

# Why does my JavaScript code receive a "No 'Access-Control-Allow-Origin' header is present on the requested resource" error, while Postman does not?

Asked 10 years, 4 months ago   Modified 8 months ago   Viewed 6.3m times



3307



**Mod note:** This question is about why `XMLHttpRequest` / `fetch` /etc. on the browser are subject to the Same Access Policy restrictions (you get errors mentioning CORB or CORS) while Postman is not. This question is **not** about how to fix a "No 'Access-Control-Allow-Origin'..." error. It's about why they happen.

Please stop posting:

- CORS configurations for every language/framework under the sun. Instead [find your relevant language/framework's question](#).
- 3rd party services that allow a request to circumvent CORS
- Command line options for turning off CORS for various browsers

I am trying to do authorization using [JavaScript](#) by connecting to the [RESTful API](#) built-in [Flask](#). However, when I make the request, I get the following error:

```
XMLHttpRequest cannot load http://myApiUrl/login.  
No 'Access-Control-Allow-Origin' header is present on the requested resource.  
Origin 'null' is therefore not allowed access.
```

I know that the API or remote resource must set the header, but why did it work when I made the request via the Chrome extension [Postman](#)?

This is the request code:

```
$.ajax({  
  type: 'POST',  
  dataType: 'text',  
  url: api,  
  username: 'user',  
  password: 'pass',  
  crossDomain: true,  
  xhrFields: {  
    withCredentials: true,  
  },  
})  
.done(function (data) {  
  console.log('done');  
})  
.fail(function (xhr, textStatus, errorThrown) {  
  alert(xhr.responseText);  
})
```

```
alert(textStatus);
});
```

[javascript](#)[jquery](#)[cors](#)[postman](#)[same-origin-policy](#)[Share](#) [Improve this question](#) [Follow](#)

edited Mar 28, 2023 at 15:50

[pizzaisdavid](#)

469 ● 3 ● 14

asked Nov 17, 2013 at 19:29

[Mr Jedi](#)

34.2k ● 8 ● 32 ● 40

56 Are you doing the request from localhost or directly executing HTML? – [MD. Sahib Bin Mahboob](#) Nov 17, 2013 at 19:31

2 @MD.SahibBinMahboob If I understand your question I do request from localhost - I have page on my computer and just run it. When I deploy site on hosting it's gave same result. – [Mr Jedi](#) Nov 17, 2013 at 19:43

16 For anyone looking for more reading, MDN has a good article all about ajax and cross origin requests: [developer.mozilla.org/en-US/docs/Web/HTTP/Access\\_control\\_CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS) – [Sam Eaton](#) Jun 18, 2015 at 15:22

1 An answer to this question (now deleted and only visible to 10K'ers) is the subject of meta question [Why was this upvoted answer deleted once, and deleted again when reposted?](#) – [Peter Mortensen](#) Oct 4, 2021 at 9:50

A related CORS deep dive into this same error but to do with cache and headers from S3 / Cloudfront triggering it is also here: [stackoverflow.com/questions/44800431/](https://stackoverflow.com/questions/44800431/) – [OG Sean](#) Jun 22, 2022 at 23:11

15 Answers

Sorted by:

Highest score (default)



1662

If I understood it right you are doing an [XMLHttpRequest](#) to a different domain than your page is on. So the browser is blocking it as it usually allows a request in the same origin for security reasons. You need to do something different when you want to do a cross-domain request.



When you are using Postman they are not restricted by this policy. Quoted from [Cross-Origin XMLHttpRequest](#):



Regular web pages can use the XMLHttpRequest object to send and receive data from remote servers, but they're limited by the same origin policy. Extensions aren't so limited. An extension can talk to remote servers outside of its origin, as long as it first requests cross-origin permissions.

[Share](#) [Improve this answer](#) [Follow](#)

edited Sep 12, 2022 at 2:44

[Max von Hippel](#)


2,934 ● 3 ● 29 ● 46

answered Nov 17, 2013 at 19:49

[MD. Sahib Bin Mahboob](#)

20.4k ● 2 ● 23 ● 45

252 The browser is not blocking the request. The only browsers that outright block cross-origin ajax requests is IE7 or older. All browsers, other than IE7 and older, implement the CORS spec (IE8 & IE9 partially). All you need to do is opt-in to CORS requests on your API server by returning the proper headers based on the

request. You should read up on CORS concepts at [mzl.la/VOFrSz](https://mzl.la/VOFrSz). Postman sends requests via XHR as well. If you are not seeing the same problem when using postman, this means that you are unknowingly not sending the same request via postman. – Ray Nicholas Nov 17, 2013 at 20:01 

- 22 @MD.SahibBinMahboob Postman is NOT sending a request "from your java/python" code. It is sending the request directly from the browser. [XHR in Chrome extensions does work a bit differently, especially when cross-origin requests are involved](#). – Ray Nicholas Nov 17, 2013 at 20:08
- 2 Found a detailed example on this post: [stackoverflow.com/questions/66486610/...](https://stackoverflow.com/questions/66486610/...) – Rayed Mar 24, 2023 at 2:23
- 1 Happens to us locally only when certain request param equal 10 rofl? Works in all other scenarios. what the heck? :D – trainoasis Sep 13, 2023 at 13:09



369



**WARNING:** Using `Access-Control-Allow-Origin: *` can make your API/website vulnerable to [cross-site request forgery](#) (CSRF) attacks. Make certain you [understand the risks](#) before using this code.

It's very simple to solve if you are using [PHP](#). Just add the following script in the beginning of your PHP page which handles the request:

```
<?php header('Access-Control-Allow-Origin: *'); ?>
```

If you are using [Node-red](#) you have to allow [CORS](#) in the `node-red/settings.js` file by un-commenting the following lines:

```
// The following property can be used to configure cross-origin resource sharing
// in the HTTP nodes.
// See https://github.com/troygoode/node-cors#configuration-options for
// details on its contents. The following is a basic permissive set of options:
httpNodeCors: {
  origin: "*",
  methods: "GET,PUT,POST,DELETE"
},
```

If you are using [Flask](#) same as the question; you have first to install `flask-cors`

```
pip install -U flask-cors
```

Then include the Flask `cors` package in your application.

```
from flask_cors import CORS
```

A simple application will look like:

```
from flask import Flask
from flask_cors import CORS
```

```
app = Flask(__name__)
CORS(app)

@app.route("/")
def helloWorld():
    return "Hello, cross-origin-world!"
```

For more details, you can check the [Flask documentation](#).

Share Improve this answer Follow

edited Sep 4, 2022 at 21:26



Peter Mortensen

31k ● 22 ● 109 ● 132

answered Dec 3, 2014 at 20:24



Shady Mohamed Sherif

15.5k ● 4 ● 47 ● 57

241 You shouldn't *turn off* CORS because you don't know what its for. This leaves your users in a fundamentally unsafe state. – [user229044](#) ♦ Dec 30, 2014 at 6:12 ✎

182 Even though it might not be secure, the question was not about security, but how to accomplish the task. This is one of the options that a developer has to choose from when dealing with cross-domain AJAX requests. It helped me resolve the issue, and for my application, I don't care where the data came from. I sanitize all the input with PHP on the destination domain, so, if someone wants to post some junk to it, let them try. The main point here is, cross-domain AJAX can be allowed from the destination domain. +1 for the answer. – [ZurabWeb](#) Feb 26, 2015 at 16:37 ✎

5 @meagar Agreeing with you that we shouldn't turn of CORS but at times we need to test the application while developing it and for that, the easiest way is to turn of CORS and check if everything works fine. Many times frontend devs don't have access to the backend system where they can change things or they need to write a proxy for the same. The best way to add a chrome extension that turns off CORS for development purposes, as written in the answer which is deleted. – [shruti](#) Sep 29, 2021 at 15:05

1 It should be much helpful if the answer (or the edit with the WARNING on top) would explain to whom is risky if using that header() script in php. The question here is about a foreign site where we have no control, and that only allows us to navigate and see it from a browser, while if we need to access the resources from our server instead it launches the CORS protection (to not let us make too much inquiries per second). Therefore, my question still stands, what dangers do we visitors have if using in OUR server that header() script ?? Did the editor confused the visitor (us) with the host? – [Eve](#) Nov 30, 2021 at 21:37

1 @ShadyMohamedSherif so isn't true that there was a time, historically, when people were allowed to 'consume' a website exposed at an IP /through a certain port/ with any means (browsers, apps, whatever), and after abuses occurred the website owners started to limit to the browsers only any connection to their page? In other words, when people just requested slowly a webpage (on a browser). I mean, even a ddos cannot happen through a browser, to not speak here about other devious ways to inquiry a website content that probably exist – [Eve](#) Dec 10, 2021 at 3:21



Because

`$.ajax({type: "POST" - calls OPTIONS`

107

`$.post( - calls POST`



Both are different. [Postman](#) calls "POST" properly, but when we call it, it will be "OPTIONS".



For C# web services - [Web API](#)



Please add the following code in your `web.config` file under the `<system.webServer>` tag. This will work:

```
<httpProtocol>
  <customHeaders>
    <add name="Access-Control-Allow-Origin" value="*" />
  </customHeaders>
</httpProtocol>
```

Please make sure you are not doing any mistake in the Ajax call.

## jQuery

```
$.ajax({
  url: 'http://mysite.microsoft.sample.xyz.com/api/mycall',
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded'
  },
  type: "POST", /* or type:"GET" or type:"PUT" */
  dataType: "json",
  data: {
  },
  success: function (result) {
    console.log(result);
  },
  error: function () {
    console.log("error");
  }
});
```

**Note:** If you are looking for downloading content from a third-party website then this will not help you. You can try the following code, but not JavaScript.

```
System.Net.WebClient wc = new System.Net.WebClient();
string str =
wc.DownloadString("http://mysite.microsoft.sample.xyz.com/api/mycall");
```

Share Improve this answer Follow

edited Sep 4, 2022 at 21:27



Peter Mortensen

31k ● 22 ● 109 ● 132

answered Dec 13, 2016 at 13:02



George

2,962 ● 2 ● 15 ● 11



## Deep

82



In the below investigation as API, I use <http://example.com> instead of <http://myApiUrl/login> from your question, because this first one working. I assume that your page is on <http://my-site.local:8088>.



**NOTE:** The API and your page have different domains!

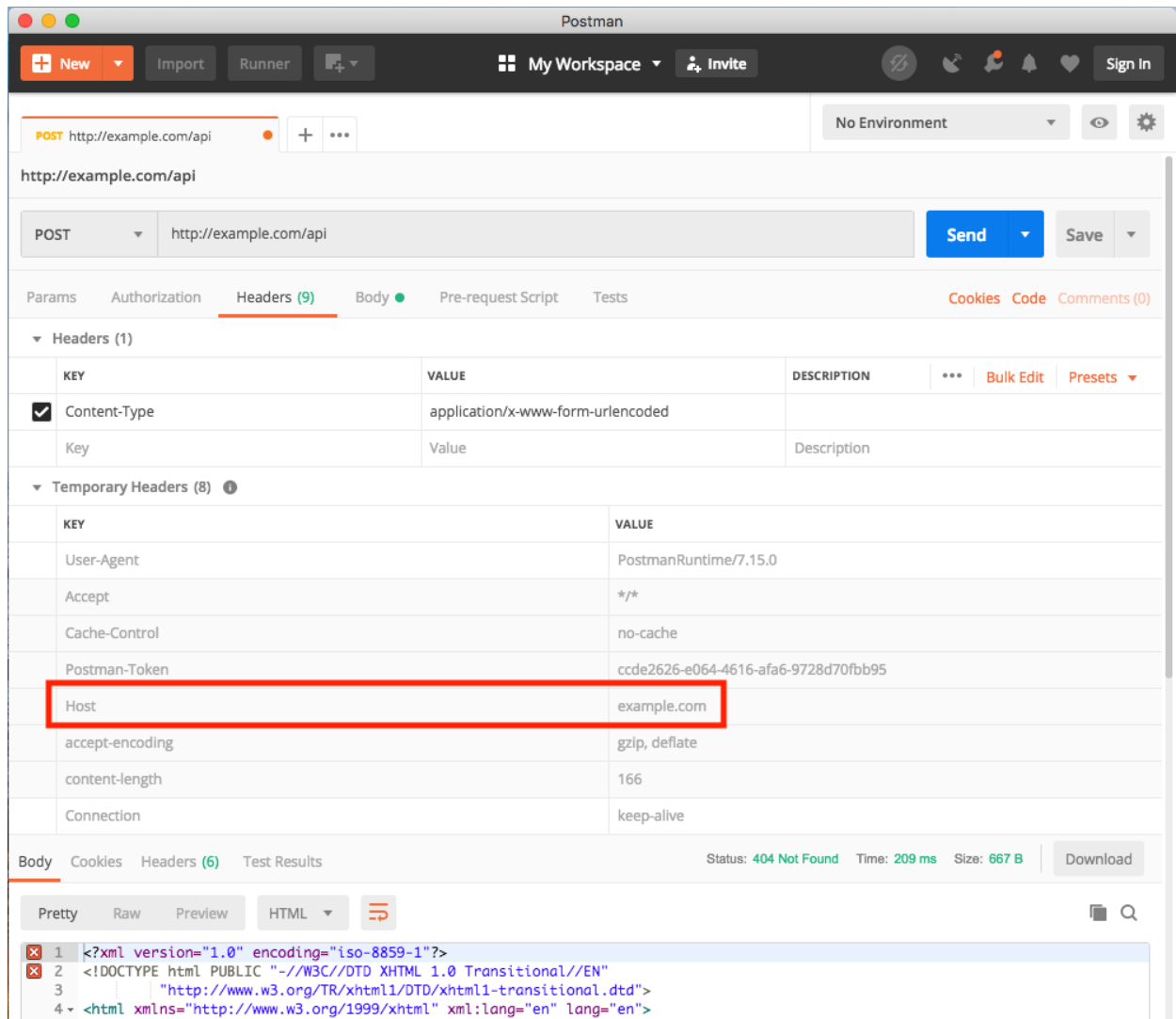


The reason why you see different results is that Postman:

- set header `Host=example.com` (your API)
- NOT set header `Origin`

- Postman actually not use your website url at all (you only type your API address into Postman)
  - he only send request to API, so he assume that website has same address as API (browser not assume this)

This is similar to browsers' way of sending requests when the site and API has the same domain (browsers also set the header item `Referer=http://my-site.local:8088`, however I don't see it in Postman). When `origin` header is *not* set, usually servers allow such requests by default.



This is the standard way how Postman sends requests. But a browser sends requests differently when **your site and API have different domains**, and then [CORS](#) occurs and the browser automatically:

- sets header `Host=example.com` (yours as API)
- sets header `Origin=http://my-site.local:8088` (your site)

(The header `Referer` has the same value as `Origin`). And now in Chrome's *Console & Networks* tab you will see:

✖ GET http://my-site.local:8088/favicon.ico 404 (Not Found) favicon.ico:1

2 ▶ POST http://example.com/api net::ERR\_ABORTED 404 (Not Found) (index):41

✖ Access to fetch at 'http://example.com/api' from origin 'http://my-site.local:8088' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

✖ Uncaught (in promise) TypeError: Failed to fetch (index):1

>

▼ General

Request URL: http://example.com/api

Request Method: POST

Status Code: 404 Not Found

Remote Address: 93.184.216.34:80

Referrer Policy: no-referrer-when-downgrade

▶ Response Headers (6)

▼ Request Headers view source

Accept: \*/\*

Accept-Encoding: gzip, deflate

Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7

Connection: keep-alive

Content-Length: 0

Host: example.com

Origin: http://my-site.local:8088

Referer: http://my-site.local:8088/

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.79 Safari/537.3

When you have **Host != Origin** this is CORS, and when the server detects such a request, it usually blocks it by default.

**Origin=null** is set when you open HTML content from a local directory, and it sends a request. The same situation is when you send a request inside an `<iframe>`, like in the below snippet (but here the **Host** header is not set at all) - in general, everywhere the HTML specification says opaque origin, you can translate that to **Origin=null**. More information about this you can find [here](#).

```
fetch('http://example.com/api', {method: 'POST'});
```

Look on chrome-console > network tab



Run code snippet

[Expand snippet](#)

If you do not use a simple CORS request, usually the browser automatically also sends an OPTIONS request before sending the main request - more information is [here](#). The snippet below shows it:

```
fetch('http://example.com/api', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' }
});
```

Look in chrome-console -> network tab to 'api' request.  
This is the OPTIONS request (the server does not allow sending a POST request)



Run code snippet

[Expand snippet](#)

You can change the configuration of your server to allow CORS requests.

Here is an example configuration which turns on **CORS on nginx** (nginx.conf file) - be very careful with setting `always/"$http_origin"` for nginx and `"*"` for Apache - this will unblock CORS from any domain (in production instead of stars use your concrete page adres which consume your api)

► [Show code snippet](#)

Here is an example configuration which turns on **CORS on Apache** (.htaccess file)

► [Show code snippet](#)

Share Improve this answer Follow

edited Feb 19, 2022 at 19:32

answered Dec 16, 2019 at 9:06



Kamil Kielczewski

88.8k ●31 ●380 ●353

10 Great great explained and easy to catch up! Thank u! – Nam G VU Feb 18, 2022 at 18:54

1 This answer clearly demonstrates what [this comment](#) describes: **If you are not seeing the same problem when using postman, this means that you are unknowingly not sending the same request via postman**. Well done! – Ray Jasson Jan 2, 2023 at 17:39 ✎

This answer seems to be somewhat misleading. While the server might block requests in which the 'Host' and 'Origin' are different, it might also not. But as long as it doesn't explicitly send a header telling it's okay to complete the request, the **browser** will block it to protect the user. This protection does not exist in Postman because it has a different purpose and different conditions. – Noam Sep 28, 2023 at 21:25



Applying a CORS restriction is a security feature defined by a server and implemented by a **browser**.

46



The browser looks at the CORS policy of the server and respects it.



However, the Postman tool does not bother about the CORS policy of the server.



That is why the CORS error appears in the browser, but not in Postman.





Peter Mortensen

31k ● 22 ● 109 ● 132



Gopinath

4,599 ● 1 ● 14 ● 19



39



The error you get is due to the CORS standard, which sets some restrictions on how JavaScript can perform ajax requests.

The CORS standard is a client-side standard, implemented in the browser. So it is the browser which prevent the call from completing and generates the error message - not the server.

Postman does not implement the CORS restrictions, which is why you don't see the same error when making the same call from Postman.

*Why* doesn't Postman implement CORS? CORS defines the restrictions relative to the origin (URL domain) of the page which initiates the request. But in Postman the requests doesn't originate from a page with an URL so CORS does not apply.



JacquesB

42.1k ● 13 ● 72 ● 88

8 @MrJedi: The accepted answer does not explain why the request succeeds in Postman, which was the original question. – JacquesB Jan 11, 2021 at 10:24

The servers originally were meant to send streams to clients (browser software programs) not to various desktop or server applications instead that could behave in twisted ways. A browser establishes a handshake protocol with the server, receives the confirmation in regard to the connection then the data stream resumes. There were (DDOS) situations where bot farms servers sent millions of inquiries and the host committed many resources (opened processes) to each of these stalled connections that eventually never occurred - thus blocking its ability to answer to other legit requests – Eve Nov 30, 2021 at 22:09

@Eve: CORS is a *client side* measure implemented in the browser and therefore not something which can prevent DDOS attacks from bot farms. – JacquesB Oct 28, 2022 at 14:08

2 this is actually the best explanation, and a 'aha moment' to realize that CORS is a browser standard! – Aris Dec 1, 2022 at 8:28



17



## Solution & Issue Origins

You are making a [XMLHttpRequest](#) to different domains, example:

1. Domain one: `some-domain.com`
2. Domain Two: `some-different-domain.com`

This difference in domain names triggers CORS ([Cross-Origin Resource Sharing](#)) policy called SOP ([Same-Origin Policy](#)) that enforces the use of same domains (hence *Origin*) in [Ajax](#), XMLHttpRequest and other HTTP requests.

Why did it work when I made the request via the Chrome extension Postman?

A client (most **Browsers** and **Development Tools**) has a choice to enforce the Same-Origin Policy.

Most browsers enforce the policy of Same-Origin Policy to prevent issues related to **CSRF** ([Cross-Site Request Forgery](#)) attack.

**Postman** as a development tool chooses not to enforce SOP while some browsers enforce, this is why you can send requests via Postman that you cannot send with XMLHttpRequest via JS using the browser.

Share Improve this answer Follow

answered Jan 3, 2022 at 11:36



Stas Sorokin

3,343 ● 27 ● 19



For browser testing purposes:

7

Windows - Run:



```
chrome.exe --user-data-dir="C://Chrome dev session" --disable-web-security
```



The command above will disable chrome web security. So for example if you work on a local project and encounter CORS policy issue when trying to make a request, you can skip this type of error with the above command. Basically it will open a new chrome session.

Share Improve this answer Follow

edited Sep 5, 2022 at 8:37

answered Jan 10, 2022 at 12:28



Daniel Iftimie

225 ● 4 ● 7

1 Could you please explain it ? – [ZebraCoder](#) Jun 21, 2022 at 13:04 ✎

3 @ZebraCoder The command above will disable chrome web security. So for example if you work on a local project and encounter CORS policy issue when trying to make a request, you can skip this type of error with the above command. Basically it will open a new chrome session. – [Daniel Iftimie](#) Jun 23, 2022 at 10:22 ✎

Can you [add](#) the information to the answer itself? Comments may be deleted at any time. (But \*\*\*\*\* **without** \*\*\*\*\* "Edit:", "Update:", or similar - the answer should appear as if it was written today) – [Peter Mortensen](#) Sep 4, 2022 at 20:50 ✎



6

You might also get this error if your gateway timeout is too short and the resource you are accessing takes longer to process than the timeout. This may be the case for complex database queries etc. Thus, the above error code can be disguising this problem. Just check if the error code is 504 instead of 404 as in [Kamil's answer](#) or something else. If it is 504, then increasing the gateway timeout might fix the problem.



In my case the CORS error could be removed by disabling the same origin policy (CORS) in the [Internet Explorer](#) browser, see [How to disable same origin policy Internet Explorer](#). After doing this, it was a pure 504 error in the log.

Share Improve this answer Follow

edited Sep 4, 2022 at 21:33

answered Dec 9, 2021 at 12:34



Peter Mortensen

31k ● 22 ● 109 ● 132



NDM

549 ● 5 ● 6

If you gettimeout you doesn't get CORS error – Mr Jedi Dec 9, 2021 at 19:45

Well, I did in trouble shooting a system and the CORS error threw me off, that it was just the timeout that was too short, which resulted in a closed connection. After increasing the timeout, the system performed perfectly. So yes the timeout caused a No 'Access-Control-Allow-Origin' error which got me into this thread in the first place. So this might be helpful to others having this thrown along with a 504. – NDM Dec 12, 2021 at 16:49

It rather mean something wrong is with your app config. You shouldn't get this error on timeout – Mr Jedi Dec 14, 2021 at 20:49

- 1 I also was getting a confusing CORS 504 error when nginx, in my case, timed out. Increasing timeout got the service back online without CORS errors. Thanks for the hint. – ogie Aug 22, 2022 at 18:05



3



To resolve this issue, write this line of code in your `doGet()` or `doPost()` function whichever you are using in backend

```
response.setHeader("Access-Control-Allow-Origin", "*");
```

Instead of `"*"` you can type in the website or API URL endpoint which is accessing the website else it will be public.

Share Improve this answer Follow

edited Nov 11, 2022 at 6:12

answered Apr 13, 2022 at 19:35



lunarzshine

387 ● 3 ● 10



1



Your IP address is not whitelisted, so you are getting this error. Ask the backend staff to whitelist your IP address for the service you are accessing.

[Access-Control-Allow-Headers](#)

Share Improve this answer Follow

edited Sep 4, 2022 at 20:53

answered Feb 14, 2022 at 1:29



Peter Mortensen

31k ● 22 ● 109 ● 132



Bapan Biswas

33 ● 4



1



For me I got this issue for different reason, the remote domain was added to origins the deployed app works perfectly except one end point I got this issue:

```
Origin https://mai-frontend.vercel.app is not allowed by Access-Control-Allow-Origin. Status code: 500
```

and

Fetch API cannot load

<https://sciigo.herokuapp.com/recommendations/recommendationsByUser/8f1bb29e-8ce6-4df2-b138-ffe53650dbab> due to access control checks.

I discovered that my Heroku database table does not contains all the columns of my local table after updating Heroku database table everything worked well.

Share Improve this answer Follow

answered Nov 28, 2022 at 0:19



DINA TAKLIT

7,750 ● 10 ● 78 ● 84



1



You can allow the **CORS** by adding below scripts in **web.config** file in server.

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="Access-Control-Allow-Origin" value="*" />
      <add name="Access-Control-Allow-Methods" value="*" />
      <add name="Access-Control-Allow-Headers" value="*" />
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

Share Improve this answer Follow

answered Apr 12, 2023 at 15:58



Pranav MS

2,249 ● 2 ● 23 ● 53



-4



If you have the backend built using Node.js (Express.js), try using `npm i cors`

```
....
const cors = require("cors");
.....

app.use(cors());
...
```

Share Improve this answer Follow

answered Jul 29, 2023 at 16:40



Anmol Pal

92 ● 1 ● 4

1 This doesn't answer the question. – [dragonx](#) Aug 4, 2023 at 21:45



-7



It works for me by applying this middleware in globally:

```
<?php
namespace App\Http\Middleware;
```



```
use Closure;

class Cors {
    public function handle($request, Closure $next) {
        return $next($request)
            ->header('Access-Control-Allow-Origin', '*')
            ->header('Access-Control-Allow-Methods', 'GET, POST, PUT,
DELETE, OPTIONS')
            ->header('Access-Control-Allow-Headers',
"Accept, authorization, Authorization, Content-Type");
    }
}
```

Share Improve this answer Follow

edited Sep 4, 2022 at 21:17

answered Aug 11, 2022 at 5:44



Peter Mortensen

31k ● 22 ● 109 ● 132



Shohedul

179 ● 1 ● 1 ● 6

- 
- 1 What do you mean by *"applying this middleware in globally"*? Can you [elaborate](#)? – Peter Mortensen Sep 4, 2022 at 21:13
- 
- 1 How does this answer the question? The question is tagged with [JavaScript](#), [jQuery](#), and [CORS](#). If it does, it ought to be [explained](#). The class has the name "Cors", but how does that relate? What is the gist? – Peter Mortensen Sep 4, 2022 at 21:14
- 



**Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.