.htaccess Cheat Sheet

All the important Apache .htaccess web server rules and config options

Welcome to our fast loading one page .htaccess cheat sheet with all major .htaccess rules listed.

We have no ads, no javascript. Just plain HTML (and a .css file), so it should load super fast. Coming here and a quick cmd+f/ctrl+f should be faster than finding the answer on stackexchange :)

> Remember that for most rules you must have the `RewriteEngine on` rule in your .htaccess file!!!

Rewrite and Redirection

- Serve All Requests With One PHP File
- WordPress .htaccess for permalinks
- Force www
- Force www in a Generic Way
- Force non-www
- Force non-www in a Generic Way
- Force HTTPS
- Force HTTPS Behind a Proxy
- Force Trailing Slash
- Remove Trailing Slash
- Redirect a Single Page
- Alias a Single Directory
- Alias Paths To Script
- Redirect an Entire Site
- Alias "Clean" URLs

Security

- Deny All Access
- Deny All Access Except Yours (Only allow certain IPs)
- Block IP Address
- Allow access only from LAN
- Deny Access To Certain User Agents (bots)
- Deny Access to Hidden Files and Directories
- Deny Access To Certain Files
- Deny Access to Backup and Source Files
- Disable Directory Browsing
- Enable Directory Listings
- Disable Listing Of Certain Filetypes (if Indexes is not disabled)
- Disable Image Hotlinking
- Redirect hotlinkers and show a different image
- Deny Access from certain referrers
- Password Protect a Directory

Performance

Compress Text Files (gzip/deflate output)

Set Expires Headers

Turn eTags Off

Limit Upload File Size

Miscellaneous

Server Variables for mod_re足write

Set PHP Variables

Custom Error Pages

Redirect users to a maintenance page while you update

Force Downloading

Disable Showing Server Info (Server Signature)

Prevent Downloading

Allow Cross-Domain Fonts

Auto UTF-8 Encode

Set Server Timezone (to UTC, or other time zone)

Switch to Another PHP Version

Locating your .htaccess file on diferent hosting platforms

.htaccess is an Apache file that only appears on Apache server. For those who are using A2 Hosting, SiteGround, and InMotion Hosting - these hosts run on Apache, the .htaccess file should be located at your domain's root folder. Stop searching if your host is running on a different web server software, for instances - Microsoft IIS and NGINX. Please refer to this web hosting list to check the type of server and control panel offerd by each company.

Also, please remember to double check and verify any rules that you use. We accept no responsibility for your use of these rules - use them at your own risk. Please get in touch if you want us to add a rule!

Rewrite and Redirection Rules

(Note: It is assumed that you have `mod_rewrite` installed and enabled. The first line should be 'RewriteEngine on' to enable this)

Serve All Requests With One PHP File with .htaccess

perm link

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^([^?]*)$ /index.php [NC,L,QSA]
```

WordPress .htaccess for permalinks with .htaccess perm link

(This is the only rule in this section that includes the RewriteEngine on rule)

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
 RewriteEngine On
 RewriteBase /
 RewriteCond %{REQUEST_FILENAME} !-f
 RewriteCond %{REQUEST_FILENAME} !-d
 RewriteRule . /index.php [L]
</IfModule>
# END WordPress
```

Force www with .htaccess perm link

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^example\.com [NC]
RewriteRule ^(.*)$ https://www.example.com/$1 [L,R=30
```

Force www in a Generic Way with .htaccess perm link

```
RewriteCond %{HTTP_HOST} !^$
RewriteCond %{HTTP_HOST} !^www\. [NC]
RewriteCond %{HTTPS}s ^on(s)|
RewriteRule ^ http%1://www.%{HTTP_HOST}%{REQUEST_URI}
```

This works for any domain. Source

Force non-www with .htaccess perm link

It's still open for debate whether www or non-www is the master race, so if you happen to be a fan or bare domains, here you go:

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^www\.example\.com [NC]
RewriteRule ^(.*)$ https://example.com/$1 [L,R=301]
```

Force non-www in a Generic Way with .htaccess perm link

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^www\.
RewriteCond %{HTTPS}s ^on(s)|off
RewriteCond http%1://%{HTTP_HOST} ^(https?://)(www\.)
RewriteRule ^ %1%3%{REQUEST_URI} [R=301,L]
```

Force HTTPS with .htaccess

perm link

Use this to redirect non HTTPS requests to a HTTPS request. I.e. if you go to https://example.com/ it will redirect to https://example.com.

```
RewriteEngine on
RewriteCond %{HTTPS} !on
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

It is recommended to use HSTS (read about it on Wikipedia) though.

"HTTP Strict Transport Security (HSTS) is a web security policy mechanism which is necessary to protect secure HTTPS websites against downgrade attacks, and which greatly simplifies protection against cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol. HSTS is an IETF standards track protocol and is specified in RFC 6797."

Force HTTPS Behind a Proxy with .htaccess

perm link

Useful if you have a proxy in front of your server performing TLS termination.

```
RewriteCond %{HTTP:X-Forwarded-Proto} !https
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

Force Trailing Slash with .htaccess

perm link

Use the follow .htaccess rule to redirect any urls to the same url (but with a trailing slash) for any requests that do not end with a trailing slash. I.e. redirect from https://example.com/your-page to https://example.com/your-page/

```
RewriteCond %{REQUEST_URI} /+[^\.]+$
RewriteRule ^(.+[^/])$ %{REQUEST_URI}/ [R=301,L]
```

Remove Trailing Slash with .htaccess

perm link

Use this to remove any trailing slash (it will 301 redirect to the non trailing slash url)

```
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)/$ /$1 [R=301,L]
```

Redirect a Single Page with .htaccess

perm link

Redirect a single URL to a new location

```
Redirect 301 /oldpage.html https://www.yoursite.com/n
Redirect 301 /oldpage2.html https://www.yoursite.com/
```

Source

Alias a Single Directory with .htaccess

perm link

```
RewriteEngine On
RewriteRule ^source-directory/(.*) target-directory/$
```

Alias Paths To Script with .htaccess

perm link

```
RewriteEngine On
RewriteRule ^$ index.fcgi/ [QSA,L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.fcgi/$1 [QSA,L]
```

This example has an `index.fcgi` file in some directory, and any requests within that directory that fail to resolve a filename/directory will be sent to the `index.fcgi` script. It's good if you want `baz.foo/some/cool/path` to be handled by `baz.foo/index.fcgi` (which also supports requests to `baz.foo`) while maintaining `baz.foo/css/style.css` and the like.

Redirect an Entire Site with .htaccess

perm link

Use the following .htaccess rule to redirect an entire site to a new location/domain

```
Redirect 301 / https://newsite.com/
```

This way does it with links intact. That is `www.oldsite.com/some/crazy/link.html` will become `www.newsite.com/some/crazy/link.html`. This is extremely helpful when you are just "moving" a site to a new domain.

Source

Alias "Clean" URLs with .htaccess

perm link

This snippet lets you use "clean URLs" -- those without a PHP extension, e.g. `example.com/users` instead of `example.com/users.php`.

```
RewriteEngine On
RewriteCond %{SCRIPT_FILENAME} !-d
RewriteRule ^([^.]+)$ $1.php [NC,L]
```

Source

Security Rules

Deny All Access with .htaccess

perm link

If you want to prevent apache serving any files at all, use the following.

Apache 2.2:

```
Deny from all
```

Apache 2.2:

```
# Require all denied
```

This will stop you from accessing your website. If you want to deny all access but still be able to view it yourself please read the next rule:

Deny All Access Except Yours (Only allow certain IPs) with .htaccess

perm link

Use this to ONLY allow certain IP addresses to access your website.

Apache 2.2

```
Order deny,allow
Deny from all
Allow from xxx.xxx.xxx.xxx
```

Apache 2.4

```
# Require all denied
# Require ip xxx.xxx.xxx.xxx
```

`xxx.xxx.xxx.xxx` is your IP. If you replace the last three digits with 0/12 for example, this will specify a range of IPs within the same network, thus saving you the trouble to list all allowed IPs separately. Source

Please see the next rule for the 'opposite' of this rule!

Block IP Address with .htaccess

perm link

This will allow access to all IPs EXCEPT the ones listed. You can use this to allow all access Except Spammer's IP addresses.

Replace xxx.xxx.xxx.xxx and xxx.xxx.xxx.xxy with the IP addresses you want to block.

Apache 2.2

```
Order deny,allow
Allow from all
Deny from xxx.xxx.xxx.xxx
Deny from xxx.xxx.xxx.xxy
```

Apache 2.4

```
# Require all granted
# Require not ip xxx.xxx.xxx.xxx
# Require not ip xxx.xxx.xxx.xxy
```

Allow access only from LAN with .htaccess

perm link

```
order deny,allow
deny from all
allow from 192.168.0.0/24
```

Deny Access To Certain User Agents (bots) with .htaccess

perm link

Use this .htaccess rule to block/ban certain user agents

```
RewriteCond %{HTTP_USER_AGENT} ^User\ Agent\ 1 [OR]
RewriteCond %{HTTP_USER_AGENT} ^Another\ Bot\ You\ Wa
RewriteCond %{HTTP_USER_AGENT} ^Another\ UA
RewriteRule ^.* - [F,L]
```

Deny Access to Hidden Files and Directories with .htaccess

perm link

Hidden files and directories (those whose names start with a dot `.`) should most, if not all, of the time be secured. For example: `.htaccess`, `.htpasswd`, `.git`, `.hg`...

```
RewriteCond %{SCRIPT_FILENAME} -d [OR]
RewriteCond %{SCRIPT_FILENAME} -f
RewriteRule "(^|/)\." - [F]
```

Alternatively, you can just raise a `Not Found` error, giving the attacker dude no clue:

```
RedirectMatch 404 /\..*$
```

Deny Access To Certain Files with .htaccess

perm link

Use this to block or deny access to certain files

```
<files your-file-name.txt>
order allow,deny
deny from all
</files>
```

Deny Access to Backup and Source Files with .htaccess

perm link

These files may be left by some text/html editors (like Vi/Vim) and pose a great security danger, when anyone can access them.

```
<FilesMatch "(\.(bak|config|dist|fla|inc|ini|log|psd|
    ## Apache 2.2
```

```
        Order allow,deny
        Deny from all
        Satisfy All

        ## Apache 2.4
        # Require all denied
    </FilesMatch>
```

Source

Disable Directory Browsing with .htaccess

perm link

```
Options All -Indexes
```

Enable Directory Listings with .htaccess

perm link

```
Options All +Indexes
```

Disable Listing Of Certain Filetypes (if Indexes is not disabled) with .htaccess

perm link

Use this to exclude certain file types from being listed in Apache directory listing. You could use this to stop .pdf files, or video files showing up.

```
IndexIgnore *.zip *.mp4 *.pdf
```

Disable Image Hotlinking with .htaccess

perm link

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http(s)?://(www\.)?your
RewriteRule \.(jpg|jpeg|png|gif)$ - [NC,F,L]
```

Redirect hotlinkers and show a different image with .htaccess

perm link

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^https://(www\.)?your-we
```

```
RewriteRule \.(gif|jpg|png)$ https://www.your-website
```

Deny Access from certain referrers with .htaccess

perm link

Use this rule to block access to requests that include a referrer from a certain domain.

```
RewriteCond %{HTTP_REFERER} block-this-referer\.com [
RewriteCond %{HTTP_REFERER} and-block-traffic-that-th
RewriteRule .* - [F]
```

Password Protect a Directory with .htaccess

perm link

First you need to create a `.htpasswd` file somewhere in the system. Run the following command at the command line:

```
htpasswd -c /home/hidden/directory/here/.htpasswd the
```

Then you can use it for authentication. In your .htaccess file you need something like the following code, but make sure the AuthUserFile is the file path to the .htpasswd you just created. You should keep the .htpasswd in a directory not accesible via the web. So don't put it in your /public_html/ or /www/ directory.

```
AuthType Basic
AuthName "Password Protected Dir Title"
AuthUserFile /home/hidden/directory/here/.htpasswd
Require valid-user
```

Password Protect a File or Several Files with .htaccess

perm link

```
AuthName "Password Protected Directory Title"
AuthType Basic
AuthUserFile /home/hidden/directory/here/.htpass

<Files "/a-private-file.txt">
Require valid-user
</Files>

<FilesMatch ^((one|two|three)-rings?\.o)$>
Require valid-user
</FilesMatch>
```

Performance Rules

Compress Text Files (gzip/deflate output) with .htaccess

perm link

```
<IfModule mod_deflate.c>

        # Force compression for mangled headers.
        # https://developer.yahoo.com/blogs/ydn/
        <IfModule mod_setenvif.c>
                <IfModule mod_headers.c>
                        SetEnvIfNoCase ^(Accept-
                        RequestHeader append Acc
                </IfModule>
        </IfModule>

        # Compress all output labeled with one o
        # (for Apache versions below 2.3.7, you
        #    and can remove the `<IfModule mod_f
        #    as `AddOutputFilterByType` is still
        <IfModule mod_filter.c>
            AddOutputFilterByType DEFLATE applic
              application/javascript \
              application/json \
              application/rss+xml \
              application/vnd.ms-fontobject \
              application/x-font-ttf \
              application/x-web-app-manifest+jso
              application/xhtml+xml \
              application/xml \
              font/opentype \
              image/svg+xml \
              image/x-icon \
              text/css \
              text/html \
              text/plain \
              text/x-component \
              text/xml
        </IfModule>

</IfModule>
```

Source

Set Expires Headers with .htaccess

perm link

Expires headers tell the browser whether they should request a specific file from the server or just grab it from the cache. It is advisable to set static content's expires headers to something far in the future.

If you don't control versioning with filename-based cache busting, consider lowering the cache time for resources like CSS and JS to something like 1 week. Source

```
<IfModule mod_expires.c>
        ExpiresActive on
        ExpiresDefault

    # CSS
        ExpiresByType text/css

    # Data interchange
        ExpiresByType application/json
        ExpiresByType application/xml
        ExpiresByType text/xml

    # Favicon (cannot be renamed!)
        ExpiresByType image/x-icon

    # HTML components (HTCs)
        ExpiresByType text/x-component

    # HTML
        ExpiresByType text/html

    # JavaScript
        ExpiresByType application/javascript

    # Manifest files
        ExpiresByType application/x-web-app-manifest+
        ExpiresByType text/cache-manifest

    # Media
        ExpiresByType audio/ogg
        ExpiresByType image/gif
        ExpiresByType image/jpeg
        ExpiresByType image/png
        ExpiresByType video/mp4
        ExpiresByType video/ogg
        ExpiresByType video/webm

    # Web feeds
        ExpiresByType application/atom+xml
        ExpiresByType application/rss+xml

    # Web fonts
        ExpiresByType application/font-woff2
        ExpiresByType application/font-woff
        ExpiresByType application/vnd.ms-fontobject
        ExpiresByType application/x-font-ttf
        ExpiresByType font/opentype
        ExpiresByType image/svg+xml
</IfModule>
```

Turn eTags Off with .htaccess

perm link

By removing the ETag header, you disable caches and browsers from being able to validate files, so they are forced to rely on your Cache-

Control and Expires header. Source

```
<IfModule mod headers.c>
        Header unset ETag
</IfModule>
FileETag None
```

Limit Upload File Size with .htaccess perm link

Put the file size in bytes. See here for a conversion tool. The code below limits it to 1mb.

```
LimitRequestBody 1048576
```

Miscellaneous Rules

Server Variables for mod_re足write with .htaccess perm link

```
%{API VERSION}
%{DOCUMENT ROOT}
%{HTTP ACCEPT}
%{HTTP COOKIE}
%{HTTP FORWARDED}
%{HTTP HOST}
%{HTTP PROXY CONNECTION}
%{HTTP REFERER}
%{HTTP USER_AGENT}
%{HTTPS}
%{IS SUBREQ}
%{REQUEST FILENAME}
%{REQUEST URI}
%{SERVER ADDR}
%{SERVER ADMIN}
%{SERVER NAME}
%{SERVER PORT}
%{SERVER PROTOCOL}
%{SERVER SOFTWARE}
%{THE_REQUEST}
```

Set PHP Variables with .htaccess perm link

```
php_value <key> <val>
```

For example:

```
php_value upload_max_filesize 50M
php_value max_execution_time 240
```

Custom Error Pages with .htaccess

perm link

```
ErrorDocument 500 "Houston, we have a problem."
ErrorDocument 401 https://error.yourdomain.com/mordor
ErrorDocument 404 /errors/halflife3.html
```

Redirect users to a maintenance page while you update with .htaccess

perm link

This will redirect users to a maintenance page but allow access to your IP address. Change 555.555.555.555 to your IP, and YourMaintenancePageFilenameOrFullUrlUrl.html to your error page (or a whole URL, on a different domain).

```
ErrorDocument 403 YourMaintenancePageFilenameOrFullUr
Order deny,allow
Deny from all
Allow from 555.555.555.555
```

Force Downloading with .htaccess

perm link

Sometimes you want to force the browser to download some content instead of displaying it. The following snippet will help.

```
<Files *.md>
        ForceType application/octet-stream
        Header set Content-Disposition attachment
</Files>
```

Disable Showing Server Info (Server Signature) with .htaccess

perm link

While many people consider this pointless (especially with regards to security), if you want to stop your server from giving away server info (the sever OS etc), use this:

```
ServerSignature Off
```

Prevent Downloading with .htaccess

perm link

Sometimes you want to force the browser to display some content instead of downloading it. The following snippet will help.

```
<FilesMatch "\.(tex|log|aux)$">
        Header set Content-Type text/plain
</FilesMatch>
```

Allow Cross-Domain Fonts with .htaccess

perm link

CDN-served webfonts might not work in Firefox or IE due to CORS. The following snippet from alrra should make it happen.

```
<IfModule mod_headers.c>
        <FilesMatch "\.(eot|otf|ttc|ttf|woff|woff2)$"
                Header set Access-Control-Allow-Origi
        </FilesMatch>
</IfModule>
```

Auto UTF-8 Encode with .htaccess

perm link

To have Apache automatically encode your content in UTF-8, use the following code. You can also swap the utf-8 for another character set if required:

```
# Use UTF-8 encoding for anything served text/plain o
AddDefaultCharset utf-8

# Force UTF-8 for a number of file formats
AddCharset utf-8 .atom .css .js .json .rss .vtt .xml
```

Source

Set Server Timezone (to UTC, or other time zone) with .htaccess

perm link

```
SetEnv TZ UTC
```

See a list of time zones here. To set it to Los Angeles time zone:

```
SetEnv TZ America/Los_Angeles
```

Switch to Another PHP Version with .htaccess

perm link

If you're on a shared host, chances are there are more than one version of PHP installed, and sometimes you want a specific version for your website. For example, Laravel requires PHP >= 5.4. The following snippet should switch the PHP version for you.

```
AddHandler application/x-httpd-php55 .php
```

Alternatively, you can use AddType

```
AddType application/x-httpd-php55 .php
```

Disable Internet Explorer Compatibility View

Compatibility View in IE may affect how some websites are displayed. The following snippet should force IE to use the Edge Rendering Engine and disable the Compatibility View.

```
<IfModule mod_headers.c>
    BrowserMatch MSIE is-msie
    Header set X-UA-Compatible IE=edge env=is-msie
</IfModule>
```

Execute PHP with a different file extension with .htaccess

perm link

The following code will run files ending in .ext with php:

```
AddType application/x-httpd-php .ext
```

Serve WebP Images Automatically If They Exist

If WebP images are supported and an image with a .webp extension and the same name is found at the same place as the jpg/png image that is going to be served, then the WebP image is served instead.

```
RewriteEngine On
RewriteCond %{HTTP_ACCEPT} image/webp
RewriteCond %{DOCUMENT_ROOT}/$1.webp -f
RewriteRule (.+)\.(jpe?g|png)$ $1.webp [T=image/webp,
```

The Basics of .htaccess

The .htaccess file is a system configuration file that's seen in many web servers, including the popular Apache server software used by most commercial hosting service providers. The .htaccess file is powerful and controls many aspects in a web server.

HTTP server Software is also called web server software and is not to be mistaken with the operating system. The operating system controls the server hardware, while the web server software manages the files that serve up the pages of your website.

Traditionally, Apache held the lion’s share of the web server software market but over the past ten years it has steadily lost ground to other brands, primarily Microsoft. Today, Apache is used on ~40% of all web servers, hosting approximately 350 million websites.

Additional Resources: Other .htaccess Cheatsheets From Around the Web

- Another htaccess cheatsheet
- Another cheatsheet - but in .pdf format
- Apache Rewrite Cheatsheet
- Mod Rewrite Cheatsheet
- Apache Docs for mod_rewrite

Credits

Content published on this site is based heavily (on its first version) from phanan/htaccess.

Snippets with specified source belong to their respective owners and have their own license(s), whenever appropriate.

Other content belongs to the public domain. Refer to Unlicense for more information.

This website is hosted on SiteGround hosting.

Support Us

If you use this website please consider linking back to https://htaccesscheatsheet.com/ or share us on Twitter

This website is 100% free and one of the fastest loading **Apache .htaccess cheatsheet** webpages on the web. It is all on one page, and optimised to help it quickly load and for you to easily find the .htaccess rules you need. Please get in touch if you have any questions.

Home • Contact/About us • Terms and Conditions • We do not set any cookies on our website.