Stream: Internet Engineering Task Force (IETF)
RFC: 9309
Category: Standards Track
Published: September 2022
ISSN: 2070-1721
Authors: M. Koster, *Google LLC*; G. Illyes, *Google LLC*; H. Zeller, *Google LLC*; L. Sassman, *Google LLC*

RFC 9309
Robots Exclusion Protocol

Abstract

This document specifies and extends the "Robots Exclusion Protocol" method originally defined by Martijn Koster in 1994 for service owners to control how content served by their services may be accessed, if at all, by automatic clients known as crawlers. Specifically, it adds definition language for the protocol, instructions for handling errors, and instructions for caching.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9309.

Copyright Notice

Table of Contents

1. Introduction

This document applies to services that provide resources that clients can access through URIs as defined in [RFC3986]. For example, in the context of HTTP, a browser is a client that displays the content of a web page.

Crawlers are automated clients. Search engines, for instance, have crawlers to recursively traverse links for indexing as defined in [RFC8288].

It may be inconvenient for service owners if crawlers visit the entirety of their URI space. This document specifies the rules originally defined by the "Robots Exclusion Protocol" [ROBOTSTXT] that crawlers are requested to honor when accessing URIs.

These rules are not a form of access authorization.

1.1. Requirements Language

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.

2. Specification

2.1. Protocol Definition

The protocol language consists of rule(s) and group(s) that the service makes available in a file named "robots.txt" as described in Section 2.3:

Rule: A line with a key-value pair that defines how a crawler may access URIs. See Section 2.2.2.

Group: One or more user-agent lines that are followed by one or more rules. The group is terminated by a user-agent line or end of file. See Section 2.2.1. The last group may have no rules, which means it implicitly allows everything.

2.2. Formal Syntax

Below is an Augmented Backus-Naur Form (ABNF) description, as described in [RFC5234].

```
robotstxt = *(group / emptyline)
group = startgroupline                  ; We start with a user-agent
                                        ; line
       *(startgroupline / emptyline)    ; ... and possibly more
                                        ; user-agent lines
       *(rule / emptyline)              ; followed by rules relevant
                                        ; for the preceding
                                        ; user-agent lines

startgroupline = *WS "user-agent" *WS ":" *WS product-token EOL

rule = *WS ("allow" / "disallow") *WS ":"
      *WS (path-pattern / empty-pattern) EOL

; parser implementors: define additional lines you need (for
; example, Sitemaps).

product-token = identifier / "*"
path-pattern = "/" *UTF8-char-noctl ; valid URI path pattern
empty-pattern = *WS

identifier = 1*(%x2D / %x41-5A / %x5F / %x61-7A)
comment = "#" *(UTF8-char-noctl / WS / "#")
emptyline = EOL
EOL = *WS [comment] NL ; end-of-line may have
                       ; optional trailing comment
NL = %x0D / %x0A / %x0D.0A
WS = %x20 / %x09

; UTF8 derived from RFC 3629, but excluding control characters

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

UTF8-tail = %x80-BF
```

2.2.1. The User-Agent Line

Crawlers set their own name, which is called a product token, to find relevant groups. The product token **MUST** contain only uppercase and lowercase letters ("a-z" and "A-Z"), underscores ("_"), and hyphens ("-"). The product token **SHOULD** be a substring of the identification string that the crawler sends to the service. For example, in the case of HTTP [RFC9110], the product token **SHOULD** be a substring in the User-Agent header. The identification string **SHOULD** describe the purpose of the crawler. Here's an example of a User-Agent HTTP request header with a link pointing to a page describing the purpose of the ExampleBot crawler, which appears as a substring in the User-Agent HTTP header and as a product token in the robots.txt user-agent line:

Figure 1: Example of a User-Agent HTTP header and robots.txt user-agent line for the ExampleBot product token

Note that the product token (ExampleBot) is a substring of the User-Agent HTTP header.

Crawlers **MUST** use case-insensitive matching to find the group that matches the product token and then obey the rules of the group. If there is more than one group matching the user-agent, the matching groups' rules **MUST** be combined into one group and parsed according to Section 2.2.2.

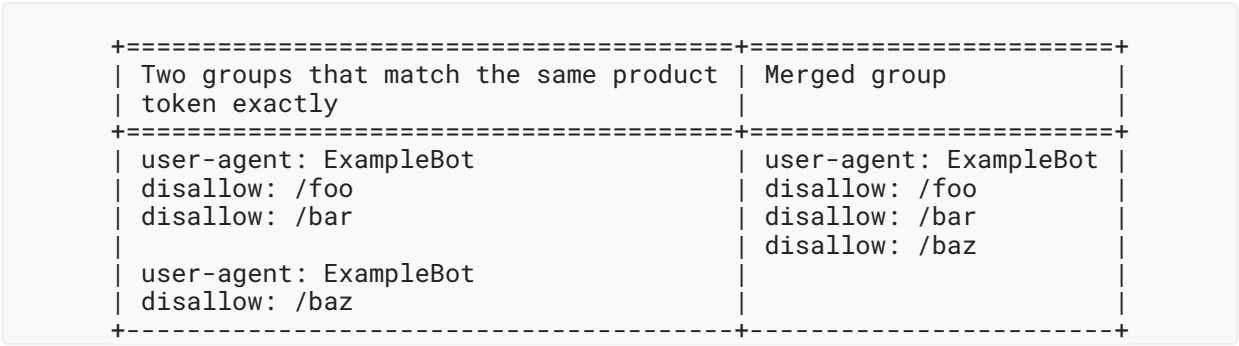

Figure 2: Example of how to merge two robots.txt groups that match the same product token

If no matching group exists, crawlers **MUST** obey the group with a user-agent line with the "*" value, if present.

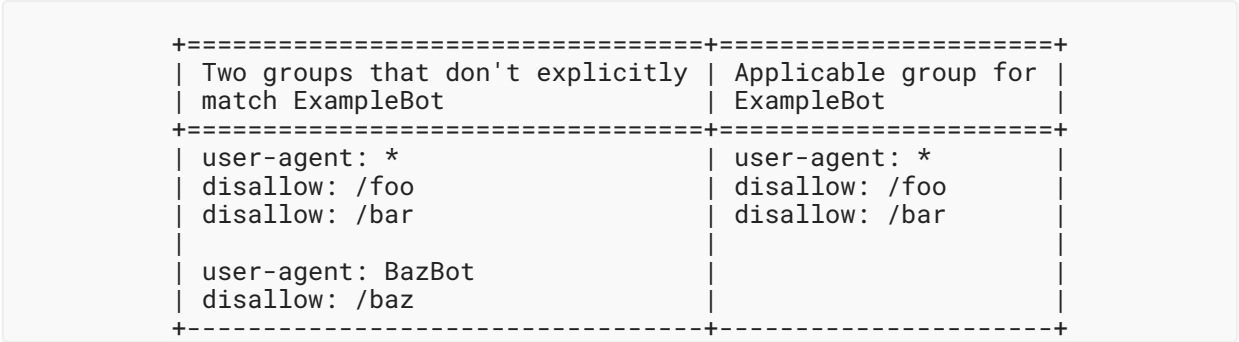

Figure 3: Example of no matching groups other than the "" for the ExampleBot product token*

If no group matches the product token and there is no group with a user-agent line with the "*" value, or no groups are present at all, no rules apply.

2.2.2. The "Allow" and "Disallow" Lines

These lines indicate whether accessing a URI that matches the corresponding path is allowed or disallowed.

To evaluate if access to a URI is allowed, a crawler **MUST** match the paths in "allow" and "disallow" rules against the URI. The matching **SHOULD** be case sensitive. The matching **MUST** start with the first octet of the path. The most specific match found **MUST** be used. The most specific match is the match that has the most octets. Duplicate rules in a group **MAY** be deduplicated. If an "allow" rule and a "disallow" rule are equivalent, then the "allow" rule **SHOULD** be used. If no match is found amongst the rules in a group for a matching user-agent or there are no rules in the group, the URI is allowed. The /robots.txt URI is implicitly allowed.

Octets in the URI and robots.txt paths outside the range of the ASCII coded character set, and those in the reserved range defined by [RFC3986], **MUST** be percent-encoded as defined by [RFC3986] prior to comparison.

If a percent-encoded ASCII octet is encountered in the URI, it **MUST** be unencoded prior to comparison, unless it is a reserved character in the URI as defined by [RFC3986] or the character is outside the unreserved character range. The match evaluates positively if and only if the end of the path from the rule is reached before a difference in octets is encountered.

For example:

Path	Encoded Path	Path to Match
/foo/bar?baz=quz	/foo/bar?baz=quz	/foo/bar?baz=quz
/foo/bar?baz=https://foo.bar	/foo/bar?baz=https%3A%2F%2Ffoo.bar	/foo/bar?baz=https%3A%2F%2Ffoo.bar
/foo/bar/U+E38384	/foo/bar/%E3%83%84	/foo/bar/%E3%83%84
/foo/bar/%E3%83%84	/foo/bar/%E3%83%84	/foo/bar/%E3%83%84
/foo/bar/%62%61%7A	/foo/bar/%62%61%7A	/foo/bar/baz

Figure 4: Examples of matching percent-encoded URI components

The crawler **SHOULD** ignore "disallow" and "allow" rules that are not in any group (for example, any rule that precedes the first user-agent line).

Implementors **MAY** bridge encoding mismatches if they detect that the robots.txt file is not UTF-8 encoded.

2.2.3. Special Characters

Crawlers **MUST** support the following special characters:

Character	Description	Example
#	Designates a line comment.	allow: / # comment in line # comment on its own line
$	Designates the end of the match pattern.	allow: /this/path/exactly$
*	Designates 0 or more instances of any character.	allow: /this/*/exactly

Figure 5: List of special characters in robots.txt files

If crawlers match special characters verbatim in the URI, crawlers **SHOULD** use "%" encoding. For example:

Percent-encoded Pattern	URI
/path/file-with-a-%2A.html	https://www.example.com/path/file-with-a-*.html
/path/foo-%24	https://www.example.com/path/foo-$

Figure 6: Example of percent-encoding

2.2.4. Other Records

Crawlers **MAY** interpret other records that are not part of the robots.txt protocol -- for example, "Sitemaps" [SITEMAPS]. Crawlers **MAY** be lenient when interpreting other records. For example, crawlers may accept common misspellings of the record.

Parsing of other records **MUST NOT** interfere with the parsing of explicitly defined records in Section 2. For example, a "Sitemaps" record **MUST NOT** terminate a group.

2.3. Access Method

The rules **MUST** be accessible in a file named "/robots.txt" (all lowercase) in the top-level path of the service. The file **MUST** be UTF-8 encoded (as defined in [RFC3629]) and Internet Media Type "text/plain" (as defined in [RFC2046]).

As per [RFC3986], the URI of the robots.txt file is:

"scheme:[//authority]/robots.txt"

For example, in the context of HTTP or FTP, the URI is:

```
https://www.example.com/robots.txt

ftp://ftp.example.com/robots.txt
```

2.3.1. Access Results

2.3.1.1. Successful Access

If the crawler successfully downloads the robots.txt file, the crawler **MUST** follow the parseable rules.

2.3.1.2. Redirects

It's possible that a server responds to a robots.txt fetch request with a redirect, such as HTTP 301 or HTTP 302 in the case of HTTP. The crawlers **SHOULD** follow at least five consecutive redirects, even across authorities (for example, hosts in the case of HTTP).

If a robots.txt file is reached within five consecutive redirects, the robots.txt file **MUST** be fetched, parsed, and its rules followed in the context of the initial authority.

If there are more than five consecutive redirects, crawlers **MAY** assume that the robots.txt file is unavailable.

2.3.1.3. "Unavailable" Status

"Unavailable" means the crawler tries to fetch the robots.txt file and the server responds with status codes indicating that the resource in question is unavailable. For example, in the context of HTTP, such status codes are in the 400-499 range.

If a server status code indicates that the robots.txt file is unavailable to the crawler, then the crawler **MAY** access any resources on the server.

2.3.1.4. "Unreachable" Status

If the robots.txt file is unreachable due to server or network errors, this means the robots.txt file is undefined and the crawler **MUST** assume complete disallow. For example, in the context of HTTP, server errors are identified by status codes in the 500-599 range.

If the robots.txt file is undefined for a reasonably long period of time (for example, 30 days), crawlers **MAY** assume that the robots.txt file is unavailable as defined in Section 2.3.1.3 or continue to use a cached copy.

2.3.1.5. Parsing Errors

Crawlers **MUST** try to parse each line of the robots.txt file. Crawlers **MUST** use the parseable rules.

2.4. Caching

Crawlers **MAY** cache the fetched robots.txt file's contents. Crawlers **MAY** use standard cache control as defined in [RFC9111]. Crawlers **SHOULD NOT** use the cached version for more than 24 hours, unless the robots.txt file is unreachable.

2.5. Limits

Crawlers **SHOULD** impose a parsing limit to protect their systems; see Section 3. The parsing limit **MUST** be at least 500 kibibytes [KiB].

3. Security Considerations

The Robots Exclusion Protocol is not a substitute for valid content security measures. Listing paths in the robots.txt file exposes them publicly and thus makes the paths discoverable. To control access to the URI paths in a robots.txt file, users of the protocol should employ a valid security measure relevant to the application layer on which the robots.txt file is served -- for example, in the case of HTTP, HTTP Authentication as defined in [RFC9110].

To protect against attacks against their system, implementors of robots.txt parsing and matching logic should take the following considerations into account:

Memory management: Section 2.5 defines the lower limit of bytes that must be processed, which inherently also protects the parser from out-of-memory scenarios.

Invalid characters: Section 2.2 defines a set of characters that parsers and matchers can expect in robots.txt files. Out-of-bound characters should be rejected as invalid, which limits the available attack vectors that attempt to compromise the system.

Untrusted content: Implementors should treat the content of a robots.txt file as untrusted content, as defined by the specification of the application layer used. For example, in the context of HTTP, implementors should follow the Security Considerations section of [RFC9110].

4. IANA Considerations

This document has no IANA actions.

5. Examples

5.1. Simple Example

The following example shows:

: A group that's relevant to all user agents that don't have an explicitly defined matching group. It allows access to the URLs with the /publications/ path prefix, and it restricts access to the URLs with the /example/ path prefix and to all URLs with a .gif suffix. The "" character designates any character, including the otherwise-required forward slash; see Section 2.2.

foobot: A regular case. A single user agent followed by rules. The crawler only has access to two URL path prefixes on the site -- /example/page.html and /example/allowed.gif. The rules of the group are missing the optional space character, which is acceptable as defined in Section 2.2.

barbot and bazbot: A group that's relevant for more than one user agent. The crawlers are not allowed to access the URLs with the /example/page.html path prefix but otherwise have unrestricted access to the rest of the URLs on the site.

quxbot: An empty group at the end of the file. The crawler has unrestricted access to the URLs on the site.

```
User-Agent: *
Disallow: *.gif$
Disallow: /example/
Allow: /publications/

User-Agent: foobot
Disallow:/
Allow:/example/page.html
Allow:/example/allowed.gif

User-Agent: barbot
User-Agent: bazbot
Disallow: /example/page.html

User-Agent: quxbot

EOF
```

5.2. Longest Match

The following example shows that in the case of two rules, the longest one is used for matching. In the following case, /example/page/disallowed.gif **MUST** be used for the URI example.com/example/page/disallow.gif.

```
User-Agent: foobot
Allow: /example/page/
Disallow: /example/page/disallowed.gif
```

6. References

6.1. Normative References

[RFC2046]

Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[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/info/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/info/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[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>.

6.2. Informative References

[KiB] "Kibibyte", Simple English Wikipedia, the free encyclopedia, 17 September 2020, <https://simple.wikipedia.org/wiki/Kibibyte>.

[ROBOTSTXT] "The Web Robots Pages (including /robots.txt)", 2007, <https://www.robotstxt.org/>.

[SITEMAPS] "What are Sitemaps? (Sitemap protocol)", April 2020, <https://www.sitemaps.org/index.html>.

Authors' Addresses

Martijn Koster
Stalworthy Manor Farm
Suton Lane
Wymondham, Norfolk
NR18 9JG
United Kingdom
Email: m.koster@greenhills.co.uk

Gary Illyes
Google LLC
Brandschenkestrasse 110
CH-8002 Zürich
Switzerland
Email: garyillyes@google.com

Henner Zeller
Google LLC
1600 Amphitheatre Pkwy
Mountain View, CA 94043
United States of America
Email: henner@google.com

Lizzi Sassman
Google LLC
Brandschenkestrasse 110
CH-8002 Zürich
Switzerland
Email: lizzi@google.com