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

L. Peiyuan
Cond迪 Nast

20 April 2025

Protocol Extension for Automation Control
draft-liao-aipref-autoctl-ext-01

Abstract

This document specifies extensions to [CORE-SPEC], providing a wider range of controls for server-side automation permissions. It supports rate limiting, automation technology restrictions, API permissions, session requirements, and HTML asset annotations. These extensions enable service owners to exercise more granular control over automated interactions.

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-ext/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-liao-aipref-autoctl-ext/.

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.

Table of Contents

1. Introduction

Often, the behaviors of automated systems may exhibit a certain pattern that is hard to characterize with only the HTTP method and user-agent used by [CORE-SPEC]. This document introduces a set of extensions for service owners to specify more complex and granular automation policies involving rate limiting, particularly prevalent automation technologies, API permissions, session requirements, and annotations on HTML assets.

1.1. Relationship to Core Specification

All directives and mechanisms defined in the core specification remain valid and are not redefined here. This document assumes familiarity with [CORE-SPEC] and uses its terminology and concepts throughout.

The extensions defined in this document are OPTIONAL for both servers and clients. Implementations that support only the core specification are considered compliant with the automation-preferences.txt protocol, though they will not benefit from the granular controls defined here.

When both core and extended directives are present in an automation-preferences.txt file, parsers that do not support the extensions defined in this document MUST ignore the unrecognized directives, as specified in the core specification’s extension-mechanism.

2. Conventions and Terminology

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.

This document uses the terminology defined in [CORE-SPEC]. The following additional terms are introduced:

* Rate limiting: Constraints on the frequency or concurrency of automated requests.

* Automation technology: Specific tools or frameworks used for automation, such as headless browsers or browser automation protocols.

* XHR/Fetch: XMLHttpRequest or Fetch API calls performed programmatically.

* Session validation: Mechanisms to verify that automated requests are part of a legitimate user session.

* Asset annotation: Metadata embedded within HTML documents to specify automation policies for individual content elements.

3. Extended Protocol Specification

This section defines additional directives that extend the automation-preferences.txt protocol. These directives may be used alongside the core directives in any group within the automation-preferences.txt file.

Unless stated otherwise, the empty form of any comma-separated list means "explicitly none", distinct from omitting the directive.

3.1. Rate Limiting

The following directives are defined:

* request-limit: Specifies the maximum number of requests allowed within a time period, expressed as a count followed by a time unit (e.g., "60/minute"). Supported time units are "second", "minute", "hour", and "day".

* concurrent-limit: Specifies the maximum number of concurrent connections allowed from a single client.

Example:

```
request-limit: 60/minute
concurrent-limit: 5
```

Figure 1

Rate limiting directives apply to all requests within the scope of the group, regardless of HTTP method. If no rate limiting directives are specified, clients SHOULD NOT assume any specific rate limits.

3.2. Automation Technology Restrictions

Automation technology directives specify whether particular automation capabilities are permitted. Automation mechanisms fall into two orthogonal layers: _protocol_ (the wire format exchanged with the browser) and _runtime_ (whether the browser runs headless or headed). Tokens defined in Section 3.2.1 and Section 3.2.2 MAY be mixed in any order.

The following directive is defined:

* allowed-automations: A comma-separated list of tokens representing permitted automation mechanisms. If this directive is present but empty, it implies that no standard tokens are explicitly permitted. If the directive is absent, clients SHOULD NOT assume any mechanism is allowed.

Default behavior: when the allowed-automations directive is:

present and empty no standard tokens are permitted;

absent the server makes no statement, and clients SHOULD assume the most restrictive stance (i.e., that no automation mechanism is authorised).

This aligns the silent default with that of api-automation, allow-xhr, and disallow-fetch-from, ensuring a single, predictable "fail-closed" rule across all extension directives.

A list-valued directive that appears as name: followed only by whitespace or a comment MUST be parsed as an empty list, not a syntax error. This applies to allowed-automations and all other comma-separated lists in the core or extension specs.

Example (allowing WebDriver traffic in headless mode):

allowed-automations: webdriver, headless

Figure 2

If a token is not listed in a present allowed-automations directive, clients SHOULD assume that the use of that mechanism is not permitted within the specified scope. Implementations SHOULD respect these directives when applicable, even though detection techniques may differ.

The directive accepts any string token, but the following identifiers are suggested for interoperability.

This list is not exhaustive. Future specifications MAY define additional standard tokens.

3.2.1. Protocol Tokens

* webdriver: Automation using the W3C WebDriver wire protocol [WEBDRIVER].

* cdp: Automation using the Chrome DevTools Protocol [CDP].

* webkit-remote: Automation using the WebKit Remote Debugging Protocol [WKREMOTE].

* firefox-remote: Automation using the Firefox Remote Debug Protocol via GeckoDriver [FFREMOTE].

3.2.2. Runtime Tokens

* headless: Browser runs with no visible UI [HEADLESS].

* headed: Browser UI is presented to the user.

3.3. API and XHR Permissions

API and XHR permission directives specify rules for API usage and automated use of XMLHttpRequest, Fetch, or AJAX. The following directives are defined:

* api-automation: Indicates how API endpoints may be accessed by automated clients. Valid values are:

  - _none_: No API automation is permitted.

  - _with-key-only_: API automation is permitted only with proper authentication.

  - _open_: API automation is generally permitted.

* allow-xhr: Indicates how XMLHttpRequest or Fetch API may be used by automated clients. Valid values are:

  - _none_: No XHR/Fetch automation is permitted.

  - _read-only_: Only GET requests are permitted via XHR/Fetch.

  - _open_: XHR/Fetch automation is generally permitted.

* disallow-fetch-from: Comma-separated list of URL patterns from which automated XHR/Fetch requests are prohibited. Wildcards MAY be used.

Example:

```
api-automation: with-key-only
allow-xhr: read-only
disallow-fetch-from: /account/*, /checkout/*, /admin/*
```

Figure 3

If API and XHR permission directives are not specified, clients SHOULD assume the most restrictive value (i.e., "none" for api-automation and allow-xhr).

3.4. Session Requirements

Session requirement directives specify whether automated requests must be part of a legitimate user session. The following directives are defined:

* require-human-initiated-session: Boolean value indicating whether automated requests must be part of a session that was initiated by a human user.

* session-validation: Specifies the method used to validate sessions. Valid values are:

  - _cookie-based_: Sessions are validated using HTTP cookies.

  - _token-based_: Sessions are validated using authentication tokens.

  - _oauth_: Sessions are validated using OAuth.

  - _none_: No session validation is required.

* session-ttl: Specifies the maximum time-to-live for a session, expressed as a duration (e.g., "30m", "2h", "1d").

  A duration literal consists of a sequence of decimal digits immediately followed by a single unit character. Units are mapped as follows:

  - s - seconds (1 <= value <= 86400)

  - m - minutes (1 <= value <= 1440)

  - h - hours (1 <= value <= 168)

  - d - days (1 <= value <= 365)

  Implementations MUST reject values outside these ranges with a client or server error appropriate to their role.

Example:

```
require-human-initiated-session: true
session-validation: cookie-based
session-ttl: 1h
```

Figure 4

If session requirement directives are not specified, clients SHOULD NOT assume any specific session requirements, but SHOULD include a valid User-Agent header in all requests.

3.5. HTML Asset Annotation

Automation preferences MAY be embedded directly within HTML documents to annotate individual assets. Authors SHOULD use structured data markup using JSON-LD [JSON-LD11] embedded in a <script> element. The JSON object SHOULD use a defined type and include relevant fields that mirror those used in automation-preferences.txt.

Note that unlike site-wide directives, asset-level annotations SHOULD NOT include HTTP method restrictions, request limits, or concurrency limits, as these concepts apply to endpoints and services rather than to individual content assets.

Example:

```
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "AutomationPolicyAnnotation",
  "allowedAutomations": [],
  "allowedPurposes": [],
}
</script>
```

Figure 5

When both an automation-preferences.txt file and HTML asset annotations are present, the more specific rule (typically the HTML annotation) SHALL be applied to the corresponding content asset. Clients supporting HTML asset annotations SHOULD parse and respect these annotations when present.

The annotation schema MAY include any directives defined in the core or extension specifications. Fields in the annotation SHOULD use camelCase naming to align with JSON-LD conventions (e.g., allowedAutomations for allowed-automations, requestLimit for request-limit), while maintaining semantic equivalence to the corresponding directives in the automation-preferences.txt file.

4. Formal Syntax

Below is an Augmented Backus-Naur Form (ABNF) description, incorporating the extensions defined in this document. It extends the grammar defined in the core specification Part formal-syntax of [CORE-SPEC].

Rules inherited or unmodified from the core specification are included for completeness. Rules added or modified by this extension are indicated with comments (‘; Extension directive‘ or ‘; Extended rule‘).

```
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 /
                   ; Extended rule: Added extension directives
                   request-limit-directive / concurrent-limit-directive /
                   allowed-automations-directive / api-automation-directive /
                   allow-xhr-directive / disallow-fetch-from-directive /
                   require-human-session-directive / session-validation-directive /
                   session-ttl-directive

; --- Core directives (from CORE-SPEC) ----------------------

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

; --- Extension directives ---------------------------------

request-limit-directive = *WS "request-limit"  *WS ":" *WS rate-spec EOL
concurrent-limit-directive = *WS "concurrent-limit" *WS ":" *WS count EOL
allowed-automations-directive = *WS "allowed-automations" *WS ":" *WS automation-list
EOL
api-automation-directive = *WS "api-automation" *WS ":" *WS api-automation-value EOL
allow-xhr-directive = *WS "allow-xhr" *WS ":" *WS allow-xhr-value EOL
disallow-fetch-from-directive = *WS "disallow-fetch-from" *WS ":" *WS url-pattern-list
 EOL
require-human-session-directive = *WS "require-human-initiated-session" *WS ":" *WS bo
olean EOL
session-validation-directive = *WS "session-validation" *WS ":" *WS session-validation
-value EOL
session-ttl-directive = *WS "session-ttl" *WS ":" *WS duration EOL
```

```
   ; --- Directive value syntax (Core) ------------------------

   url-pattern     = 1*( VCHAR / UTF8-char-noctl )
   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   = identifier / "*"
   identifier      = 1*( %x2D / %x41-5A / %x5F / %x61-7A )

   ; --- Directive value syntax (Extension) -------------------

   rate-spec       = count "/" time-unit
   count           = 1*DIGIT
   time-unit       = "second" / "minute" / "hour" / "day"

   automation-list = *( automation-token *( *WS "," *WS automation-token ) )
   automation-token= 1*VCHAR ; e.g., "cdp", "headless", "selenium"

   api-automation-value = "none" / "with-key-only" / "open"

   allow-xhr-value = "none" / "read-only" / "open"

   url-pattern-list= url-pattern *( *WS "," *WS url-pattern )

   boolean         = "true" / "false"

   session-validation-value = "cookie-based" / "token-based" / "oauth" / "none"

   duration        = 1*DIGIT time-unit-char
   time-unit-char  = "s" / "m" / "h" / "d"  ; s=second, m=minute,
                                            ; h=hour, d=day

   ; --- Lexical primitives (from CORE-SPEC) ------------------

   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. Backward Compatibility

All directives defined in this document are OPTIONAL. Implementations that support only the core specification can safely ignore these directives.

Implementations supporting these extensions SHOULD degrade gracefully when interacting with servers or clients that support only the core specification.

* Servers supporting extensions SHOULD still process all core directives correctly, even if extended directives are also present.

* Clients supporting extensions SHOULD still honor all core directives, even if they do not recognize extended directives in a file.

* When HTML asset annotations are not supported by a client, the client SHOULD fall back to the site-level automation-preferences.txt file for guidance.

6. Implementation and Enforcement

Servers implementing the extensions defined in this document SHOULD:

* Employ detection mechanisms (e.g., CDP fingerprinting, headless browser detection) to identify automated clients using specific technologies.

* Implement rate limiting according to the specified directives.

* Validate sessions as required by the session requirement directives.

* Process HTML asset annotations when interpreting automation policies for specific content.

* Respond with appropriate HTTP status codes [RFC9110] for non-compliant requests, such as:

  - 429 Too Many Requests for rate limit violations.

  - 403 Forbidden for unauthorized automation technology use.

  - 401 Unauthorized for missing or invalid authentication.

Clients supporting these extensions SHOULD:

* Honor rate limiting directives by self-throttling requests.

* Respect automation technology restrictions by only using tools listed in the allowed-automations directive, if present.

* Adhere to API and XHR permissions as specified (api-automation, allow-xhr, disallow-fetch-from).

* Establish and maintain valid sessions when required (require-human-initiated-session, session-validation, session-ttl).

* Parse and respect HTML asset annotations when present.

Both servers and clients MAY implement additional detection and enforcement mechanisms beyond those explicitly described in this document, as long as they maintain compatibility with the specified directives.

7. Security Considerations

In addition to the security considerations mentioned in [CORE-SPEC], the extensions defined in this document introduce the following considerations:

* Rate Limiting: Implementations of rate limiting SHOULD use secure methods to track request counts and prevent circumvention through IP spoofing or other means.

* Technology Detection: Methods used to detect specific automation technologies can be circumvented by sophisticated clients. Servers SHOULD employ multiple detection approaches and adapt to evolving evasion techniques.

* Session Validation: Session validation mechanisms SHOULD be resistant to replay attacks and session hijacking attempts.

* HTML Asset Annotations: Parsing of JSON-LD annotations MUST be performed securely to prevent injection attacks or denial-of-service through malformed input.

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[CORE-SPEC] Liao, P., "Protocol for Automation Control", Work in Progress, Internet-Draft, draft-liao-aipref-autoctl-core-01, April 2025, <https://datatracker.ietf.org/doc/html/draft-liao-aipref-autoctl-core-01>.

[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>.

[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>.

9.2. Informative References

[CDP] Google, "Chrome DevTools Protocol", 2025, <https://chromedevtools.github.io/devtools-protocol/>.

[FFREMOTE] Mozilla, "GeckoDriver Remote Protocol", 2025, <https://firefox-source-docs.mozilla.org/testing/geckodriver/index.html>.

[HEADLESS] Google, "Chrome Headless Mode", 2024, <https://developer.chrome.com/docs/chromium/headless>.

[JSON-LD11] Kellogg, G., Champin, P.-A., and D. Longley, "JSON-LD 1.1", W3C Recommendation REC-json-ld11-20200716, 16 July 2020, <https://www.w3.org/TR/json-ld11/>.

[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>.

[WEBDRIVER] W3C, "WebDriver", 2018, <https://www.w3.org/TR/webdriver/>.

[WKREMOTE] WebKit Project, "WebKit Remote Debugging Protocol", 2015, <https://webkit.org/debugging-webkit/>.

Sample Extended automation-preferences.txt File

The following is an example of an automation-preferences.txt file that includes both core and extended directives:

```
<!-- Automation preferences for example.com -->
<!-- Version: 2.0 (incorporating extensions) -->
<!-- Last updated: 2025-04-20 -->

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

<!-- Extended directives for Group 1 -->
request-limit: 60/minute
concurrent-limit: 5
<!-- Empty - forbids all automation technologies -->
allowed-automations:
api-automation: with-key-only
require-human-initiated-session: true
session-validation: cookie-based
session-ttl: 1h


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

<!-- Extended directives for admin path (Group 2) -->
request-limit: 10/minute
concurrent-limit: 2
require-human-initiated-session: true
session-validation: token-based
session-ttl: 30m


<!-- Group 3: Default for /admin/ for other user agents (less specific than Group 2) -->
user-agent: *
host: example.com
scope: /admin/
allowed-methods: GET
```

Figure 6

Author’s Address

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