AI Preferences
Internet-Draft
Intended status: Experimental
Expires: 22 October 2025

L. Peiyuan
Cond迪 Nast

20 April 2025

Protocol for Automation Control
draft-liao-aipref-autoctl-core-01

Abstract

This document specifies a machine-readable protocol for server-side automation permissions in the light of recent advances in AI-driven web automation. Building upon RFC9309, this protocol addresses a broader range of state-changing activities that service owners may wish to control. It defines the file format, HTTP method restrictions, and purpose requirements.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://datatracker.ietf.org/doc/draft-liao-aipref-autoctl-core/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-liao-aipref-autoctl-core/.

Discussion of this document takes place on the AI Preferences Working Group mailing list (mailto:ai-control@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ai-control/. Subscribe at https://www.ietf.org/mailman/listinfo/ai-control/.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 October 2025.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The evolution of web automation has outpaced the capabilities of existing standards that only provide for read-only crawler permissions. Sophisticated, AI-driven bots are now able to interact with web servers in a complex, human-like manner, allowing them to deeply acquire and modify content. This document introduces a protocol that enables service owners to declare policies governing such interactions, notably state-changing HTTP methods.

1.1. Applicability

The automation-preferences.txt _file_ applies to automated systems interacting with web servers, especially those driven by AI models. Content owners may use this file to specify acceptable automation behaviors, and developers of automated systems may use these directives to ensure compliance.

1.2. Relationship to Extension Specification

A separate document, "Protocol Extension for Automation Control," extends this document with additional directives and capabilities, focusing on a wider range of state-changing web requests.

Implementations conforming to only this specification are considered compliant with the protocol.

This protocol _augments_, and does not relax, the directives of [RFC9309]. If a path is disallowed in robots.txt, an automation-preferences.txt directive MUST NOT be interpreted to expand access. Conversely, when robots.txt explicitly allows any HTTP method on a path, an automation-preferences.txt directive that disallows state-changing requests to that same path SHALL take precedence; the more restrictive rule MUST be enforced.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document:

* Automation: Programmatic interactions with a web server.

* State-changing requests: HTTP methods that alter server state (e.g., POST, PUT, DELETE, PATCH).

* Automation purpose: The declared intent or use case for which automation is being performed.

All terminology defined in this core specification applies to the extension specification without redefinition. The extension specification may introduce additional terms for concepts not covered in this document.

3. Protocol Specification

3.1. File Location and Format

The automation-preferences.txt file MUST be hosted at the root of the domain, in the same manner as [RFC9309]. The file is structured as a series of key-value pairs that specify automation permissions.

The file MUST be served with the media type text/plain; charset=utf-8. Lines beginning with the hash symbol (#) are considered comments and MUST be ignored by parsers. Each directive consists of a field name, followed by a colon, followed by a value. Multiple values MAY be separated by commas. Parsers MUST silently ignore any directives they do not recognize.

Implementations SHOULD reject files that contain raw control bytes (code points < U+0020) other than the permitted CR (U+000D) and LF (U+000A) line breaks.

3.2. HTTP Method Restrictions

The protocol MUST explicitly list allowed HTTP methods using the allowed-methods directive. Typically, GET and HEAD are permitted while methods such as POST, PUT, DELETE, and PATCH are disallowed for automated processing. Methods that are not listed are assumed to be disallowed.

Example:

```
allowed-methods: GET, HEAD
<!-- Note: This example shows basic method allowance -->
```

Figure 1

If a group omits the allowed-methods directive, _all_ HTTP methods are considered disallowed for automated processing within that scope.

3.3. Purpose Declaration

The protocol allows server owners to accept or reject user agents with specific purposes with the allowed-purposes directive, which accepts a comma-separated list of permitted purposes using standardized vocabulary terms.

The specific vocabulary terms for automation purposes are not defined in this document. Instead, this protocol provides a mechanism for expressing allowed and disallowed purposes, which can be compatible with an accepted vocabulary standard in the future.

Example:

```
allowed-purposes: PLACEHOLDER_PURPOSE1, PLACEHOLDER_PURPOSE2
<!-- Note: Placeholder purposes are used here -->
```

Figure 2

3.4. Scope and Applicability

The automation-preferences.txt file is divided into groups, each of which applies to a specific subset of content. Each group begins with one or more scope directives that define the target of the preferences. The following directives MAY be used within a group:

* scope: Specifies the URL pattern (e.g., /admin/) to which the group applies. Wildcards MAY be used to indicate variable components of the URL.

* host: Specifies a subdomain or host. If present, the group applies only to the indicated subdomain; if omitted, the group is assumed to apply to the entire host.

* user-agent: Specifies one or more automation user-agent tokens to which the group applies. If omitted, the group applies to all user agents.

Groups are processed in order of specificity. Specificity is determined by (1) an _exact host match_ over a wildcard or absent host; (2) the _longest matching scope directive_; (3) the _most specific user-agent token_ (an exact token outranks the wildcard *); then (4) appearance order, where the group appearing later in the file SHALL take precedence when all preceding criteria are equal.

A group MUST contain at least one scope directive and MAY omit all other directives.

Example:

```
<!-- Group 1: Applies to the entire site -->
user-agent: *
host: example.com
scope: /
allowed-methods: GET, HEAD

<!-- Group 2: Specific preferences for the /admin/ path -->
user-agent: *
host: example.com
scope: /admin/
allowed-methods: GET
```

Figure 3

4. Formal Syntax

Below is an Augmented Backus-Naur Form (ABNF) description, as defined in [RFC5234], with references to [RFC3629], [RFC3986], and [RFC9309].

```
automation-preferences = *( group )

group            = 1*scope-directive           ; at least one <scope>
                   *( directive / emptyline )
                   1*emptyline                 ; blank line terminates group

directive        = scope-directive / host-directive /
                   user-agent-directive /
                   method-directive / purpose-directive

; --- individual directives ---------------------------------

scope-directive   = *WS "scope"            *WS ":" *WS url-pattern    EOL
host-directive    = *WS "host"             *WS ":" *WS host-pattern   EOL
method-directive  = *WS "allowed-methods"  *WS ":" *WS method-list    EOL
purpose-directive = *WS "allowed-purposes" *WS ":" *WS purpose-list   EOL
user-agent-directive  = *WS "user-agent" *WS ":" *WS product-token
                        *( *WS "," *WS product-token ) EOL

; --- directive value syntax --------------------------------

; url-pattern: visible ASCII or UTF-8, implementers SHOULD interpret it
;              as an RFC 3986 path/authority matcher with wildcards
url-pattern     = 1*( VCHAR / UTF8-char-noctl )

; host-pattern: UTF-8 hostname or wildcard; servers SHOULD match IDNA U-labels
host-pattern    = 1*( ALPHA / DIGIT / "-" / "." / UTF8-char-noctl )

method-list     = method *( *WS "," *WS method )
```

```
   method          = "GET" / "HEAD" / "POST" / "PUT" /
                     "DELETE" / "PATCH" / "OPTIONS" /
                     "TRACE" / "CONNECT"

   purpose-list    = purpose-token *( *WS "," *WS purpose-token )
   purpose-token   = 1*VCHAR   ; placeholder for future vocabulary

   ; product-token: derived from RFC 9309
   product-token         = identifier / "*"
   identifier            = 1*( %x2D / %x41-5A / %x5F / %x61-7A )

   ; --- lexical primitives ------------------------------------

   comment         = "#" *( UTF8-char-noctl / WS / "#" )

   emptyline       = *WS [comment] EOL
   EOL             = *WS [comment] NL
   NL              = CRLF / LF / CR
   CRLF            = CR LF
   CR              = %x0D
   LF              = %x0A
   WS              = SP / HTAB
   SP              = %x20
   HTAB            = %x09

   ; --- core ABNF terminals (RFC 5234) -------------------------

   ALPHA           = %x41-5A / %x61-7A
   DIGIT           = %x30-39
   VCHAR           = %x21-7E

   ; --- UTF-8 (derived from RFC 3629) --------------------------

   UTF8-char-noctl = UTF8-1-noctl / UTF8-2 / UTF8-3 / UTF8-4
   UTF8-1-noctl    = %x21 / %x22 / %x24-7F
   UTF8-2          = %xC2-DF UTF8-tail
   UTF8-3          = %xE0 %xA0-BF UTF8-tail
                   / %xE1-EC UTF8-tail-2
                   / %xED %x80-9F UTF8-tail
                   / %xEE-EF UTF8-tail-2
   UTF8-4          = %xF0 %x90-BF UTF8-tail-2
                   / %xF1-F3 UTF8-tail-3
                   / %xF4 %x80-8F UTF8-tail-2
   UTF8-tail       = %x80-BF
   UTF8-tail-2     = UTF8-tail UTF8-tail
   UTF8-tail-3     = UTF8-tail UTF8-tail UTF8-tail
```

5. Extension Mechanism

The automation-preferences.txt file defines a forward-compatible approach where implementations:

* MUST process all recognized directives according to this specification
* MUST silently ignore any unrecognized directives
* MUST NOT fail or produce errors when encountering extended directives

This enables future extensions to add new capabilities, which may include:

* Rate limiting controls
* Automation technology restrictions
* API and XHR permissions
* Session requirements
* Asset-level annotation methods

6. Implementation and Enforcement

Servers implementing this protocol SHOULD:

* Verify incoming requests against automation-preferences.txt.
* Respond with appropriate HTTP status codes (e.g., 403 Forbidden) [RFC9110] for non-compliant requests.

Clients consuming this protocol SHOULD:

* Fetch and parse automation-preferences.txt before performing automated operations.
* Honor the HTTP method restrictions specified.
* Declare their automation purpose.
* Respect the scope directives when performing operations on different paths.

Implementations MAY cache automation-preferences.txt in accordance with [RFC9111]; all freshness calculations are governed solely by standard HTTP cache-control semantics.

7. Security Considerations

The use of automation-preferences.txt introduces security considerations that SHOULD be assessed by implementors:

* Parsing of automation-preferences.txt MUST be performed securely to prevent vulnerabilities such as buffer overruns and denial-of-service attacks.

* Care SHOULD be taken to avoid exposing sensitive scope details that could be exploited by adversaries.

* The protocol does not provide authentication or cryptographic verification mechanisms for the file content. Servers SHOULD ensure the file is served via secure connections to prevent tampering.

* The protocol does not enforce client compliance; it relies on good-faith adherence by automation providers. Servers SHOULD implement additional detection and enforcement mechanisms when needed.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC9309] Koster, M., Illyes, G., Zeller, H., and L. Sassman, "Robots Exclusion Protocol", RFC 9309, DOI 10.17487/RFC9309, September 2022, <https://www.rfc-editor.org/info/rfc9309>.

9.2. Informative References

Sample automation-preferences.txt File

The following is an example of an automation-preferences.txt file that adheres to this specification:

```
<!-- Automation preferences for example.com -->
<!-- Version: 1.0 -->
<!-- Last updated: 2025-04-20 -->

<!-- Group 1: Applies to the entire site -->
scope: /
user-agent: *
host: example.com
allowed-methods: GET, HEAD
allowed-purposes: PLACEHOLDER_PURPOSE1, PLACEHOLDER_PURPOSE2

<!-- Group 2: Specific preferences for the /admin/ path -->
scope: /admin/
user-agent: ExampleBot
host: example.com
allowed-methods: GET
allowed-purposes: PLACEHOLDER_PURPOSE1
```

Figure 4

Author’s Address

Liao Peiyuan
Cond迪 Nast
United States of America
Email: peiyuan_liao@condenast.com