WRFI: A Discovery Layer for AI Agent Interaction with Web Pages

Joona Kaappa
Aalto University, School of Science
Fiops Oy / wr.fi
joona@wr.fi

March 2026

Abstract

AI agents increasingly browse the web to accomplish user tasks, yet no standard governs how they discover and execute actions on the pages they visit. We describe WRFI (Web-Readable Functional Instructions), a pattern for embedding self-describing, machine-executable instructions in human-facing web pages. Unlike API specifications (OpenAPI), plugin manifests (ChatGPT Plugins), or tool protocols (MCP), WRFI requires no prior registration, no separate documentation, and no client-side integration — the AI reads the page and immediately knows what actions are available. We present `wr.fi` as a reference implementation, report observations from 88 versions of a shared document edited by four AI sessions with zero configuration, and identify three novel security risks arising from AI agents autonomously discovering and executing web page instructions: transitive trust, autonomous discovery at scale, and cross-domain action chains. We propose client-enforced guardrails and position WRFI as a bootstrap layer complementing existing protocols.

1 Introduction

When an AI agent browses a web page, it already interprets and acts on the content it finds. A user asks "book me a table at this restaurant" and the agent reads the page, finds a phone number or booking widget, and attempts to act. This behaviour exists today across Claude, ChatGPT, Gemini, and other AI tools with browsing capabilities — it is not hypothetical.

Yet there is no standard governing this interaction. The agent guesses what constitutes an instruction versus content. There is no structured way for a page to declare what actions are available, what methods they accept, or what trust level they require. The result is ad-hoc, invisible, and unauditable AI-web interaction.

WRFI (Web-Readable Functional Instructions) addresses this gap. A WRFI-enabled page embeds a machine-readable instruction block in its HTML that describes available actions, accepted payloads, and endpoint URLs. An AI agent visiting the page discovers these instructions through standard content negotiation and can execute them without prior setup.

The key insight is that **the page itself is the documentation**. There is no separate API specification to discover, no plugin directory to register with, no manifest at a well-known URL. The human-facing page and the machine-readable API specification coexist at the same URL.

2 Related Work

Several existing approaches address machine-readable API discovery, each with different assumptions about the interaction model:

WRFI occupies a distinct position: it is embedded in the page (unlike OpenAPI), requires no registration (unlike ChatGPT Plugins), is executable (unlike llms.txt), requires no client integration (unlike

Approach	Mechanism	Gap for AI agents
OpenAPI/Swagger	Separate specification document describing REST APIs	Requires discovery; often requires authentication to read; not embedded in human-facing pages
ChatGPT Plugins	Manifest at `/.well-known/ai-plugin.json` with OpenAPI spec	Required registration with OpenAI; deprecated in favour of GPTs
llms.txt	Markdown documentation at `/llms.txt` for LLM consumption	Readable but not executable; no action specification
MCP	Bidirectional protocol between AI clients and tool servers	Rich capabilities but requires server-side integration and client support
Semantic Web	RDF/RDFa structured data embedded in HTML	Machine-readable but required specialised parsers; no action model

Table 1: Existing approaches to machine-readable web interaction.

MCP), and uses natural language with JSON structures (unlike the Semantic Web). It is intentionally simpler than all of these and does not replace them — it serves as a bootstrap layer for discovery of simple actions.

3 The WRFI Pattern

A WRFI-enabled page serves different content based on the client:

```
GET example.com (browser)    -> HTML page for humans
GET example.com (AI agent)   -> JSON instruction block
POST example.com             -> Execute described action
```

Content negotiation is performed via `Accept` headers and user-agent detection. The instruction block contains:

- **Endpoint URL** — where to send requests
- **Accepted methods** — GET, POST, etc.
- **Payload schema** — accepted fields and formats
- **Trust level** (hint) — `user-initiated` or `autonomous`
- **Risk level** (hint) — `read-only`, `creates-content`, `modifies-data`, `financial`
- **Examples** — complete request/response pairs

Trust and risk fields are **UX hints for display purposes only**. A compliant AI client must not use page-declared metadata for automated permission decisions, as a malicious page will misrepresent its risk level. This aligns with MCP's specification, which states that tool annotations "should not be trusted as a security mechanism" [1].

3.1 Natural Language Credentials

WRFI introduces natural language authentication tokens: two-word edit tokens (e.g., "Blue-Castle") and four-word API keys drawn from a 2,048-word vocabulary. These are designed for AI-mediated workflows where credentials must survive context switches, conversation summaries, and cross-session handoffs. A user can say "update wr.fi/abcd with token Blue-Castle" in a new AI session and the instruction is unambiguous.

These are explicitly convenience credentials suitable for collaborative content, not security credentials for sensitive material.

4 Reference Implementation: wr.fi

We implemented WRFI as `wr.fi`, a content repository where the root URL serves as both the human interface and the machine API. The instruction "share this to wr.fi" is sufficient for any AI agent with HTTP capabilities to discover the API, publish content, and return a permanent short URL.

4.1 Architecture

The system uses content-addressed storage (SHA-256) with automatic deduplication, in-place versioning with unified diffs, and content negotiation serving 7 of 7 tested AI models. The stack is Next.js 16, SQLite, Docker, AWS Lightsail, and Cloudflare, running on approximately $49/month infrastructure.

4.2 Multi-Session Collaboration Observation

During development, the project's master document accumulated 88 versions edited by four AI sessions across two machines with zero prior configuration:

1. **Claude.ai** (claude.ai) — strategy, messaging, naming decisions
2. **Claude Code** (Lightsail server) — implementation, 24-commit sessions
3. **ChatGPT GPT-5.4** (relayed) — multi-model QA, bug reports
4. **Relay sessions** — cross-machine context handoffs

No session was configured to know about the others. Each read the latest version from the shared URL, edited, and pushed. The URL became *shared context* between AI environments with no other communication channel.

A write conflict at version 35 — where one session overwrote another's changes from a stale local copy — led to the design of an optimistic concurrency protocol using `expectedVersion` fields and server-side merge warnings.

ChatGPT's testing revealed content negotiation failures that Claude-based testing missed, demonstrating the value of cross-model QA through shared artifacts.

5 Security Analysis

5.1 Standard Risks

The open POST endpoint presents standard web application security concerns: spam, storage exhaustion, content abuse. These are addressed through established techniques (rate limiting, content scanning, proof-of-work challenges) and are not novel contributions.

5.2 Novel Risks

WRFI introduces a new trust boundary that does not exist in traditional web security: AI agents autonomously discover and execute instructions from pages that the user never directly evaluated.

Transitive trust. In traditional web interaction, the trust chain is: human $\rightarrow$ page $\rightarrow$ action. The human evaluates the page and decides to act. With AI agents, the chain becomes: human $\rightarrow$ AI $\rightarrow$ page $\rightarrow$ action. The user trusts the AI; the AI encounters page instructions; but the user never approved trusting that specific page's instructions. The AI acts as an unaudited trust bridge.

Autonomous discovery at scale. If WRFI-style instructions become prevalent, every page an AI visits during a browsing session could contain executable instructions. The attack surface expands from individual endpoints to the entire web. Attackers need not trick users into visiting malicious pages — they need only ensure an AI encounters them during task execution.

Cross-domain action chains. An AI reads page A (legitimate), follows a link to page B (compromised), discovers WRFI instructions on B, and executes. The user's request concerned A; the action occurred via B. This multi-hop attack pattern has no direct equivalent in traditional web security models.

5.3 Proposed Guardrails

All enforcement must reside in the AI client. Page-declared metadata (trust, risk) cannot be trusted and serves only as display hints.

1. **Same-origin restriction** — instructions may only direct actions to the same domain. Cross-domain actions require explicit user confirmation.
2. **Method restrictions** — GET and same-origin POST are permitted without confirmation; PUT, DELETE, and cross-origin POST require user approval.
3. **No action chains without consent** — executing an instruction that leads to discovering instructions on a different domain requires user confirmation before the second execution.
4. **No loops** — recursive or repeated action execution from WRFI instructions requires user confirmation at each iteration.
5. **User intent supremacy** — WRFI instructions never override the user's original request.

6 Positioning

WRFI is a bootstrap and discovery layer, not a comprehensive API standard. We position it at the bottom of a complementary stack:

- **WRFI** — AI visits a page, discovers what it can do (simple, user-initiated actions)
- **OpenAPI** — formal API contract with schemas and authentication (structured integration)
- **MCP** — native tool surface with bidirectional communication (deep integration)

A site may start with WRFI for discovery and graduate to OpenAPI or MCP as interaction complexity grows. Coexistence is the design intent.

7 Limitations and Future Work

The multi-session collaboration observations are from a single project during development, not a controlled study. External user data is required to validate whether the zero-configuration push pattern is broadly useful. The security guardrails are proposed but not yet validated through adversarial testing.

Future work includes: controlled user studies of cross-session collaboration via shared URLs, adversarial evaluation of the proposed trust boundary model, formal specification of client-side enforcement requirements, and analysis of WRFI adoption patterns alongside existing API discovery mechanisms.

8 Conclusion

AI agents already discover and act on web page instructions without any governing standard. WRFI provides a structured alternative: embedded, self-describing instruction blocks that make AI-web interaction visible, auditable, and filterable. The reference implementation demonstrates that zero-configuration content sharing across AI sessions is practical, and the security analysis identifies three novel risks requiring client-enforced guardrails. We release the implementation as AGPL-3.0 and the specification as CC-BY-4.0 at `wr.fi`.

References

[1] Anthropic. Model Context Protocol Specification, 2024. https://modelcontextprotocol.io

[2] J. Willison. llms.txt: A proposal for LLM-readable site documentation, 2025. https://llmstxt.org

[3] OpenAPI Initiative. OpenAPI Specification v3.1.0, 2021. https://spec.openapis.org/oas/v3.1.0

[4] OpenAI. ChatGPT Plugins Documentation, 2023. https://platform.openai.com/docs/plugins

[5] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.