

# OnPREMISE API REST

## Introduction

All OnPremise series software uses a REST API to deploy all its functionality and the JSON format to receive and deliver the data models it exchanges.

All calls and data models used by the API are self-documented via Swagger 2.0 by the same HTTP server that deploys the back-end. From the same local network of the device with the software installed, such documentation can be accessed from any browser using the URL:

[http://local\\_ip\\_device/swagger](http://local_ip_device/swagger) .

The Swagger interface not only documents each component, but can also be used to test those components. From the same web site you can download the doc.json file to be imported into Postman for detailed testing by any front-end developer.

The screenshot shows the Swagger UI interface for the OnPremise SRT Edge API. At the top, there's a navigation bar with the Swagger logo, the URL '/swagger/doc.json', and a green 'Explore' button. Below the header, the title 'OnPremise SRT Edge API <sup>1.0</sup>' is displayed, along with the base URL '[ Base URL: 192.168.1.103:80/api/v1 ] /swagger/doc.json'. A brief description follows: 'This is the API Service to control OnPremise SRT Edge'. Below this, links for 'Terms of service', 'Contact API Support', and 'Apache 2.0' are provided. To the right of the main content area is a green 'Authorize' button with a lock icon. The main content area is titled 'network' and lists several API endpoints with their methods, URLs, descriptions, and authentication requirements (indicated by padlock icons):

- GET /ancillary** Get the Ancillary config
- PUT /ancillary** Update the Ancillary config
- GET /ddns** Get the DDNS config
- PUT /ddns** Update the DDNS config
- GET /network** Get the Network config
- PUT /network** Update the Network config
- GET /timezones** Get all possible times zones in this OS

API calls that are locked require the use of authentication for proper operation.

## Authentication process

Before we start using any authentication-protected API call, we have to get the apiKey, which we will use in each call, in the header of each request, like here:

Authorization: apiKey

To do this, we will first use the unprotected call in the **auth** section: POST /login.

The screenshot shows the 'auth' section of an API documentation. It contains two endpoints: a green 'POST /login' button labeled 'Authentication entry point' and a red 'DELETE /logout' button labeled 'Authentication exit point'. A lock icon is also present next to the delete endpoint.

Using the api.Login model, we pass the user name and password that are stored at that time (defaults: admin/admin).

The screenshot shows a 'Curl' command to log in:

```
curl -X POST "http://192.168.1.103:80/api/v1/login" -H "accept: application/json" -H "Content-Type: application/json" -d "{\"pass\": \"admin\", \"user\": \"admin\"}"
```

The screenshot shows the 'Server response' details for a 200 status code. It includes a 'Code' section with a 'Details' link and a 'Response body' section containing a JSON object:

```
{  
  "apikey": "SuGxqJY0XCzptuc3c1odevYqg8"  
}
```

A 'Download' button is available for the response body. Below it, the 'Response headers' section lists:

```
access-control-allow-credentials: true  
access-control-allow-headers: *  
access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE, HEAD  
access-control-allow-origin: *  
access-control-expose-headers: *  
authorization: SuGxqJY0XCzptuc3c1odevYqg8  
content-length: 40  
content-type: application/json  
date: Fri, 17 Dec 2021 02:29:29 GMT  
server: TodoStreaming WebServer
```

In this example, we receive as a response inside the session.APIKey model, the value of the apiKey SuGxqJY0XCzptuc3c1odevYqg8, which we will use in the locked calls, passing them in the header:  
Authorization: SuGxqJY0XCzptuc3c1odevYqg8

The generic response model is api.HTTPResponse with this composition:

The screenshot shows the generic response model composition, which is defined as:

```
{  
  "message": "string",  
  "type": "string"  
}
```

Let's see as an example of an authenticated call, the call in the **network** section GET /network.

Curl

```
curl -X GET "http://192.168.1.103:80/api/v1/network" -H "accept: application/json" -H "Authorization: SuGxqJY0XCzptuc3c1odevYqg8"
```

Which returns the following response in the body (metal.Network model) and in the headers:

Server response

Code	Details
200	<p>Response body</p> <pre>{     "interface": [         {             "index": 2,             "name": "enp1s0",             "mac": "84:47:09:08:4b:36",             "ipv4": "192.168.1.103/24",             "ipv6": "",             "gateway4": "192.168.1.1",             "gateway6": "",             "mode4": "dhcp",             "mode6": ""         }     ],     "dns1": "212.230.135.2",     "dns2": "212.230.135.1",     "dns3": "",     "mdnsif": "enp1s0",     "monif": "enp1s0" }</pre> <p>Download</p> <p>Response headers</p> <pre>access-control-allow-credentials: true access-control-allow-headers: * access-control-allow-methods: POST, GET, OPTIONS, PUT, DELETE, HEAD access-control-allow-origin: * access-control-expose-headers: * authorization: SuGxqJY0XCzptuc3c1odevYqg8 content-length: 262 content-type: application/json date: Fri, 17 Dec 2021 02:31:16 GMT server: TodoStreaming WebServer</pre>

Every session has a duration of 1440 seconds (24 minutes), which is updated in every new API call. If we are that time making no calls, that update will timeout, the session will expire automatically and we will need a new apiKey, with the same process described above.

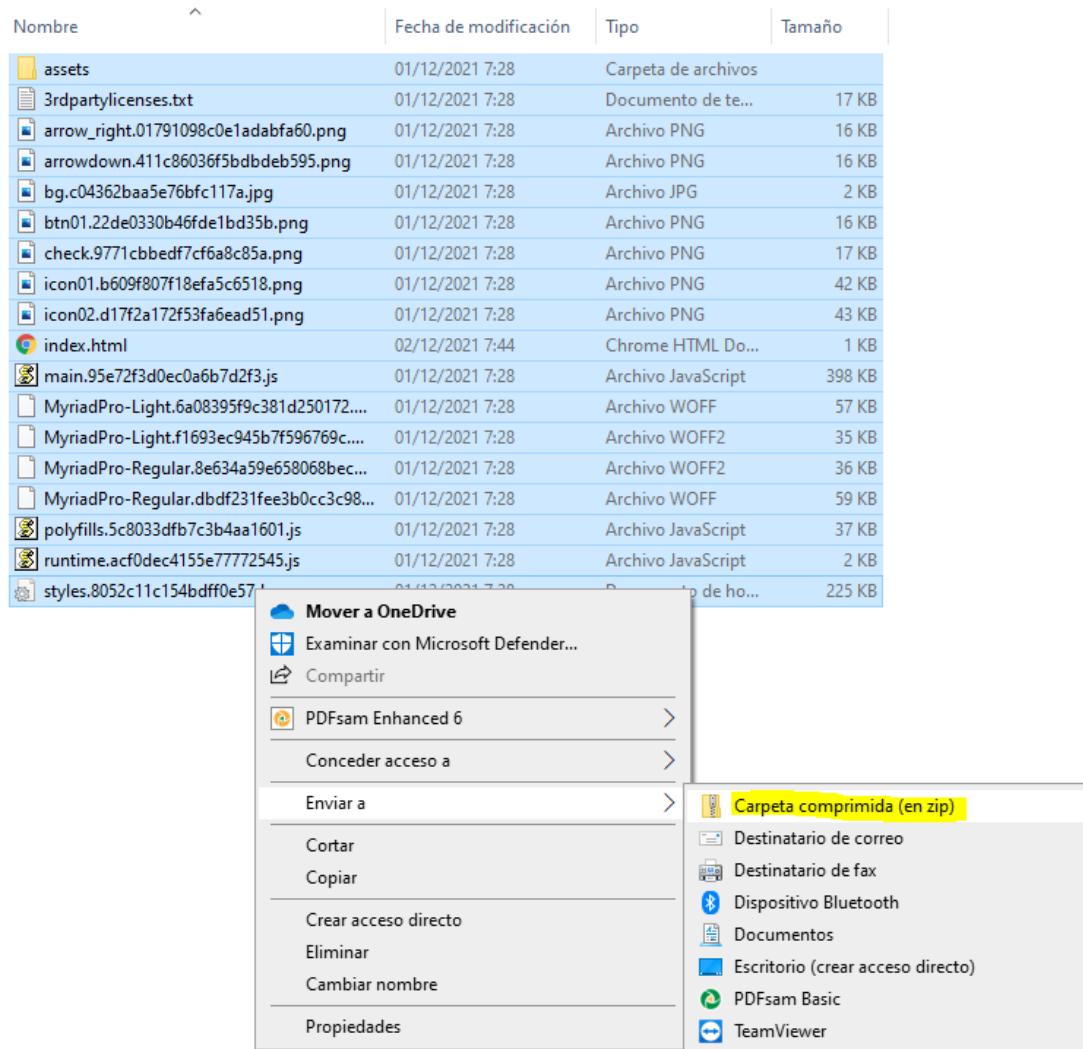
We can also voluntarily logout by using the call DELETE /logout .

The HTTP server is CORS-compliant, so you can create a front-end application that calls the full OnPremise REST API from the outside.

## Install your own panel on your device

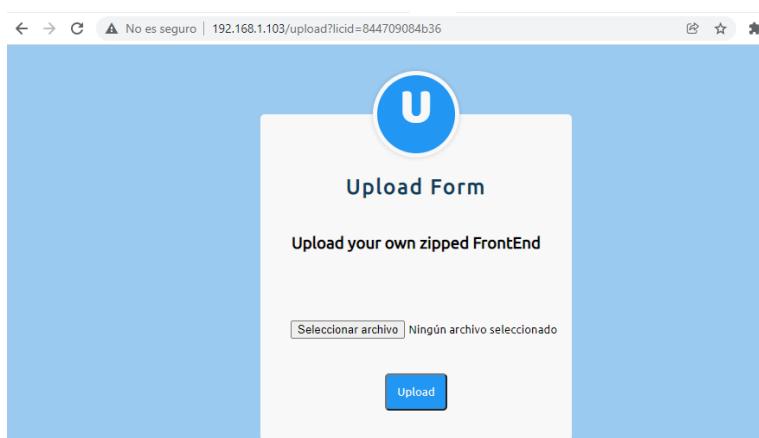
Apart from being able to use the API from external applications that call the device remotely, you can also create your own front-end panel with your own design and branding. In fact you can test it beforehand remotely, and if everything is ok, just upload it to the device as follows:

- 1.- We compress in a zip file, the root of the front-end.

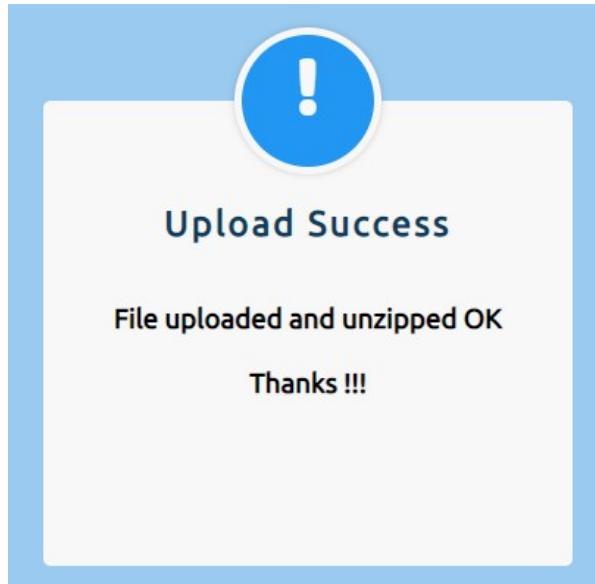


- 2.- We access the Upload Form of the device that is in the URL:

[http://ip\\_local\\_device/upload?licid=licid\\_device](http://ip_local_device/upload?licid=licid_device)



Select the zip file that we created in step 1, and click Upload.  
If everything goes fine, a window like this one will be shown:



3.- We can now load the new app again in our browser, refreshing the cache using Ctrl+F5 several times..

The original panel installed inside OnPremise software, delivers the most basic and generic functionalities in the API, and has been created with Angular JS in typescript. If you are interested in creating your own panel and do not have front-end developer staff, you can contact us for quotation.