Barkhausen AI

NOTE

Protocols for agent access: the landscape at concept level

The top of the Barkhausen Ladder (BL-8) describes an entity that exposes a machine-actionable interface an agent can invoke directly, rather than one an agent reaches by parsing a rendered page.

“Every leap begins as a crackle.”

KIND	Note
DATE	2026
SUBJECTS	Engines & documentation; Conventions & policy
LICENSE	CC-BY-4.0
AUTHOR	Barkhausen AI

ABSTRACT

The top of the Barkhausen Ladder (BL-8) describes an entity that exposes a machine-actionable interface an agent can invoke directly, rather than one an agent reaches by parsing a rendered page. This note maps, at concept level, the emerging protocols by which a site can expose such an interface — the Model Context Protocol, Microsoft's NLWeb, and OpenAPI-described endpoints of the kind OpenAI's GPT Actions consume — and marks the boundary that separates them from the page-parsing agents described elsewhere in this publication and from robots.txt, which is a crawler-access mechanism rather than an agent protocol. Every characterization is drawn from the projects' own documentation. The note takes no position on which protocol will prevail and reports that no public adoption statistics exist to quantify any of them.

CONTENTS

There are two ways software can obtain something from a website. It can read the page a person would read — the rendered markup, or the accessibility tree beneath it, or a screenshot of the result — and extract what it needs. Or the site can expose a structured interface the software calls directly, receiving data or performing an action without a page in the loop at all. The first is the subject of the whitepaper on machine readers (BA-W-2026-03) and the companion note on reading the accessibility tree: page-parsing agents, which cope with the web as published. The second is the top of the Barkhausen Ladder — BL-8, whose marker is a machine-actionable interface an agent can invoke without parsing a web page. This note maps the protocols that occupy that second layer, at the level of what each one is, not how to implement it. It is a photograph of a young and unsettled landscape, and it is descriptive throughout: it defines no requirements and ranks nothing.

1 The distinction that organizes the layer

The whitepaper draws the line this note starts from: "BL-8's marker is a machine-actionable interface an agent can invoke without parsing a web page at all; the agents documented here are doing precisely the page-parsing that BL-8 endpoints exist to make unnecessary." Page-parsing agents and invokable endpoints are not competing implementations of one idea; they are different layers. A page-parsing agent works against a site that was built for human readers, reconstructing structure from presentation. An invokable endpoint is structure the site publishes on purpose, addressed to software. The protocols below are all mechanisms for the second — ways a site can declare, in a form an AI system can discover and call, what data and actions it offers.

2 Model Context Protocol

The Model Context Protocol (MCP) is documented as "an open-source standard for connecting AI applications to external systems" [1]. Its unit is a server that a site or service runs to expose its data and tools; an AI application acting as a client connects to that server and can then, in the documentation's terms, reach "data sources (e.g. local files, databases), tools (e.g. search engines, calculators) and workflows (e.g. specialized prompts)" [1]. The documentation offers a deliberately plain analogy: "Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to connect AI applications to external systems" [1]. What MCP standardizes, then, is the shape of the connection — how a server advertises the tools and resources it holds and how a client invokes them — rather than any particular data or vocabulary. It is documented as "an open protocol supported across a wide range of clients and servers," with the AI assistants Claude and ChatGPT among the clients its own page names as supporting it [1]. This is the protocol the Ladder cites as its public example of a BL-8 interface (BA-C-1).

3 NLWeb

NLWeb is described by Microsoft as "an open project developed by Microsoft that aims to make it simple to create a rich, natural language interface for websites" [2]. Where MCP standardizes the connection abstractly, NLWeb is pitched at the level of a website's existing published data: its repository documents it as "a collection of open protocols and associated open source tools" that leans on "Schema.org and related semi-structured formats like RSS" — the structured markup many sites already publish — to build a natural-language endpoint over a site's own content [3]. Its relationship to MCP is explicit and worth recording, because it shows the layer composing rather than fragmenting: Microsoft states that "Every NLWeb instance is also a Model Context Protocol (MCP) server, allowing websites to make their content discoverable and accessible to agents" [2]. The project frames its own ambition in protocol-stack terms —

"NLWeb is to MCP/A2A what HTML is to HTTP" [3] — positioning itself as a content-and-query layer above the connection layer MCP provides. Its documentation is candid that the shipped code is "proof-of-concept demonstrations showing one possible approach" rather than a finished standard [3].

4 OpenAPI-described endpoints

The third pattern predates the current wave and reuses an established specification. Here a site exposes an ordinary web API described by an OpenAPI schema, and an agent platform reads that schema to learn how to call the API. OpenAI's GPT Actions are the documented instance: OpenAI's crawler documentation states that "ChatGPT users may also interact with external applications via GPT Actions" [4], and the Actions documentation makes the mechanism concrete — "A GPT Action requires an Open API schema to describe the parameters of the API call, which is a standard for describing APIs" [5]. The schema is what the model reads to decide which call to make and with what parameters; the endpoint itself is a conventional RESTful API. Conceptually this is the oldest and most familiar of the three approaches — an API plus a machine-readable description of it — repositioned as an agent-facing interface. It differs from MCP and NLWeb in provenance rather than in kind: all three let an agent discover and invoke a structured interface a site publishes deliberately.

5 Why robots.txt is not on this list

It is tempting to file robots.txt alongside these protocols, since it is the other machine-readable file a site publishes for automated visitors. It belongs to a different layer. Robots.txt is a crawler-access mechanism: it lets a site request that named crawlers not fetch certain paths, and its governing specification states that its rules "are not a form of access authorization" [6]. It exposes nothing for an agent to invoke; it grants or withholds fetch access to pages, which is the page-parsing world, not the invokable-endpoint one. The crawler taxonomy (BA-C-6) sharpens the point from the measurement side: agent traffic is largely not identifiable from user-agent strings at all, so robots.txt cannot even reliably address agents as a class, let alone offer them a structured interface. The protocols above are the structured alternative to a mechanism that was never built for this purpose — a way to publish an interface rather than to permit or refuse a fetch.

6 Limitations

This is a landscape photograph taken on 2026-07-10, and the landscape is early. The three protocols are young, their documentation changes without notice, and the characterizations here are drawn from each project's own published material, which describes intent and design more than deployment. No public adoption statistics exist for any of them — no verified count of sites running an MCP server, an NLWeb instance, or an agent-facing OpenAPI endpoint — and none is estimated here; a number offered without a source would be a fabrication, not a measurement. The set is also not exhaustive: it covers the protocols whose official documentation supports a concrete description, and other named efforts exist that the public record does not yet let this note characterize. Which of these approaches, if any, becomes the layer's convention is not something the current documentation decides, and this note does not predict it. The specifics above should be assumed perishable.

REFERENCES

1. Model Context Protocol (open-source specification). *What is the Model Context Protocol (MCP)? — Introduction* (2026). https://modelcontextprotocol.io/docs/getting-started/intro Accessed 2026-07-10. [archived]

2. Microsoft (Microsoft Source). *Introducing NLWeb: Bringing conversational interfaces directly to the web* (2025). https://news.microsoft.com/source/features/company-news/introducing-nlweb-bringing-conversational-interfaces-directly-to-the-web/ Accessed 2026-07-10. [archived]
3. Microsoft. *NLWeb — README (github.com/microsoft/NLWeb)* (2026). https://github.com/microsoft/NLWeb Accessed 2026-07-10. [archived]
4. OpenAI. *Overview of OpenAI Crawlers* (2026). https://developers.openai.com/api/docs/bots Accessed 2026-07-10. [archived]
5. OpenAI. *Getting started with GPT Actions* (2026). https://developers.openai.com/api/docs/actions/getting-started Accessed 2026-07-10. [archived]
6. M. Koster, G. Illyes, H. Zeller, and L. Sassman, IETF. *RFC 9309: Robots Exclusion Protocol* (2022). https://www.rfc-editor.org/rfc/rfc9309.html Accessed 2026-07-08. [archived]

HOW TO CITE

Barkhausen AI (2026). Protocols for agent access: the landscape at concept level. https://barkhausen.ai/notes/agent-protocol-landscape/

Barkhausen AI · Est. MMXXVI · *Every leap begins as a crackle.*