A WORKING SPECIFICATION

The AI Wayfinding Checklist

Seven required sections, one cross-link rule, and the three traps that silently break a deployment. Everything below is what I enforce across a federation of more than 200 sites.

Brent Passmore · brentpassmore.com

1. The seven sections every `llms.txt` needs

Same seven, same order, every site. Uniformity is not an aesthetic preference. It is what makes a missing section visible from a list instead of invisible until something misroutes.

Each row is required. A file missing any row is a file with a known failure mode.

SECTION	WHAT IT DOES	WHAT IT PREVENTS
Single H1 display name	Names the publisher, exactly once	Agents guessing your entity name from the URL
Blockquote summary	One paragraph on what this site is	Summaries assembled from whatever page ranked
Metadata	Version, Last Updated, Status	Agents citing a file nobody has touched in two years
Targeted Content	Three to six links you want cited	A crawler picking your six least useful pages
Related Resources	Where to go for adjacent needs	Dead ends
Authority and Scope	Positive *and* negative statements	Misrouting, which is the expensive one
Full Index link	Points to `llms-full.txt`	Agents assuming the short file is the whole map

> **The half everyone skips.** Authority and Scope needs both directions. Everyone writes "we are authoritative for X." Almost nobody writes "we are not authoritative for Y, go here instead." The negative statement is the single highest-value line in the file. It is what stops a research center from being cited as the place to book an appointment.

Three to six links, not thirty

The short file is a lobby directory, not the building. Volume belongs in `llms-full.txt`, and that file should be generated from your CMS, not typed. If you are hand-maintaining a full index, it is already out of date.

2. The cross-link rule

If you only take one thing from this page, take this one. It is the rule that decides whether an AI agent finds an answer or circles.

> **The pillar root is the hub. Departments link up, never sideways.**

Every department or subsite file declares exactly three kinds of links:

1. **An up-link to its pillar root.** Always. No exceptions. This is what gives the graph a center.
2. **A clinical or primary destination link**, if the unit has public-facing operations that live somewhere else. Pure research and administrative units skip this.
3. **One or two topic-specific cross-pillar links**, only where a genuine subject relationship exists across pillars.

> **And one prohibition.** Peer links inside the same pillar are forbidden. If two departments under the same root need to be connected, that connection lives at the root, not in either department's file. Reciprocity belongs at the hub. Departments do not owe each other links.

This will feel wrong the first time you apply it to two closely related units. Apply it anyway. Letting departments link to whoever they collaborate with is generous, human, and produces a graph with no center, where an agent can wander laterally without ever meeting the authority that could answer the question.

---

3. Three traps in `robots.txt`

The file everyone assumes is solved. In a federation it is not.

Trap one: it is host-scoped, not site-scoped

One host, one `robots.txt`, served from the host root. If each of your sites is its own hostname, each genuinely gets its own crawler policy. If your sites are subdirectories of one host, they all share a single file, and `example.com/dept/robots.txt` is read by nothing. Every rule for every subdirectory has to live in the one host-root file.

Trap two: groups do not inherit

A crawler obeys exactly one group, the most specific one matching its name, and ignores every other group including `*`. Add a `Disallow` under `User-agent: *` while you have a named `GPTBot` block, and GPTBot never sees it. Whatever you want a named agent to respect must be repeated inside that agent's own group.

This is the most common robots.txt mistake I see, and it fails silently. Nothing errors. The rule is simply not there.

Trap three: a static file in the web root wins

If you serve wayfinding files dynamically, a leftover static file on disk will beat your rewrite rules. The web server answers from disk before your application ever runs. Your routes are registered, your logic is correct, and none of it matters.

4. Before you ship

Run this on every file, every time. Most of it takes under a minute.

- [ ] All seven sections present, in order, with the exact headings
- [ ] Authority and Scope has both a positive and a negative statement
- [ ] Three to six links in Targeted Content, no more
- [ ] Every URL in the file actually resolves, and resolves to the page it claims
- [ ] Anchors in `llms.txt` match real headings in `llms-full.txt`
- [ ] Version and Last Updated reflect this change, and the two files agree
- [ ] Cross-links follow the hub rule, with no peer links inside the pillar
- [ ] No static file in the web root shadowing dynamic delivery
- [ ] Served as `text/plain; charset=utf-8`, not HTML
- [ ] The trailing-slash form does not redirect-loop
- [ ] Named crawler groups each carry the rules you expect them to obey

> **The one that is not technical.** Somebody owns the review date. Version and Last Updated fields going stale is a governance failure, not a bug, and no script will catch it. If nobody owns the date, the file quietly becomes a lie.

Brent Passmore writes about accessibility, AI wayfinding, and the unglamorous work of governing a lot of websites at once. This checklist comes from running it across a federation of more than 200 public properties at an academic medical center.

The current version of this checklist lives at **brentpassmore.com/resources/ai-wayfinding-checklist/**

Version 1.0 · Free to copy, adapt, and use inside your own organization. Attribution appreciated, not required.