Robots.txt Specifications

Abstract

This document details how Google handles the robots.txt file that allows you to control how Google's website crawlers crawl and index publicly accessible websites.

What changed

On July 1, 2019, Google announced (https://webmasters.googleblog.com/2019/07/rep-id.html) that the robots.txt protocol is working towards becoming an Internet standard (https://tools.ietf.org/html/draft-koster-rep-00). Those changes are reflected in this document.

List of changes

Here's what changed:

- Removed the "Requirements Language" section in this document because the language is Internet draft specific.
- Robots.txt now accepts all URI-based (https://en.wikipedia.org/wiki/Uniform_Resource_Identifier) protocols.
- Google follows at least five redirect hops. Since there were no rules fetched yet, the redirects are followed for at least five hops and if no robots.txt is found, Google treats it as a 404 for the robots.txt. Handling of logical redirects for the robots.txt file based on HTML content that returns 2xx (frames, JavaScript, or meta refresh-type redirects) is discouraged and the content of the first page is used for finding applicable rules.
- For 5xx, if the robots.txt is unreachable for more than 30 days, the last cached copy of the robots.txt is used, or if unavailable, Google assumes that there are no crawl restrictions.
- Google treats unsuccessful requests or incomplete data as a server error.
- "Records" are now called "lines" or "rules", as appropriate.
- Google doesn't support the handling of `<field>` elements with simple errors or typos (for example, "useragent" instead of "user-agent").
- Google currently enforces a size limit of 500 kibibytes (https://en.wikipedia.org/wiki/Kibibyte) (KiB), and ignores content after that limit.

- Updated formal syntax to be valid Augmented Backus-Naur Form (ABNF) per RFC5234 (https://tools.ietf.org/html/rfc5234) and to cover for UTF-8 characters in the robots.txt.
- Updated the definition of "groups" to make it shorter and more to the point. Added an example for an empty group.
- Removed references to the deprecated Ajax Crawling Scheme.

Basic definitions

Definitions	
Crawler	A crawler is a service or agent that crawls websites. Generally speaking, a crawler automatically and recursively accesses known URLs of a host that exposes content which can be accessed with standard web-browsers. As new URLs are found (through various means, such as from links on existing, crawled pages or from Sitemap files), these are also crawled in the same way.
User-agent	A means of identifying a specific crawler or set of crawlers.
Directives	The list of applicable guidelines for a crawler or group of crawlers set forth in the robots.txt file.
URL	Uniform Resource Locators as defined in RFC 1738 (http://www.ietf.org/rfc/rfc1738.txt).
Google-specific	These elements are specific to Google's implementation of robots.txt and may not be relevant for other parties.

Applicability

The guidelines set forth in this document are followed by all automated crawlers at Google. When an agent accesses URLs on behalf of a user (for example, for translation, manually subscribed feeds, malware analysis), these guidelines do not need to apply.

File location and range of validity

The robots.txt file must be in the top-level directory of the host, accessible through the appropriate protocol and port number. Generally accepted protocols for robots.txt are all URI-based

(https://en.wikipedia.org/wiki/Uniform_Resource_Identifier), and for Google Search specifically (for example, crawling of websites) are "http" and "https". On http and https, the robots.txt file is fetched using a HTTP non-conditional GET request.

Google-specific: Google also accepts and follows robots.txt files for FTP sites. FTP-based robots.txt files are accessed via the FTP protocol, using an anonymous login.

The directives listed in the robots.txt file apply only to the host, protocol and port number where the file is hosted.

RL for the robots.txt file is - like other URLs - case-sensitive.

Examples of valid robots.txt URLs

Robots.txt URL examples	
`http://example.com/robots.txt`	**Valid for:** • `http://example.com/` • `http://example.com/folder/file` **Not valid for:** • `http://other.example.com/` • `https://example.com/` • `http://example.com:8181/` This is the general case. It is not valid for other subdomains, protocols or port numbers. It is valid for all files in all subdirectories on the same host, protocol and port number.

Robots.txt URL examples	
`http://www.example.com/robots.txt`	👍 **Valid for:** `http://www.example.com/` 👎 **Not valid for:** • `http://example.com/` • `http://shop.www.example.com/` • `http://www.shop.example.com/` 💡 A robots.txt on a subdomain is only valid for that subdomain.
`http://example.com/folder/robots.txt`	Not a valid robots.txt file. Crawlers don't check for robots.txt files in subdirectories.
`http://www.müller.eu/robots.txt`	👍 **Valid for:** • `http://www.müller.eu/` • `http://www.xn--mller-kva.eu/` 👎 **Not valid for:** `http://www.muller.eu/` 💡 IDNs are equivalent to their punycode versions. See also RFC 3492 (http://www.ietf.org/rfc/rfc3492.txt).
`ftp://example.com/robots.txt`	👍 **Valid for:** `ftp://example.com/` 👎 **Not valid for:** `http://example.com/` Google-specific: We use the robots.txt for FTP resources.
`http://212.96.82.21/robots.txt`	👍 **Valid for:** `http://212.96.82.21/` 👎 **Not valid for:** `http://example.com/` (even if hosted on 212.96.82.21) 💡 A robots.txt with an IP-address as the host name is only valid for crawling of that IP-address as host name. It isn't automatically valid for all websites hosted on that IP-address (though it is possible that the robots.txt file is shared, in which case it would also be available under the shared host name).

Robots.txt URL examples	
`http://example.com:80/robots.txt`	👍 **Valid for:** • `http://example.com:80/` • `http://example.com/` 👎 **Not valid for:** `http://example.com:81/` 💡 Standard port numbers (80 for http, 443 for https, 21 for ftp) are equivalent to their default host names. See also **`[portnumbers]`**.
`http://example.com:8181/robots.txt`	👍 **Valid for:** `http://example.com:8181/` 👎 **Not valid for:** `http://example.com/` 💡 Robots.txt files on non-standard port numbers are only valid for content made available through those port numbers.

Handling HTTP result codes

There are generally three different outcomes when robots.txt files are fetched:

- full allow: All content may be crawled.
- full disallow: No content may be crawled.
- conditional allow: The directives in the robots.txt determine the ability to crawl certain content.

Handling HTTP result codes	
2xx (successful)	HTTP result codes that signal success result in a "conditional allow" of crawling.
3xx (redirection)	Google follows at least five redirect hops as defined by RFC 1945 (http://www.ietf.org/rfc/rfc1945.txt) for HTTP/1.0 and then stops and treats it as a 404. Handling of robots.txt redirects to disallowed URLs is discouraged; since there were no rules fetched yet, the redirects are followed for at least five hops and if no robots.txt is found, Google treats it as a 404 for the robots.txt. Handling of logical redirects for the robots.txt file based on HTML content that returns 2xx (frames, JavaScript, or meta refresh-type redirects) is discouraged and the content of the first page is used for finding applicable rules.

Handling HTTP result codes	
4xx (client errors)	All 4xx errors are treated the same way and it's assumed that no valid robots.txt file exists. It is assumed that there are no restrictions. This is a "full allow" for crawling. ★ This includes 401 "Unauthorized" and 403 "Forbidden" HTTP result codes.
5xx (server error)	Server errors are seen as temporary errors that result in a "full disallow" of crawling. The request is retried until a non-server-error HTTP result code is obtained. A 503 (Service Unavailable) error results in fairly frequent retrying. If the robots.txt is unreachable for more than 30 days, the last cached copy of the robots.txt is used. If unavailable, Google assumes that there are no crawl restrictions. To temporarily suspend crawling, it is recommended to serve a 503 HTTP result code. Google-specific: If we are able to determine that a site is incorrectly configured to return 5xx instead of a 404 for missing pages, we treat a 5xx error from that site as a 404.
Unsuccessful requests or incomplete data	Handling of a robots.txt file which cannot be fetched due to DNS or networking issues, such as timeouts, invalid responses, reset or hung up connections, and HTTP chunking errors, is treated as a server error (#server-error).
Caching	robots.txt content is generally cached for up to 24 hours, but may be cached longer in situations where refreshing the cached version is not possible (for example, due to timeouts or 5xx errors). The cached response may be shared by different crawlers. Google may increase or decrease the cache lifetime based on max-age Cache-Control (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9.3) HTTP headers.

File format

The expected file format is plain text encoded in UTF-8 (http://en.wikipedia.org/wiki/UTF-8). The file consists of lines separated by CR, CR/LF, or LF.

Only valid lines are considered; all other content is ignored. For example, if the resulting document is an HTML page, only valid text lines are taken into account, the rest are discarded without warning or error.

If a character encoding is used that results in characters being used which are not a subset of UTF-8, this may result in the contents of the file being parsed incorrectly.

An optional Unicode BOM (http://en.wikipedia.org/wiki/Byte_order_mark) (byte order mark) at the beginning of the robots.txt file is ignored.

Each valid line consists of a field, a colon, and a value. Spaces are optional (but recommended to improve readability). Comments can be included at any location in the file using the "#" character; all content after the start of a comment until the end of the line is treated as a comment and ignored. The general format is **<field>:<value><#optional-comment>**. Whitespace at the beginning and at the end of the line is ignored.

The **<field>** element is case-insensitive. The <value> element may be case-sensitive, depending on the <field> element.

Handling of **<field>** elements with simple errors or typos (for example, "useragent" instead of "user-agent") is not supported.

A maximum file size may be enforced per crawler. Content which is after the maximum file size is ignored. Google currently enforces a size limit of 500 kibibytes (https://en.wikipedia.org/wiki/Kibibyte) (KiB). To reduce the size of the robots.txt file, consolidate directives that would result in an oversized robots.txt file. For example, place excluded material in a separate directory.

Formal syntax / definition

Here is an Augmented Backus-Naur Form (ABNF) description, as described in RFC 5234 (https://www.ietf.org/rfc/rfc5234.txt)

```
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; see 3.2.2
empty-pattern = *WS

identifier = 1*(%x2d / %x41-5a / %x5f / %x61-7a)
```

```
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
```

Grouping of lines and rules

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. The last group may have no rules, which means it implicitly allows everything.

Example groups:

```
user-agent: a
disallow: /c

user-agent: b
disallow: /d

user-agent: e
user-agent: f
disallow: /g

user-agent: h
```

There are four distinct groups specified, one for "a" and one for "b" as well as one for both "e" and "f". Except for the last group, each group has its own group-member line. The last group is empty. Note the optional use of white-space and empty lines to improve readability.

Order of precedence for user-agents

Only one group is valid for a particular crawler. The crawler must determine the correct group of lines by finding the group with the most specific user-agent that still matches. All other groups are ignored by the crawler. The user-agent is case-sensitive. All non-matching text is ignored (for example, both **`googlebot/1.2`** and **`googlebot*`** are equivalent to **`googlebot`**). The order of the groups within the robots.txt file is irrelevant.

If there's more than one group for a specific user-agent, all the rules from the groups applicable to a specific user-agent are combined.

Example

Assuming the following robots.txt file:

```
user-agent: googlebot-news
(group 1)

user-agent: *
(group 2)

user-agent: googlebot
(group 3)
```

This is how the crawlers would choose the relevant group:

Group followed per crawler	
Googlebot News	The group followed is group 1. Only the most specific group is followed, all others are ignored.
Googlebot (web)	The group followed is group 3.
Googlebot Images	The group followed is group 3. There is no specific **`googlebot-images`** group, so the more generic group is followed.
Googlebot News (when crawling images)	>The group followed is group 1. These images are crawled for and by Googlebot News, therefore only the Googlebot News group is followed.
Otherbot (web)	The group followed is group 2.
Otherbot (News)	The group followed is group 2. Even if there is an entry for a related crawler, it is only valid if it is specifically matching.

Also see Google's crawlers and user-agent strings (https://support.google.com/webmasters/answer/1061943).

Group-member rules

Only standard group-member rules are covered in this section. These rules are also called "directives" for the crawlers. These directives are specified in the form of **`directive: [path]`** where **`[path]`** is optional. By default, there are no restrictions for crawling for the designated crawlers. Directives without a **`[path]`** are ignored.

The **`[path]`** value, if specified, is to be seen relative from the root of the website for which the robots.txt file was fetched (using the same protocol, port number, host and domain names). The path value must start with "/" to designate the root. The path is case-sensitive. More information can be found in the section "URL matching based on path values" below.

disallow

The **`disallow`** directive specifies paths that must not be accessed by the designated crawlers. When no path is specified, the directive is ignored.

Usage:

```
disallow: [path]
```

allow

The **`allow`** directive specifies paths that may be accessed by the designated crawlers. When no path is specified, the directive is ignored.

Usage:

```
allow: [path]
```

URL matching based on path values

The path value is used as a basis to determine whether or not a rule applies to a specific URL on a site. With the exception of wildcards, the path is used to match the beginning of a URL (and any valid URLs that start with the same path). Non-7-bit ASCII characters in a path may be included as UTF-8 characters or as percent-escaped UTF-8 encoded characters per RFC 3986 (http://www.ietf.org/rfc/rfc3986.txt).

Google, Bing, and other major search engines support a limited form of "wildcards" for path values. These are:

- `*` designates 0 or more instances of any valid character.
- `$` designates the end of the URL.

Example path matches

`/`	Matches the root and any lower level URL
`/*`	Equivalent to `/`. The trailing wildcard is ignored.
`/fish`	👍 **Matches:** • `/fish` • `/fish.html` • `/fish/salmon.html` • `/fishheads` • `/fishheads/yummy.html` • `/fish.php?id=anything` 👎 **Does not match:** • `/Fish.asp` • `/catfish` • `/?id=fish` ★ Note the case-sensitive matching.

Example path matches	
`/fish*`	Equivalent to `/fish`. The trailing wildcard is ignored. 👍 **Matches:** • `/fish` • `/fish.html` • `/fish/salmon.html` • `/fishheads` • `/fishheads/yummy.html` • `/fish.php?id=anything` 👎 **Does not match:** • `/Fish.asp` • `/catfish` • `/?id=fish`
`/fish/`	The trailing slash means this matches anything in this folder. 👍 **Matches:** • `/fish/` • `/fish/?id=anything` • `/fish/salmon.htm` 👎 **Does not match:** • `/fish` • `/fish.html` • `/Fish/Salmon.asp`

Example path matches	
`/*.php`	👍 **Matches:** • `/filename.php` • `/folder/filename.php` • `/folder/filename.php?parameters` • `/folder/any.php.file.html` • `/filename.php/` 👎 **Does not match:** • `/` (even if it maps to /index.php) • `/windows.PHP`
`/*.php$`	👍 **Matches:** • `/filename.php` • `/folder/filename.php` 👎 **Does not match:** • `/filename.php?parameters` • `/filename.php/` • `/filename.php5` • `/windows.PHP`
`/fish*.php`	👍 **Matches:** • `/fish.php` • `/fishheads/catfish.php?parameters` 👎 **Does not match:** `/Fish.PHP`

Google-supported non-group-member lines

Google, Bing,and other major search engines support `sitemap`, as defined by sitemaps.org (http://sitemaps.org).

Usage:

```
sitemap: [absoluteURL]
```

[absoluteURL] points to a Sitemap, Sitemap Index file, or equivalent URL. The URL does not have to be on the same host as the robots.txt file. Multiple **sitemap** entries may exist. As non-group-member lines, these are not tied to any specific user-agents and may be followed by all crawlers, provided it is not disallowed.

Order of precedence for group-member lines

At a group-member level, in particular for **allow** and **disallow** directives, the most specific rule based on the length of the **[path]** entry trumps the less specific (shorter) rule. In case of conflicting rules, including those with wildcards, the least restrictive rule is used.

Sample situations	

`http://example.com/page`	`allow`: `/p`
	`disallow`: `/`
	Verdict: `allow`

`http://example.com/folder/page`	`allow`: `/folder`
	`disallow`: `/folder`
	Verdict: `allow`

`http://example.com/page.htm`	`allow`: `/page`
	`disallow`: `/*.htm`
	Verdict: `undefined`

`http://example.com/`	`allow`: `/$`
	`disallow`: `/`
	Verdict: `allow`

`http://example.com/page.htm`	`allow`: `/$`
	`disallow`: `/`
	Verdict: `disallow`

Last updated 2020-01-29.