Cooperative Content Negotiation for the Agentic Web: Extending robots.txt for AI Agents

David Hurley
Plasmate Labs

March 2026

Abstract

The Robots Exclusion Protocol, first proposed informally in 1994 and formalized as RFC 9309 in 2022, provides a binary access-control mechanism for web crawlers: allow or disallow. This mechanism is insufficient for the emerging relationship between web publishers and AI agents, which do not merely index content but read, reason over, extract from, and act upon it. We observe that publishers are responding to the rise of AI crawlers with broad blocking rules that are simultaneously too aggressive (rendering their content invisible to AI-powered search and discovery) and too blunt (offering no intermediate option between full access and total exclusion). We propose a set of purely additive directives for robots.txt that enable cooperative content negotiation: `SOM-Endpoint`, `SOM-Format`, `SOM-Scope`, `SOM-Freshness`, and `SOM-Token-Budget`. These directives allow publishers to advertise semantic representations of their content through the Semantic Object Model (SOM), giving AI agents access to structured, token-efficient page representations while preserving publisher control over what is exposed, in what form, and at what cost. We present complementary signaling mechanisms via HTML meta tags, Schema.org types, and HTTP headers, forming a layered discovery architecture. We analyze the current publisher-agent conflict through empirical data on AI bot blocking patterns across major websites, demonstrate backward compatibility with RFC 9309, discuss security considerations including endpoint poisoning and privacy leakage, and outline an adoption pathway through the W3C Web Content Browser for AI Agents Community Group.

1 Introduction

The World Wide Web is experiencing a structural conflict between content publishers and AI agents. This conflict manifests in robots.txt files across the internet, where publishers deploy increasingly aggressive blocking rules against AI crawlers while simultaneously losing visibility in AI-powered search, recommendation, and discovery systems.

The root cause is a mismatch between the expressive capacity of the Robots Exclusion Protocol and the complexity of the publisher-agent relationship. robots.txt offers exactly two options: allow a crawler to access a resource, or disallow it. For traditional search engine crawlers, this binary was sufficient. A publisher either wanted to appear in Google's search index or did not.

AI agents present a fundamentally different proposition. When an AI agent accesses a web page, it may be doing so for any of several purposes: answering a user's question, summarizing content for a research task, extracting structured data for analysis, comparing prices across vendors, or autonomously completing a multi-step workflow. The publisher's interests vary across these use cases, yet robots.txt provides no vocabulary to express this variation.

The result is a lose-lose equilibrium. Publishers who block AI agents (as the majority of major news organizations now do) protect their content from uncontrolled extraction but become invisible to the fastest-growing content discovery channel in history. Publishers who allow AI agents gain visibility but surrender control over how their content is consumed, receive no attribution or compensation, and absorb the bandwidth costs of full-page rendering for machine consumption.

This paper proposes a way out of this binary. We introduce a set of directives for robots.txt that enable publishers to say not just "yes" or "no" to AI agents, but "yes, and here is a better way to consume my content." These directives point agents to Semantic Object Model (SOM) endpoints [10] that provide structured, token-efficient representations of web pages, giving publishers control over what is exposed while giving agents higher-quality, lower-cost access to content.

The proposed directives are purely additive to RFC 9309 [3]. They do not modify the semantics of existing Allow and Disallow rules. Crawlers that do not understand them will ignore them, as RFC 9309 Section 2.2.4 explicitly permits. This backward compatibility is essential: any extension to robots.txt that breaks existing parsers would face insurmountable adoption barriers.

The remainder of this paper is organized as follows. Section 2 provides background on the Robots Exclusion Protocol. Section 3 analyzes the publisher-agent conflict with empirical data. Section 4 presents the proposed SOM directives. Section 5 describes complementary signaling mechanisms. Section 6 analyzes stakeholder benefits. Section 7 addresses security considerations. Section 8 outlines an adoption pathway. Section 9 surveys related work. Section 10 concludes.

2 Background: The Robots Exclusion Protocol

2.1 History

The Robots Exclusion Protocol originated in June 1994, when Martijn Koster proposed a convention for web crawlers to consult a file at `/robots.txt` before accessing a site's resources [1]. The proposal arose from practical necessity: early web crawlers were consuming excessive server bandwidth, and site administrators needed a lightweight mechanism to request restraint.

The original convention was never formally standardized through the IETF process. It existed as a de facto standard documented on robotstxt.org [2], implemented voluntarily by major crawlers including Googlebot, Bingbot, and Yahoo Slurp. Despite its informal status, robots.txt achieved near-universal adoption: virtually every major website publishes one, and every major crawler respects it.

In September 2022, the IETF published RFC 9309, authored by Koster, Illyes, Zeller, and Sassman [3]. This document formalized the protocol for the first time, defining its syntax in ABNF, specifying error-handling behavior, establishing caching rules, and clarifying the semantics of the User-agent, Allow, and Disallow directives. RFC 9309 elevated robots.txt from an informal convention to an Internet Standards Track document.

2.2 Current Mechanism

The Robots Exclusion Protocol operates through a simple text file served at the well-known URI `/robots.txt`. The file contains groups of rules, each prefixed by one or more `User-agent` lines that identify which crawlers the rules apply to. Within each group, `Allow` and `Disallow` lines specify URL path patterns that the identified crawlers may or may not access.

The formal syntax from RFC 9309 defines:

```
robotstxt = *(group / emptyline)
group     = startgroupline
            *(startgroupline / emptyline)
            *(rule / emptyline)
startgroupline = *WS "user-agent" *WS ":" *WS product-token EOL
rule = *WS ("allow" / "disallow") *WS ":"
       *WS (path-pattern / empty-pattern) EOL
```

Key operational rules include: crawlers MUST use case-insensitive matching for user-agent names; the most specific (longest) path match takes precedence; if no matching group exists, the wildcard (`*`) group applies; if no rules match, access is implicitly allowed [3].

Several extensions have been adopted in practice without formal standardization. The `Sitemap` directive, which points crawlers to XML sitemap files [32], is widely supported. The `Crawl-delay` directive, which requests a minimum interval between requests, is recognized by some crawlers but not universally supported. Both demonstrate that the robots.txt ecosystem can absorb new directives without disruption.

2.3 Limitations for the Agent Era

The Robots Exclusion Protocol was designed for a specific relationship: a search engine crawler indexing pages to build a search index. This relationship has several properties that robots.txt handles well:

1. **Binary intent**: The crawler either indexes the page or does not.
2. **Uniform purpose**: All major search crawlers serve the same fundamental purpose.
3. **Mutual benefit**: Publishers want to appear in search results; crawlers want to index content.
4. **Path-based granularity**: Publishers can control access at the URL-path level.

AI agents violate all four assumptions:

1. **Graduated intent**: An agent may want to read, extract, summarize, compare, or act on content. "Allow" and "disallow" cannot capture this range.
2. **Diverse purpose**: GPTBot trains models, ChatGPT-User answers real-time questions, PerplexityBot builds search results, ClaudeBot may be performing research. These purposes have different implications for publishers.
3. **Misaligned incentives**: Publishers receive little direct benefit from AI content extraction. Unlike search engines, which drive traffic back to the publisher's site, AI agents often consume and synthesize content without generating referral traffic.
4. **Content-level granularity**: Publishers may want to expose article text but not comments, product descriptions but not pricing. Path-based rules are too coarse for these distinctions.

These limitations have produced the current crisis: publishers cannot express their actual preferences, so they default to the only strong signal available—total blocking.

3 The Publisher-Agent Conflict

3.1 Current Blocking Patterns

To characterize the state of publisher-agent relations, we surveyed the robots.txt files of major websites in March 2026. The results reveal pervasive blocking of AI-specific crawlers.

The New York Times (nytimes.com) blocks at least twelve AI-related user agents with blanket `Disallow: /` rules:

```
User-agent: GPTBot
Disallow: /

User-agent: ChatGPT-User
Disallow: /

User-agent: ClaudeBot
Disallow: /

User-agent: anthropic-ai
Disallow: /

User-agent: CCBot
Disallow: /

User-agent: Google-Extended
Disallow: /

User-agent: cohere-ai
Disallow: /

User-agent: PerplexityBot
Disallow: /

User-agent: Bytespider
Disallow: /
```

CNN (cnn.com) takes an even broader approach, listing over thirty AI and data-extraction user agents in a single group, including anthropic-ai, Bytespider, CCBot, ChatGPT-User, ClaudeBot, Claude-SearchBot, Claude-Web, cohere-ai, GPTBot, Google-Extended, GoogleOther, and PerplexityBot.

The Washington Post (washingtonpost.com) blocks anthropic-ai, CCBot, ClaudeBot, and PerplexityBot with blanket disallow rules. **The Guardian** (theguardian.com) blocks CCBot, Bytespider, PerplexityBot, anthropic-ai, ClaudeBot, and Claude-SearchBot.

3.2 The Scale of Blocking

Empirical studies confirm that blocking is widespread and accelerating:

- As of August 2024, 35.7% of the world's top 1,000 websites blocked OpenAI's GPTBot, representing a seven-fold increase from the 5% blocking rate when the crawler launched in August 2023 [4].
- Tollbit's Q2 2025 report documented a 336% increase in sites blocking AI crawlers over the preceding year [5].

- A BuzzStream study in December 2025 found that 71% of news publisher sites block AI retrieval bots, with PerplexityBot blocked by 67% [6].
- Arc XP analysis found that nearly half (49.4%) of news sites disallow GPTBot outright [7].
- Only 14% of publishers block all AI bots comprehensively, while 18% do not block any, leaving the majority in an inconsistent middle ground [6].

3.3 The Economic Tension

The blocking behavior reflects a genuine economic tension. Publishers face two unpalatable options:

The cost of blocking. When a publisher blocks AI agents, their content becomes invisible to AI-powered systems. AI search engines (Perplexity, SearchGPT, Google AI Overviews) cannot cite or summarize the publisher's content. AI assistants (ChatGPT, Claude, Gemini) cannot reference the publisher's work. As AI-mediated discovery grows [8], blocked publishers face declining reach.

The cost of allowing. When a publisher allows AI agents unrestricted access, they face training data extraction without compensation (motivating the New York Times lawsuit against OpenAI [9]), content synthesis without attribution, bandwidth costs from full-page rendering for machine consumption, and loss of control over which portions of their content are accessible to agents.

3.4 Why Binary Control Fails

The fundamental problem is that publishers have legitimate interests that fall between "block everything" and "allow everything." A news publisher might reasonably want to:

- Allow AI search engines to cite headlines and brief excerpts (driving discovery traffic)
- Prevent AI systems from reproducing full article text (protecting subscription revenue)
- Permit AI agents to access structured data (author, date, topic) for indexing
- Exclude paywalled content from any AI access
- Provide a clean, ad-free representation of public content for AI consumption
- Rate-limit AI access to prevent bandwidth abuse

None of these preferences can be expressed in the current robots.txt vocabulary. The SOM directives proposed in this paper provide mechanisms for several of these use cases, with the potential to support others through authentication and scope controls at the SOM endpoint level.

4 Proposed Extension: SOM Directives

4.1 Overview

We propose five new directives for robots.txt that enable publishers to advertise semantic representations of their web content. These directives point AI agents to Semantic Object Model (SOM) endpoints [10] where they can obtain structured, token-efficient representations of pages instead of (or in addition to) fetching raw HTML.

The directives are summarized in Table 1.

Table 1: Proposed SOM directives for robots.txt

Directive	Description	Required
SOM-Endpoint	Base URL of the SOM service	Yes (to enable SOM)
SOM-Format	Format of the semantic representation	No (default: SOM/1.0)
SOM-Scope	Content coverage of the representation	No (default: full-page)
SOM-Freshness	Max age in seconds of a cached representation	No (default: 86400)
SOM-Token-Budget	Suggested max token count	No

4.2 Directive Syntax

Each directive follows the key-value pair syntax established by RFC 9309 for non-group records (Section 2.2.4). Directives are case-insensitive in their names but may have case-sensitive values.

4.2.1 SOM-Endpoint

```
SOM-Endpoint: <URL>
```

Specifies the base URL of a SOM service. AI agents construct a request by appending a `url` query parameter with the target page URL. For example, given `SOM-Endpoint: https://cache.example.com` an agent seeking the SOM representation of `https://example.com/article/123` would request:

```
GET https://cache.example.com/v1/som?url=https://example.com/article/123
```

The endpoint MUST return a valid SOM document as specified in the SOM specification [10]. The endpoint MAY require authentication, rate-limit requests, or return different representations based on the requesting agent's identity.

4.2.2 SOM-Format

```
SOM-Format: <format-identifier>
```

Specifies the format of the semantic representation. Defined values include: `SOM/1.0` (Semantic Object Model version 1.0, JSON), `markdown` (Markdown representation), and `accessibility-tree` (browser accessibility tree representation). If omitted, agents SHOULD assume `SOM/1.0`.

4.2.3 SOM-Scope

```
SOM-Scope: <scope-identifier>
```

Specifies content coverage. Defined values: `full-page` (entire page including navigation, headers, footers), `main-content` (primary content region only), `article-body` (article/post body only, excluding metadata and comments). Default: `full-page`.

4.2.4 SOM-Freshness

```
SOM-Freshness: <seconds>
```

Maximum age in seconds of a cached SOM representation. Default: 86400 (24 hours), consistent with RFC 9309's caching recommendation. Values of 0 indicate agents should always fetch fresh representations.

4.2.5 SOM-Token-Budget

```
SOM-Token-Budget: <integer>
```

Advisory maximum token count for representations, measured in cl100k_base tokens [11]. Helps agents estimate costs before fetching.

4.3 Complete Examples

Basic configuration:

```
User-agent: *
Allow: /

# Semantic Object Model endpoint
SOM-Endpoint: https://cache.example.com/v1/som
SOM-Format: SOM/1.0
SOM-Scope: main-content
SOM-Freshness: 3600
```

Publisher blocking raw crawling but offering SOM:

```
User-agent: GPTBot
Disallow: /

User-agent: ClaudeBot
Disallow: /

User-agent: PerplexityBot
Disallow: /

User-agent: *
Allow: /

# AI agents blocked above can still access
# content via the SOM endpoint
SOM-Endpoint: https://api.publisher.com/som
SOM-Format: SOM/1.0
SOM-Scope: article-body
SOM-Freshness: 1800
SOM-Token-Budget: 3000
```

This last example illustrates a critical design decision: SOM directives operate independently of Allow/Disallow rules. A publisher can block raw HTML crawling for specific AI agents while still offering them access to curated SOM representations. The publisher says "do not scrape my HTML, but here is a representation I am willing to share."

4.4 Backward Compatibility

The proposed directives are fully backward compatible with RFC 9309. Section 2.2.4 explicitly states:

> "Crawlers MAY interpret other records that are not part of the robots.txt protocol—for example, 'Sitemaps'. Crawlers MAY be lenient when interpreting other records. [...] Parsing of other records MUST NOT interfere with the parsing of explicitly defined records." [3]

Crawlers that do not recognize SOM directives will ignore them. The Sitemap directive followed this same adoption path: never part of the formal specification but achieving widespread support through voluntary adoption.

SOM directives do not modify the semantics of User-agent, Allow, or Disallow. They do not change group structure, matching rules, or precedence logic. A robots.txt file containing SOM directives is valid under RFC 9309.

4.5 Agent Discovery and Usage Flow

When an AI agent encounters a website, the following discovery process applies:

1. **Fetch robots.txt**: The agent retrieves `/robots.txt` as required by RFC 9309.
2. **Parse access rules**: The agent evaluates Allow/Disallow rules for its user-agent.
3. **Check for SOM directives**: If `SOM-Endpoint` is present, the agent notes the endpoint URL.
4. **Decide on access method**: For each page:
   - If allowed, the agent MAY fetch raw HTML or the SOM representation.
   - If disallowed but a SOM endpoint is available, the agent SHOULD use the SOM endpoint.
   - If no SOM endpoint exists and the agent is disallowed, it MUST NOT access the page.
5. **Fetch SOM representation**: Request the endpoint with the target URL.
6. **Cache management**: Cache per `SOM-Freshness` and HTTP caching headers.

5 Complementary Signaling Mechanisms

While robots.txt SOM directives provide site-wide signaling, finer-grained control requires per-page mechanisms. We propose three complementary layers.

5.1 HTML Meta Tags

Per-page SOM availability can be signaled via HTML meta tags:

```
<meta name="som-endpoint"
      content="https://cache.example.com/v1/som" />
<meta name="som-format" content="SOM/1.0" />
<meta name="som-scope" content="article-body" />
```

Meta tags override robots.txt directives for the specific page on which they appear.

5.2 Schema.org WebPageSemanticRepresentation

We have proposed a new Schema.org type, `WebPageSemanticRepresentation` [12], for per-page structured data markup:

```
{
  "@context": "https://schema.org",
  "@type": "WebPageSemanticRepresentation",
  "sourceUrl": "https://example.com/article/123",
  "representationUrl": "https://cache.example.com/som?url=...",
  "representationFormat": "SOM/1.0",
  "tokenCount": 1200,
  "compressionRatio": 14.2,
  "contentScope": "article-body"
}
```

This integrates with the existing Schema.org ecosystem, providing richer metadata including token counts and compression ratios.

5.3 HTTP Link Headers

For dynamic content, SOM availability can be signaled via HTTP Link headers [13]:

```
Link: <https://cache.example.com/v1/som?url=...>;
      rel="semantic-representation";
      type="application/som+json"
```

5.4 Layered Discovery Architecture

The three mechanisms operate at different levels of specificity, as shown in Table 2.

Table 2: Layered discovery architecture for SOM signaling

Layer	Mechanism	Scope	Discovery Cost
Site-wide	robots.txt directives	Entire domain	One request per domain
Per-page	HTML meta tags	Individual page	Requires page fetch
Per-response	HTTP Link headers	Individual response	Zero (in response)
Structured data	Schema.org JSON-LD	Individual page	Requires page parsing

Precedence order: HTTP Link header > HTML meta tag > Schema.org markup > robots.txt directive. More specific signals override less specific ones.

6 Benefits for Stakeholders

6.1 For Publishers

Control without blocking. SOM directives give publishers a middle ground between total access and total blocking. Instead of choosing between "GPTBot can read everything" and "GPTBot can read nothing," a publisher can say "GPTBot cannot scrape my HTML, but it can access a curated representation of my article content through this endpoint."

Bandwidth reduction. A typical news article page is 500KB–2MB of HTML, CSS, JavaScript, and embedded resources. A SOM representation of the same content is 5–50KB. Publishers save

90–99% of bandwidth costs for AI traffic.

Content curation. SOM endpoints can be configured to exclude advertisements, paywalled sections, user-generated comments, and tracking scripts. Publishers control exactly what agents receive.

Analytics. SOM requests go through publisher-controlled endpoints, enabling logging, rate limiting, and analysis of AI agent traffic.

6.2 For Agent Developers

Token efficiency. SOM representations achieve 16.6x mean token compression compared to raw HTML [10], translating to approximately 94% reduction in LLM input token costs.

Higher extraction quality. SOM documents are structured JSON with typed semantic elements. No HTML parsing, CSS resolution, or DOM traversal required.

No headless browser. Fetching a SOM endpoint is a single HTTP GET request, eliminating the 300–500MB memory overhead per browser instance and 2–5 second rendering latency per page.

Explicit permission. SOM directives constitute an explicit signal that AI agent consumption of SOM content is permitted, reducing legal ambiguity.

6.3 For the Web Ecosystem

Cooperative dynamics. SOM directives create a cooperative channel where both publishers and agents benefit, replacing the current adversarial pattern of blocking and circumvention.

Measurable improvements. Token compression, bandwidth savings, and latency reduction are all quantifiable, creating positive feedback loops.

Interoperability. A standardized discovery mechanism means agent developers can build a single pipeline that works across all SOM-enabled sites.

7 Security and Abuse Considerations

7.1 SOM Endpoint Poisoning

An attacker who compromises a publisher's infrastructure could serve poisoned SOM documents containing false information or prompt injection payloads. SOM endpoints SHOULD be served over HTTPS. Agents SHOULD verify endpoint domain authorization. SOM documents SHOULD include cryptographic signatures verifiable against the publisher's public key.

7.2 Denial of Service

On-demand SOM generation could be exploited for DoS attacks. SOM endpoints SHOULD pre-generate and cache representations. The `SOM-Freshness` directive enables publisher-side caching. Endpoints SHOULD implement rate limiting and MAY require authentication.

7.3 Privacy and Content Leakage

A poorly configured SOM endpoint could expose paywalled or restricted content. SOM endpoints MUST enforce the same access controls as the underlying content. Publishers SHOULD audit endpoints to prevent leakage of paywalled, private, or restricted material. SOM endpoints

MUST NOT include personally identifiable information from user-generated content without appropriate data-processing agreements.

7.4 Spoofing and Trust Verification

For agent identity verification, publishers MAY require verifiable credentials. For endpoint authenticity, agents SHOULD verify that the SOM-Endpoint URL is under the publisher's domain or explicitly authorized. Cross-origin SOM endpoints SHOULD be validated through DNS TXT records or `.well-known` files.

7.5 Prompt Injection

SOM documents contain text from web pages that could include prompt injection attempts. This threat exists equally for raw HTML consumption. SOM's structured format makes content isolation easier, reducing the attack surface. Agent developers SHOULD treat all SOM content as untrusted input, consistent with RFC 9309's security guidance [3].

8 Adoption Pathway

8.1 W3C Community Group

The SOM directives proposal is being developed within the W3C Web Content Browser for AI Agents Community Group [15], established in March 2026. The Community Group process provides lightweight governance for incubating specifications.

8.2 Reference Implementation

Plasmate [16] provides an open-source reference implementation under the Apache 2.0 license, available via crates.io, PyPI, npm, and Docker.

8.3 CDN and Platform Integration

For broad adoption, SOM endpoints need edge deployment. We propose integration pathways for Cloudflare Workers, Vercel Edge Functions, AWS CloudFront with Lambda@Edge, and Fastly Compute. These integrations allow publishers to add SOM support with minimal infrastructure investment.

8.4 Agent Framework Adoption

SOM discovery needs integration into major agent frameworks: LangChain [17], LlamaIndex [18], CrewAI [19], AutoGen [20], and Browser Use [21].

8.5 Proposed Timeline

9 Related Work

9.1 Content Negotiation in HTTP

HTTP content negotiation (RFC 9110 [22]) allows clients and servers to agree on resource representations through Accept headers. SOM directives extend this concept to the discovery layer: while HTTP negotiation operates per-request, robots.txt SOM directives operate site-wide, allowing agents to discover alternative representations before making individual requests.

Table 3: Proposed adoption timeline

Phase	Timeline	Milestone
Specification	Q1–Q2 2026	SOM directives draft in W3C CG
Reference Impl.	Q2 2026	Plasmate SOM endpoint available
Early Adopter	Q3–Q4 2026	10+ publishers deploy endpoints
Framework Integration	Q3–Q4 2026	LangChain, LlamaIndex add support
CDN Integration	Q1 2027	Cloudflare, Vercel offer SOM
Broad Adoption	2027–2028	1,000+ sites with SOM directives
Formal Standard	2028+	Potential IETF or W3C WG

9.2 Structured Data

Schema.org [23], JSON-LD [24], and Microdata [25] embed structured metadata within HTML pages. SOM differs fundamentally: structured data annotates existing content with metadata, while SOM replaces HTML with a purpose-built machine representation. The two approaches are complementary.

9.3 AI-Specific Proposals

llms.txt [26] specifies a file at `/llms.txt` containing LLM-friendly Markdown about a website, focusing on static site-level documentation rather than per-page content delivery.

ai.txt proposals provide consent frameworks for AI training data usage. SOM directives differ in focusing on content delivery rather than permission management, answering "how should AI agents consume my content?" rather than "may AI agents use my content?"

9.4 Prior robots.txt Extensions

The `Sitemap` directive [32] was never part of the formal specification but achieved widespread adoption through voluntary implementation. The `Crawl-delay` directive saw partial adoption (Bing, Yandex) without universal support. The SOM directives follow the Sitemap model: purely additive, backward compatible, and initially supported by a subset of agents.

9.5 Content Extraction Services

Jina Reader [27], Firecrawl [28], Crawl4AI [29], and Mozilla Readability [30] convert web content to machine-friendly formats. These operate as intermediaries; SOM directives enable publishers to serve optimized representations directly, preserving publisher sovereignty.

10 Conclusion

The Robots Exclusion Protocol has served the web well for over thirty years, but its binary access-control model is inadequate for the relationship between publishers and AI agents. The current state, where most major publishers block most AI crawlers, reflects not a preference for invisibility but an absence of alternatives.

SOM directives for robots.txt provide that alternative. By enabling publishers to advertise semantic representations of their content, these directives transform the publisher-agent relationship from adversarial to cooperative. Publishers gain control over what AI agents receive without sacrificing discoverability. Agents gain access to higher-quality, more token-efficient content without resorting to aggressive scraping.

The proposed directives are backward compatible with RFC 9309, require no changes to existing robots.txt parsers, and follow a proven adoption path established by the Sitemap directive. Complementary mechanisms via HTML meta tags, Schema.org types, and HTTP headers provide a complete, layered discovery architecture.

The challenge ahead is adoption. Like all web standards, SOM directives require coordinated action by both publishers and agent developers. The W3C Community Group provides a neutral forum for this coordination, and the reference implementation in Plasmate lowers the barrier to entry. But the ultimate driver of adoption will be the same force that drove robots.txt adoption in 1994: mutual self-interest. Publishers want their content discovered and consumed on their terms; agents want efficient, high-quality access. SOM directives align these interests.

We invite publishers, agent developers, browser vendors, CDN providers, and the broader web community to participate in developing and adopting these extensions through the W3C Web Content Browser for AI Agents Community Group.

References

[1] M. Koster, "A Standard for Robot Exclusion," 1994. https://www.robotstxt.org/orig.html

[2] "The Web Robots Pages," 2007. https://www.robotstxt.org/

[3] M. Koster, G. Illyes, H. Zeller, and L. Sassman, "Robots Exclusion Protocol," RFC 9309, Internet Engineering Task Force, September 2022. https://www.rfc-editor.org/rfc/rfc9309

[4] "OpenAI revises ChatGPT crawler documentation with significant policy changes," PPC Land, December 2025. https://ppc.land/openai-revises-chatgpt-crawler-documentation-with-significant-policy-changes/

[5] "Publishers say no to AI scrapers, block bots at server level," The Register, December 2025. https://www.theregister.com/2025/12/08/publishers_say_no_ai_scrapers/

[6] "Which News Sites Block AI Crawlers in 2025?," BuzzStream, December 2025. https://www.buzzstream.com/blog/publishers-block-ai-study/

[7] "How AI Bots Crawl News Content," Arc XP, 2025. https://dev.arcxp.com/blog/ai-bot-traffic-trends/

[8] Gartner, "Gartner Predicts Search Engine Volume Will Drop 25% by 2026," October 2024.

[9] *The New York Times Company v. Microsoft Corporation et al.*, Case No. 1:23-cv-11195, U.S. District Court, S.D.N.Y., December 2023.

[10] D. Hurley, "The Semantic Object Model: A Token-Efficient Web Representation for AI Agents," Plasmate Labs, March 2026.

[11] OpenAI, "tiktoken: BPE tokeniser for use with OpenAI's models," 2023. https://github.com/openai/tiktoken

[12] "Proposal: WebPageSemanticRepresentation type for AI agent consumption," Schema.org Issue #4786, 2026. https://github.com/schemaorg/schemaorg/issues/4786

[13] M. Nottingham, "Web Linking," RFC 8288, Internet Engineering Task Force, October 2017. https://www.rfc-editor.org/rfc/rfc8288

[14] D. Hurley, "The Agentic Web: Rethinking Web Infrastructure for Machine Consumption," Plasmate Labs, March 2026.

[15] W3C, "Web Content Browser for AI Agents Community Group," 2026. https://www.w3.org/community/web-content-browser-ai/

[16] Plasmate Labs, "Plasmate: Semantic Object Model compiler and runtime," 2026. https://github.com/plasmate-labs/plasmate

[17] LangChain, "LangChain: Building applications with LLMs through composability," 2023. https://github.com/langchain-ai/langchain

[18] LlamaIndex, "LlamaIndex: Data framework for LLM applications," 2023. https://github.com/run-llama/llama_index

[19] J. Moura, "CrewAI: Framework for orchestrating role-playing, autonomous AI agents," 2024. https://github.com/joaomdmoura/crewAI

[20] Microsoft, "AutoGen: A framework for building AI agent systems," 2023. https://github.com/microsoft/autogen

[21] "Browser Use: Make websites accessible for AI agents," 2024. https://github.com/browser-use/browser-use

[22] R. Fielding, M. Nottingham, and J. Reschke, "HTTP Semantics," RFC 9110, Internet Engineering Task Force, June 2022. https://www.rfc-editor.org/rfc/rfc9110

[23] Schema.org Community, "Schema.org," 2011. https://schema.org/

[24] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindstrom, "JSON-LD 1.1," W3C Recommendation, July 2020. https://www.w3.org/TR/json-ld11/

[25] "HTML Microdata," W3C Working Group Note, 2021. https://www.w3.org/TR/microdata/

[26] J. Howard, "llms.txt: A proposal for LLM-friendly website content," 2024. https://llmstxt.org/

[27] Jina AI, "Jina Reader: Convert any URL to LLM-friendly text," 2024. https://jina.ai/reader/

[28] Firecrawl, "Firecrawl: Turn entire websites into LLM-ready markdown," 2024. https://firecrawl.dev/

[29] "Crawl4AI: Open-source LLM-friendly web crawler," 2024. https://github.com/unclecode/crawl4ai

[30] Mozilla, "Readability.js," 2015. https://github.com/mozilla/readability

[31] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," RFC 7231, Internet Engineering Task Force, June 2014. https://www.rfc-editor.org/rfc/rfc7231

[32] "Sitemaps XML format," sitemaps.org, 2008. https://www.sitemaps.org/