405 Method Not Allowed: What It Is and How to Fix It

[January 18, 2018](https://airbrake.io/blog/http-errors/405-method-not-allowed) | [Andrew Powell-Morse](https://airbrake.io/blog/author/andrewpomo) | [HTTP Errors](https://airbrake.io/blog/category/http-errors)

The **405 Method Not Allowed** is an [HTTP response status code](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status) indicating that the specified request [HTTP method](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods) was received and recognized by the server, but the server has rejected that particular method for the requested resource. This message is distinctly different from the [404 Not Found](https://airbrake.io/blog/http-errors/404-not-found-error) code that we [looked at a while back](https://airbrake.io/blog/http-errors/404-not-found-error). A 404 Not Found error indicates that the requested resource could not be found, and is often the result of an incorrect URL, or trying to access a resource for which the client has inadequate permissions. Conversely, a 405 code response confirms that the requested resource is valid and exists, *but* the client has used an unacceptable HTTP method during the request.

Like most HTTP response codes — especially for those that indicate an error — the cause of a 405 Method Not Allowed reponse can be difficult to track down and fix. With a potential pool of [*over 50* status codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes) that represent the complex relationship between the client, a web application, a web server, and often multiple third-party web services, determining the cause of a particular status code can be a challenge under the best of circumstances.

In this article we’ll examine the 405 Method Not Allowed in more detail by looking at what might cause a message, along with a handful of tips for diagnosing and debugging the appearance of this error within your own application. We’ll even examine a number of the most popular content management systems (CMSs) for potential problem areas that could cause your own website to be generating a 405 Method Not Allowed unexpectedly. Let’s dive in!

Server- or Client-Side?

All HTTP response status codes that are in the 4xx category are considered client error responses. This category contrasts with 5xx classification errors, such as the [503 Service Unavailable Error](https://airbrake.io/blog/http-errors/503-service-unavailable) we [explored a few months ago](https://airbrake.io/blog/http-errors/503-service-unavailable), which are considered server error responses. That said, the appearance of a 4xx error doesn’t necessarily mean the issue is on the client side, where the “client” is the web browser or device being used to access the application. Oftentimes, if you’re trying to diagnose an issue within your own application, you can immediately ignore most client-side code and components, such as HTML, cascading style sheets (CSS), client-side JavaScript, and so forth. This doesn’t apply *solely* to web sites, either. Many smart phone apps, which implement a modern looking user interface, are actually powered by a normal web application behind the scenes that is simply hidden from the user.

On the other hand, this doesn’t entirely rule out the server as the actual *cause* of a 405 Method Not Allowed. In some cases, the server may be misconfigured and handling requests improperly, which can result in 405 code responses and other problematic traffic routing issues. We’ll explore some of these scenarios (and potential solutions) down below, but be aware that, even though the 405 Method Not Allowed is considered a client error response, it doesn’t inherently mean we can rule out either the client nor the server as the culprit in this scenario. In these scenarios, the server is still the network object that is producing the 405 Method Not Allowed and returning it as the HTTP response code to the client, but it could be that the client is causing the issue in some way.

Start With a Thorough Application Backup

As with anything, it’s better to have played it safe at the start than to screw something up and come to regret it later on down the road. As such, it is *critical* that you perform a full backup of your application, database, and all other components of your website or application *before* attempting any fixes or changes to the system. Even better, if you have the capability, create a complete copy of the application and stick the copy on a secondary staging server that isn’t active or is inaccessible to the public. This will give you a clean testing ground with which to test all potential fixes to resolve the issue, without threatening the security or sanctity of your live application.

Diagnosing a 405 Method Not Allowed

As discussed in the introduction, a 405 Method Not Allowed indicates that the user agent (the web browser, in most cases) has requested a valid resource using an invalid HTTP method. This could happen in a few different circumstances:

* The user agent is accidentally sending an incorrect HTTP method.
* The server is expecting only a handful of valid HTTP methods for the requested resource.

There are currently [9 possible HTTP methods](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods) that can be used, though some of them are far more prevalent than others. For example, the [GET](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET) method handles the *vast majority* of requests made on the Internet, as it is used solely to *retrieve* data (i.e. “get” a page or resource). The [POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST) method is the second-most common, and it’s typically used to send data to the server (such as login credentials).

Since each of the possible HTTP methods have their own intended uses, it often doesn’t make sense for a server to accept requests using certain methods for particular resources. For example, a resource might exist at the URL https://airbrake.io/users/create, which is used by the server to *create* a new user when valid credentials are sent via a POST HTTP method request. Therefore, in such a scenario, it makes no sense for the server to accept a GET request at that resource/URL, so it may respond with a 405 Method Not Allowed code when such a request is made.

Troubleshooting on the Client-Side

Since the 405 Method Not Allowed is a client error response code, it’s best to start by troubleshooting any potential client-side issues that could be causing this error. Here are a handful of tips to try on the browser or device that is giving you problems.

Check the Requested URL

The most common cause of a 405 Method Not Allowed is simply inputting an incorrect URL. As discussed before, many web servers are tightly secured to disallow access to improper URLs that the server isn’t prepared to provide access to. This could be anything from trying to access a file directory via a URL to attempting to gain access to a private page meant for other users. Since 405 codes are not as common as 404 codes, the appearance of a 405 *could* means that the requested URL is valid, but is typically used for . Either way, it’s a good idea to double-check the exact URL that is returning the 405 Method Not Allowed error to make sure it is intended resource.

Debugging Common Platforms

If you’re running common software packages on the server that is responding with the 405 Method Not Allowed, you may want to start by looking into the stability and functionality of those platforms first. The most common content management systems — like WordPress, Joomla!, and Drupal — are all typically well-tested out of the box, but once you start making modifications to the underlying extensions or PHP code (the language in which nearly all modern content management systems are written in), it’s all too easy to cause an unforeseen issue that results in a 405 Method Not Allowed.

There are a few tips below aimed at helping you troubleshoot some of these popular software platforms.

Rollback Recent Upgrades

If you recently updated the content management system itself just before the 405 Method Not Allowed appeared, you may want to consider rolling back to the previous version you had installed when things were working fine. Similarly, any extensions or modules that you may have recently upgraded can also cause server-side issues, so reverting to previous versions of those may also help. For assistance with this task, simply Google “downgrade [PLATFORM\_NAME]” and follow along. In some cases, however, certain CMSs don’t really provide a version downgrade capability, which indicates that they consider the base application, along with each new version released, to be extremely stable and bug-free. This is typically the case for the more popular platforms, so don’t be afraid if you can’t find an easy way to revert the platform to an older version.

Uninstall New Extensions, Modules, or Plugins

Depending on the particular content management system your application is using, the exact name of these components will be different, but they serve the same purpose across every system: improving the capabilities and features of the platform beyond what it’s normally capable of out of the box. But be warned: such extensions can, more or less, take full control of the system and make virtually any changes, whether it be to the PHP code, HTML, CSS, JavaScript, or database. As such, it may be wise to uninstall any new extensions that may have been recently added. Again, Google the extension name for the official documentation and assistance with this process.

Check for Unexpected Database Changes

It’s worth noting that, even *if* you uninstall an extension through the CMS dashboard, this doesn’t *guarantee* that changes made by the extension have been fully reverted. This is particularly true for many WordPress extensions, which are given carte blanche within the application, including full access rights to the database. Unless the extension author explicitly codes such things in, there are scenarios where an extension may modify database records that don’t “belong” to the extension itself, but are instead created and managed by other extensions (or even the base CMS itself). In those scenarios, the extension may not know how to revert alterations to database records, so it will ignore such things during uninstallation. Diagnosing such problems can be tricky, but I’ve personally encountered such scenarios multiple times, so your best course of action, assuming you’re reasonably convinced an extension is the likely culprit for the 405 Method Not Allowed, is to open the database and manually look through tables and records that were likely modified by the extension.

Above all, don’t be afraid to Google your issue. Try searching for specific terms related to your issue, such as the name of your application’s CMS, along with the 405 Method Not Allowed. Chances are you’ll find someone who has experienced the same issue.

Troubleshooting on the Server-Side

If you aren’t running a CMS application — or even if you are, but you’re confident the 405 Method Not Allowed isn’t related to that — here are some additional tips to help you troubleshoot what might be causing the issue on the server-side of things.

Confirm Your Server Configuration

Your application is likely running on a server that is using one of the two most popular web server softwares, Apache or nginx. At the time of publication, both of these web servers make up [84% of the world’s web server software](https://w3techs.com/technologies/overview/web_server/all)! Thus, one of the first steps you can take to determine what might be causing these 405 Method Not Allowed response codes is to check the configuration files for your web server software for unintentional redirect or request handling instructions.

To determine which web server your application is using you’ll want to look for a key file. If your web server is Apache then look for an .htaccess file within the root directory of your website file system. For example, if your application is on a shared host you’ll likely have a username associated with the hosting account. In such a case, the application root directory is typically found at the path of /home/<username>/public\_html/, so the .htaccess file would be at /home/<username>/public\_html/.htaccess.

If you located the .htaccess file then open it in a text editor and look for lines that use RewriteXXX directives, which are part of the [mod\_rewrite](http://httpd.apache.org/docs/current/mod/mod_rewrite.html) module in Apache. Covering exactly how these rules work is well beyond the scope of this article, however, the basic concept is that a RewriteCond directive defines a text-based pattern that will be matched against entered URLs. If a matching URL is requested by a visitor to the site, the RewriteRule directive that follows one or more RewriteCond directives is used to perform the actual redirection of the request to the appropriate URL.

For example, here is a simple RewriteRule that matches all incoming GET requests to https://airbrake.io/users/create and responds with a 405 Method Not Allowed error code:

RewriteEngine on

RewriteCond %{REQUEST\_URI} ^/users/create/?.\*$

RewriteCond %{REQUEST\_METHOD} =GET

RewriteRule ^(.\*)$ http://airbrake.io/users/new$1 [R=405,L]

Notice the R=405 flag at the end of the RewriteRule, which explicitly states that the response code should be 405, indicating to user agents that the resource exists, but the provided HTTP method is not allowed. Thus, if you find any strange RewriteCond or RewriteRule directives in the .htaccess file that don’t seem to belong, try temporarily commenting them out (using the # character prefix) and restarting your web server to see if this resolves the issue.

On the other hand, if your server is running on nginx, you’ll need to look for a completely different configuration file. By default this file is named nginx.conf and is located in one of a few common directories: /usr/local/nginx/conf, /etc/nginx, or /usr/local/etc/nginx. Once located, open nginx.conf in a text editor and look for directives that are using the 405 response code flag. For example, here is a simple block directive (i.e. a named set of directives) that configures a virtual server for airbrake.io and ensures that a POST request to https://airbrake.io/users/create fails and is responded with a 405 response code:

server {

listen 80;

listen 443 ssl;

server\_name airbrake.io;

location /users/create {

if ($request\_method = POST) {

return 405 https://airbrake.io/users/create$request\_uri;

}

}

}

Have a look through your nginx.conf file for any abnormal directives or lines that include the 405 flag. Comment out any abnormalities before restarting the server to see if the issue was resolved.

Configuration options for each different type of web server can vary dramatically, so we’ll just list a few popular ones to give you some resources to look through, depending on what type of server your application is running on:

* [Apache](https://httpd.apache.org/docs/2.4/configuring.html)
* [Nginx](http://nginx.org/en/docs/beginners_guide.html)
* [IIS](https://docs.microsoft.com/en-us/iis/configuration/)
* [Node.js](https://nodejs.org/api/)
* [Apache Tomcat](https://tomcat.apache.org/tomcat-9.0-doc/setup.html)

Look Through the Logs

Nearly every web application will keep some form of server-side logs. Application logs are typically the history of what the application did, such as which pages were requested, which servers it connected to, which database results it provides, and so forth. Server logs are related to the actual hardware that is running the application, and will often provide details about the health and status of all connected services, or even just the server itself. Google “logs [PLATFORM\_NAME]” if you’re using a CMS, or “logs [PROGRAMMING\_LANGUAGE]” and “logs [OPERATING\_SYSTEM]” if you’re running a custom application, to get more information on finding the logs in question.

Debug Your Application Code or Scripts

If all else fails, it may be that a problem in some custom code within your application is causing the issue. Try to diagnose where the issue may be coming from through manually debugging your application, along with parsing through application and server logs. Ideally, make a copy of the entire application to a local development machine and perform a step-by-step debug process, which will allow you to recreate the exact scenario in which the 405 Method Not Allowed occurred and view the application code at the moment something goes wrong.

No matter the cause — and even if you managed to fix it this time — the appearance of an issue like the 405 Method Not Allowed within your own application is a good indication you may want to implement an error management tool, which will help you automatically detect errors and will alert you the very moment they occur. [Airbrake’s error monitoring software](https://airbrake.io/account/new?utm_source=blog&utm_medium=end-post&utm_campaign=airbrake-405-method-not-allowed&utm_content=405%20Method%20Not%20Allowed:%20What%20It%20Is%20and%20How%20to%20Fix%20It) provides real-time error monitoring and automatic exception reporting for all your development projects. Airbrake’s state of the art web dashboard ensures you receive round-the-clock status updates on your application’s health and error rates. No matter what you’re working on, Airbrake easily integrates with all the most popular languages and frameworks. Plus, Airbrake makes it easy to customize exception parameters, while giving you complete control of the active error filter system, so you only gather the errors that matter most.

Check out [Airbrake’s error monitoring software](https://airbrake.io/account/new?utm_source=blog&utm_medium=end-post&utm_campaign=airbrake-405-method-not-allowed&utm_content=405%20Method%20Not%20Allowed:%20What%20It%20Is%20and%20How%20to%20Fix%20It) today and see for yourself why so many of the world’s best engineering teams use Airbrake to revolutionize their exception handling practices!