

CORS

By implementing this policy, you can enable and configure Cross Origin Resource Sharing on an API. This is a method to define access to resources outside of the originating domain. Principally, this is a security mechanism to prevent the loading of resources from unexpected domains, for instance via cross site scripting (xss) injection attacks.

General Remarks

The CORS policy works only for public APIs. If the API is private, the API Key is checked at first stage. However, the browser will not send the API Key during a preflight request. So the CORS request is blocked before it can reach the CORS policy.

API Management sets the CORS headers in the following order:

1. CORS headers from the CORS policy have the highest priority.
2. If no CORS policy has been defined, CORS headers from the external API are used.



For detailed explanations about Cross-Origin Resource Sharing (CORS) visit the [official Mozilla documentation](#).

On this Page:

- [General Remarks](#)
- [Configuration Options](#)
 - [Basic Configuration](#)
 - [Advanced Configuration](#)

Related Pages:

- [Policies](#)
 - [Attaching Policies](#)

Configuration Options

The screenshot shows the CORS configuration interface. It is divided into two main sections: Basic Configuration and Advanced Configuration. The Basic Configuration section includes four input fields: Access-Control-Allow-Origin, Access-Control-Allow-Headers, Access-Control-Expose-Headers, and Access-Control-Allow-Methods. Each field has a description of its function. The Advanced Configuration section includes two checkboxes: Terminate on CORS Error (checked) and Access-Control-Allow-Credentials (unchecked). There is also a field for Access-Control-Max-Age. At the bottom, there are buttons for Cancel, Back, and Next.

Basic Configuration

Option	Description	Possible Values	Default
Access-Control-Allow-Origin	<p>A list of origin URLs that are permitted to make CORS requests through the gateway. By default, same-origin is permitted, cross-origin is forbidden. An entry of * permits all CORS requests.</p> <p> Confirm each field input with Enter to create various list entries.</p>	a string	-
Access-Control-Allow-Headers	<p>A list of headers that can be used during the actual request. Will be provided as a response to a preflight request.</p> <p> Confirm each field input with Enter to create various list entries.</p>	a valid HTTP header	-
Access-Control-Expose-Headers	<p>Determines which non-simple headers the browser may expose during CORS.</p> <p> Confirm each field input with Enter to create various list entries.</p>	a string	-

Access-Control-Allow-Methods	Defines the HTTP methods that can be used during the actual request. Click the field to display a drop-down list and select all methods you want to use. They will be provided as a response to a preflight request.	<ul style="list-style-type: none"> • GET • HEAD • POST • PUT • DELETE • CONNECT • OPTIONS • TRACE • PATCH 	-
-------------------------------------	--	--	---

Advanced Configuration

Option	Description	Possible Values	Default
Terminate on CORS error	When enabled, any request that fails the CORS validation will be terminated with an appropriate error. When disabled, the request will still be sent to the backend API but it will be left to the browser to enforce the CORS failure. In both cases valid CORS headers will be set.	<ul style="list-style-type: none"> • enabled • disabled 	enabled
Access-Control-Allow-Credentials	This response header tells browsers whether to expose the response to the frontend JavaScript code.	<ul style="list-style-type: none"> • enabled • disabled 	disabled
Access-Control-Max-Age	Value in seconds how long a browser may cache a preflight request before it expires.	delta in seconds	0