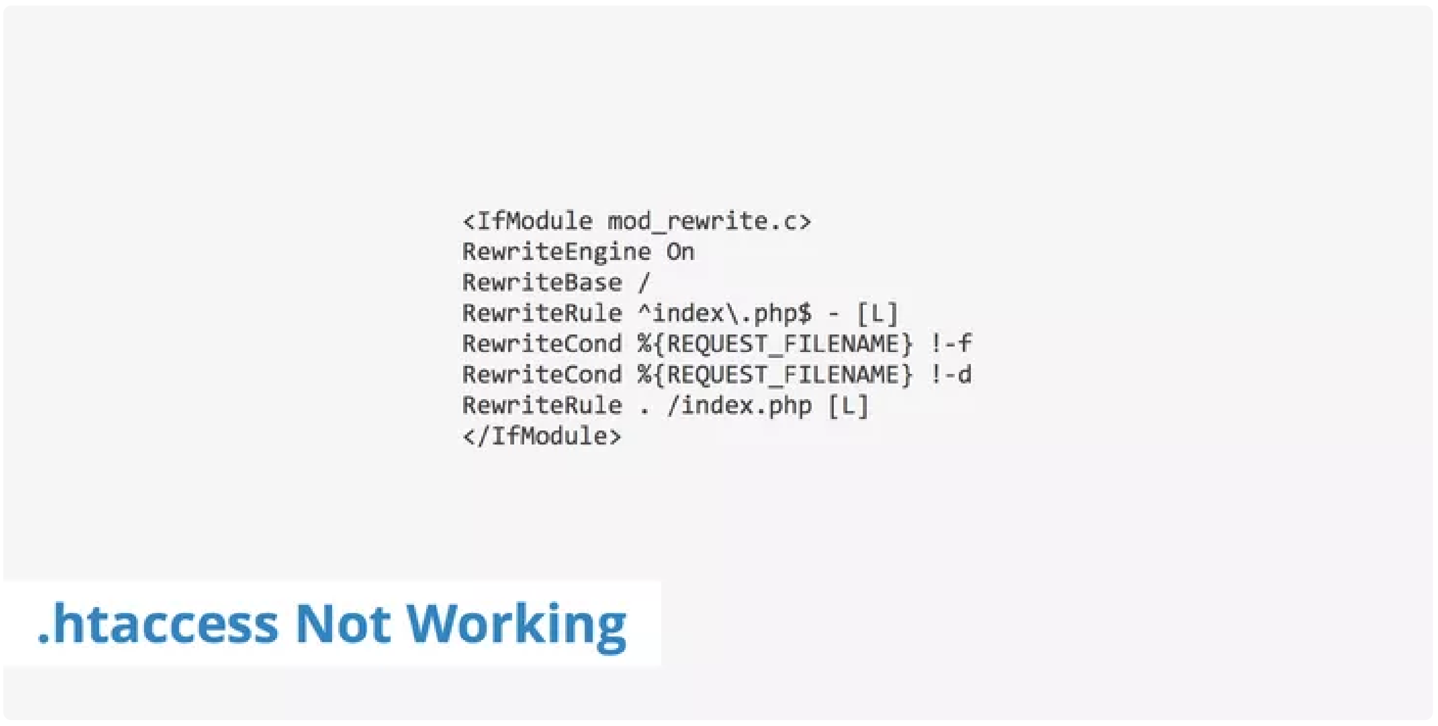
Support

Find answers, guides, and tutorials to supercharge your content delivery.

.htaccess Not Working - How to Troubleshoot and Fix

Updated on March 14, 2020



```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>
```

.htaccess Not Working

Getting Started

Zones

Integrations

Account Management

Troubleshooting

Tutorials

Glossary

Table of contents

Hypertext Access File, or most known as `.htaccess`, is a configuration file for Apache web servers that can be used to **define very specific configuration options**. Configurations can become quite granular with the use of regex however, most users typically stick to using popular `.htaccess` examples such as redirecting web pages or setting custom headers.

Although `.htaccess` can be quite useful, it can also be somewhat of a challenge to figure out where the issue lies given you are faced with an `.htaccess` not working. This post provides a few tips for helping to resolve this issue by identifying a few common `.htaccess` problems that you can check in your own `.htaccess` file, as well as a few troubleshooting techniques.

Common `.htaccess` problems

The following includes a few common `.htaccess` problems that are easy to fix and worth trying if you are experiencing issues with your `.htaccess` file not working.

`.htaccess` needs to be enabled with `AllowOverride`

This is the first thing that should be verified. If the `AllowOverride` directive is set to `None` then this will disable all `.htaccess` files. In order to verify this, you must open the Apache configuration file (typically either called `httpd.conf` or `apache.conf`) and check that the `AllowOverride` directive is set to `AllowOverride All`. If you needed to make changes to your Apache config, remember to save the file and **restart Apache**.

```
sudo service apache2 restart
```

The filename is misspelled or does not begin with a period

If you are creating an `.htaccess` file from scratch (i.e. you are not using a CMS which comes with an `.htaccess` file included) then you must ensure that the filename is correct and that it begins with a period (`.`). Without the period at the beginning, Apache will ignore the file - same goes for if the file is misspelled. Additionally, double check that the **filename is all lowercase**. Your `.htaccess` file should be named exactly as `.htaccess`.

The location of your rules needs to be above or below others

Certain `.htaccess` rules may be sensitive to where they are located within the `.htaccess` file and therefore cause an `.htaccess` not working issue. If upon adding an `.htaccess` rule you notice that it is not taking effect, try moving it above the previous rule or to the very beginning of your file.

Conflicting `.htaccess` files

Although most users simply use one `.htaccess` file, you have the ability to use multiple. Since `.htaccess` file rules apply to the directory that they live in, as well as all other subdirectories, it can happen that two or more `.htaccess` files are conflicting with one another. To verify this, try disabling each additional `.htaccess` file you have one-by-one in order to see where the issue is.

Improper syntax being used

It is quite common for a syntax error to be the reason for an `.htaccess` file not working. If you are familiar with how to read and configure `.htaccess` rules, double check your configuration. Otherwise, you can use the troubleshooting tips mentioned in the next section to help determine why you are experiencing an issue.

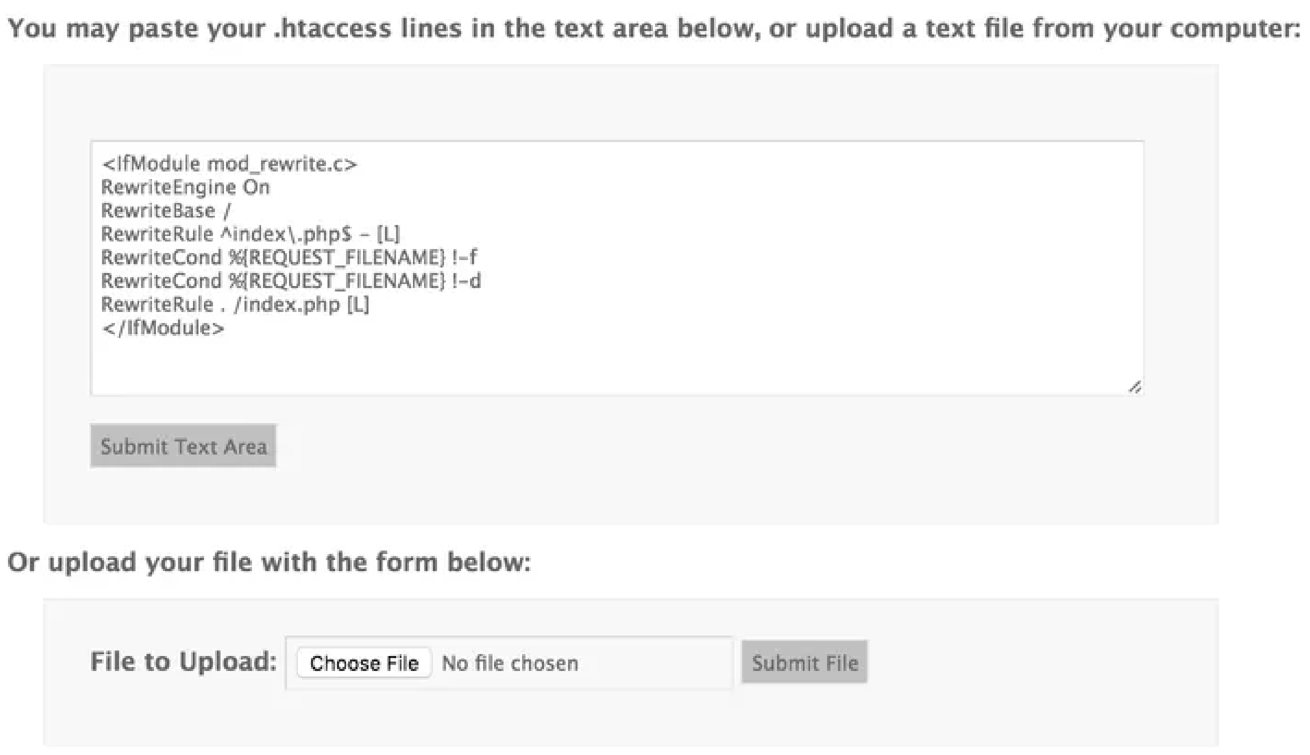
How to troubleshoot `.htaccess` not working

There are a few options available for troubleshooting an `.htaccess` not working. Depending upon the type of issue you are trying to solve, you may need to use a **combination of the suggestions** mentioned below to determine what steps need to be taken to rectify the issue.

Using an `.htaccess` validator

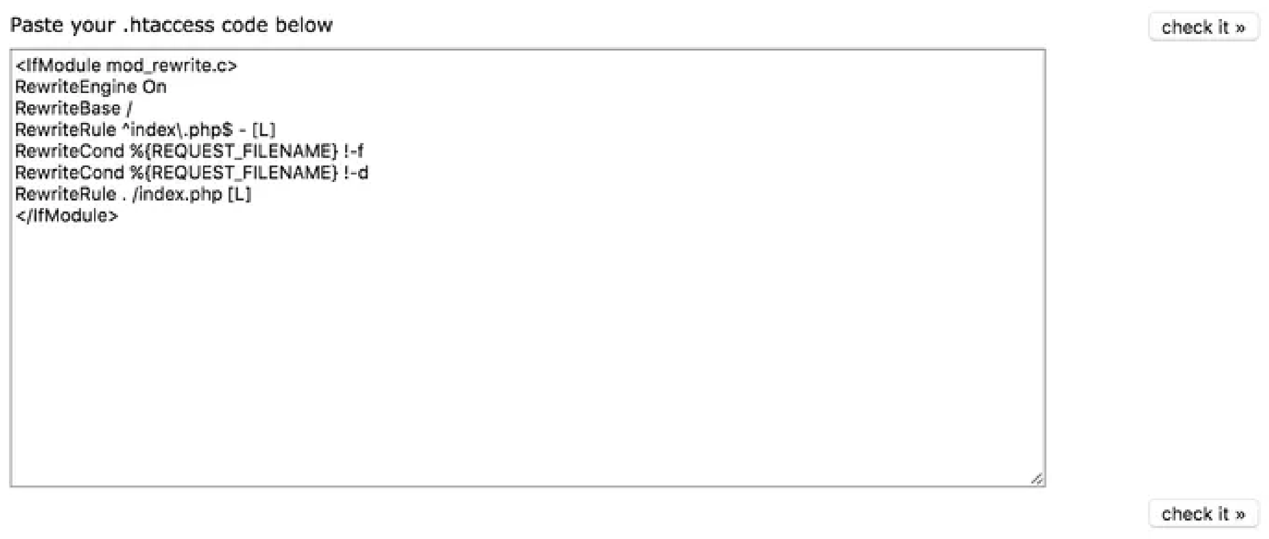
If you're having issues with the actual syntax of your `.htaccess` file, you can run its contents through an `.htaccess` validator. The following tools will check your syntax and report back any errors that they find.

- .htaccess Check - This first tool gives you two options for validating your `.htaccess` file. You can either copy and paste the contents of your file directly into the tool or upload an `.htaccess` file. The tool will then check your syntax and **highlight any lines that it finds errors on**.


You may paste your .htaccess lines in the text area below, or upload a text file from your computer:

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>
```

- Lyxx - You can also use the `.htaccess` syntax validator offered by Lyxx. This tool is similar to the one mentioned above except it does not have the option to upload an `.htaccess` file, you can only copy and paste your syntax.


Paste your .htaccess code below

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>
```

Checking the Apache error log

If upon making changes to your `.htaccess` file your website breaks, you can also check the **Apache error log** for additional debugging information. The Apache error log file is typically located in the `/var/log/apache2/` directory. Therefore, let's say for example we have the following content in our `.htaccess` file.

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

This is some gibberish
```

As you can see, I've added "This is some gibberish" to intentionally throw an error. Once this file is saved and the page is reloaded, we can check the error logs with the following command.

```
sudo tail /var/log/apache2/error.log
```

In this case, Apache throws the following error:

```
/var/www/wordpress/.htaccess: Invalid command 'This', perhaps misspelled or defined by
```

We can use this information to then go back to our `.htaccess` file and remove or modify any parts of the file that were flagged in the error log.

Debugging with the Apache configuration file

Lastly, you can also debug the content of your `.htaccess` file by inserting it into your Apache configuration file instead. All of the `.htaccess` rules will still apply in your configuration file, however now **Apache will parse and check the configuration file**. Be sure to include all the contents of your `.htaccess` in the `<Directory>` directive. Using the same example as above, the `<Directory>` portion of your config file may look similar to the following.

```
<Directory /var/www/wordpress>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all

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

    This is gibberish
</Directory>
```

Once saved, you can run the following command to check the syntax of your config file.

```
sudo apache2ctl -t
```

In this case, Apache returns an error message that says there is a syntax error on line 72 of my configuration file. If you use this method you may also want to **verify the error logs** in the event that any additional information was recorded there.

Debugging `mod_rewrite` with logs

If you're using the Apache `mod_rewrite` module you can also enable the rewrite log to provide you with more debugging details. To do this, you need to have **access to your Apache web server configuration file**. Start by opening the configuration file and adding the appropriate directive values as required. For example:

```
LogLevel alert rewrite:trace6
```

The above snippet will log all `mod_rewrite` errors up to the the "alert" level in your `error.log` file. Check out Apache's log level directive to learn more. It should be noted that the higher trace log level you define, the slower this will make your Apache web server.

Summary

`.htaccess` files are **extremely useful** in many cases for users who either do not have root permissions or for users who simply aren't comfortable in making changes in their web server's configuration file. Trying to debug `.htaccess` not working isn't always the easiest thing to do, however, hopefully by checking the above mentioned `.htaccess` common problems as well as the troubleshooting tips, you'll have a better grasp on what you may have to modify to get your `.htaccess` file running smoothly.

Additionally, if you would like to do some further testing, give the htaccess tester tool a try. It allows you to specify a certain URL as well as the rules you would like to include and then shows which rules were tested, which ones met the criteria, and which ones were executed.