Guide for configuring and working with the Emulator integration module

ACFA PSIM 1.1

Last update 02/09/2024

Table of Contents

1 Introduction into the Guide for configuring and working with the Emulator integration module

On the page:

1.1 Purpose of the document

The *Guide for configuring and working with the Emulator integration module* is a reference and information manual and is intended for configuration specialists and operators of the FSA/ACS and PID. This integration module is a part of *ACFA PSIM*.

The Guide has the following information:

1. General information about the *Emulator* integration module.
2. Configuring the *Emulator* integration module.
3. Working with the *Emulator* integration module.

1.2 General information about the Emulator integration module

The *Emulator* program module is a part of *ACFA PSIM* and allows you to perform the following actions:

- simulate the modes and states of the FSA/ACS and PID hardware;
- simulate the actions of the FSA/ACS and PID operator;
- simulate triggering of hardware and/or change of hardware state to which the operator must respond.

2 Supported hardware and licensing of the Emulator module

Module licensing

Per one sensor, or per one reader, or per one fence.

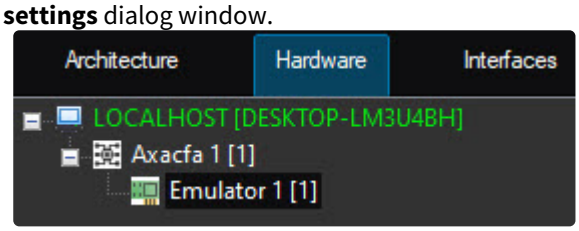
3 Configuring the Emulator module

3.1 Configuring the parent object of the Emulator module

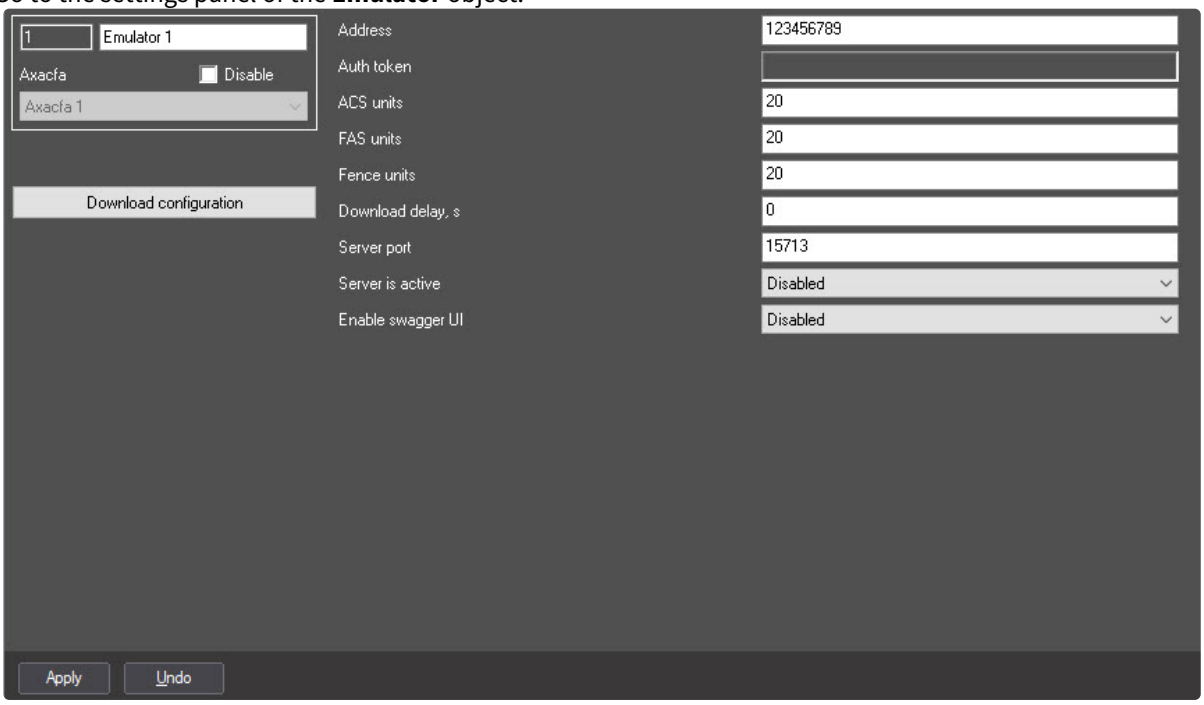
To work with the *Emulator* integration module, you must install and configure the *AxACFA* feature. For more details, see Connecting and configuring the AxACFA feature.

To configure the parent object of the *Emulator* module, do the following:

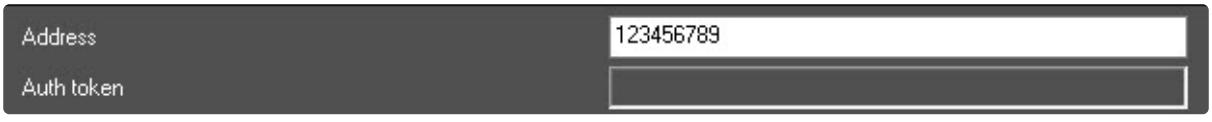
1. Create the **Emulator** parent object on the basis of the **Axacfa** object on the **Hardware** tab of the **System settings** dialog window.



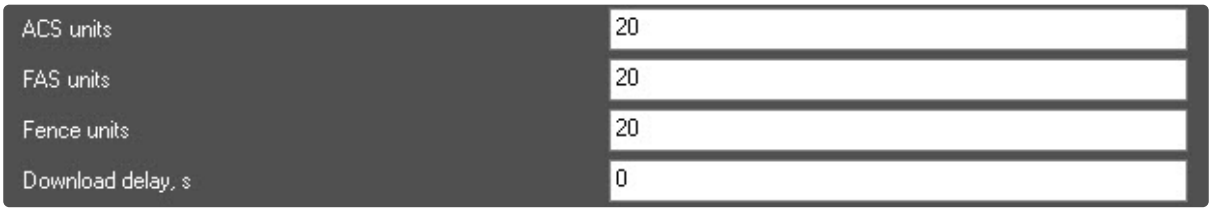
2. Go to the settings panel of the **Emulator** object.



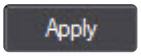
3. The **Address** field automatically contains the local IP address of the computer that you can change if necessary. In future versions, it will be possible to specify the IP address of the computer for remote connection and control of the emulator.

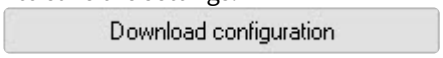


4. In the **Auth token** field, specify a token for logging into Swagger software—a framework that allows you to automatically describe APIs based on its code.
5. In the **ACS units** field, specify the number of ACS (Access Control System) objects you want to create. The default value is 20. One object consists of one lock and two readers (child objects). You cannot change the number of child readers of one lock.

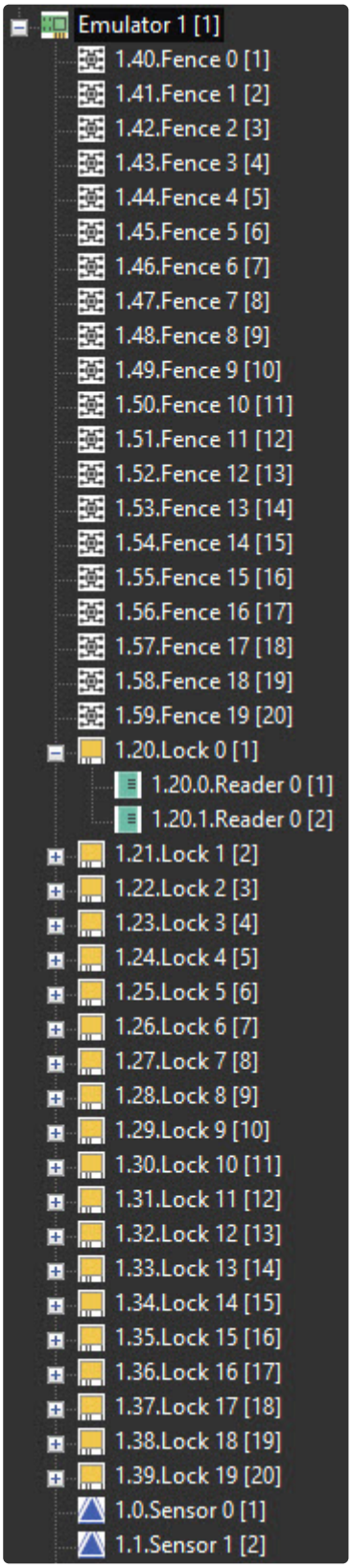


6. In the **FSA units** field, specify the number of FSA (Fire and Security Alarm) of the **Sensor** object. The default value is 20.
7. In the **Fence units** field, specify the number of PID (Perimeter Intrusion Detection System) of the **Fence** object. The default value is 20.
8. In the **Download delay, s** field, specify the interval in seconds between the creation of two objects. The default value is 0. We don't recommend changing it.
9. In the **Server port** field, specify the port number via which Swagger software will be available.

10. From the **Server is active** drop-down list, select **Enabled** to activate Swagger software. By default, Swagger is disabled (the **Disabled** value is selected).
11. From the **Enable swagger UI** drop-down list, select **Enabled** to activate Swagger software in the browser. By default, Swagger is disabled in the browser (the **Disabled** value is selected).
12. Click the **Apply** 

button to save the settings.
13. Click the **Download configuration** 

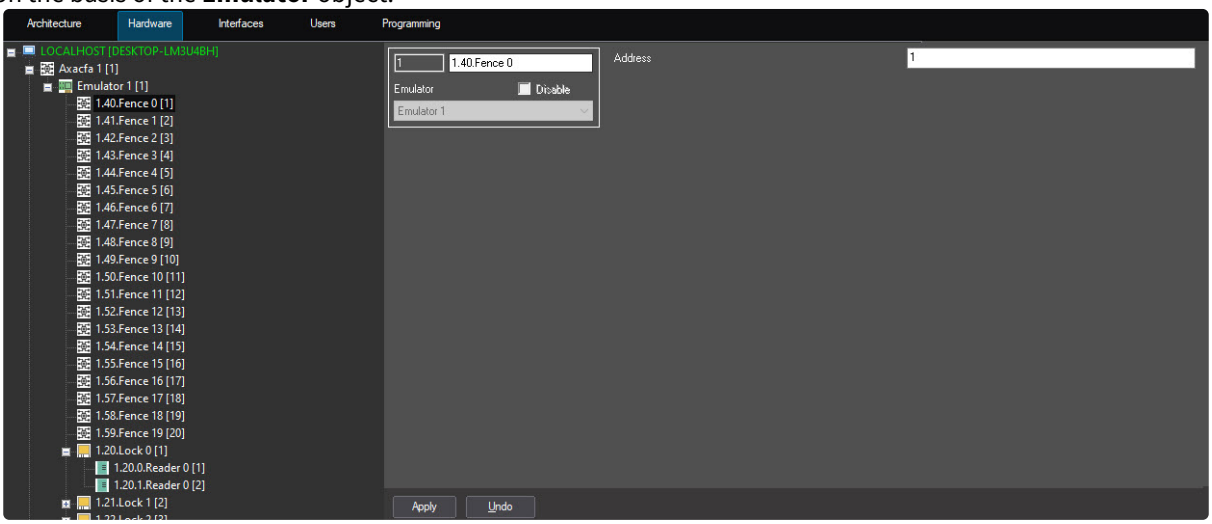
button to automatically build the hardware tree according to the configuration specified in the previous steps:


Emulator 1 [1]
1.40.Fence 0 [1]
1.41.Fence 1 [2]
1.42.Fence 2 [3]
1.43.Fence 3 [4]
1.44.Fence 4 [5]
1.45.Fence 5 [6]
1.46.Fence 6 [7]
1.47.Fence 7 [8]
1.48.Fence 8 [9]
1.49.Fence 9 [10]
1.50.Fence 10 [11]
1.51.Fence 11 [12]
1.52.Fence 12 [13]
1.53.Fence 13 [14]
1.54.Fence 14 [15]
1.55.Fence 15 [16]
1.56.Fence 16 [17]
1.57.Fence 17 [18]
1.58.Fence 18 [19]
1.59.Fence 19 [20]
1.20.Lock 0 [1]
1.20.0.Reader 0 [1]
1.20.1.Reader 0 [2]
1.21.Lock 1 [2]
1.22.Lock 2 [3]
1.23.Lock 3 [4]
1.24.Lock 4 [5]
1.25.Lock 5 [6]
1.26.Lock 6 [7]
1.27.Lock 7 [8]
1.28.Lock 8 [9]
1.29.Lock 9 [10]
1.30.Lock 10 [11]
1.31.Lock 11 [12]
1.32.Lock 12 [13]
1.33.Lock 13 [14]
1.34.Lock 14 [15]
1.35.Lock 15 [16]
1.36.Lock 16 [17]
1.37.Lock 17 [18]
1.38.Lock 18 [19]
1.39.Lock 19 [20]
1.0.Sensor 0 [1]
1.1.Sensor 1 [2]

3.2 Configuring fence, lock and sensor of the Emulator module

Configuration of the fence, lock and sensor will be shown by the example of the **Fence** object. You can configure the **Lock** and **Sensor** objects of the *Emulator* module in the same way.

1. Go to the settings panel of the **Fence** object that is created automatically when you download configuration on the basis of the **Emulator** object.

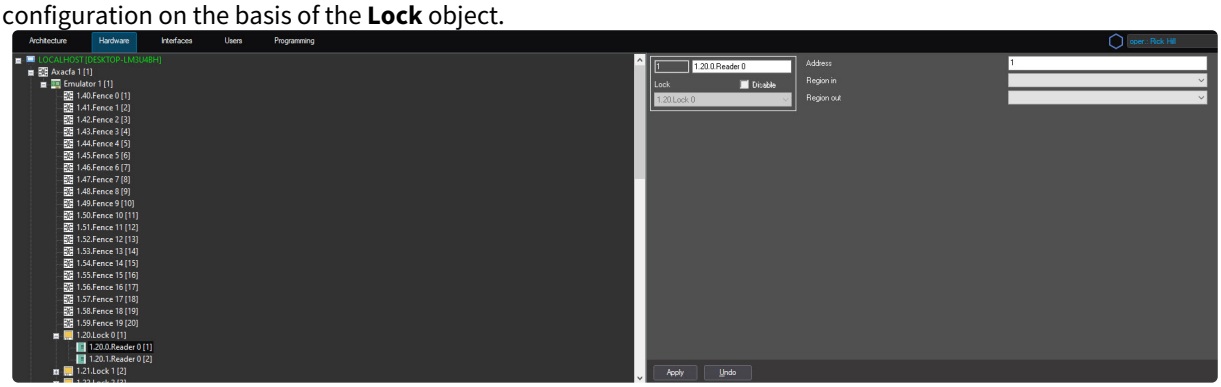


2. The **Address** field automatically contains the address of the device that you can change if necessary.
3. Click the **Apply** button to save the settings.

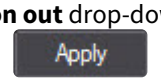
3.3 Configuring the reader of the Emulator module

To configure the reader of the *Emulator* module, do the following:

1. Go to the settings panel of the **Reader** object that is created automatically when you download configuration on the basis of the **Lock** object.



2. The **Address** field automatically contains the address of the device that you can change if necessary.

3. From the **Region in** drop-down list, select the area on the reader exit side.
4. From the **Region out** drop-down list, select the area on the reader entry side.
5. Click the **Apply**  button to save the settings.

4 Working with the Emulator module

4.1 General information about working with the Emulator module

The following interface objects are used to work with the *Emulator* module:

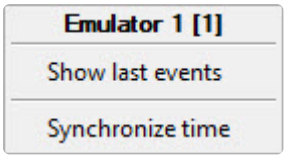
1. **Map**.
2. **Event Viewer**.

For the information on configuring these interface objects, see the *Axxon PSIM* Administrator's Guide.

For the information on working with these interface objects, see the *Axxon PSIM* Operator's Guide.

4.2 Working with the parent object of the Emulator module

You can manage the parent object of the *Emulator* module in the **Map** interactive window using the function menu of the **Emulator** object.



Command to manage the parent object of the *Emulator* module:

- Synchronize time—write the current time to all devices.

The parent object of the *Emulator* module can have the following states:

	Connected
	Disconnected
	Cracked

4.3 Working with the fence of the Emulator module

You can manage the fence of the *Emulator* module in the **Map** interactive window using the function menu of the **Fence** object.

Commands for managing the fence of the *Emulator* module are described in the table:

Function menu command	**Function**
Generate alarm	Generation of an alarm event, executed only after the **Arm** command
Handle alarm	Confirmation of an alarm by an operator
Arm	Arm the fence
Disarm	Disarm the fence

The fence of the *Emulator* module can have the following states:

	Disarm
	Arm
	Alarm

4.4 Working with the lock of the Emulator module

You cannot manage the **Lock** object of the *Emulator* module in the **Map** interactive window.

The **Lock** object of the *Emulator* module can have the following states:

	Locked
	Unlocked

4.5 Working with the reader of the Emulator module

You can manage the reader of the *Emulator* module in the **Map** interactive window using the function menu of the **Reader** object.

Commands for managing the reader of the *Emulator* module are described in the table:

Function menu command	Function
Handle alarm	Confirmation of an alarm by an operator. Generation of an alarm is possible when you send the **Open** or **Pass** commands only after the **Block** command
Open temporary	Temporary set the reader to the **Open** state
Block	Set the reader to the **Block** state
Pass	Generation of a pass event
Unblock	Set the reader to the **Unblock** state
Close	Set the reader to the **Close** state
Open	Set the reader to the **Open** state

The reader of the *Emulator* module can have the following states:

	Closed
	Opened

	Locked
	Unlocked
	Alarm

4.6 Working with the sensor of the Emulator module

You can manage the sensor of the *Emulator* module in the **Map** interactive window using the function menu of the **Sensor** object.

Commands for managing the sensor of the *Emulator* module are described in the table:

Function menu command	Function
Generate alarm	Generation of an alarm event, executed only after the **Arm** command
Handle alarm	Confirmation of alarm by an operator
Arm	Arm the facility
Disarm	Disarm the facility

The sensor of the *Emulator* module can have the following states:

	Disarm
	Arm

	Alarm

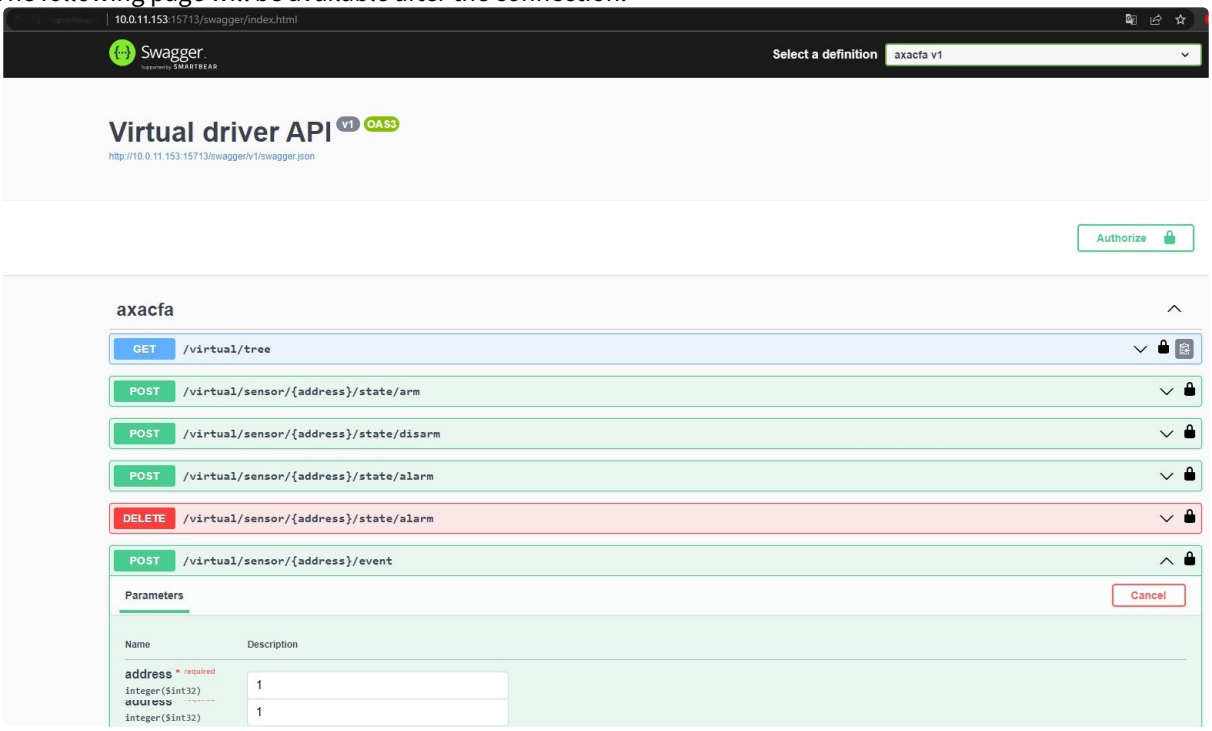
4.7 Working with the Swagger software

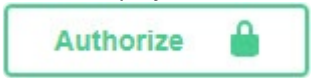
In the *Emulator* integration module, you can work with the Swagger software directly via POST requests. To do this, do the following:

1. Activate the Swagger software and Swagger UI (see Configuring the parent object of the Emulator module).
2. Connect to the Swagger software in a web browser using the string:

```
http://{ip}:{port}/swagger
```

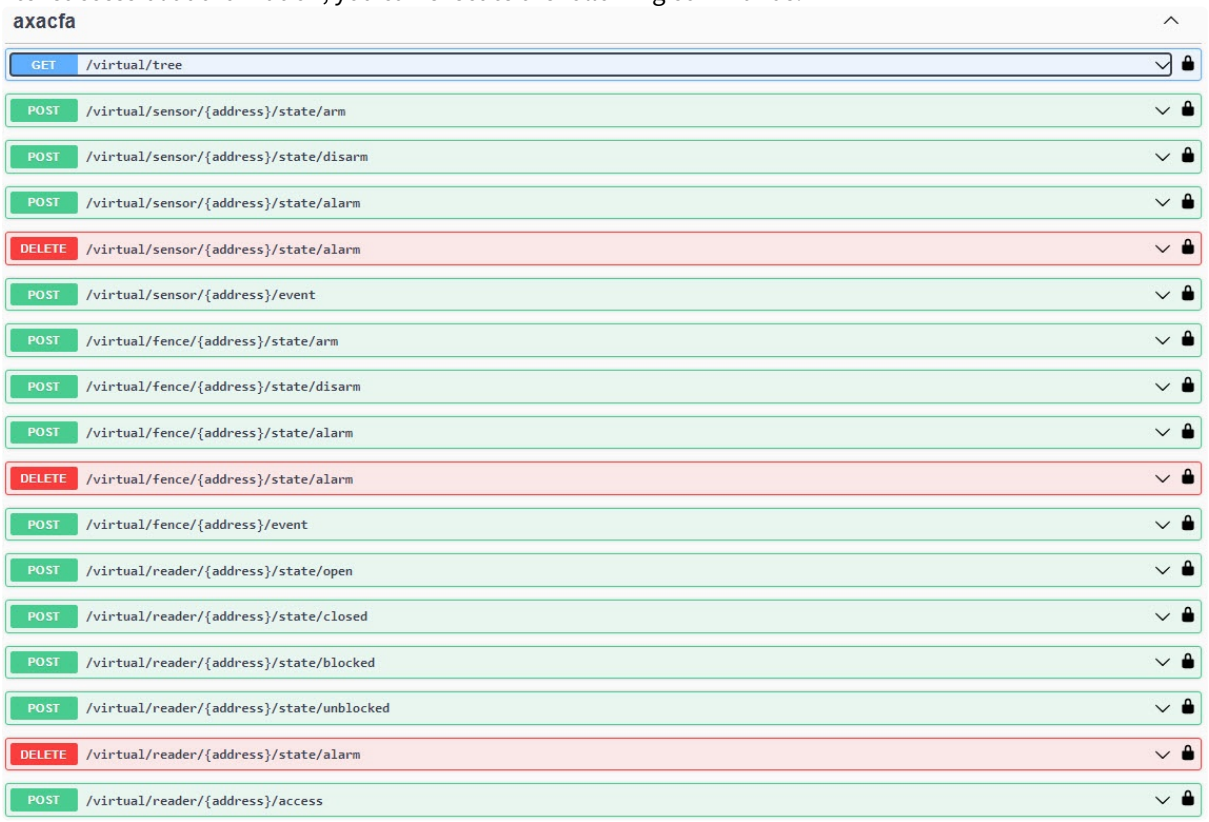
The following page will be available after the connection:



3. You can authorize by the token, which was displayed in the settings of the parent object of the *Emulator* module, by clicking the **Authorize** 

button in the upper right corner of the page.

4. After successful authorization, you can execute the following commands:



Examples of commands:

a. Read the hardware tree with the current states. To do this, select the GET /virtual/tree request, first click the **Try it Out** button and then click the **Execute** button.
b. Switch the sensor with the specified address to the **arm** state by selecting the corresponding request and following the same steps as in step 4a.
c. Switch the sensor with the specified address to the **disarm** state by selecting the corresponding request and following the same steps as in step 4a.
d. Switch the sensor with the specified address from the **arm** state to the **alarm** state by selecting the corresponding request and following the same steps as in step 4a.
e. Process the alarm and switch the sensor to the **disarm** state by selecting the corresponding request and following the same steps as in step 4a.
f. And so on, similarly to the described commands.