Engineer

# Gopher

a menu-based protocol for retrieving internet documents

## What it is

Gopher is a protocol for fetching documents and menus over the internet, in which a client opens a TCP connection, sends one line naming what it wants, reads the answer and is done. It is formally the Internet Gopher protocol, its addresses begin with `gopher://`, and a site published over it is usually called a gopher hole. It dates from 1991, and RFC 1436 described it in March 1993.

The exchange is short enough to type by hand. The client connects to port 70, sends a selector (a string the server handed out earlier, possibly empty) followed by a carriage return and line feed, and the server answers and closes the connection. Nothing is remembered between requests: no session, no cookie, no headers. An empty selector asks for the server's top menu.

A menu is the heart of it. Each line of a menu is one item, made of a single type character, a display string, then a selector, a host and a port separated by tabs. A line holding only a full stop ends the menu. Here is a small one, with each tab written as `<TAB>`:

```
0About this server<TAB>/about.txt<TAB>example.org<TAB>70
1Reports<TAB>/reports<TAB>example.org<TAB>70
7Search the reports<TAB>/search<TAB>example.org<TAB>70
.
```

The type character tells the client what it will get before it asks. RFC 1436 defines `0` for a text file, `1` for another menu, `7` for a search server, `9` for a binary file, `g` for a GIF image and a handful more. Two types in wide use today come from practice rather than the RFC: `i` for a line of plain text shown inside a menu, and `h` for an HTML document, which clients also use to hand a web address to a browser.

The design is deliberately small. RFC 1436 has a section headed “Simplicity is intentional”, and it is an Informational RFC, not an Internet standard. The `gopher://` address format came later, in RFC 4266, where the first character of the path is the item type and the rest is the selector.

Use the concept in a decision

Put “What is Gopher?” to work in a real decision. We can help relate the concept to the system you are building or buying. A site that is quick, findable and editable by your own staff, without a bill for every change of wording.

**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. Developed 100 web applications using HTML/HTML5 — 100 web applications built

**Developed 100 web applications using HTML/HTML5** Get in touch →

## Why it matters

Gopher matters today less as a way to reach a large audience than as a clear picture of what a document needs and what it can live without. A menu is navigation reduced to its bones: every way out of a page is listed, typed and on its own line, with nothing hidden behind a hover or a script. Publishing to it forces questions that HTML lets an author postpone. What is this page called? What are its real links? Does the text still make sense once the links are taken out of it?

For an organisation, a gopher hole is an extra front door, never a replacement for the web. Readers need a dedicated client, and they are few. The cost of the door depends entirely on how it is built. Maintained by hand as a second copy of the site, it drifts the first time one copy is edited and the other is forgotten. Generated from the same source by the same build, it costs a template and some disk space, and it serves mainly as a stress test of the writing.

The second thing to understand is security. Gopher has no encryption layer. Every selector, and so every page a reader asks for, crosses the network in clear text, and so does every answer. RFC 1436 has nothing for signing in, and the words a reader types into a search item travel in clear text as well. That is acceptable for public documents and wrong for anything else, and it means the operator’s own logs deserve as much thought as the content.

It is also cheap to debug. With no TLS, no headers and no state, a single command line shows the exact bytes a client receives, which makes a gopher copy one of the easiest outputs of a site to check.

## What it looks like in practice

This site is a gopher hole as well as a website. The Engineer in a terminal page gives the address, `gopher://engineer.company/`, describes Gopher as a menu protocol from 1991 with plain TCP, no encryption and no scripting, and says the site there is a set of menus, one per directory. It names Lynx, Bombadillo and gopher-cli as clients and shows a request made by hand with `printf` and `nc` on port 70, the entire protocol in one line.

The build behind it is described in the write-up of mirroring the site to Gemini and Gopher. The same build that writes the HTML also wrote 777 Gopher menus at the recorded implementation stage, deployed to the same tree and served by a small daemon on the same host. A menu line is a set of tab-separated fields, so there is no way to put a link inside a sentence, and prose is hard-wrapped at sixty-eight columns while the site is built. Working within that sharpened the writing.

The templates show how each page becomes a menu. Inline links keep their words and lose their targets, headings lose their hash marks, bullets become indented asterisks and the prose is wrapped at 68 columns. Code is the exception and passes through unwrapped, because a code line broken at column 68 is a different program. Categories and services become type `1` items pointing at their own menus, and the foot of each page carries an `h` item whose `URL:` selector leads to the web version of the same page.

Running a gopher hole also taught the site something about logs. The Privacy page explains that since September 2026 the site counts only how many requests the Gemini and Gopher copies answered each day, with no address, no page and no visitor. The day counting began, it turned out that the operating system starts the Gopher copy afresh for each visitor and names that moment after the address the visitor came from, in a log of its own. That log now expires after thirty days, like everything else the site keeps.

# Where it goes wrong

## A page stops halfway and lists the directory

A reader opens a page and gets the first half of it, then a listing of the files in the folder. Servers such as Gophernicus read a gophermap before sending it, and a line without tabs whose first character is one of their directives is treated as an instruction: `#` hides it as a comment, `=` includes another file, and `*` or `.` stops the menu. This site met it when a line of CSS in its Ukrainian note about Safari underlines began with the star that closes a CSS comment, and the live server cut the page there and printed index.html, index.md and index.gmi instead. Write every such line as an explicit `i` item, and fail the build on any bare line that still begins with a directive character.

## Markup and entities arrive as text

A paragraph shows up as a single very long line full of angle brackets, or with an ampersand still spelled out as its HTML entity. The converter copied HTML into a format that has no idea what HTML is. Strip tags while keeping the anchor text, decode entities, and leave inline code alone so its symbols survive.

## Lines run off the right edge

Prose spills past the screen, or display strings are cut mid-word. Gopher clients are not obliged to reflow anything, and RFC 1436 asks for display strings of under 70 characters. Wrap prose during the build, truncate menu labels deliberately, and keep code out of the wrapper.

## A link fetches the wrong thing

A `gopher://` address opens an error or an unrelated item. RFC 4266 reads the first character of the path as the item type, so `gopher://example.org/reports` asks for type `r` and the selector `eports`. Put the type in front of the selector, as in `gopher://example.org/1/reports`, and copy addresses from a working menu rather than typing them.

## The reading list travels in the clear

Every page a reader opens is visible to anyone on the network path, and possibly in logs nobody meant to keep. Publish only what would be fine on a postcard, read the logs of the whole machine and not just the daemon’s, and give each one an expiry.

# Further reading

- RFC 1436: The Internet Gopher Protocol — the original description: the request and response cycle, menu lines, item types and the reasons for keeping it simple.
- RFC 4266: The gopher URI Scheme — how a `gopher://` address encodes host, port, item type and selector.
- IANA Service Name and Transport Protocol Port Number Registry — the official record of port 70 as the gopher port.
- Gophernicus — source and documentation of a current Gopher server, including the gophermap format it reads.

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 Gopher?” 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/gopher/