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

Copyright (c) 2022 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
   1.1. Requirements Language

2. Specification
   2.1. Protocol Definition
   2.2. Formal Syntax
      2.2.1. The User-Agent Line
      2.2.2. The "Allow" and "Disallow" Lines
      2.2.3. Special Characters
      2.2.4. Other Records
   2.3. Access Method
      2.3.1. Access Results
         2.3.1.1. Successful Access
         2.3.1.2. Redirects
         2.3.1.3. "Unavailable" Status
         2.3.1.4. "Unreachable" Status
         2.3.1.5. Parsing Errors
   2.4. Caching
   2.5. Limits

3. Security Considerations

4. IANA Considerations

5. Examples
   5.1. Simple Example
   5.2. Longest Match

6. References
   6.1. Normative References
   6.2. Informative References

Authors' Addresses
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].
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:
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.

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

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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Encoded Path</th>
<th>Path to Match</th>
</tr>
</thead>
</table>

**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:

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

*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:

<table>
<thead>
<tr>
<th>Percent-encoded Pattern</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>/path/foo-%24</td>
<td><a href="https://www.example.com/path/foo-$">https://www.example.com/path/foo-$</a></td>
</tr>
</tbody>
</table>

*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:
6. References

6.1. Normative References

[RFC2046]


6.2. Informative References


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