How to Enable CORS on SAP NetWeaver Platform

February 8, 2017 | 7,300 Views |

Dong Pan

more by this author

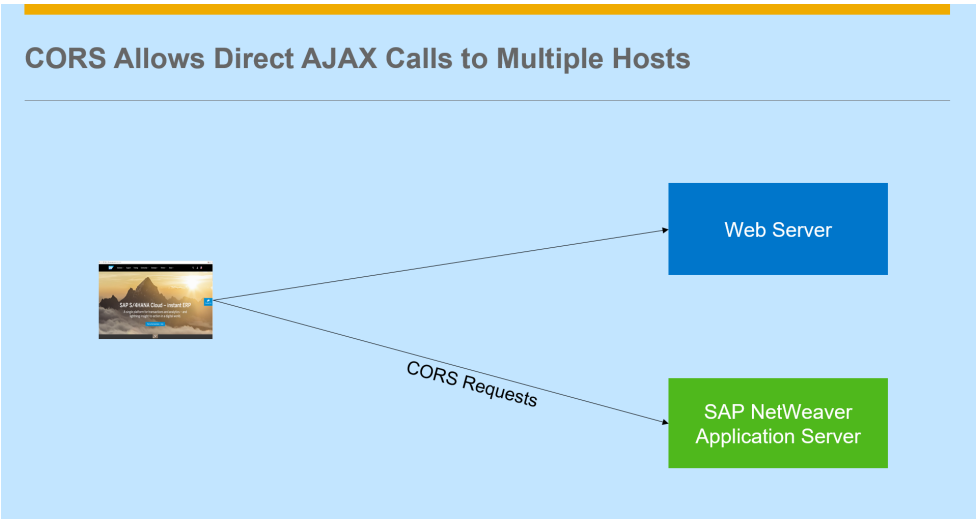
SAP NetWeaver Application Server

NW ABAP Gateway (OData) | SAP Analytics Cloud | SAP Fiori | SAPUI5 | ajax | cors | cross origin resource sharing | html5 | reverse proxy | sap web dispatcher | SAPSAC_Connectivity

Cross-Origin Resource Sharing (CORS) is a W3C specification that allows cross-domain communication from the browser. By building on top of the AJAX/XMLHttpRequest object, CORS allows developers to work in the same coding paradigm as with same-domain requests. CORS has started to play a more and more important role in today's web and cloud based applications, while our web applications are trending towards system/data integration across domains. Web application servers that support CORS make it possible for a clean architecture, without using reverse proxies or other forms of middle tier.

A majority of SAP applications reside on top of the SAP NetWeaver Application Server platform, from which many web applications retrieve data. If the data retrieval needs to happen in the web browser with AJAX calls, the traditional method to bypass web browser's Same Origin Policy is to setup a reverse proxy in front of both the web server and the SAP NetWeaver Application Server, so that they appear to the web browser as if they shared the same host name. While this may be a handy workaround, it does not only have a higher TCO for maintaining the solution, but also causes implications on SSL, authentication and Single Sign-On options.

But as a matter of fact, it is technically possible to configure SAP NetWeaver Application Server to support CORS, so that your web application landscape can be greatly simplified as below:



The trick is simple. Add a rewrite rule for NetWeaver's ICM component, so that it returns the necessary CORS headers.

First, configure the NetWeaver Application Server's Default profile, enable HTTP rewriting and point to the action/rewrite file. In the below example, the action file is the rewrite.txt file in the system profiles' folder.

```
icm/HTTP/mod_0 = PREFIX=/,FILE=$(DIR_PROFILE)/rewrite.txt
```

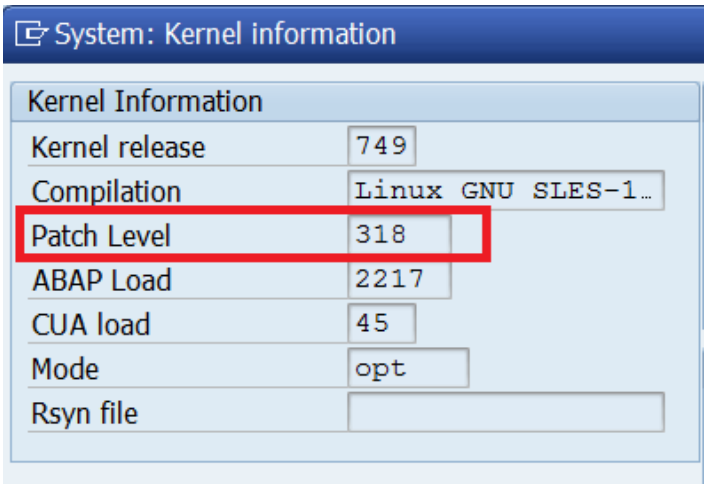
In the action file, maintain the following settings to inject the necessary CORS headers. Make sure you specify your web server's URL as the value of the Access-Control-Allow-Origin header.

```
#Author: Dong Pan, dong.pan@sap.com
if %{HEADER:ORIGIN} stricmp https://webserver1 [OR]
if %{HEADER:ORIGIN} stricmp https://webserver2 [OR]
if %{HEADER:ORIGIN} stricmp https://webserver3
  begin
        SetResponseHeader Access-Control-Allow-Origin %{HEADER:ORIGIN}
        SetResponseHeader Access-Control-Allow-Credentials true
        SetResponseHeader Access-Control-Allow-Methods "GET, POST, PUT
        SetResponseHeader Access-Control-Allow-Headers "X-Csrf-Token,
        SetResponseHeader Access-Control-Expose-Headers "x-csrf-token"
        SetResponseHeader Access-Control-Max-Age 600
  end
```

Restart the NetWeaver Application Server, and you are all set. The above settings will turn on CORS support and allow CORS requests originated from the three web servers.

*__Prerequisites__: The NetWeaver system's kernel patch level must be higher than the following in order for the above settings to take effect:

- Kernel 7.49 PL 315



Turn on CORS per application

The above settings will turn on CORS support for the entire NetWeaver application server. If you want to enable CORS for a specific application/path only, you can create the rewrite rules like below:

```
#Author: Dong Pan, dong.pan@sap.com
if %{HEADER:ORIGIN} stricmp https://webserver1 [AND]
if %{PATH} regimatch /app1/path1/*
begin
    SetResponseHeader Access-Control-Allow-Origin %{HEADER:ORIGIN}
    SetResponseHeader Access-Control-Allow-Credentials true
    SetResponseHeader Access-Control-Allow-Methods "GET, POST, PUT, OPT
    SetResponseHeader Access-Control-Allow-Headers "X-Csrf-Token, x-csr
    SetResponseHeader Access-Control-Expose-Headers "x-csrf-token"
end
```

With the above setting, only the application under the /app1/path1/* path is CORS-enabled. You can even fine-tune the allowed HTTP request methods to reflect what request methods the application supports. Pretty cool, right?

Handling Stateful Applications

Many SAP web applications implement “URL rewriting” to achieve stateful session management. In this case, the web browser/JavaScript is supposed to add a long session ID dynamically to the URL, for example, the request to /sap/bw/ina would change to something like below after authentication:

/sap(cz1TSUQlM2FBTk9OJTNhQlc3NTBfQjc1XzAwJTNhMFR2WnAxdzRoanVuU2gzODRHbk9ydUF5aTRDYzd4WWZrdFEzMlBjRi1BVFQ=)/bw/ina/GetResponse

The part in red is the dynamic session ID that keeps changing from request to request in the entire dialog between the Web Browser and NetWeaver application server. This is done via two HTTP response headers:

1. sap-rewriteurl
2. sap-url-session-id

The JavaScript in the web browser needs to be able to pick up the values of the above HTTP response headers, in order to construct the dynamic URLs containing the session ID according to the NetWeaver applications server’s

instruction. Since these headers come from a CORS response, we need to add them to the Access-Control-Expose-Headers list.

We also need to fine-tune the condition expression so that it continues to match the application's path, even when the session ID is injected to it. We will leverage the powerful regular expression support here. Below is a sample rules file:

```
#Author: Dong Pan, dong.pan@sap.com
if %{HEADER:ORIGIN} stricmp https://webserver1 [OR]
if %{HEADER:ORIGIN} stricmp https://webserver2 [AND]
if %{PATH} regimatch /sap(\(.+\))*/app1/path1/*
  begin
    SetResponseHeader Access-Control-Allow-Origin %{HEADER:ORIGIN}
    SetResponseHeader Access-Control-Allow-Credentials true
    SetResponseHeader Access-Control-Allow-Methods "GET, POST, PUT, OPT
    SetResponseHeader Access-Control-Allow-Headers "X-Csrf-Token, x-csr
    SetResponseHeader Access-Control-Expose-Headers "x-csrf-token, sap-
  end
```

Take Care of Preflight CORS Requests

The above settings will work well with Simple CORS requests only. A Simple CORS request is a CORS request that makes use of HTTP request methods GET, POST, HEAD only, and carries a limited set of HTTP headers. If a CORS request uses any other HTTP request method, such as DELETE or PUT, or if it carries HTTP headers outside of the limited set headers mentioned above, the request must be preceded with a Preflight Request, which is in essence an HTTP OPTIONS request with certain CORS-specific headers. Unfortunately the Preflight CORS request will fail with the above settings. If you would like to get a deeper understanding of Simple CORS requests and Preflight Requests, see more details here.

But why? We have already configured ICM to issue all the CORS response headers, why is the Preflight Request failing? The issue lies in how a Preflight Request is constructed. According to the CORS specification, the Preflight Request must NOT carry any user credential. As most applications on NetWeaver require user authentication, the Preflight Request will get an "HTTP 401 Unauthorized" error message, thus failing the request.

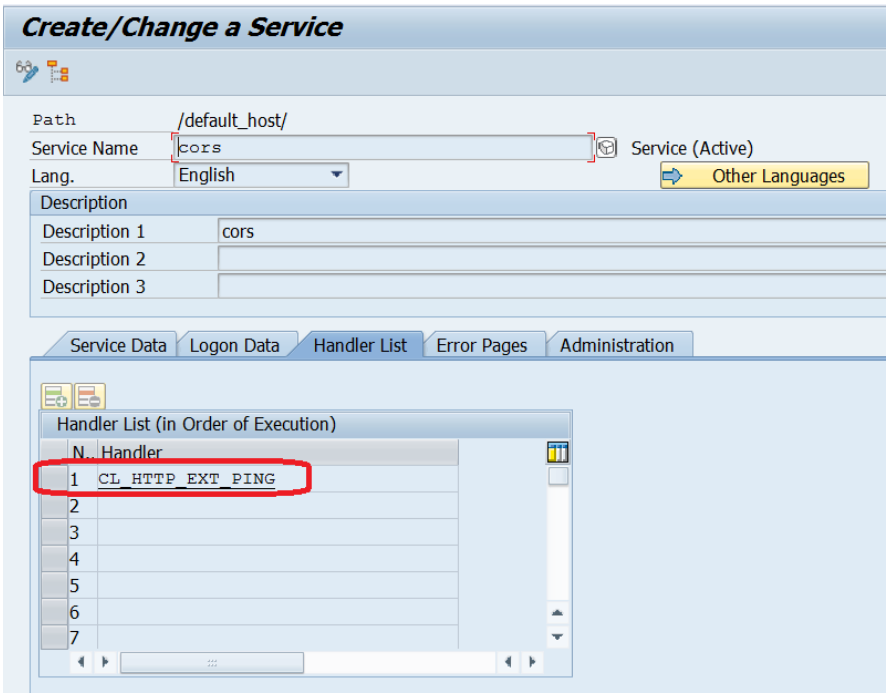
There are multiple ways to address the issue, but the cleanest solution that solves the issue without introducing any extra infrastructure components looks

like below:

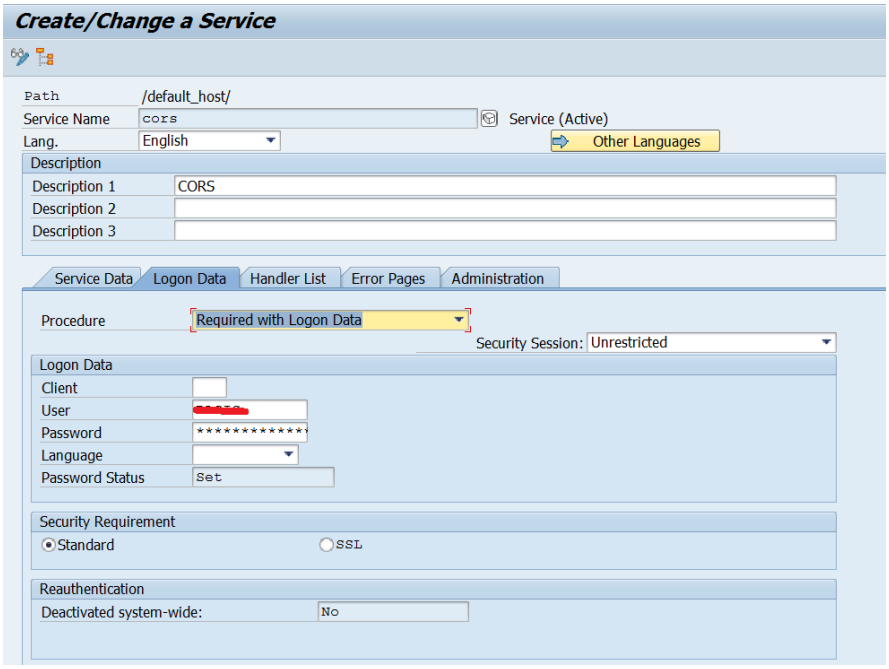
1. Create a dummy node with SICF that allows anonymous access
2. In ICM rewrite rules, redirect Preflight Requests to the dummy node that allows anonymous access
3. Make sure that the originally-requested URL is preserved, so that CORS response headers are issued according to the application.

In this way, the Preflight Request will get the desired CORS response headers based on the requested application without being turned down due to authentication errors.

For Step 1, you can create a dummy node in SICF and name it “cors’ for example. Serve the node with the handler **CL_HTTP_EXT_PING**, which exists on any NetWeaver ABAP system and produces minimum amount of traffic.



Make sure anonymous access is allowed for this node.



For step 2&3, we will redirect the Preflight OPTIONS request to the dummy node created above, preserve the originally-requested path in an HTTP header, and issue the CORS response headers accordingly. Here goes a sample rewirte rules file that enables CORS for the BW Info Access (InA) service:

```
#Author: Dong Pan, dong.pan@sap.com
SetHeader OriginalPath ""

if %{HEADER:ORIGIN} stricmp https://webserver1 [OR]
if %{HEADER:ORIGIN} stricmp https://webserver2 [AND]
if %{REQUEST_METHOD} stricmp OPTIONS
  begin
    RegRewriteUrl ^/.* /cors?%{QUERY_STRING} [noescape]
    SetHeader OriginalPath %{PATH}?%{QUERY_STRING}
  end

if %{HEADER:ORIGIN} stricmp https://webserver1 [OR]
if %{HEADER:ORIGIN} stricmp https://webserver2 [AND]
if %{HEADER:OriginalPath} regimatch /sap(\(.+\))*/bw/ina/* [AND]
if %{PATH} regmatch /cors
  begin
    SetResponseHeader Access-Control-Max-Age 600
    SetResponseHeader Access-Control-Allow-Origin %{HEADER:ORIGIN}
    SetResponseHeader Access-Control-Allow-Credentials true
    SetResponseHeader Access-Control-Allow-Methods "GET, POST, PUT, OPT
    SetResponseHeader Access-Control-Allow-Headers "X-Csrf-Token, x-csr
    SetResponseHeader vary "Origin"
    RemoveResponseHeader set-cookie
    RemoveResponseHeader expires
  end

if %{HEADER:ORIGIN} stricmp https://webserver1 [OR]
if %{HEADER:ORIGIN} stricmp https:///webserver2 [AND]
if %{PATH} regimatch /sap(\(.+\))*/bw/ina/*
  begin
```

```
    SetResponseHeader Access-Control-Allow-Origin %{HEADER:ORIGIN}
    SetResponseHeader Access-Control-Allow-Credentials true
    SetResponseHeader Access-Control-Expose-Headers "x-csrf-token, sap-
    SetResponseHeader vary "Origin"
  end
```

Restart the NetWeaver system, and it is now able to serve preflighted CORS requests as well. With some simple configuration steps, your NetWeaver system is now able to provide full support for CORS! Hooray!!! Isn't that cool?!

On a side note: if you are sending Preflighted CORS requests to a NetWeaver Java system, you can skip step 1. As there are a number of anonymous HTML pages on a vanilla NetWeaver Java system, e.g. the home page, you can make use of any of them, or create a dummy anonymous page that produces minimum traffic.

Turn on CORS on SAP Web Dispatcher

In a large deployment of SAP NetWeaver landscape, it is often the case that there are multiple server nodes and a load balancer such as SAP Web Dispatcher sits in front of the multiple server nodes. In this case, you can turn on CORS support on the SAP Web Dispatcher instead of on each and every server node. With the latest version of Web Dispatcher (7.49 PL112 or above), you can use exactly the same rewrite rules as above.

Start to enjoy the beauty and simplicity of CORS!

[Update 2017.10] With NetWeaver AS ABAP 7.52, SAP provides built-in native support for CORS. See details here. You can continue to use the method in this blog post to enable CORS if your NetWeaver AS ABAP system is of a version lower than 7.52, or if you are using NetWeaver AS Java.

Alert Moderator

15 Comments

You must be Logged on to comment or reply to a post.

Achim Braemer

July 11, 2017 at 5:19 pm

Great article. The method described here should work fine for simple CORS requests, i.e. ones that do not require a pre-flight request. I assume it does not work for pre-flight requests, though.

The good news is: With AS ABAP release 7.52 we will ship real CORS support built into the server.

Regards, Achim
(Product owner for ABAP connectivity)

Dong Pan Post author

July 12, 2017 at 11:28 pm

Thanks Achim for your comments! Simple CORS requests, plus the services that can be globally configured about its allowed HTTP request methods, should work with the above settings. I believe this should work for the BW InA services, for example.

I look forward to the full-fledged CORS support in 7.52, and I also hope that this important feature can be downported to lower versions of AS ABAP.

Dong Pan Post author

October 16, 2017 at 10:31 pm

Hi Achim,

I've updated the blog post with the CORS feature in NetWeaver AS ABAP 7.52. At the same time, I've also enhanced the blog post with updated configuration so that it covers preflighted CORS requests too, so that lower NetWeaver versions can benefit from full CORS support as well.

Cheers,

Dong

Roland Kramer

November 17, 2017 at 2:06 pm

Hello Achim
Thank you for your Update.

Unfortunately with BW/4 we only have SAP Application Server 7.50 with SAP_ABA 7.5A available

Is it anyway possible to activate CORS successfully?
Can some of the functionality be down ported to lower SAP_BASIS Versions?

See also my current finding in the Document – SAP First Guidance – Implement SAP BW/4HANA in the Azure Cloud

Best Regards Roland Kramer

Achim Braemer

December 6, 2017 at 9:42 am

We are currently in the process of downporting CORS to 7.50, it should become available with 7.50 SP 12 (delivery in June 2018). (Usual disclaimers regarding future shipments apply).

Regards, Achim

Roland Kramer

December 6, 2017 at 3:52 pm

Hello Achim
in the meantime we found a solution also to activate the CORS solution also on BW/4HANA with SAP BASIS 7.50

The procedure will be published also in the Document – SAP First Guidance – Implement SAP BW/4HANA in the Azure Cloud

Best Regards Roland
PM BW/EDW, SAP SE

Patrick Weber

July 24, 2017 at 12:43 pm

Hallo,

thanks for sharing this.

I have an aspx page which needs data from SAP. Therefore I created an oDATA service on my SAP Gateway and tried to call this service in ASPX as Ajax request.

I had a lot of trouble related to this CORS stuff, your article helped me to overcome one problem. On HTTP Response Header, all access-control-* fields were missing.
I tried to add them on my abap code with no success. Then I found your article. Now the Response Header looks good.
But I still have problems with preflight check. I'm trying to GET data from Service. As I see in Fiddler, the initial request is Method OPTIONS, which always response as 401 unauthorized. If I call the service URL in browser, everything works great.

Do you know how to use authentication for OPTIONS method (is it possible at all?)? Do you know how to avoid OPTIONS Request and send GET?

Patrick Weber

July 25, 2017 at 9:20 am

Hello,

it's me again. As I learned meanwhile, OPTIONS will never use Authorization Headers. Do you know how to configure SAP Gateway to accept OPTIONS Method? I always get 401 unauthorized error.

Dong Pan Post author

October 13, 2017 at 12:00 am

Hi Patrick,

Sorry for the delay with my reply. I have updated the blog post, and it now covers Preflighted CORS requests as well. I hope it is not too late for you.

Thanks,

Dong

Deepu Sasidharan

November 7, 2017 at 8:58 pm

Great blog Dong!

Do you know which version of BW4HANA supports AS ABAP 7.52?

Deepu

Markus Belzer

November 13, 2017 at 4:45 pm

Hello,

I have a question concerning the prereqs:

What is the difference between “Kernel PL” and “disp+work package PL” ?

Thanks in advance,

Markus

Dong Pan Post author

November 15, 2017 at 9:06 pm

The disp+work package updates a portion of the kernel, which may be of a higher PL than the overall kernel PL.

Pablo Silvestro

April 5, 2018 at 5:09 pm

just a question, as per my understanding, CORS works out of the box just in case the external application can reach the NW application which have CORS support.
how can I handle support over SAC or other external APP outside my network domain?
I mean, should I open a firewall rule, create NAT or something?

Dong Pan Post author

April 23, 2018 at 9:26 pm

No, you don’t need to create any firewall rules. CORS allows web browsers to retrieve data with AJAX calls from multiple sites, so this is really a browser-side mechanism. The only rule you would need is the whitelist on the datasource site, which is essentially what the script in the blog post does.

Vadim Zaripov

May 10, 2018 at 9:41 am

Hi Dong,

I am still struggling with the Live connection from SAC with the following error on preflight:

Response for preflight has invalid HTTP status code 401.

What could be the issue?