Network Working Group   M. Koster
Internet-Draft   Stalworthy Computing, Ltd.
Intended status: Draft Standard   G. Illyes
Expires: January 9, 2020   H. Zeller
                                                         L. Harvey
                                                           Google
                                                     July 07, 2019

                      Robots Exclusion Protocol
                         draft-koster-rep-00

Abstract

   This document standardizes and extends the "Robots Exclusion
   Protocol" <http://www.robotstxt.org/> method originally defined by
   Martijn Koster in 1996 for service owners to control how content
   served by their services may be accessed, if at all, by automatic
   clients known as crawlers.

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 http://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 document may not be modified, and derivative works of it may not
   be created, except to format it for publication as an RFC or to
   translate it into languages other than English.

   This Internet-Draft will expire on January 9, 2020.

include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

This document applies to services that provide resources that clients
can access through URIs as defined in RFC3986 [1]. 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 [2].

It may be inconvenient for service owners if crawlers visit the
entirety of their URI space. This document specifies the rules that
crawlers MUST obey when accessing URIs.

These rules are not a form of access authorization.

1.1. 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.

2. Specification

2.1. Protocol definition

The protocol language consists of rule(s) and group(s):

o *Rule*: A line with a key-value pair that defines how a crawler
   may access URIs. See section The Allow and Disallow lines.

o *Group*: One or more user-agent lines that is followed by one or
   more rules. The group is terminated by a user-agent line or end
   of file. See User-agent line. 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 [3].

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

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

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

; parser implementors: add additional lines you need (for
; example Sitemaps), and be lenient when reading lines that don't
; conform. Apply Postel's law.

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 RFC3629, 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 2( UTF8-tail ) /
         %xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-tail )
UTF8-4 = %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) /
         %xF4 %x80-8F 2( UTF8-tail )

UTF8-tail = %x80-BF
```

2.2.1. The user-agent line

Crawlers set a product token to find relevant groups. The product
token MUST contain only "a-zA-Z_-" characters. The product token
SHOULD be part of the identification string that the crawler sends
to the service (for example, in the case of HTTP, the product name
SHOULD be in the user-agent header). The identification string
SHOULD describe the purpose of the crawler. Here's an example of an
HTTP header with a link pointing to a page describing the purpose of
the ExampleBot crawler which appears both in the HTTP header and as a
product token:

```
+-----------------------------------------------+----------------+
| HTTP header                                    | robots.txt     |
|                                                | user-agent line |
+-----------------------------------------------+----------------+
| user-agent: Mozilla/5.0 (compatible;           | user-agent:    |
| ExampleBot/0.1;                                | ExampleBot     |
| https://www.example.com/bot.html)              |                |
+-----------------------------------------------+----------------+
```

Crawlers MUST find the group that matches the product token exactly,
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. The matching MUST be case-insensitive. If
no matching group exists, crawlers MUST obey the first group with a
user-agent line with a "*" value, if present. If no group satisfies
either condition, 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 robot MUST match the
paths in allow and disallow rules against the URI. The matching
SHOULD be case sensitive. The most specific match found MUST be
used. The most specific match is the match that has the most octets.

If an allow and disallow rule is equivalent, the allow 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 US-
ASCII coded character set, and those in the reserved range defined by
RFC3986 [1], MUST be percent-encoded as defined by RFC3986 [1] prior
to comparison.

If a percent-encoded US-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 [1] 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=http | /foo/bar?baz=http%3A% | /foo/bar?baz=http%3A% |
| ://foo.bar        | 2F%2Ffoo.bar         | 2F%2Ffoo.bar         |
|                   |                      |                      |
| /foo/bar/U+E38384 | /foo/bar/%E3%83%84   | /foo/bar/%E3%83%84   |
|                   |                      |                      |
| /foo/bar/%E3%83%8 | /foo/bar/%E3%83%84   | /foo/bar/%E3%83%84   |
| 4                 |                      |                      |
|                   |                      |                      |
| /foo/bar/%62%61%7 | /foo/bar/%62%61%7A   | /foo/bar/baz         |
| A                 |                      |                      |
+-------------------+----------------------+----------------------+
```

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).

Implementers MAY bridge encoding mismatches if they detect that the
robots.txt file is not UTF8 encoded.

2.2.3. Special characters

Crawlers SHOULD allow the following special characters:

```
+-----------+------------------------------+----------------------+
| Character | Description                  | Example              |
+-----------+------------------------------+----------------------+
| "#"       | Designates an end of line    | "allow: / # comment  |
|           | comment.                     | in line"             |
|           |                              |                      |
|           |                              | "# comment at the    |
|           |                              | end"                 |
|           |                              |                      |
| "$"       | Designates the end of the    | "allow:              |
|           | match pattern. A URI MUST end| /this/path/exactly$" |
|           | with a $.                    |                      |
|           |                              |                      |
| "*"       | Designates 0 or more instances | "allow:            |
|           | of any character.            | /this/*/exactly"     |
+-----------+------------------------------+----------------------+
```

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

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

2.2.4. Other records

Clients MAY interpret other records that are not part of the
robots.txt protocol. For example, 'sitemap' [4].

2.3. Access method

The rules MUST be accessible in a file named "/robots.txt" (all lower
case) in the top level path of the service. The file MUST be UTF-8
encoded (as defined in RFC3629 [5]) and Internet Media Type "text/
plain" (as defined in RFC2046 [6]).

As per RFC3986 [1], the URI of the robots.txt is:

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

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

    http://www.example.com/robots.txt

         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, the crawler
   MUST follow the parseable rules.

2.3.1.2. Redirects

   The server may respond to a robots.txt fetch request with a redirect,
   such as HTTP 301 and HTTP 302. The crawlers SHOULD follow at least
   five consecutive redirects, even across authorities (for example
   hosts in case of HTTP), as defined in RFC1945 [7].

   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 is unavailable.

2.3.1.3. Unavailable status

   Unavailable means the crawler tries to fetch the robots.txt, and the
   server responds with unavailable status codes. For example, in the
   context of HTTP, unavailable status codes are in the 400-499 range.

   If a server status code indicates that the robots.txt file is
   unavailable to the client, then crawlers MAY access any resources on
   the server or MAY use a cached version of a robots.txt file for up to
   24 hours.

2.3.1.4. Unreachable status

   If the robots.txt is unreachable due to server or network errors,
   this means the robots.txt is undefined and the crawler MUST assume
   complete disallow. For example, in the context of HTTP, an
   unreachable robots.txt has a response code in the 500-599 range. For
   other undefined status codes, the crawler MUST assume the robots.txt
   is unreachable.

   If the robots.txt is undefined for a reasonably long period of time
   (for example, 30 days), clients MAY assume the robots.txt is
   unavailable or continue to use a cached copy.

2.3.1.5. Parsing errors

   Crawlers SHOULD 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 RFC2616 [8]. Crawlers
   SHOULD NOT use the cached version for more than 24 hours, unless the
   robots.txt is unreachable.

2.5. Limits

   Crawlers MAY impose a parsing limit that MUST be at least 500
   kibibytes (KiB).

2.6. Security Considerations

   The Robots Exclusion Protocol MUST NOT be used as a form of security
   measures. Listing URIs in the robots.txt file exposes the URI
   publicly and thus making the URIs discoverable.

2.7. IANA Considerations.

   This document has no actions for IANA.

3. Examples

3.1. Simple example

   The following example shows:

   o *foobot*: A regular case. A single user-agent token followed by
      rules.
   o *barbot and bazbot*: A group that's relevant for more than one
      user-agent.
   o *quxbot:* Empty group at end of file.

```
<CODE BEGINS>
User-Agent : foobot
Disallow : /example/page.html
Disallow : /example/disallowed.gif

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

User-Agent: quxbot

EOF
<CODE ENDS>
```

3.2. Longest Match

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

```
<CODE BEGINS>
User-Agent : foobot
Allow : /example/page/
Disallow : /example/page/disallowed.gif
<CODE ENDS>
```

4. References

4.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC8174]   Leiba, B., "Ambiguity of Uppercase vs Lowercase in
            RFC 2119 Key Words", BCP 14, RFC 2119, May 2017.

4.2. URIs

[1] https://tools.ietf.org/html/rfc3986

[2] https://tools.ietf.org/html/rfc8288

[3] https://tools.ietf.org/html/rfc5234

[4] https://www.sitemaps.org/index.html

[5] https://tools.ietf.org/html/rfc3629

[6] https://tools.ietf.org/html/rfc2046

   [7] https://tools.ietf.org/html/rfc1945

   [8] https://tools.ietf.org/html/rfc2616

Authors' Address

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

   Gary Illyes
   Brandschenkestrasse 110
   8002, Zurich
   Switzerland
   Email: garyillyes@google.com

   Henner Zeller
   1600 Amphitheatre Pkwy
   Mountain View, CA 94043
   USA
   Email: henner@google.com

   Lizzi Harvey
   1600 Amphitheatre Pkwy
   Mountain View, CA 94043
   USA
   Email: lizzi@google.com