Copyright (C) 2019 Gaman SP z o.o

<https://gaman-gt.com>

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Gaman

CORS Support for PCCE/UCCE Configuration API

# Table of Contents

[Table of Contents 2](#_Toc56144046)

[Introduction 3](#_Toc56144047)

[Overview 4](#_Toc56144048)

[Procedure 5](#_Toc56144049)

[Appendix A – Sample CORS Configuration File 8](#_Toc56144050)

# Introduction

Cisco Configuration API is delivered on PCCE/UCCE Logger and Distributor machines. The API is designed to be called directly – from any 3rd party application but not to be called as a part of the 3rd party web application. Here are some real-life scenarios where this integration can be used:

* Finesse gadget that needs to call the PCCE/UCCE API to pull additional agent details. Those details can be used for additional 3rd party integrations ex. CRM systems, Recorders, Statistics server.
* Finesse gadget that delivers reskilling feature published to call center supervisors. Used case scenario – automated emergency reskilling performed on button click.
* Finesse gadget that allows business administrator/supervisor to change routing in case of emergency situations in call center (ex. CRM application crash, IVR crash, etc.). The routing is change based on the dialed number call type assignment.

Looking at those examples - Finesse Gadgets – we can see that the API calls can only be made using JavaScript AJAX technology. Modern browsers are equipped with the technology that requires a specific behavior on server side to allow this type of the communication. Most problems that developer is facing when creating Finesse gadgets are related to CORS.

The following document has been prepared to show how to resolve the CORS problems in a system way when using PCCE/UCCE Configuration API by 3rd party web applications (ex. Finesse gadgets).

# Overview

To enable CORS support we need to get a general idea how the requests are processed on Loggers or Distributor servers on PCCE/UCCE platform. Below diagram shows how the requests are processed.



1. Every request made to the server is processed by IIS (Internet Information Services)
2. Then the request is forwarded by ISAPI Redirector to Tomcat server
3. Application deployed on Tomcat server performs the request treatment

Looking at this architecture you can see that we have two possibilities to add CORS support:

* On Tomcat server – enable native support for CORS on Tomcat
* On IIS server before ISAPI Redirector forwards the call

Which to choose – it is all up to you, but we think that it is better to implement it on IIS, why?

* Performance – if you look at the presented diagram you will notice that the IIS server acts as a proxy server and is not involved in request processing. The load on this server is lower than the Tomcat. This means that enabling CORS should not impact on application performance.
* Selectivity – CORS configuration on IIS can be done on multiple levels. You can enable it for all request, but you can do it in more selective way – enable it on web application level. This means that you can enable CORS support only for a specific application.
* Configuration – when you configure CORS on Tomcat Server, each time you change the configuration you need to reboot the server, so the new settings are uploaded. In PCCE/UCCE tomcat server hosts more than just Configuration API (ex. UCCE Web Admin application, Status API’s, etc.). This means that when you restart the server, you loose other features which forces you to plan Maintenance window for the environment. IIS allows to change the configuration dynamically and this operation does not require any infrastructure downtime. You can change it without losing PCCE/UCCE features or functionality.

So, the last question is how to enable the CORS on IIS. On the Internet you will find multiple answers how to approach this problem – mostly using workarounds that add CORS HTTP Headers to each response. This may work in some cases, but the main goal is to have a complex solution that supports all CORS flows including authorization requests. Having that in mind we propose to use a IIS server add-on – IIS CORS module/extension. Here is the link to the module website: [link](https://www.iis.net/downloads/microsoft/iis-cors-module). If you look at this module description, please look at this part:

*“The IIS CORS module helps with setting appropriate response headers and responding to preflight requests. Once installed, the IIS CORS module is configured via a site or application web.config and has its own cors configuration section within system.webserver.”*

This is exactly wat is needed to enable CORS on Logger or Distributor servers in PCCE/UCCE environment.

Below diagram shows the infrastructure once the proposed solution will be implemented on Loggers or Distributor servers.



As you can see the only difference compared to original design is the IIS CORS module that is needed to treat CORS requests. And that is all. The rest is the configuration applied on the server. The next chapter will describe the installation and configuration process that will enable CORS Support for PCCE/UCCE Configuration API.

# Procedure

Below you will find step-by-step procedure that enables CORS support on PCCE/UCCE Logger and Distributors servers. It describes the installation process of the IIS CORS module and configuration steps (post installation) that will enable CORS support.

**Please note**

Please execute the procedure on all UCCE servers (Loggers or Distributors) that will be used to treat Configuration API requests.

**Please note**

The sample CORS configuration file (that can be used as a reference) can be found in the chapter: [Appendix A - Sample CORS Configuration File](#_Appendix_A_–).

1. Download the newest version of the IIS CORS Module from the following site:

<https://www.iis.net/downloads/microsoft/iis-cors-module>

The module adds some features design to handle CORS requests on the server level.

1. Copy the cors.exe file on the server and run the installer – **Web Platform Installer**.



1. After short period of time the system will present detailed info about the installed package. Please click  button to begin installation process.



1. The installer will ask to install prerequisites. Please click  button to continue the installation process.



1. The installation process should be finished in less than a minute. Once done installer will present the summary page. Click  button.



1. Close the **Web Platform Installer** by clicking  button.



1. On the server navigate to the following location: **C:\icm\tomcat\bin\i386** and in this directory search for the **web.config** file and edit it.
2. In the file search for the tag **<system.webServer>.** Once found add the following section underneath it:

**<cors enabled="true" failUnlistedOrigins="true">**

**</cors>**

1. For every server that will pull the data from UCCE Configuration API add the following section to **<cors>** tag. Before you paste the code, replace the **<Server\_FQDN>** tag with the FQDN name of the Finesse server.

**<add origin=https://<Server\_FQDN> allowed="true" allowCredentials="true" maxAge="120">**

**<allowHeaders allowAllRequestedHeaders="true">**

**</allowHeaders>**

**</add>**

1. Once done save the file.
2. Repeat the steps starting from step 2 for the next UCCE server

# Appendix A – Sample CORS Configuration File

Below you will find a sample configuration file that shows the configuration of the CORS on Distributor server that was used to enable integration for Finesse gadget. The FQDN of the Finesse server is: **lab12finessea.gaman.local**. Please use it as a reference to create your individual configuration file.

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

    <system.webServer>

        <cors enabled="true" failUnlistedOrigins="true">

            <add origin="https://lab12finessea.gaman.local:8445" allowed="true" allowCredentials="true" maxAge="120">

                 <allowHeaders allowAllRequestedHeaders="true">

                 </allowHeaders>

            </add>

        </cors>

        <httpErrors errorMode="DetailedLocalOnly" existingResponse="PassThrough" />

        <directoryBrowse enabled="false" showFlags="Date, Time, Size, Extension" />

    </system.webServer>

</configuration>