

VERSION 1.0

On-Premises Solix User Guide



HW System Requirements:

- ✧ CPU Information
 - ◆ Model: Intel NUC Kit NUC7CJYH (Celeron J4005, UHD 600) Mini PC
 - ✧ Memory
 - ◆ Size: 8GiB
 - ✧ Network Interfaces
 - ◆ Ethernet Socket
 - ◆ WiFi Access Point
 - ✧ Storage:
 - ◆ Size: 40GiB
 - ✧ Operating System:
 - ◆ Ubuntu 22.04.3 LTS
-

1.1. Solix Installation:

Before proceeding with the installation of Solix On-Premise, ensure that you have downloaded the docker-compose file and that the following ports are not in use on your server machine: **4000, 5500, 8771, 3306, 1883, 9001**.

Please note that the installation steps provided below are tailored for **Ubuntu 22.04**.

1. **Connect to the Server:** Utilize **WinSCP** to establish a connection to your server.
2. **Navigate to the Home Directory:** Use the following command to navigate to the home directory on your server:

```
cd /home
```

3. **Create a new directory** where you will copy the downloaded docker-compose file:

```
mkdir solix
```

4. **Copy the downloaded docker file to the directory** on your server.

In our example, it is /home/solix

```
/home/solix
```

You should be able to see your docker-compose file in the directory.

5. **Navigate to the directory:**

```
cd /home/solix
```

6. Create environment file for your license key:

`nano .env`

7. Now, paste your license key in the environmental file (make sure the format is the same as below):

`LICENSE_KEY=your-license-key-here`

Save your file by pressing **CTRL+S** and exit the file terminal by pressing **CTRL+X**.

8. Run the docker-compose file in your terminal in the directory you copied your docker-compose file to, in our example its /home/solix:

`docker-compose up` (to see the docker logs)

`docker-compose up -d` (to run the docker in the background)

9. After the installation is done, Solix should be accessible in several minutes.

```
Creating network solix24_my_network with driver bridge
Creating solix24_solix-db_1 ... done
Creating solix24_solix-api_1 ... done
Creating solix24_solix-engine_1 ... done
Creating solix24_solix-broker_1 ... done
Creating solix24_solix-platform_1 ... done
Attaching to solix24_solix-db_1, solix24_solix-api_1, solix24_solix-broker_1, solix24_solix-engine_1, solix24_solix-platform_1
solix-broker_1 | 1710155424: mosquitto version 2.0.18 starting
solix-broker_1 | 1710155424: config loaded from /etc/mosquitto/mosquitto.conf.
solix-broker_1 | 1710155424: Warning: File /etc/mosquitto/passwd has world readable permissions. Future versions will refuse to load this file.
solix-broker_1 | To fix this, use 'chmod 0700 /etc/mosquitto/passwd'.
solix-broker_1 | 1710155424: Warning: File /etc/mosquitto/passwd owner is not mosquitto. Future versions will refuse to load this file. To fix this, use 'chown mosquitto /etc/mosquitto/passwd'.
solix-broker_1 | 1710155424: Warning: File /etc/mosquitto/passwd group is not mosquitto. Future versions will refuse to load this file.
solix-broker_1 | 1710155424: Opening IPv4 listen socket on port 1883.
solix-broker_1 | 1710155424: Opening IPv6 listen socket on port 1883.
solix-broker_1 | 1710155424: Opening websockets listen socket on port 9001.
solix-broker_1 | 1710155424: mosquitto version 2.0.18 running
solix-broker_1 | 1710155426: New connection from 172.21.0.5:52281 on port 1883.
solix-broker_1 | 1710155426: New client connected from 172.21.0.5:52281 as AoAEngine2584526-8ffc-4526-8887-d0254d9f8f92 (p2, co, k15, u'solix_mqttuser').
solix-broker_1 | 1710155426: No will message specified.
solix-broker_1 | 1710155426: Sending CONNACK to AoAEngine2584526-8ffc-4526-8887-d0254d9f8f92 (q, 0)
solix-broker_1 | 1710155426: Received SUBSCRIBE from aoaengine2584526-8ffc-4526-8887-d0254d9f8f92
solix-broker_1 | 1710155426: /Logs/Plug (QoS 0)
solix-broker_1 | 1710155426: AoAEngine2584526-8ffc-4526-8887-d0254d9f8f92 0 /Logs/Plug
solix-broker_1 | 1710155426: /Config/Response (QoS 0)
solix-broker_1 | 1710155426: AoAEngine2584526-8ffc-4526-8887-d0254d9f8f92 0 /Config/Response
solix-broker_1 | 1710155426: sending SUBACK to aoaengine2584526-8ffc-4526-8887-d0254d9f8f92
solix-db_1 | 2024-03-11 11:10:22+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.3.0-1.el8 started.
solix-db_1 | 2024-03-11 11:10:23+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
solix-db_1 | 2024-03-11 11:10:23+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.3.0-1.el8 started.
solix-db_1 | '/var/lib/mysql/mysql.sock' -> '/var/run/mysql/mysql.sock'
solix-db_1 | 2024-03-11 11:10:24.549602Z 0 [System] [MV-018121] [Server] MySQL Server -- start.
solix-db_1 | 2024-03-11 11:10:24.549602Z 0 [System] [MV-018116] [Server] /usr/sbin/mysqld (mysqld 8.3.0) starting as process 9
solix-db_1 | 2024-03-11 11:10:24.562707Z 1 [System] [MV-013576] [InnoDB] InnoDB initialization has started.
solix-db_1 | 2024-03-11 11:10:25.383697Z 1 [System] [MV-013577] [InnoDB] InnoDB initialization has ended.
solix-db_1 | 2024-03-11 11:10:25.855360Z 0 [System] [MV-018229] [Server] Starting XA crash recovery...
solix-db_1 | 2024-03-11 11:10:25.879102Z 0 [System] [MV-018232] [Server] XA crash recovery finished.
solix-db_1 | 2024-03-11 11:10:26.054325Z 0 [Warning] [MV-018068] [Server] CA certificate ca.pem is self signed.
solix-db_1 | 2024-03-11 11:10:26.056527Z 0 [System] [MV-013602] [Server] Channel mysql_native configured to support TLS. Encrypted connections are now supported for this channel.
solix-db_1 | 2024-03-11 11:10:26.065127Z 0 [Warning] [MV-013818] [Server] Insecure configuration for --pid-file: location '/var/run/mysql/' in the path is accessible to all OS users. Consider choosing a d
ifferent directory.
solix-db_1 | 2024-03-11 11:10:26.213163Z 0 [System] [MV-011323] [Server] X Plugin ready for connections. Bind address: '::' port: 33060, socket: '/var/run/mysql/mysql.sock'
solix-db_1 | 2024-03-11 11:10:26.216081Z 0 [System] [MV-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.3.0' socket: '/var/run/mysql/mysql.sock' port: 3306 MySQL Community Serve
r - GPL
solix-engine_1 | 2024-03-11 11:10:24 - Starting GatewayConfig...
solix-api_1 | info: Microsoft.Hosting.Lifetime[0]
solix-api_1 | info: Now listening on: http://0.0.0.0:5300
solix-api_1 | info: Microsoft.Hosting.Lifetime[0]
solix-api_1 | Application started. Press Ctrl+C to shut down.
solix-api_1 | info: Microsoft.Hosting.Lifetime[0]
solix-api_1 | Hosting environment: Production
```

Note: If you intend to utilize your own mosquito broker, you can modify the `config.ini` file located inside the `engine` folder, which you downloaded alongside the docker-compose file.

- a) Navigate to the `engine` folder.

`cd /home/solix/engine`

- b) Edit the `config.ini` file.

`nano config.ini`

```
[MQTT]
BROKER = solix-broker
USERNAME = solix_mqttuser
MQTT_PORT = 1883
PASSWORD = Solix_mqttPW4
GATEWAY_TOPIC = /device/filtered/staticangles
```

GATEWAY_TOPIC represents the topic through which your gateways stream angles data. Make necessary adjustments as per your requirements.

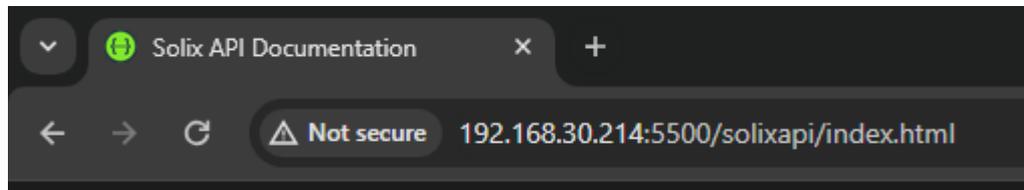
1.2. Setting up Solix:

Before proceeding, please note that setting up **Solix** requires familiarity with **API requests** and **developer knowledge**.

Now that the **Solix** is up and running, you need to first add your Gateway and Tag data and create Client and User IDs in order to use the **Solix Platform**.

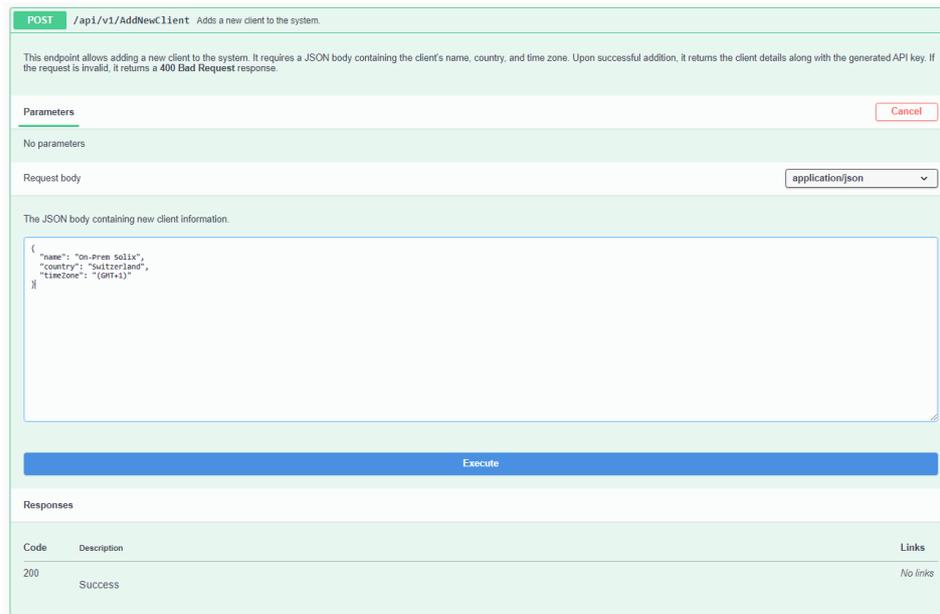
1. Access Solix API Documentation:

- a) Visit the Solix API Documentation by entering your Linux server IP followed by port **5500** and endpoint **/solixapi**.



2. Create an Admin Client:

- a) Navigate to the **POST API /api/v1/AddNewClient** under the **ClientAPI** section.
- b) Click on **Try it out** and input your credentials.
- c) **Execute** the request.



POST /api/v1/AddNewClient Adds a new client to the system.

This endpoint allows adding a new client to the system. It requires a JSON body containing the client's name, country, and time zone. Upon successful addition, it returns the client details along with the generated API key. If the request is invalid, it returns a 400 Bad Request response.

Parameters Cancel

No parameters

Request body application/json

The JSON body containing new client information.

```
{
  "name": "On-Prem Solix",
  "country": "Switzerland",
  "timezone": "(GMT+1)"
}
```

Execute

Responses

Code	Description	Links
200	Success	No links

- d) Upon receiving Success Response **200**, save the generated **apikey** and **id** in a file. These credentials will serve as your main **ClientId** and **APIKey** for further API calls.

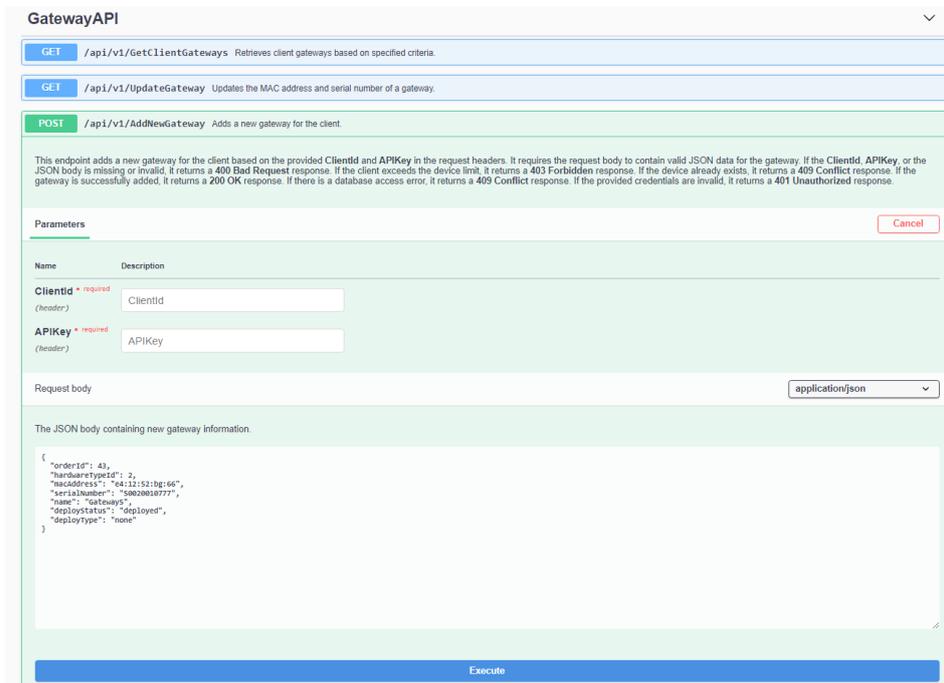
```
{
  "Message": "Success",
  "Data": [
    {
      "id": 1,
      "name": "On-Prem Solix",
      "apikey": "CZq/B+NpXw8D1XhvqzcrYWuvc5SRsL++CTPw8cYkc88="
    }
  ]
}
```

- e) Proceed to the **GET API /api/v1/GetHWID** under the **ClientAPI** section and note the IDs of your TOW-1 Tag, LON-2 Gateway, and TOK-1 Tag.

```
{
  "Message": "Success",
  "Data": [
    {
      "id": 1,
      "name": "Pinix TOW-1",
      "productType": "Tag"
    },
    {
      "id": 2,
      "name": "Zenix LON-2",
      "productType": "Gateway"
    },
    {
      "id": 3,
      "name": "Pinix TOK-1",
      "productType": "Tag"
    }
  ]
}
```

3. Add Gateway Data:

- a) Access the **POST API /api/v1/AddNewGateway** under the **GatewayAPI** section.



The screenshot shows the 'GatewayAPI' section with three endpoints. The selected endpoint is 'POST /api/v1/AddNewGateway' with the description 'Adds a new gateway for the client.' Below this, there is a 'Parameters' section with two required header fields: 'Clientid' and 'APIKey'. The 'Request body' section is set to 'application/json' and contains a JSON object with gateway details. An 'Execute' button is at the bottom.

Parameters

Name	Description
Clientid * required (header)	Clientid
APIKey * required (header)	APIKey

Request body application/json

The JSON body containing new gateway information.

```
{
  "orderId": 43,
  "hardwareTypeId": 2,
  "macaddress": "94:12:15:2b:66",
  "serialnumber": "5802081077",
  "name": "Gateways",
  "deploystatus": "deployed",
  "deploytype": "none"
}
```

Execute

- b) Fill the headers with the **Clientid** and **APIKey** you noted earlier in **step #2c**.
- c) Input Gateway **MAC**, **Serial**, and replace **hardwareTypeId** with the **LON-2 Id** you noted earlier in **step #2e** which in our example is **2**.
- d) Execute the request to add your gateway to the Solix Platform.

4. Add Tag Data:

- a) Navigate to the **POST API /api/v1/AddNewTag** under the **TagsAPI** section.

- a) Fill the headers with the **Clientid** and **APIKey** you noted earlier in **step #2c**. Input Tag **MAC, Serial** and replace **hardwareTypeid** with the appropriate ID (**TOK-1** or **TOW-1**) you noted earlier in **step #2e** which in our example is **1** for **TOW-1** and **3** for **TOK-1**.
- b) Execute the request to add your tag to the Solix Platform.

5. Add Client User:

- a) Go to the **POST API /api/v1/AddClientUser** under the **UserAPI** section.

- b) Fill the headers with the **Clientid** and **APIKey** you noted earlier in **step #2c**.
- c) Use **pagelist** to control user access to specific pages on Solix Platform.

- d) **Pagelist 0** will give user access to all the pages on Solix Platform, if you want to restrict users to have limited page access, head over to the **GET API /api/v1/GetAllPages** under **LoginAPI** section.

LoginAPI

GET /api/v1/LoginUser Logs in a user with the provided Email and Password.

GET /api/v1/GetAllPages Retrieves all available pages.

This endpoint returns a list of all available pages.

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	Success	No links

GET /api/v1/CheckUserRights Checks user rights based on the provided UserID.

- a) **Execute it** and you should see **Success Response 200** with the list of all the pages on Solix Platform.

```

{
  "Message": "Success",
  "Pages": [
    {
      "id": 1,
      "page": "Location_LiveMap"
    },
    {
      "id": 2,
      "page": "Beacons_Summary"
    },
    {
      "id": 3,
      "page": "Beacons_SensorData"
    },
    {
      "id": 4,
      "page": "Beacons_Position"
    },
    {
      "id": 5,
      "page": "Beacons_ThirdParty"
    },
    {
      "id": 6,
      "page": "Beacons_ProductionConfig"
    }
  ]
}
    
```

- b) Back to the **UserAPI** section, add the **Id** of the pages you want the user to have access to in **pagelist**, in our example we would like them to see only the **Live Map and Gateway Summary page**. Our response body should be **pagelist 1,9**:

Request body application/json

The JSON object containing user information.

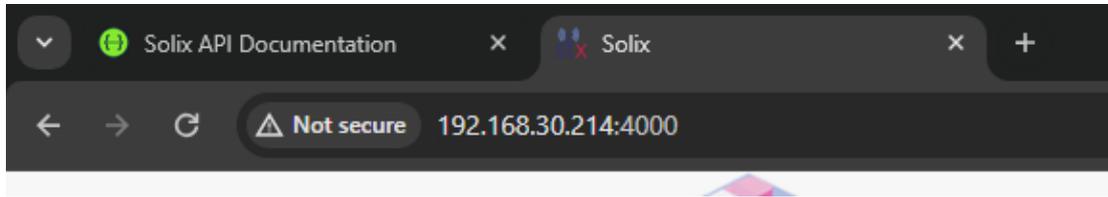
```

{
  "firstname": "John",
  "lastname": "Smith",
  "email": "john@gmail.com",
  "phone": "22222222",
  "password": "Johnsmith123",
  "pagelist": "1,9"
}
    
```

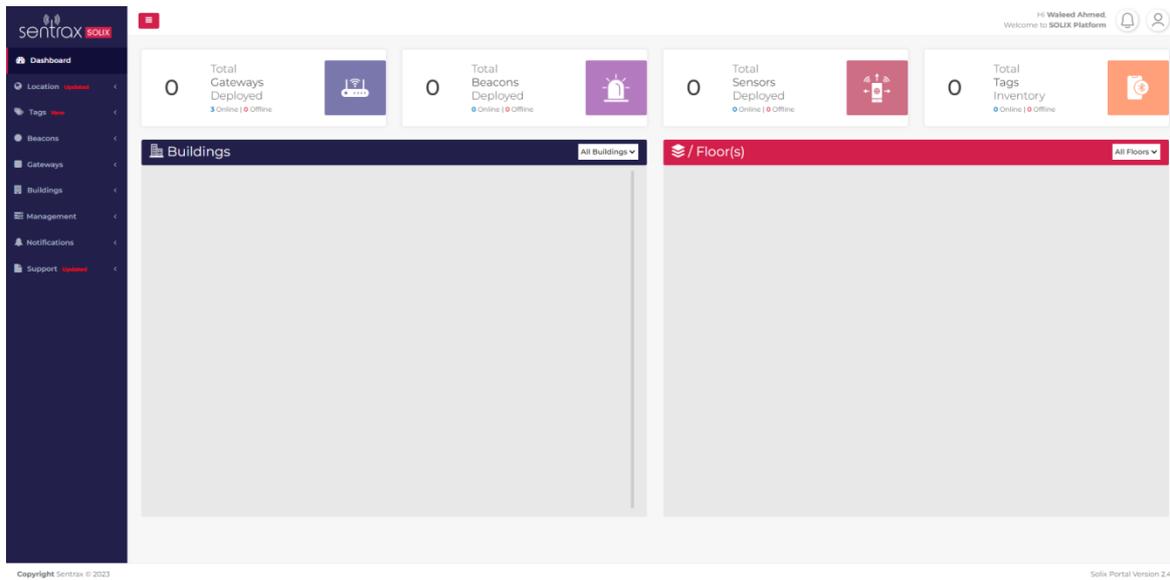
- c) **Execute it** and you should see **Success Response 200**, user has been added to your Solix Platform.

6. Access Solix Platform:

- a) Login to Solix Platform using the URL of your server IP and port **4000**.

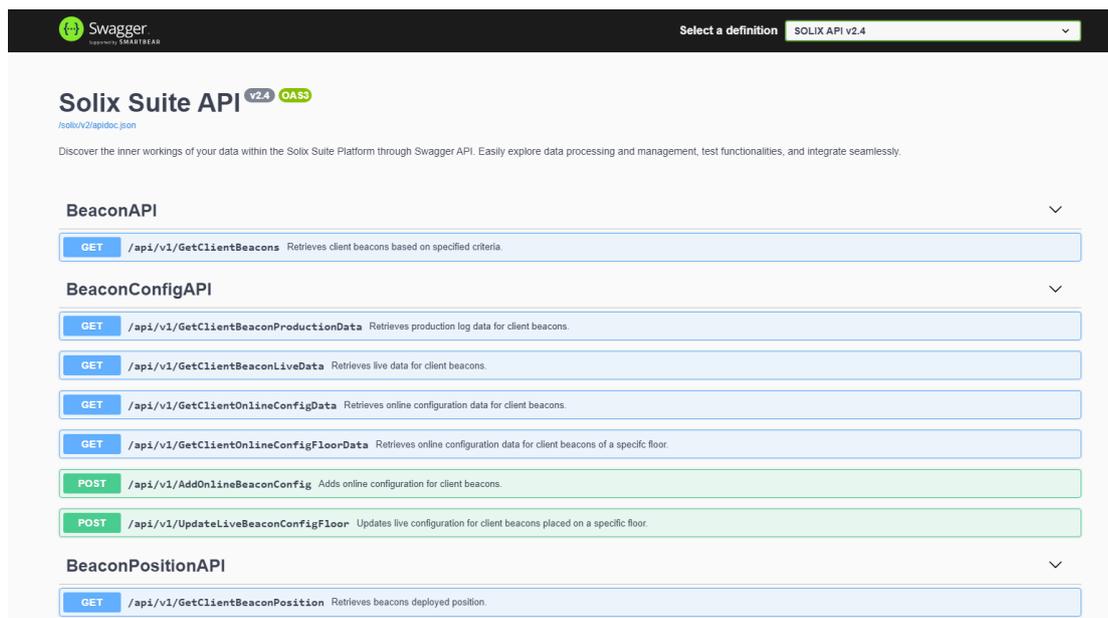


- b) Enter the **Email** and **Password** of the newly created **ClientUser** to access the Dashboard.



7. Additional Resources:

- a) You can find the **Solix User Guide** and **Data sheet** under the **Support** drop-down in the navigation bar.
- b) You can visit the **Solix API Documentation** by URL of your server IP, port **5500** and endpoint **/solixapi**



Solix Suite API v2.4 OAS3

Discover the inner workings of your data within the Solix Suite Platform through Swagger API. Easily explore data processing and management, test functionalities, and integrate seamlessly.

BeaconAPI

- GET /api/v1/GetClientBeacons Retrieves client beacons based on specified criteria.

BeaconConfigAPI

- GET /api/v1/GetClientBeaconProductionData Retrieves production log data for client beacons.
- GET /api/v1/GetClientBeaconLiveData Retrieves live data for client beacons.
- GET /api/v1/GetClientOnlineConfigData Retrieves online configuration data for client beacons.
- GET /api/v1/GetClientOnlineConfigFloorData Retrieves online configuration data for client beacons of a specific floor.
- POST /api/v1/AddOnlineBeaconConfig Adds online configuration for client beacons.
- POST /api/v1/UpdateLiveBeaconConfigFloor Updates live configuration for client beacons placed on a specific floor.

BeaconPositionAPI

- GET /api/v1/GetClientBeaconPosition Retrieves beacons deployed position.

Disclaimer:

This guide is intended for informational purposes only. If in doubt at any stage of the installation or operation of the locator/gateway always consult Sentrax's authorized dealer, distributor, or get in touch directly with Sentrax GmbH.

Given that Sentrax will continuously improve and develop the product, changes may be made to the information in this manual at any time without any obligation to notify any person of any such revisions or changes. Sentrax will make all possible efforts to secure the accuracy and integrity of this manual.

Note: Reproduction, transfer, distribution or storage of part or all the contents of this document in any form without the prior permission of Sentrax GmbH is prohibited.



CONNECT WITH US



www.sentrax.com



support@sentrax.com