practice

This site’s file is long, and almost all of it is comments. The part a crawler acts on is four lines:

```
User-agent: *
Content-Signal: search=yes, ai-input=yes, ai-train=yes
Allow: /

Sitemap: https://engineer.company/sitemap.xml
```

One group covers every crawler, and `Allow: /` matches every address. A group with no rules would allow exactly the same, since an address that no rule matches is allowed, but the line is written out because the openness is a choice. The comment above it calls the policy “Decided rather than inherited”, and the page on what AI agents need from a site describes crawl policy as one of the site’s deliberate decisions.

`Content-Signal` is not a record RFC 9309 defines. It comes from the Content Signals vocabulary, and here it consents to collection for search, for answering a question at inference time and for training. Under the RFC it is one of those other records: a crawler that knows the convention may read it, one that predates it skips the line, as the file’s own comment says, and neither may treat it as the end of the group. The `Sitemap:` line is another record of that kind and points at the site’s sitemap index. The credits page sums the file up the same way: it “allows everything, to search crawlers and AI systems alike” and names the sitemap.

What the file leaves out was decided too. Among the five recommendations this site declined, the AI readiness page lists rules per agent, keeping one robots rule for every agent “because a rule written for one crawler silently replaces the general one for it.” Had the file given one AI crawler a group of its own, that crawler would read only that group, and a `Disallow` added later under `*` would never reach it.

The rest is comments, written for whoever opens the file: the reasoning behind the policy, then the addresses of `llms.txt`, `agents.txt`, `humans.txt`, `security.txt` and other machine-readable files, kept as comments “because robots.txt has no field for them”. That makes the comments part of what ships. A `#` line in the template is served byte for byte to every crawler, and twice one carried a note meant for a maintainer, pointing at a planning document in a repository that is not public. A check over the built site now fails when a served file carries a path into the repository, and maintainer notes in the template sit in template comments, which the generator removes.

Where it goes wrong

A rule meant for everyone stops applying to one crawler. Someone adds a group for a single bot, perhaps to keep it out of one folder, and from then on that bot ignores everything under `*`, including every `Disallow` written there. RFC 9309 sends a crawler to the `*` group only when no group names it, and Google's documentation says specific groups and the `*` group "are not combined". Repeat each general rule inside every named group, or keep a single group, as this site does.

A blocked page still appears in search results. The address shows up with no description, perhaps with the words other sites used when linking to it. Google never fetched the page, so it knows only that the address exists and what others call it. A `Disallow` controls crawling, not indexing. Remove the rule and serve `noindex` on the page, or put the page behind a password; a `noindex` hidden behind a `Disallow` is never read.

The file advertises what it was meant to hide. A line such as `Disallow: /admin/` tells every reader, polite crawler or not, exactly where to look, and the file is public by design. The RFC's security section says listing paths makes them discoverable and calls for access control instead. Put private material behind authentication and leave it out of the file entirely.

A server error stops the crawl of a whole site. The pages load, but the request for `/robots.txt` returns a 5xx status or times out, and a crawler that follows the RFC must treat that as a complete disallow. Google stops crawling for the first 12 hours, then works from its last good copy for up to 30 days. The opposite mistake is quieter: Google treats a 4xx answer other than 429, including 401 and 403, as if no file existed, so every address is open. Serve the file as static text with a 200 status, and monitor `/robots.txt` itself, not only the home page.

Further reading

- RFC 9309: Robots Exclusion Protocol — the standard itself: groups, matching, error handling, caching and the security note.
- Introduction to robots.txt — what Google says the file is for, and why it does not hide a page from results.
- How Google interprets the robots.txt specification — status codes, caching, the fields Google ignores and how it picks a group.

- Block Search indexing with noindex — the rule that does keep a page out of results, and why it needs the page crawlable.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is robots.txt?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/robots-txt/