Table of contents

CORS in AMP

Many AMP components and extensions take advantage of remote endpoints by using Cross-Origin Resource Sharing (CORS) requests. This document explains the key aspects of using CORS in AMP. To learn about CORS itself, see the **W3 CORS Spec**.

Why do I need CORS for my own origin?

You might be confused as to why you'd need CORS for requests to your own origin, let's dig into that.

AMP components that fetch dynamic data (e.g., amp-form, amp-list, etc.) make CORS requests to remote endpoints to retrieve the data. If your AMP page includes such components, you'll need to handle CORS so that those requests do not fail.

Let's illustrate this with an example:

Let's say you have an AMP page that lists products with prices. To update the prices on the page, the user clicks a button, which retrieves the latest prices from a JSON endpoint (done via the amp-list component). The JSON is on your domain.

Okay, so the page is *on my domain* and the JSON is *on my domain*. I see no problem!

Ah, but how did your user get to your AMP page? Is it a cached page they access? It's quite likely that your user did not access your AMP page directly, but instead they discovered your page through another platform. For example, Google Search uses the Google AMP Cache to render AMP pages quickly; these are cached pages that are served from the Google AMP Cache, which is a *different* domain. When your user

clicks the button to update the prices on your page, the cached AMP page sends a request to your origin domain to get the prices, which is a mismatch between origins (cache -> origin domain). To allow for such cross-origin requests, you need to handle CORS, otherwise, the request fails.

Okay, what should I do?

1. For AMP pages that fetch dynamic data, make sure you test the cached version of those pages; *don't just test on your own domain*. (See **Testing CORS in AMP** section below)
2. Follow the instructions in this document for handling CORS requests and responses.

Utilizing cookies for CORS requests

Most AMP components that use CORS requests either automatically set the **credentials mode** or allow the author to optionally enable it. For example, the `amp-list` component fetches dynamic content from a CORS JSON endpoint, and allows the author to set the credential mode through the `credentials` attribute.

Example: Including personalized content in an amp-list via cookies

```
<amp-list
  credentials="include"
  src="<%host%>/json/product.json?clientId=CLIENT_ID(myCookie)
>
  <template type="amp-mustache">
    Your personal offer: ${{price}}  </template>
</amp-list>
```

By specifying the credentials mode, the origin can include cookies in the CORS request and also set cookies in the response (subject to **third-party cookie restrictions**).

Third-party cookie restrictions

The same third-party cookie restrictions specified in the browser also apply to the credentialed CORS requests in AMP. These restrictions depend on the browser and the platform, but for some browsers, the origin can only set cookies if the user has previously visited the origin in a 1st-party (top) window. Or, in other words, only after the user has directly visited the origin website itself. Given this, a service accessed via CORS cannot assume that it will be able to set cookies by default.

CORS security in AMP

To ensure valid and secure requests and responses for your AMP pages, you must:

1. **Verify the request**.
2. **Send the appropriate response headers**.

If you're using Node in your backend, you can use the **AMP CORS middleware**, which is part of the **AMP Toolbox**.

Verify CORS requests

When your endpoint receives a CORS request:

1. **Verify that the CORS `Origin` header is an allowed origin (publisher's origin + AMP caches)**.
2. **If there isn't an Origin header, check that the request is from the same origin (via `AMP-Same-Origin`)**.

1) Allow requests for specific CORS origins

CORS endpoints receive the requesting origin via the `Origin` HTTP header. Endpoints should only allow requests from: (1) the publisher's own origin; and (2) every `cacheDomain` origin listed in **https://cdn.ampproject.org/caches.json**.

For example, endpoints should allow requests from:

- Google AMP Cache subdomain: `https://<publisher's domain>.cdn.ampproject.org`
  (for example, `https://nytimes-com.cdn.ampproject.org`)

For information on AMP Cache URL formats, see these resources:

- **Google AMP Cache Overview**

2) Allow same-origin requests

For same-origin requests where the `Origin` header is missing, AMP sets the following custom header:

```
AMP-Same-Origin: true
```

This custom header is sent by the AMP Runtime when an XHR request is made on the same origin (i.e., document served from a non-cache URL). Allow requests that contain the `AMP-Same-Origin:true` header.

Send CORS response headers

After verifying the CORS request, the resulting HTTP response must contain the following headers:

Access-Control-Allow-Origin: <origin>

This header is a **W3 CORS Spec** requirement, where `origin` refers to the requesting origin that was allowed via the CORS `Origin` request header (for example, `"https://<publisher's subdomain>.cdn.ampproject.org"`).

Although the W3 CORS spec allows the value of `*` to be returned in the response, for improved security, you should:

- If the `Origin` header is present, validate and echo the value of the `Origin` header.

Processing state changing requests

Perform these validation checks *before* you process the request. This validation helps to provide protection against CSRF attacks, and avoids processing untrusted sources requests.

Before processing requests that could change the state of your system (for example, a user subscribes to or unsubscribes from a mailing list), check the following:

If the `Origin` header is set:

1. If the origin does not match one of the following values, stop and return an error response:

    - `<publisher's domain>.cdn.ampproject.org`
    - the publisher's origin (aka yours)

  where `*` represents a wildcard match, and not an actual asterisk (*).

2. Otherwise, process the request.

If the `Origin` header is NOT set:

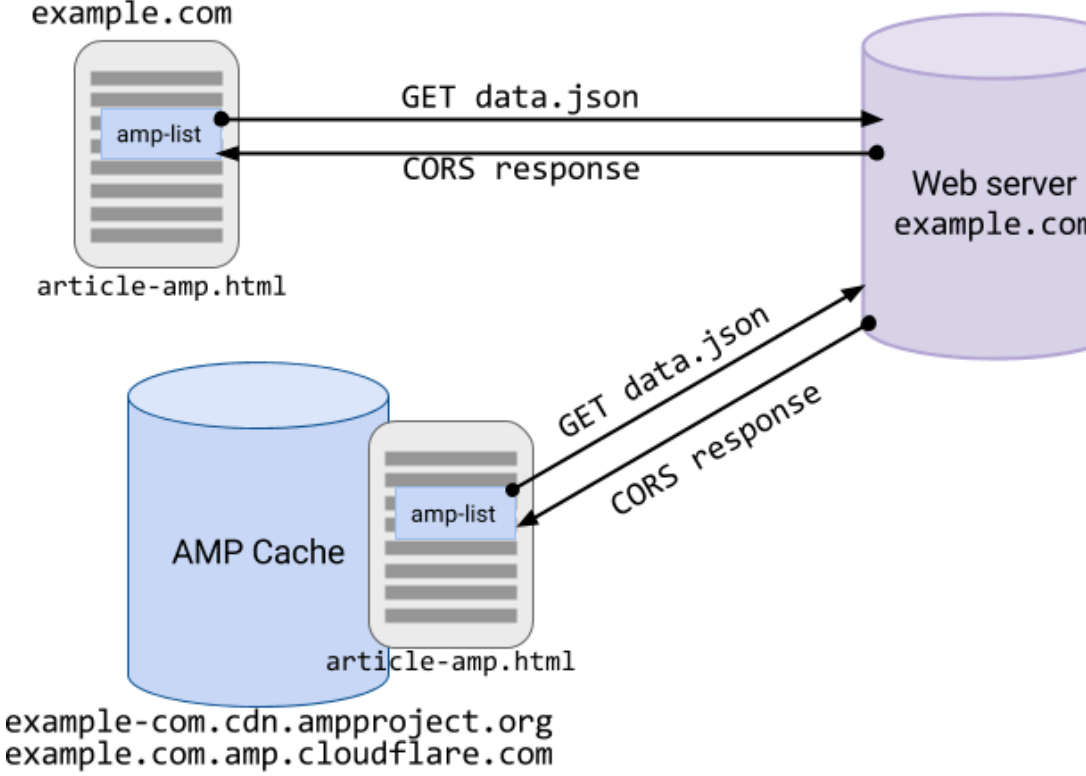
1. Verify that the request contains the `AMP-Same-Origin: true` header. If the request does not contain this header, stop and return an error response.
2. Otherwise, process the request.

Example walkthrough: Handing CORS requests and responses

There are two scenarios to account for in CORS requests to your endpoint:

1. A request from the same origin.
2. A request from a cached origin (from an AMP Cache).

Let's walk though these scenarios with an example. In our example, we manage the `example.com` site that hosts an AMP page named `article-amp.html.` The AMP page contains an `amp-list` to retrieve dynamic data from a `data.json` file that is also hosted on `example.com` . We want to process requests to our `data.json` file that come from our AMP page. These requests could be from the AMP page on the same origin (non-cached) or from the AMP page on a different origin (cached).



Allowed origins

Based on what we know about CORS and AMP (from **Verify CORS requests** above), for our example we will allow requests from the following domains:

- `example.com` --- Publisher's domain
- `example-com.cdn.ampproject.org` --- Google AMP Cache subdomain

Response headers for allowed requests

For requests from the allowed origins, our response will contain the following headers:

```
Access-Control-Allow-Origin: <origin>
```

These are additional response headers we might include in our CORS response:

```
Access-Control-Allow-Credentials: true
Content-Type: application/json
Access-Control-Max-Age: <delta-seconds>
Cache-Control: private, no-cache
```

Pseudo CORS logic

Our logic for handling CORS requests and responses can be simplified into the following pseudo code:

```
IF CORS header present
   IF origin IN allowed-origins
      allow request & send response
   ELSE
      deny request
ELSE
   IF "AMP-Same-Origin: true"
      allow request & send response
   ELSE
      deny request
```

CORS sample code

Here's a sample JavaScript function that we could use to handle CORS requests and responses:

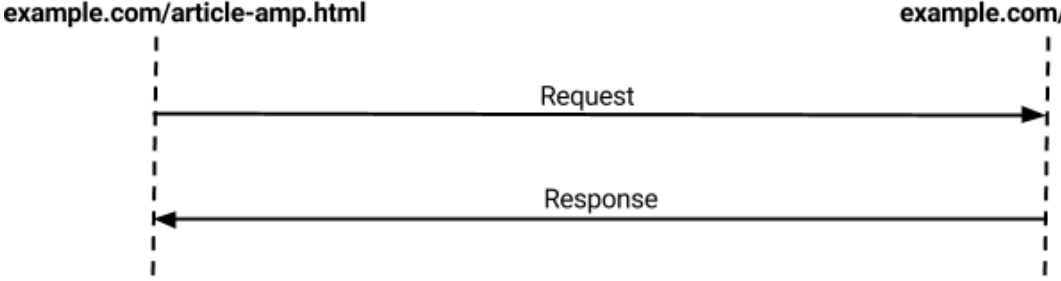
```
function assertCors(req, res, opt_validMethods, opt_exposeHeac
  var unauthorized = 'Unauthorized Request';
  var origin;
  var allowedOrigins = [
    'https://example.com',
    'https://example-com.cdn.ampproject.org',
    'https://cdn.ampproject.org',
  ];
  var allowedSourceOrigin = 'https://example.com'; //publisher
  // If same origin
  if (req.headers['amp-same-origin'] == 'true') {
    origin = sourceOrigin;
    // If allowed CORS origin & allowed source origin
  } else if (
    allowedOrigins.indexOf(req.headers.origin) != -1 &&
    sourceOrigin == allowedSourceOrigin
  ) {
    origin = req.headers.origin;
  } else {
    res.statusCode = 403;
    res.end(JSON.stringify({message: unauthorized}));
    throw unauthorized;
  }

  res.setHeader('Access-Control-Allow-Credentials', 'true');
  res.setHeader('Access-Control-Allow-Origin', origin);
}
```

Note: For a working code sample, see **amp-cors.js**.

Scenario 1: Get request from AMP page on same origin

In the following scenario, the `article-amp.html` page requests the `data.json` file; the origins are the same.



If we examine the request, we'll find:

```
Request URL: https://example.com/data.json
Request Method: GET
AMP-Same-Origin: true
```

As this request is from the same origin, there is no `Origin` header but the custom AMP request header of `AMP-Same-Origin: true` is present. We can allow this request as it's from the same origin (`https://example.com`).

Our response headers would be:

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://example.com
```

Scenario 2: Get request from cached AMP page

In the following scenario, the `article-amp.html` page cached on the Google AMP Cache requests the `data.json` file; the origins differ.

If we examine this request, we'll find:

```
Request URL: https://example.com/data.json
Request Method: GET
Origin: https://example-com.cdn.ampproject.org
```

As this request contains an `Origin` header, we'll verify that it's from an allowed origin. We can allow this request as it's from an allowed origin.

Our response headers would be:

```
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: https://example-com.cdn.ampprojec
```

Working with cached fonts

Google AMP Cache caches AMP HTML documents, images and fonts to optimize the speed of the AMP page. While making the AMP page fast, we also want to be careful in securing the cached resources. We will be making a change in how AMP cache responds it's cached resources, typically for fonts, by respecting the origin's `Access-Control-Allow-Origin` value.

Past behavior (before October 2019)

When an AMP page was loading `https://example.com/some/font.ttf` from `@font-face src` attribute, AMP Cache will cache the font file and serve the resource as below with having the wild card `Access-Control-Allow-Origin` .

- URL `https://example-com.cdn.ampproject.org/r/s/example.com/some/font.tff`
- Access-Control-Allow-Origin: *

New behavior (October 2019 and after)

While the current implementation is permissive, this could lead to unexpected use of the fonts from cross-origin sites. In this change AMP Cache will start to respond with the exact same `Access-Control-Allow-Origin` value the origin server responds. To properly load the fonts from the cached AMP document, you will need to accept the AMP Cache origin via the header.

A sample implementation would be:

START

LEARN

Actions and events

Common element attributes

AMP HTML Specification

Enable experimental features

AMP's Layout System

Validation Workflow

AMP Caches & CORS

Debug AMP Cache issues

CORS in AMP

How AMP pages are cached

AMP Cache URL Format and Request Handling

How AMP and PWA relate to each other

DEVELOP

INTEGRATE

OPTIMIZE & MEASURE

CONTRIBUTE


```
function assertFontCors(req, res, opt_validMethods, opt_expose
  var unauthorized = 'Unauthorized Request';
  var allowedOrigins = [
    'https://example.com',
    'https://example-com.cdn.ampproject.org',
  ];
  // If allowed CORS origin
  if (allowedOrigins.indexOf(req.headers.origin) != -1) {
    res.setHeader('Access-Control-Allow-Origin', req.headers.c
  } else {
    res.statusCode = 403;
    res.end(JSON.stringify({message: unauthorized}));
    throw unauthorized;
  }
}
```

As an example, if you wanted to load /some/font.ttf in `https://example.com/amp.html`, the origin server should respond with the Access-Control-Allow-Origin header as below.

If your font file is okay to be accessible from any origin, you can respond with a wild card `Access-Control-Allow-Origin`, AMP cache will also echo that value meaning it will be responding with `Access-Control-Allow-Origin: *`. If you already have this setting, there is no need in changing anything.

We are planning to make this change around mid October 2019 and would expect every AMP publishers using self-hosted fonts to check if it's affected.

Roll out plan

- 2019-09-30: release contains more precise control over which domains this change applies to. This build should roll out over the course of this week.
- 2019-10-07: test domains will be enabled for manual testing.
- 2019-10-14: (but depending on how testing goes): the feature will be rolled out generally.

Follow the related **issue here.**

Testing CORS in AMP

When you are testing your AMP pages, make sure to include tests from the cached versions of your AMP pages.

Verify the page via the cache URL

To ensure your cached AMP page renders and functions correctly:

1. From your browser, open the URL that the AMP Cache would use to access your AMP page. You can determine the cache URL format from this **tool on AMP By Example**.

   For example:
   - URL: `https://amp.dev/documentation/guides-and-tutorials/start/create/`
   - AMP Cache URL format: `https://www-ampproject-org.cdn.ampproject.org/c/s/www.ampproject.org/docs/tutorials/create.html`

2. Open your browser's development tools and verify that there are no errors and that all resources loaded correctly.

Verify your server response headers

You can use the `curl` command to verify that your server is sending the correct HTTP response headers. In the `curl` command, provide the request URL and any custom headers you wish to add.

Syntax: `curl <request-url> -H <custom-header> - I`

Test request from same origin

In a same-origin request, the AMP system adds the custom `AMP-Same-Origin:true` header.

Here's our curl command for testing a request from `https://amp.dev` to the `examples.json` file (on the same domain):

```
curl 'https://amp.dev/static/samples/json/examples.json' -H 'A
```

The results from the command show the correct response headers (note: extra information was trimmed):

```
HTTP/2 200
access-control-allow-headers: Content-Type, Content-Length, Ac
access-control-allow-credentials: true
access-control-allow-origin: https://amp.dev
access-control-allow-methods: POST, GET, OPTIONS
```

Test request from cached AMP page

In a CORS request not from the same domain (i.e., cache), the `origin` header is part of the request.

Here's our curl command for testing a request from the cached AMP page on the Google AMP Cache to the `examples.json` file:

```
curl 'https://amp.dev/static/samples/json/examples.json' -H 'c
```

The results from the command show the correct response headers:

```
HTTP/2 200
access-control-allow-headers: Content-Type, Content-Length, Ac
access-control-allow-credentials: true
access-control-allow-origin: https://ampbyexample-com.cdn.ampp
access-control-allow-methods: POST, GET, OPTIONS
```

Functionality

Examples

Design Templates

Tools

Community

Platform and Vendor Partners

Contribute

Roadmap

Events

AMP Conf 2020

AMP Contributor Summit 2019

AMP Brand Materials

Styleguide

Logos

OpenJS Foundation

The OpenJS Foundation

OpenJS Foundation Bylaws

Trademark Policy

Trademark List