

# WisNode TrackIt Helium Integration

In this example, you will learn how to register the TrackIt device in the Helium console and make integration to Datacake, where you can use the dashboard to view the data in a more user-friendly way. For example, create a map so you can see the location of the device.

Before you dive into registering the device in the Helium console and creating the Datacake integration, you need to set the device to work in [Third-Party LoRaWAN Network Server \(LNS\) Mode](#). [↗](#)

## Register the TrackIt Node in the Helium Console

1. Login into your Helium console account. If you do not have one, head to the web page and create one. Once you are logged in, you will see the Welcome window.

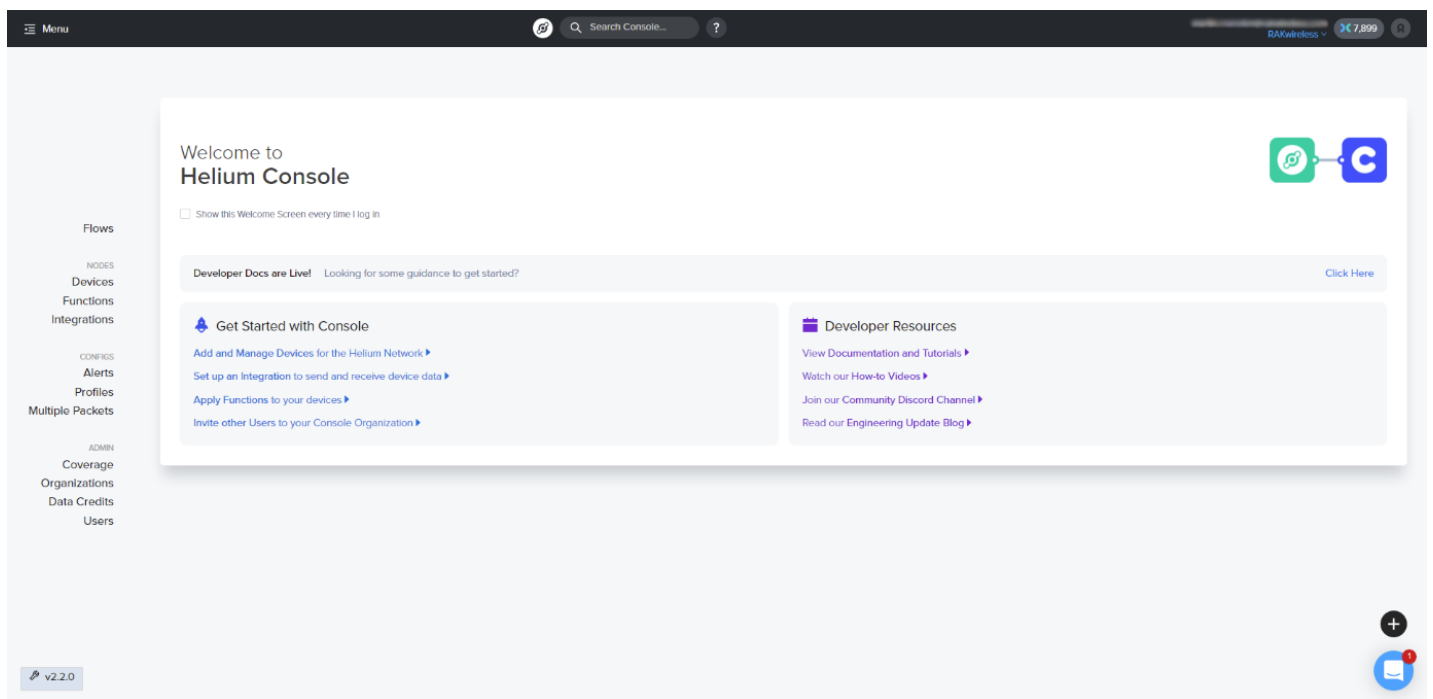


Figure 1: Helium Console Main Page

2. From the menu on the left side of the console, select **Devices** and then click on the **Add new device button** (



) to register your device.

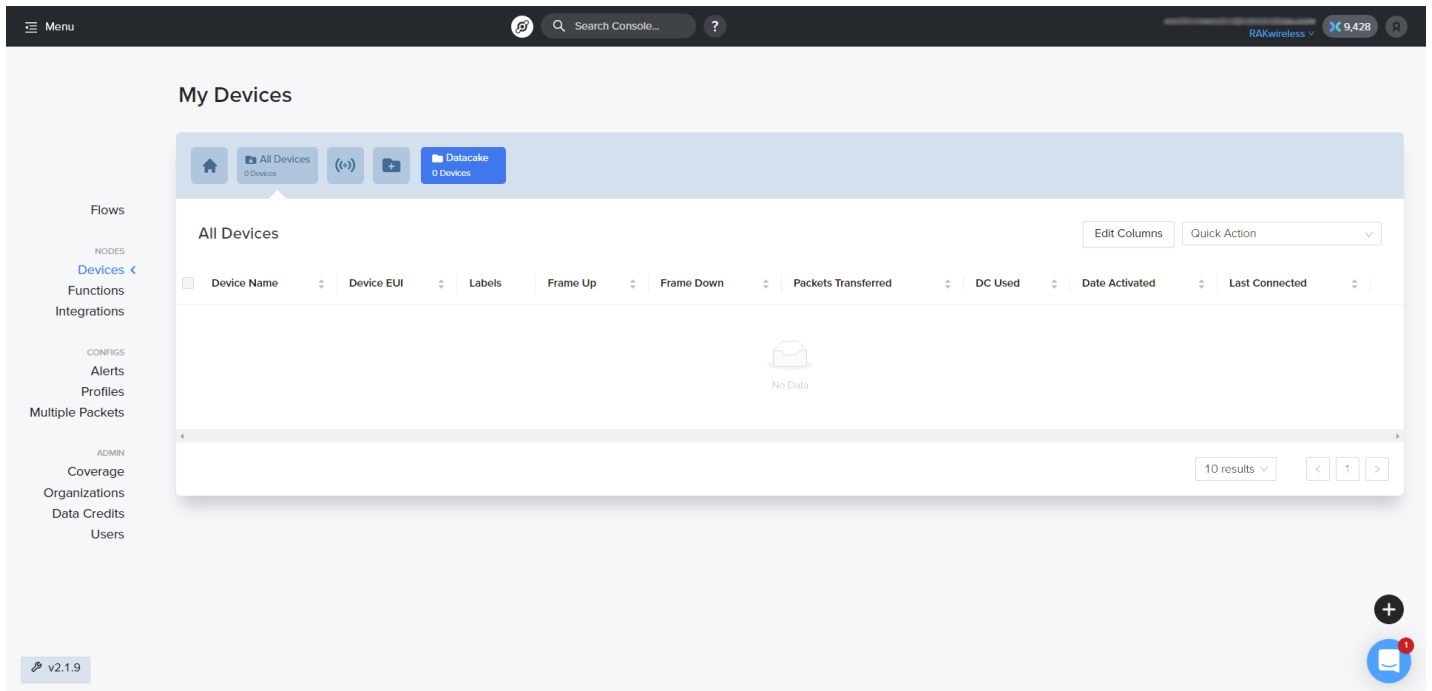


Figure 2: Helium Console Devices Page

3. In the **Name** field, type in a name for your device.

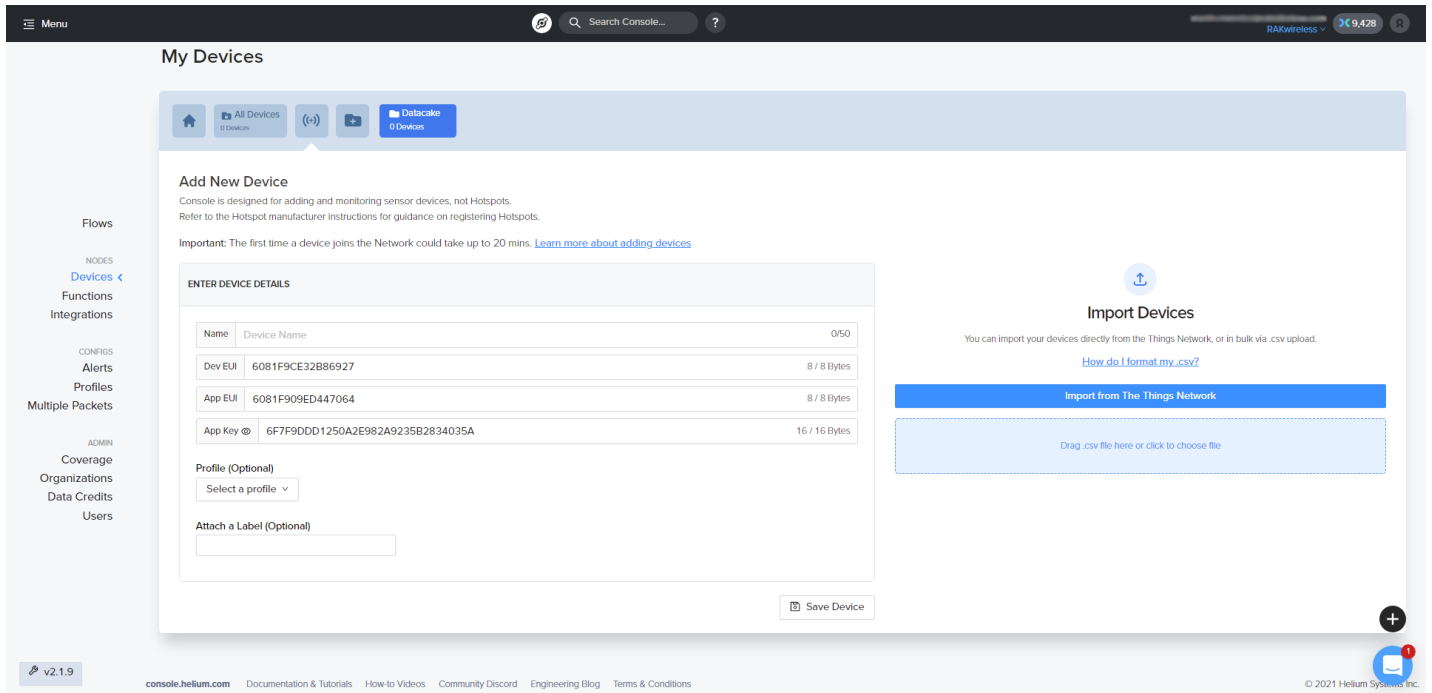


Figure 3: Add New Device Page

4. You can see that the Helium console generates random **Dev EUI**, **App EUI**, and **App Key**. As the TrackIt keys cannot be changed, you need to type your keys in these fields. These keys can be found in the TrackIt application in the LoRaWAN Working Mode (**Third Party NS**).

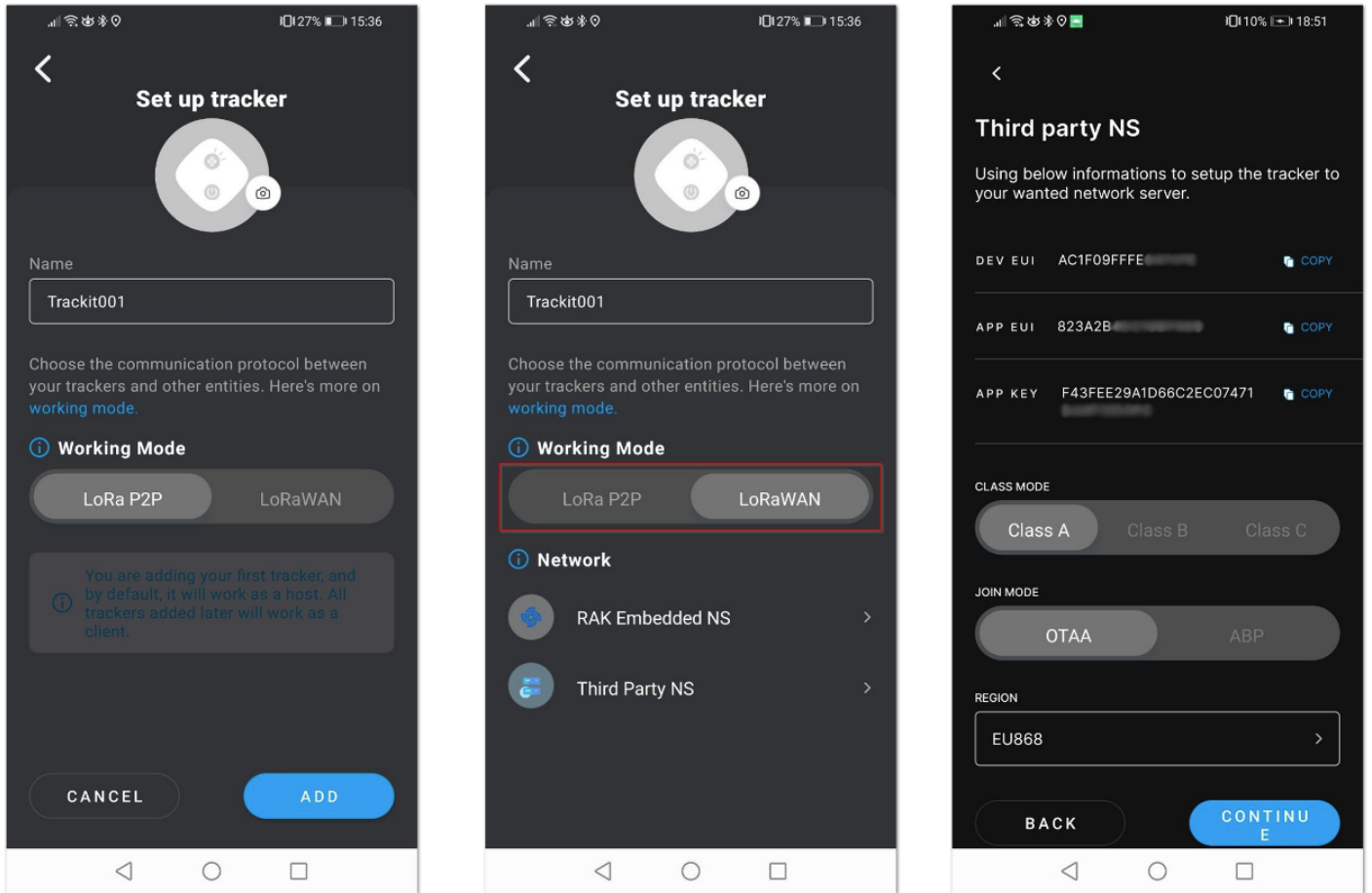


Figure 4: TrackIt Keys

5. Optionally, you can select a **Profile** and **Attach a Label**. For this tutorial, they will be skipped. Note that Profiles and Labels can be added after the device is registered.
6. After typing in the required keys, click on **Save Device** and you will see the registered device. As mentioned by the console, the initial join process takes about 20 minutes for the device to join, so be patient.

## Create Datacake Integration

1. While the device is joining, you can create the integration. Select Integrations from the menu in the left panel.

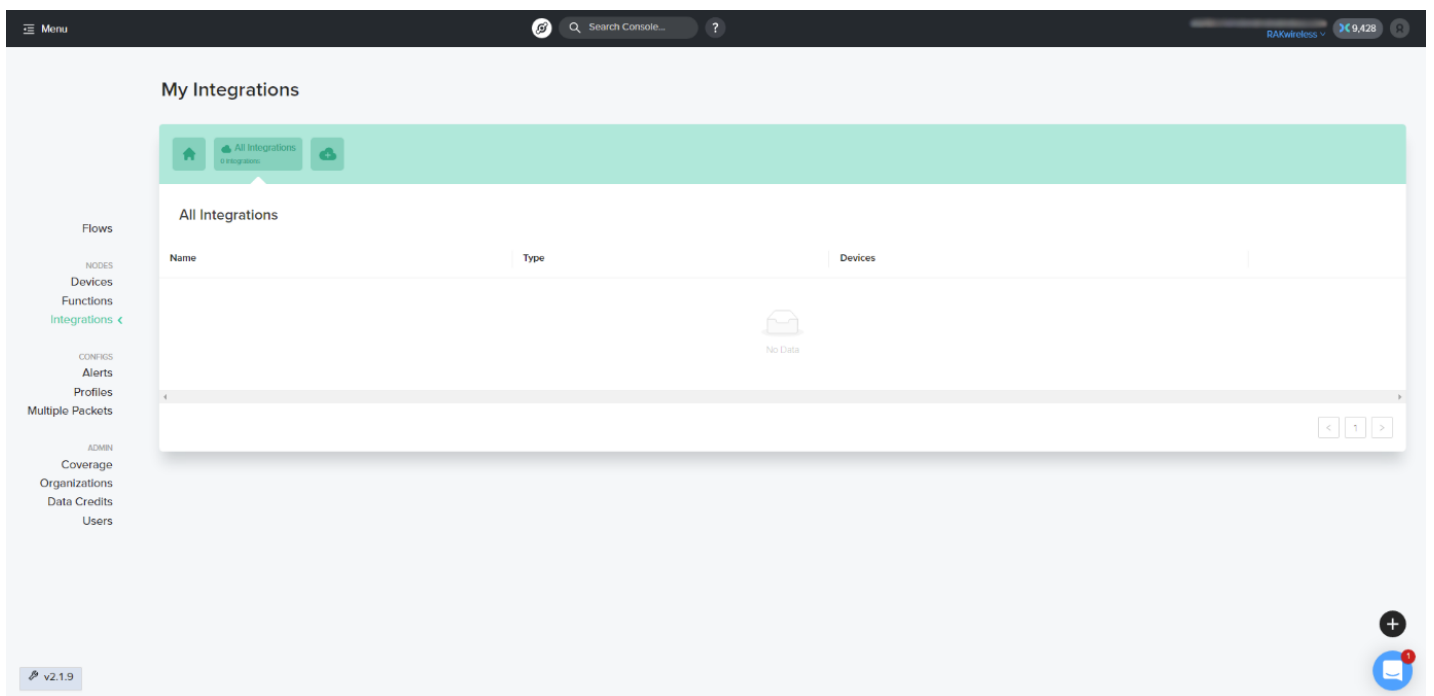

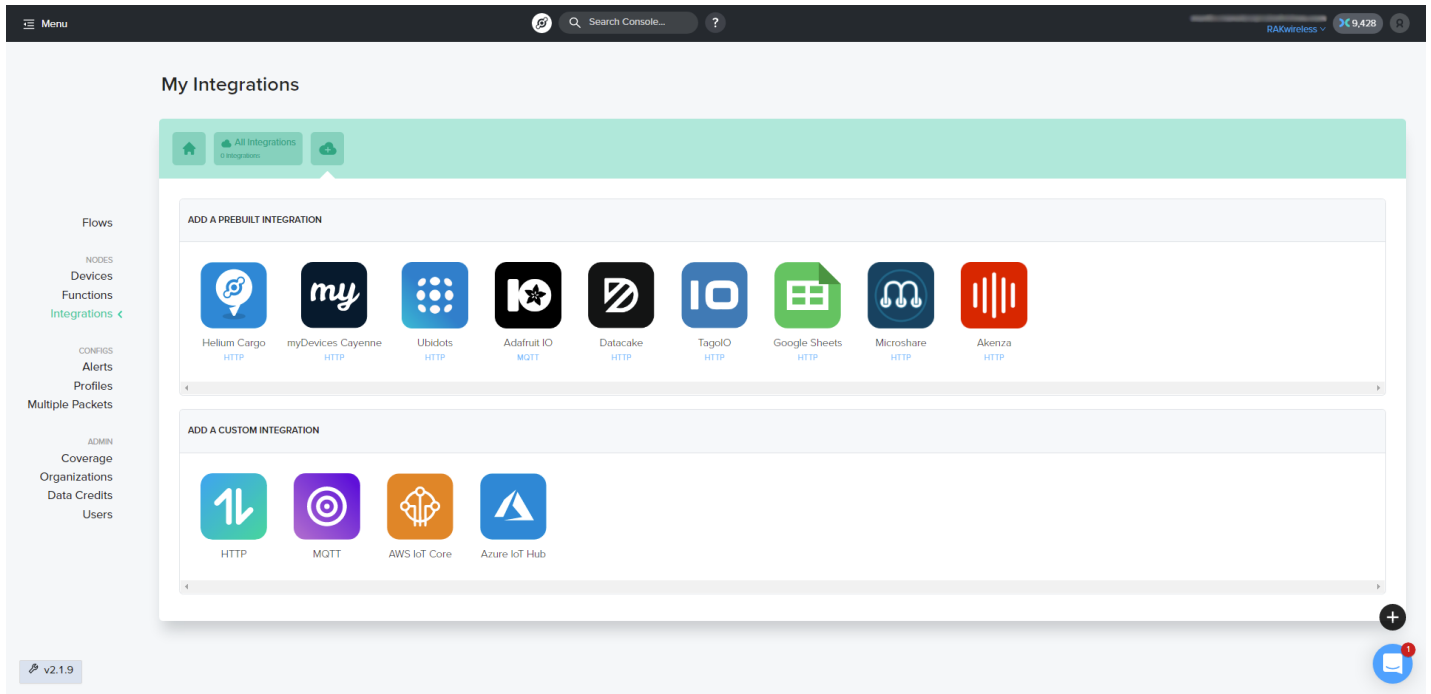


Figure 5: Integration page

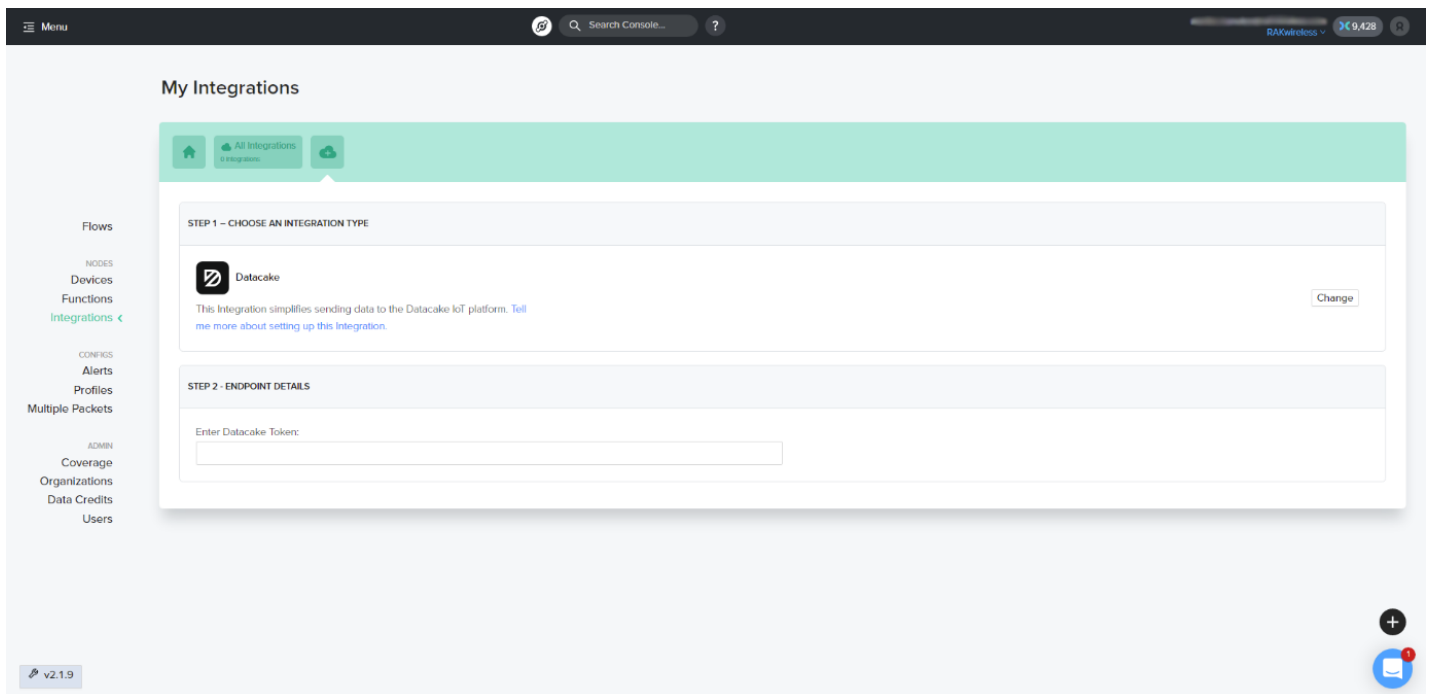


2. To create an integration, click on the **Add New Integration button** (  ). A list of the available integrations will appear. As this example is about Datacake, click on it. Here you can find information about the other integrations. You can find information about the other integrations in the [link](#) .



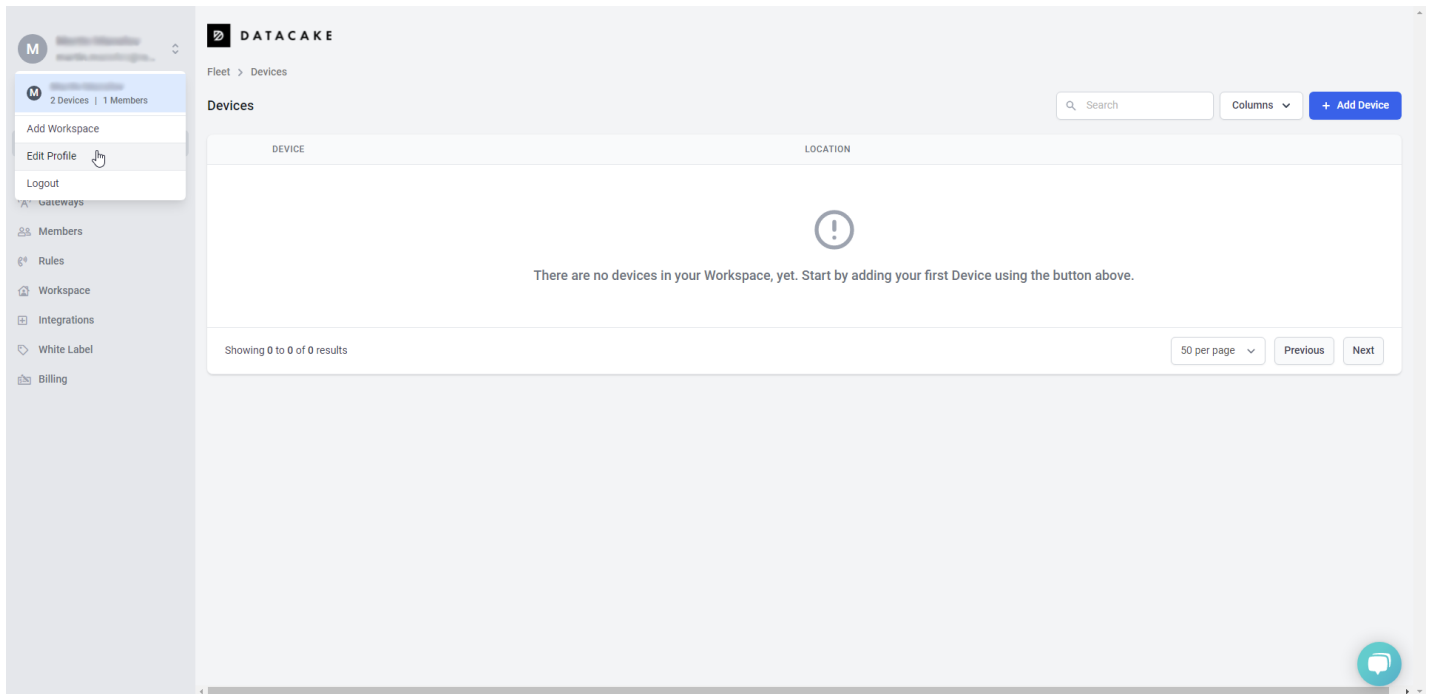
**Figure 6:** List of Available Integrations

3. On the next page, you will need a **Datacake Token**. To generate one, you will need a Datacake account. In case you don't already have an account, you can create one on [Datacake's website](#) .



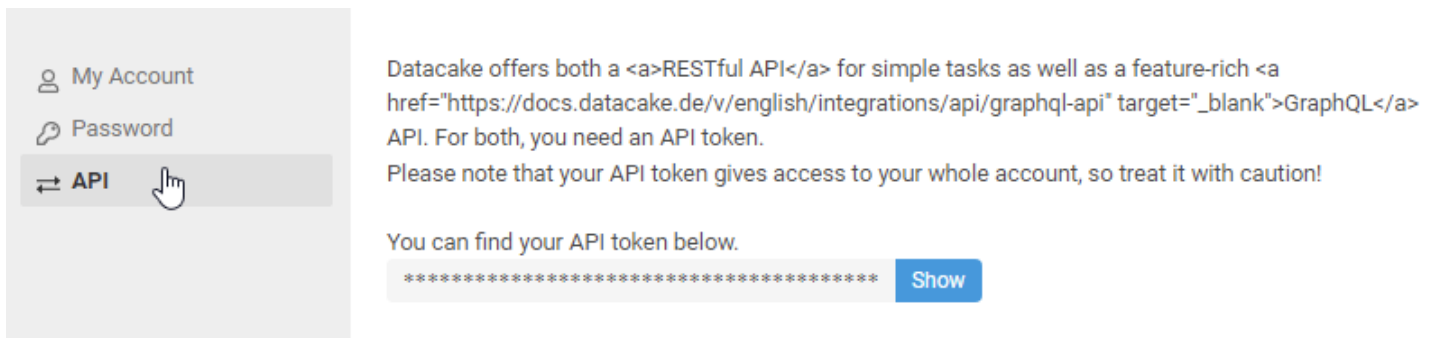
**Figure 7:** Datacake Endpoint Token

4. After you log into your Datacake account, click on your profile and select **Edit Profile**.



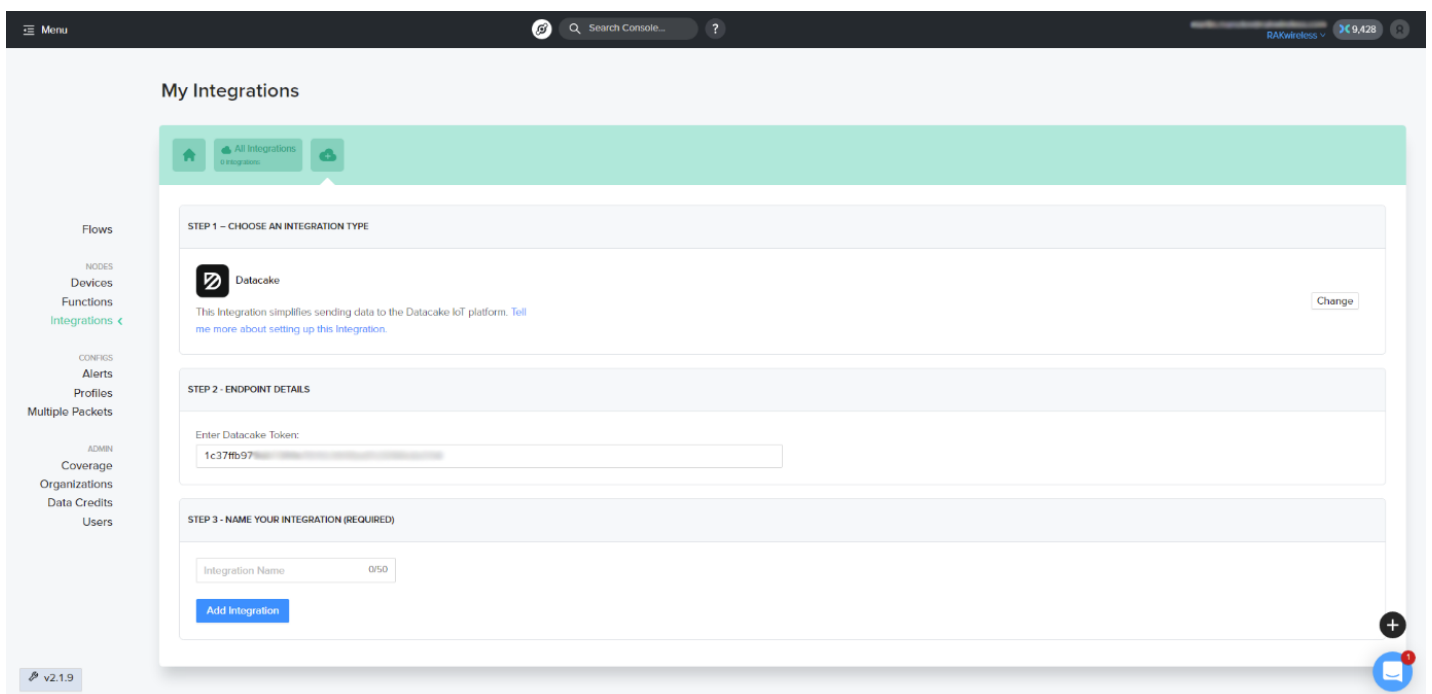
**Figure 8:** Datacake Console Page

5. In the **API** tab, you will find your API token. Copy and paste it into the **Enter Datacake Token** field in the Helium console shown in Figure 7.



**Figure 9:** Datacake Endpoint

6. When you place the token, you will need to type a name of your choice for the integration and click **Add Integration** to continue.



**Figure 10:** Adding Integration Name

## Connection Between Helium and Datacake

- Now that the device is registered and the integration created, you will need to make the connection between Helium and Datacake. Click on **Flows** from the left menu.

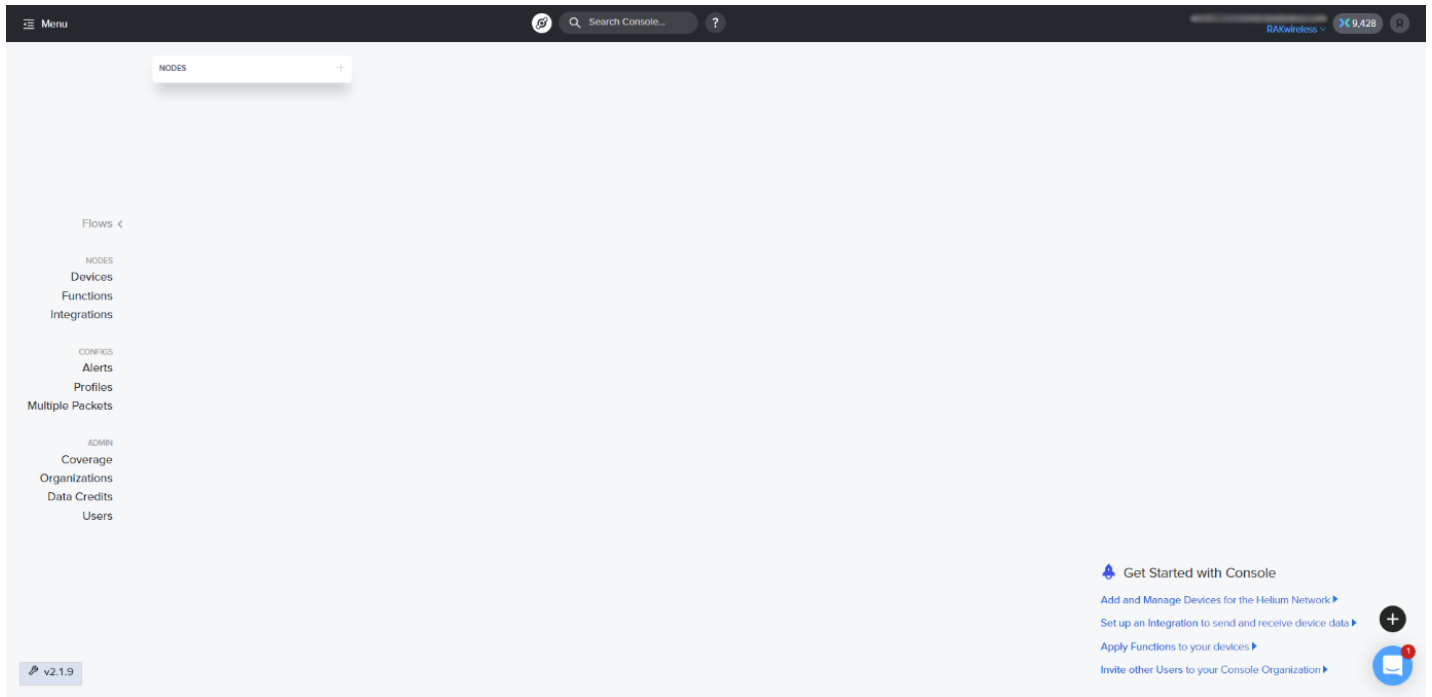


Figure 11: Helium Console Flows

- Click on the + in the **Nodes** area. A window will pop up with four menus – **Labels**, **Devices**, **Functions**, and **Integrations**. Click on **Devices**. You will see your registered device.

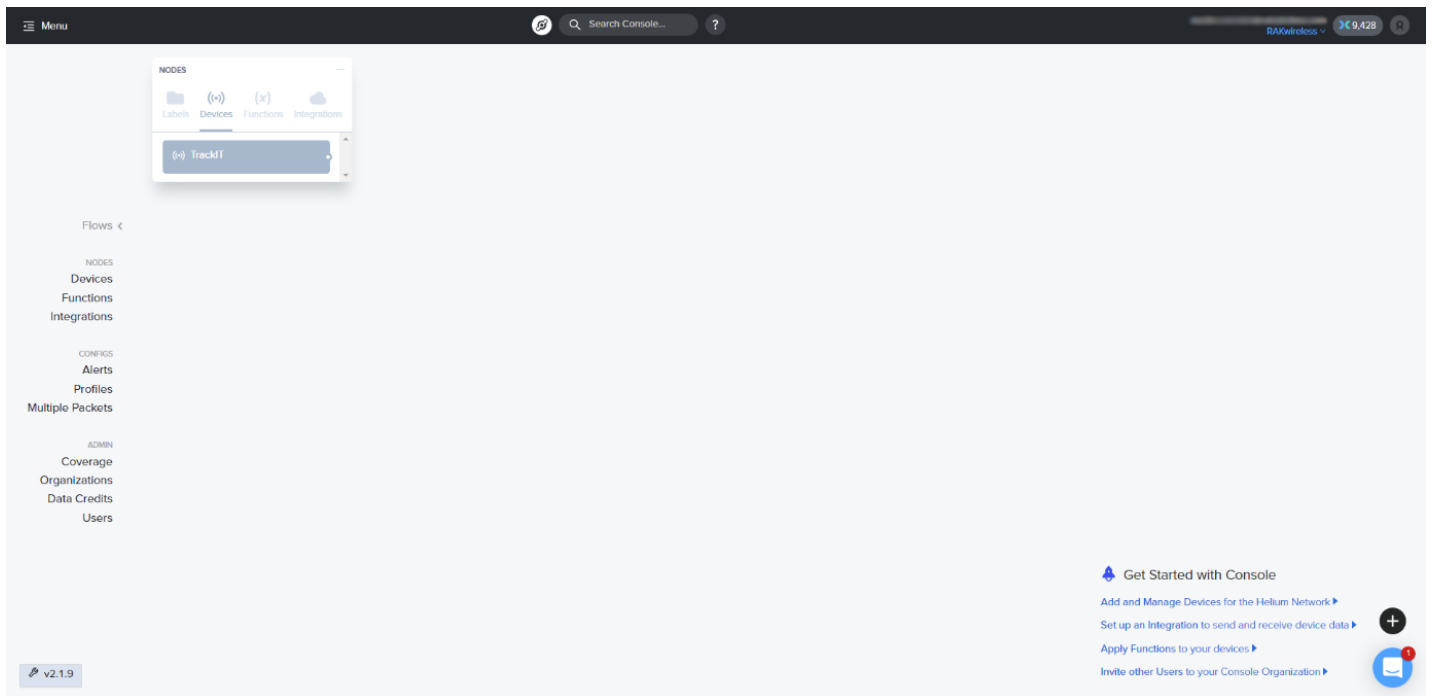


Figure 12: Nodes

- Now, click on your registered device and drag and drop it on the blank page below as shown in Figure 13.
- Do the same with the created integration. Click on the **Integrations** menu and drag and drop your created integration.

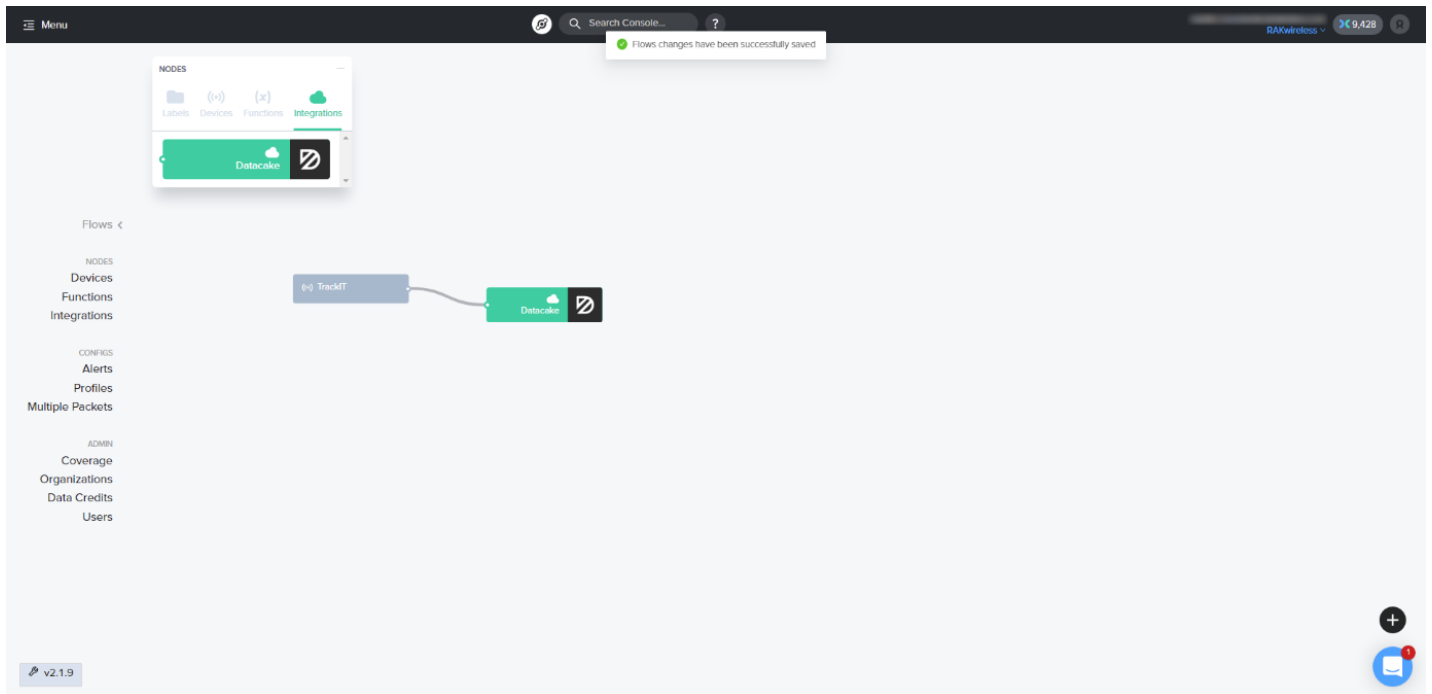


Figure 13: Creating the Connection Between the Device and the Integration

## Datacake Device Registration

1. You need to add the device to the Datacake console. Head there and click on the **+ Add Device** button.

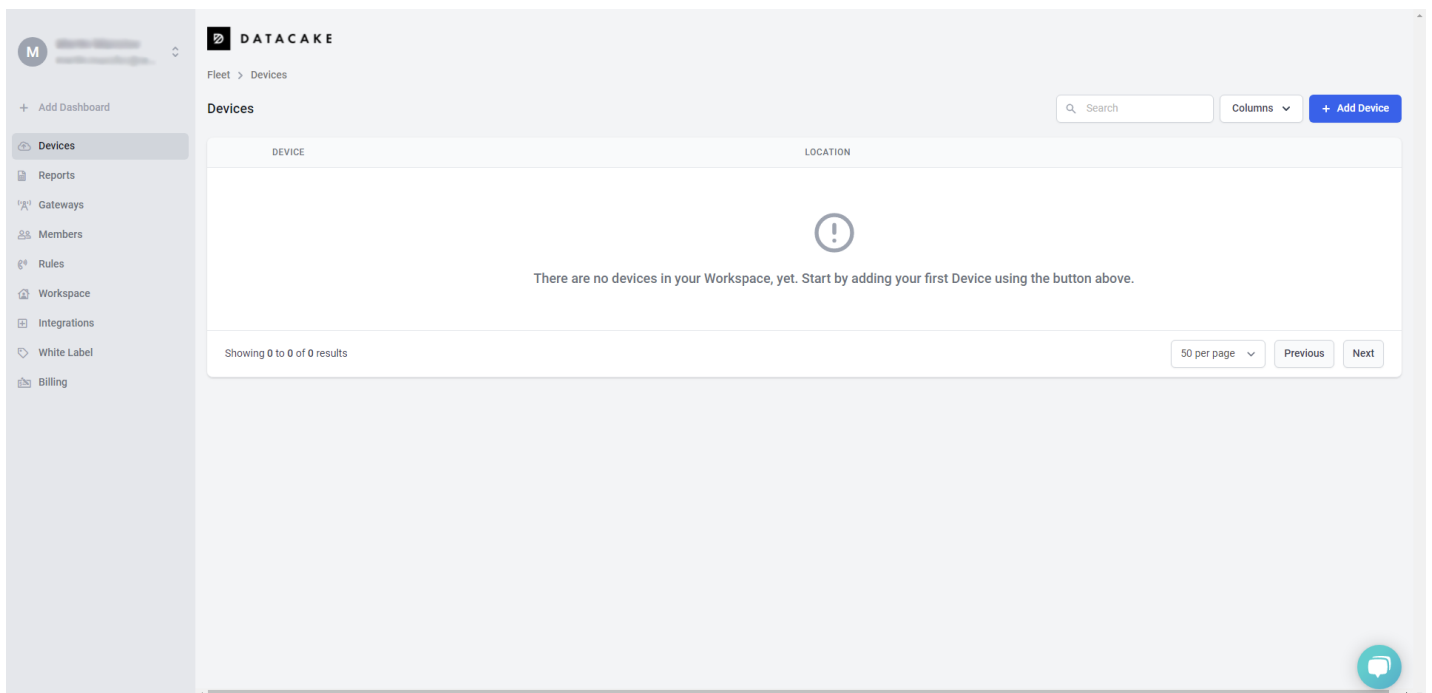
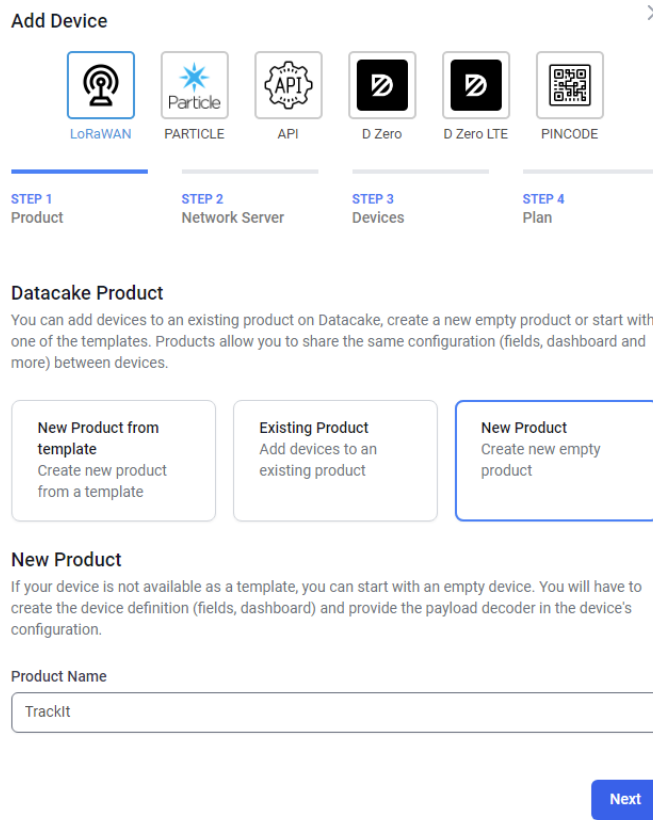


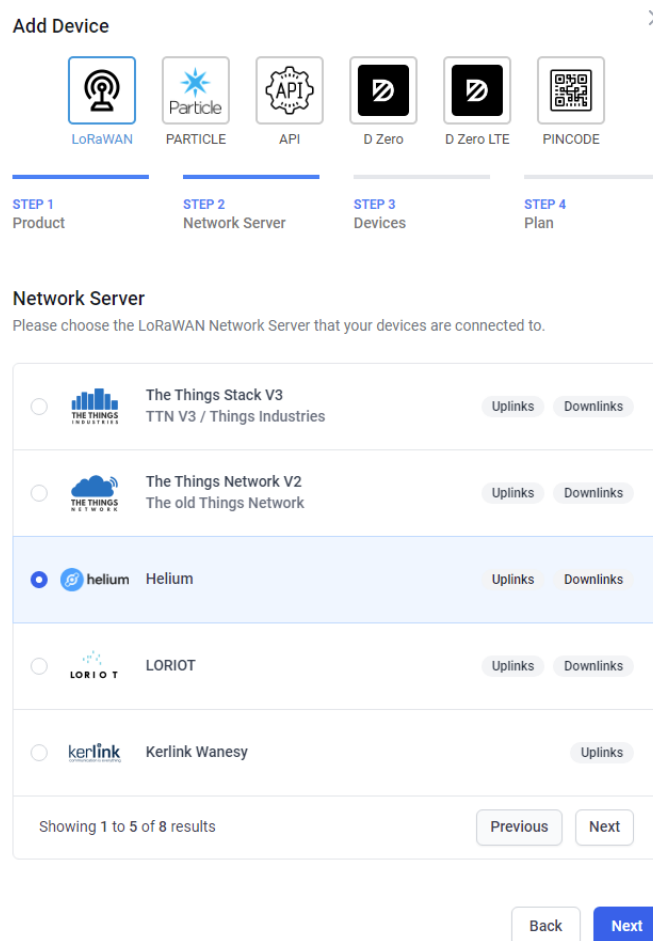
Figure 14: Datacake Console

2. Datacake has ready-to-use templates for different devices. However, as TrackIt is a new device, it doesn't have a ready-to-use template, so you need to make one on your own. To add the device, click on **New Product** and type the name of your product in the **Product Name** field.



**Figure 15:** Adding a New Product in Datacake

3. Click **Next** to continue. On the next window, for Network Server select **Helium** and click **Next** again.



**Figure 16:** Choosing Network Server

4. On the next page, you need to type the **Dev EUI** and the **Name** of your device. You can add more than one device. You can also drag and drop a .csv file and automatically add multiple devices at once. Click **Next**.



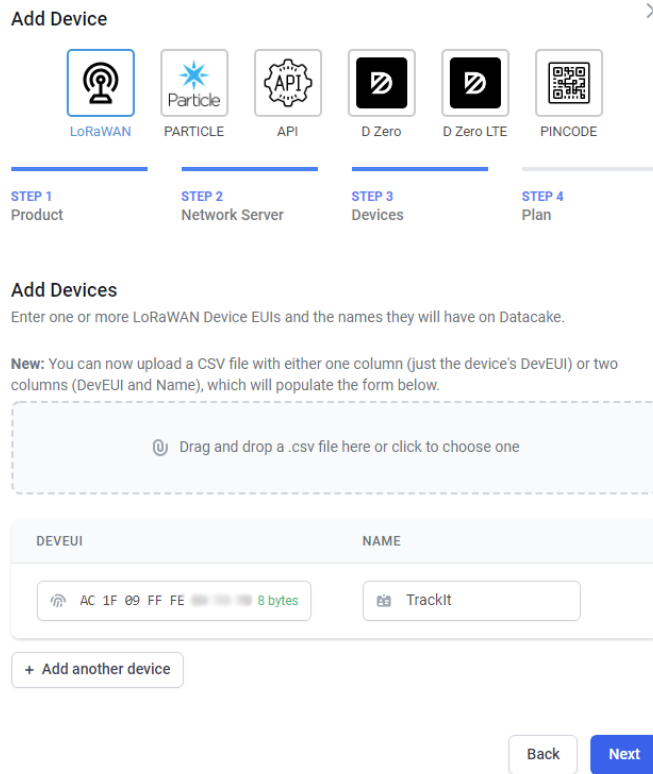


Figure 17: Adding the Device

5. On the next page, you have to choose a plan. We will choose the **Free** one for this example. Click on **Add 1 device** to finish.

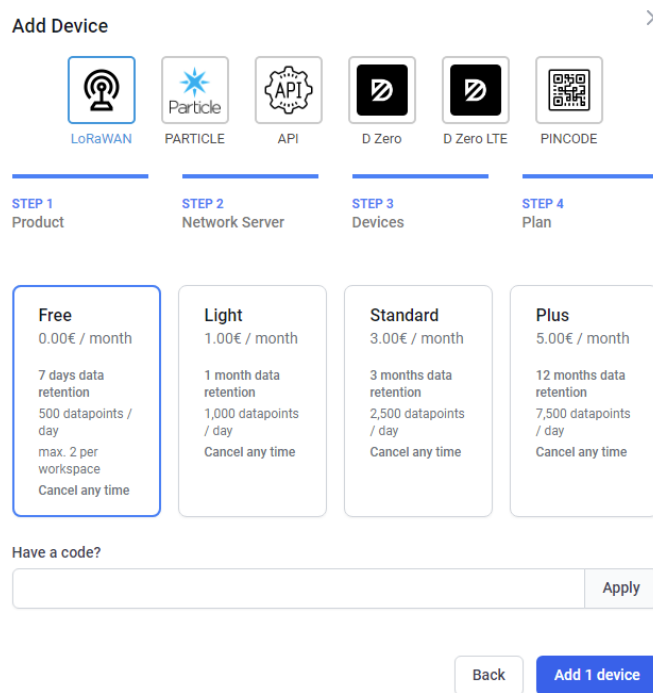


Figure 18: Choosing a plan

Now, your device is registered in Datacake.

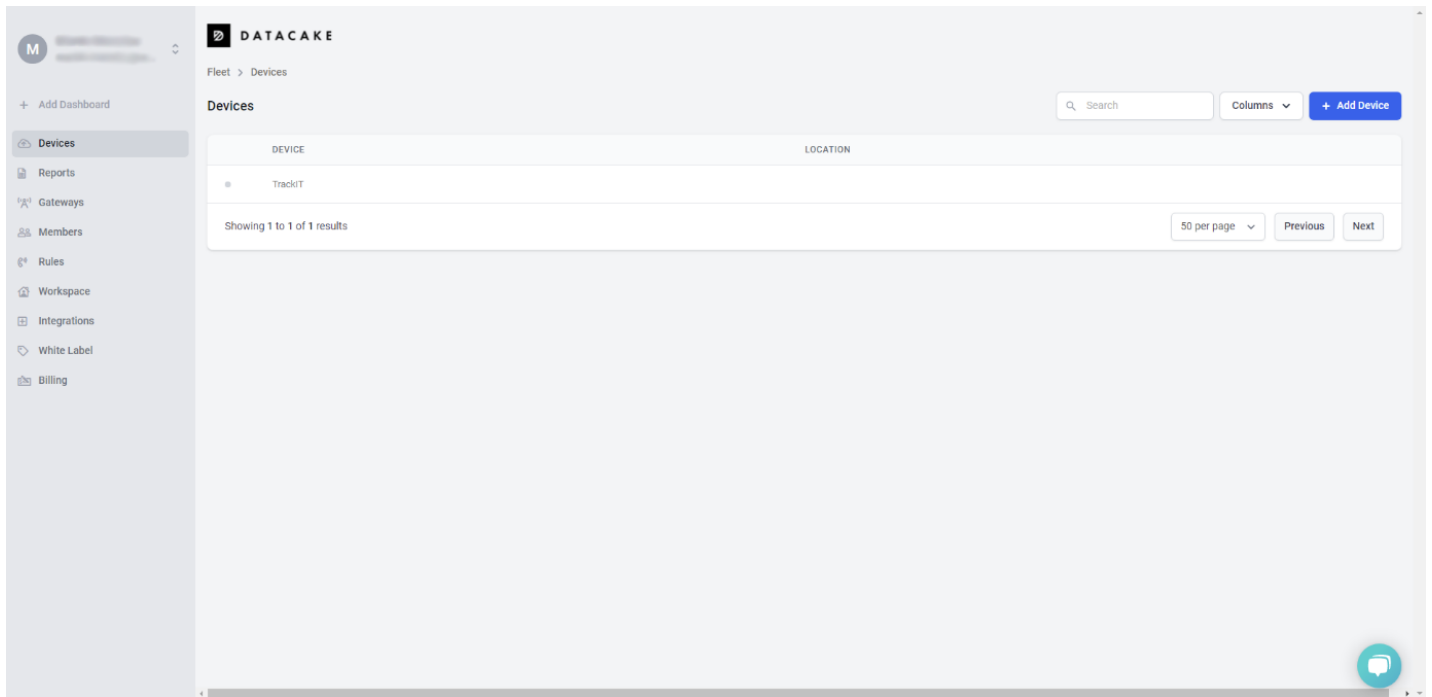


Figure 19: Successfully Registered Device

6. Before you carry on with setting up the Datacake Dashboard of the device, make sure that the device has already joined in Helium Console.

The screenshot displays the Helium Console interface for a device named "TrackIT". The page is organized into several sections:

- Header:** Includes a menu, search bar, and user information (RAKwireless, 7,899).
- Left Sidebar:** Contains navigation options like Flows, Nodes, Functions, Integrations, Configs, Alerts, Profiles, Multiple Packets, Admin, Coverage, Organizations, Data Credits, and Users.
- Device Details:**
  - Name: TrackIT
  - ID: bec1a8f1-99
  - Device EUI: AC1F09FF
  - App EUI: 6CAA59E
  - App Key: [Redacted]
  - Activation Method: OTAA
  - Profile: None
- Packets Transferred / DC Used:**
  - All Time: 227
  - Last 7 Days: 228
  - Last 30 Days: 228
  - Last 24 Hours: 0
- Labels:** 0 Labels Attached. + Add Label
- Flows:** A diagram showing a flow from TrackIT to Datacake.
- Real Time Packets:** A graph showing RSSI vs Time Past in Seconds. Legend: Integration Success (blue), Integration Error (orange), No Integration (red).
- Event Log:**
  - Expand All, Filter Events w/ Commands, Show Dropped Uplinks: Late, Inactive Device
  - Export JSON

Event	Type	No. of Hotspots	Time
+ [Failure]	Uplink	1	Jan 17, 2022 4:25:17.418 PM
+ [Success]	Join Accept	1	Jan 17, 2022 4:25:12.685 PM
+ [Success]	Join Request	2	Jan 17, 2022 4:25:10.684 PM
+ [Success]	Join Accept	1	Jan 17, 2022 4:24:59.055 PM
+ [Success]	Join Request	2	Jan 17, 2022 4:24:57.054 PM

Figure 20: Joined Device in Helium Console

## Configure the Device in Datacake

1. You need to decode the raw data that the device sends to Datacake, so you can preview it in a more user-friendly way. On the device page of the Datacake console, you can find different tabs like **Dashboard**, **History**, **Downlinks**, etc. For now, head on to the **Configuration** tab.

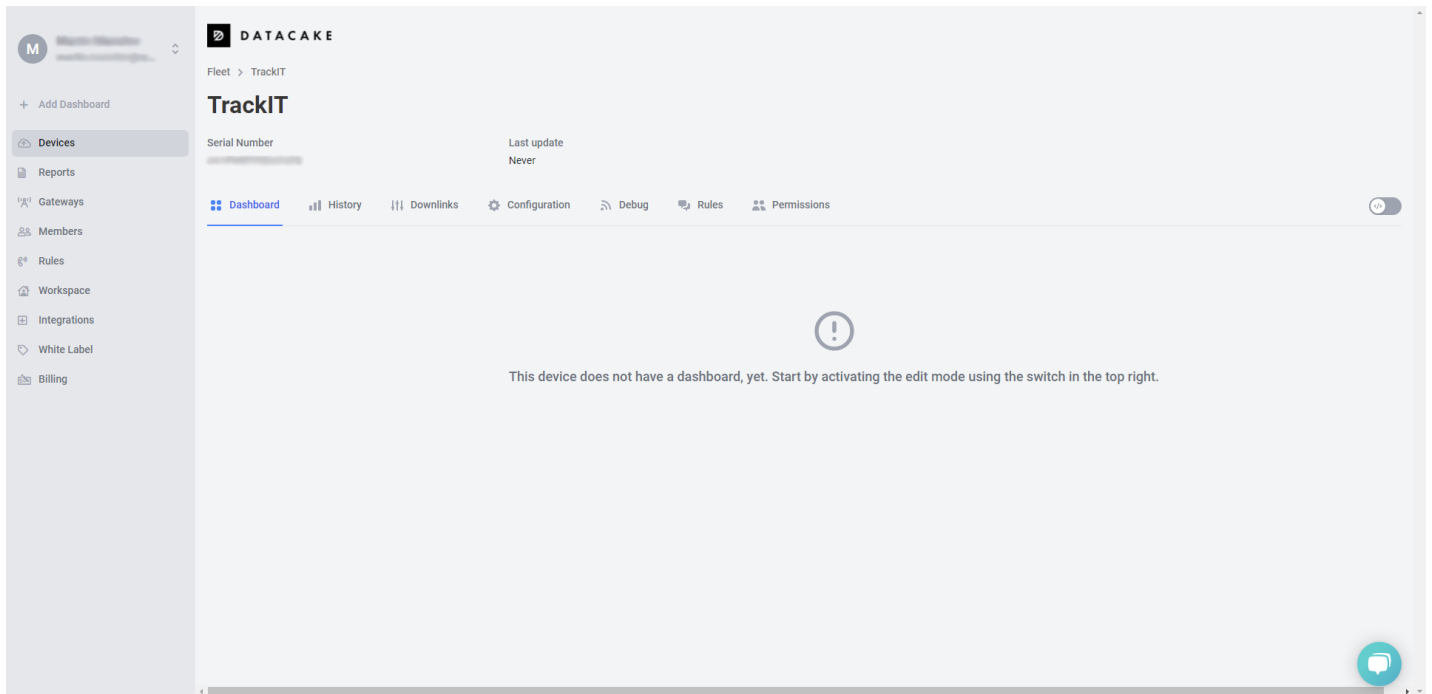


Figure 21: Device's Dashboard

2. Scroll down until you find the Payload Decoder field.

**DATA CAKE**

Fleet > TrackIT

## TrackIT

Serial Number
Last update  
Never

Dashboard
History
Downlinks
Configuration
Debug
Rules
Permissions

### General Configuration

Device Name:

Location:  Optional

Tags:

You can use tags to organize your devices and create filters on the dashboard

**Metadata**  
Metadata is displayed on the device overview and can be used in dashboards

**Online Timeout**  
 Minutes

Sets the time that has to pass without any new data coming in for the device to be considered offline.

Notify me if the device goes offline

Device can be claimed into other workspace  
When enabled, the device can be claimed into another workspace using the device's serial number and its claim code. Claiming does not remove the device from the current workspace.

**Product Slug**

Used by the MQTT broker

### LoRaWAN

**DevEUI**

**Network Server**

**Helium**

✔ Downlink configured

**Authenticate Uplink Webhooks** Product-wide setting

Learn more about securing the webhook

### Payload Decoder

Product-wide setting

When your devices sends data, the payload will be passed to the payload decoder, alongside the event's name. The payload decoder then transforms it to measurements.

```

1 function Decoder(payload, port) {
2   //
3   return [
4     {
5       | field: "TEST",
6       | value: 123
7     }
8   ]
9 }
10 }
    
```

**Payload**  **Port**

**Output**

**console.log Output**

**Recognized measurements**

### Fields

Fields describe the data the device will store.

NAME	IDENTIFIER	TYPE	CURRENT VALUE	LAST UPDATE
<p><b>No fields have been created, yet</b> <small>Create fields to define the scheme of the device.</small></p>				

### Integrations

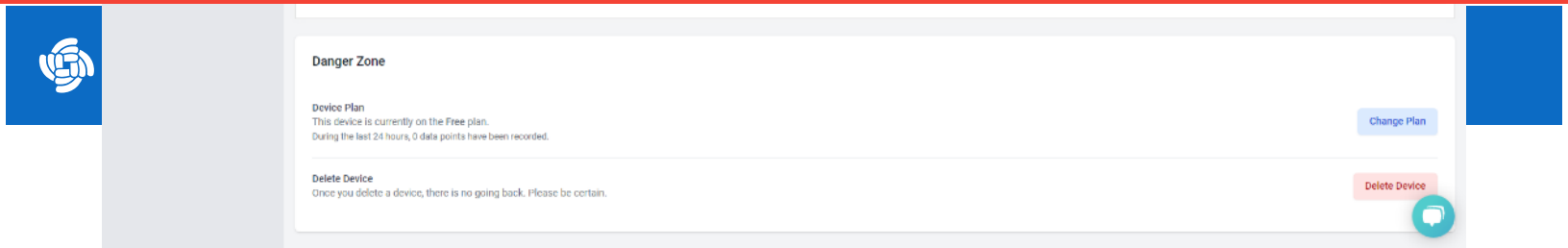


Figure 22: Device's Configuration Page

3. In the **Payload Decoder** field copy and paste the code below, then click on the **Save** button below the decoder field.

```

function Decoder(bytes, port) {
    var decoded = {};

    // adjust time zone, here Asia/Manila = +8H
    var my_time_zone = (8 * 60 * 60);

    decoded.num = bytes[1];
    decoded.app_id = (bytes[2] << 24) | (bytes[3] << 16) | (bytes[4] << 8) | bytes[5];
    decoded.dev_id = (bytes[6] << 24) | (bytes[7] << 16) | (bytes[8] << 8) | bytes[9];
    switch (bytes[0]) {
        case 0xCA: // No Location fix
            decoded.acc = 0;
            decoded.fix = 0;
            decoded.batt = bytes[10];
            decoded.time = ((bytes[11] << 24) | (bytes[12] << 16) | (bytes[13] << 8) | bytes[14])
            // adjust time zone
            decoded.time = decoded.time + my_time_zone;
            var dev_date = new Date(decoded.time * 1000);
            decoded.time_stamp = dev_date.getHours() + ":" + dev_date.getMinutes();
            decoded.date_stamp = dev_date.getDate() + "." + (dev_date.getMonth() + 1) + "." + dev
            decoded.stat = bytes[15] & 0x03;
            decoded.gps = bytes[15] & 0x0C;
            break;
        case 0xCB: // Location fix
            decoded.fix = 1;
            decoded.batt = bytes[20];
            decoded.time = ((bytes[21] << 24) | (bytes[22] << 16) | (bytes[23] << 8) | bytes[24])
            // adjust time zone
            decoded.time = decoded.time + my_time_zone;
            var dev_date = new Date(decoded.time * 1000);
            decoded.time_stamp = dev_date.getHours() + ":" + dev_date.getMinutes();
            decoded.date_stamp = dev_date.getDate() + "." + (dev_date.getMonth() + 1) + "." + dev
            decoded.stat = bytes[25] & 0x03;
            decoded.gps = bytes[25] & 0x0C;
            decoded.lng = (((bytes[10] << 24) | (bytes[11] << 16) | (bytes[12] << 8) | bytes[13])
            decoded.lat = (((bytes[14] << 24) | (bytes[15] << 16) | (bytes[16] << 8) | bytes[17])
            decoded.acc = bytes[18];
            decoded.gps_start = bytes[19];
            break;
        case 0xCC: // SOS
            decoded.sos = 1;
            decoded.lng = (((bytes[10] << 24) | (bytes[11] << 16) | (bytes[12] << 8) | bytes[13])
            decoded.lat = (((bytes[14] << 24) | (bytes[15] << 16) | (bytes[16] << 8) | bytes[17])
            if (bytes.length > 18) {
                var i;
                for (i = 18; i < 28; i++) {
                    decoded.name += bytes[i].toString();
                }
                for (i = 28; i < 40; i++) {
                    decoded.country += bytes[i].toString();
                }
                for (i = 39; i < 50; i++) {
                    decoded.phone += bytes[i].toString();
                }
            }
            break;
        case 0xCD:
            decoded.sos = 0;
            break;
        case 0xCE:
            decoded.alarm = 0x01;
            decoded.alarm_lvl = bytes[10];
    }
}

```



```
        break;
    }
    return decoded;
}
```

4. Now as you have a decoder, you can see the decoded data you receive in the **Debug tab**.

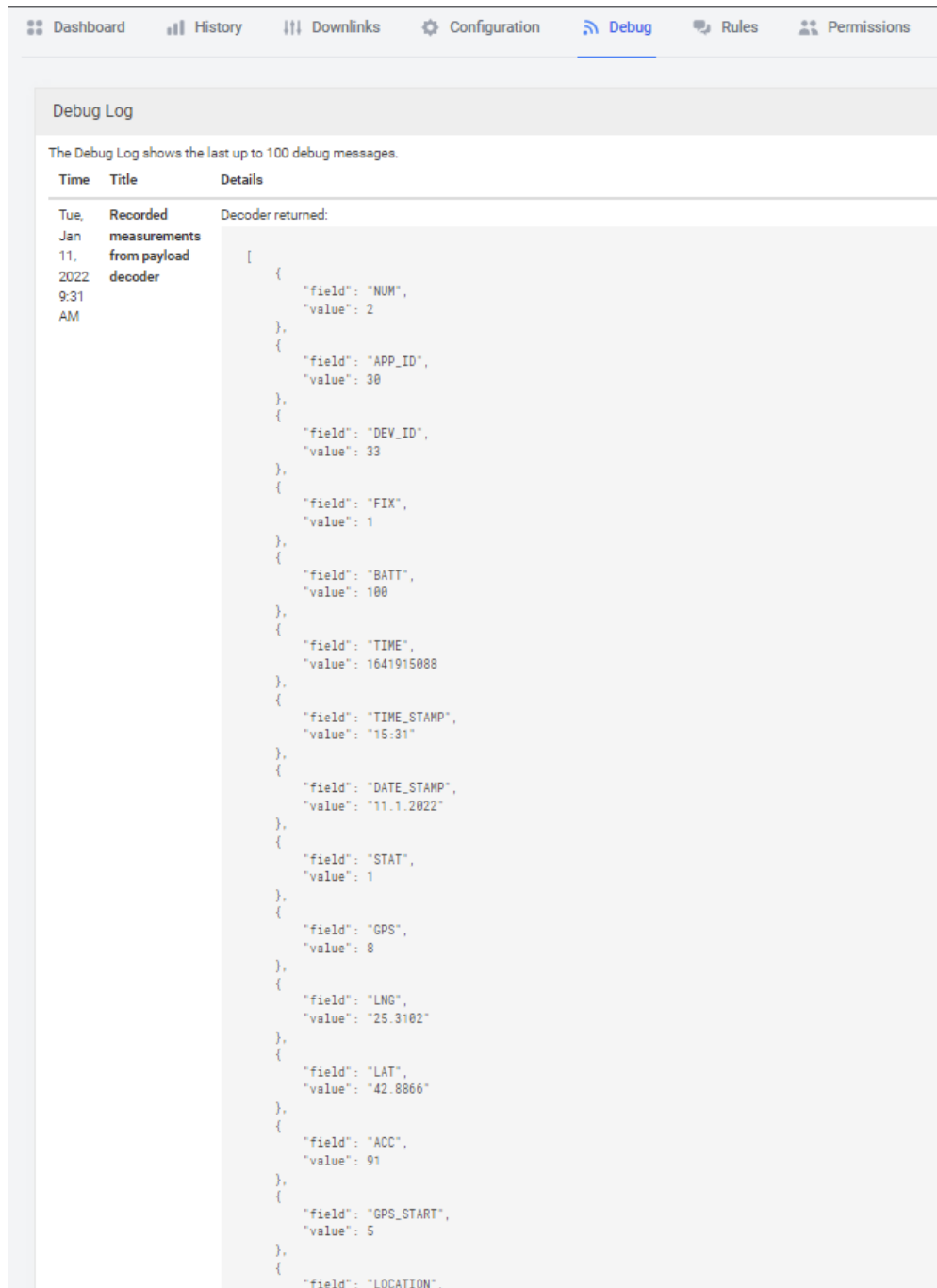


Figure 23: TrackIt Data

5. Now you need to create fields for the different data that the device sends. Below is a list of the fields you can create:

Type	Name	Identifier	Description
Integer	Frame Counter	NUM	Packets Counter
Integer	Application ID	APP_ID	The ID of the application
Integer	Device ID	DEV_ID	The ID of the device
Boolean	Fix	FIX	GPS fix



Type	Name	Identifier	Description
Integer	Battery	BATT	Battery Level
String	Time Stamp	TIME_STAMP	Time of the packet
String	Date Stamp	DATE_STAMP	Date of the packet
Integer	Status	STAT	1=sending 3=sending
Integer	GPS	GPS	GPS Status 0:open the GPS fix 4:locating 8:successful 12:failed
Integer	Accuracy	ACC	Accuracy of GPS
Geolocation	Location	LOCATION	GPS coordinates
String	Tracker ID	TRACKER_ID	The ID of the tracker
Integer	Alarm Level	ALARM_LVL	The level of the alarm set in the application
Integer	Alarm	ALARM	Shows if the alarm is triggered

6. To create a field, in the **Configuration** tab for the device in the Datacake console, scroll down to the **Fields** field. Click on the **+ Add Field** button. In the **Add Field** window, you need to fill in the following information based on the table above:

- **Type** – Type of the field.
- **Name** – Name of the field. Note that you can type a name of your choice. The above names are just a template.
- **Identifier** – The decoder decodes the data in fields (see Figure 24). The **Identifier** must be exactly the name of the field of the decoded data (e.g. **TIME\_STAMP** for the Time Stamp field). Note that field names cannot be changed.
- **Unit (optional)** – The unit of the value (e.g. V for battery (volts)).
- **Use formula** - Formulas can be used to perform calculations on values based on other fields.

**Add Field**
✕

Fields define the schema of the data the device stores.

**Type**

Float

**Name**

**Identifier**

The field identifier is a unique string that can consist of uppercase letters, numbers and underscores. Once a field has been created, the identifier can not be changed.

**Unit** Optional

**Formula** Optional

Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.

Use Formula

Cancel

Add Field

Figure 24: Adding a field

For example, here is how to create a **LOCATION** field. For **Type**, choose **Geolocation** and for **Identifier** type the name of the decoded data field: **LOCATION**.

**Add Field**
✕

Fields define the schema of the data the device stores.

**Type**

Geolocation

**Name**

Location

**Identifier**

LOCATION

The field identifier is a unique string that can consist of uppercase letters, numbers and underscores. Once a field has been created, the identifier can not be changed.

**Unit** Optional

**Formula** Optional

Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.

Use Formula

Cancel

Add Field

Figure 25: Location field

7. When you enter the required information, click **Add Field**. You will see the created field in the Field tab. Once a packet is received, the field will take the value of the identifier.

**Fields**

Fields describe the data the device will store.

+ Add Field

NAME	IDENTIFIER	TYPE	CURRENT VALUE	LAST UPDATE
Location	LOCATION	Location	42.885900, 25.311600	a few seconds ago

**Figure 26:** Successfully Created Field

Another example is if you want to create a field for the battery level. Click again on **+ Add Field**. For **Type** select **Integer**, for a name - **Battery** and for Identifier type **BATT**.

**Add Field** ✕

Fields define the schema of the data the device stores.

**Type**

Integer

**Name**

Battery

**Identifier**

BATT

The field identifier is a unique string that can consist of uppercase letters, numbers and underscores. Once a field has been created, the identifier can not be changed.

**Unit** Optional

**Formula** Optional

Formulas can be used to perform calculations on values from other fields. Fields that have a formula can not be written to from a decoder or via the API.

Use Formula

**Figure 27:** Add Battery Field

8. Do the same with the other fields. It is up to you to decide on what fields are required for your project. Make sure you select the right **Type** and type the correct **Identifier**.

9. Now that the fields are created, head to the **Dashboard** of the device. Activate the **Edit mode** using the switch (



) and click on **+ Add Widget**. Here you can create a widget to preview the data in the dashboard.

Choose the correct widget for the desired field.

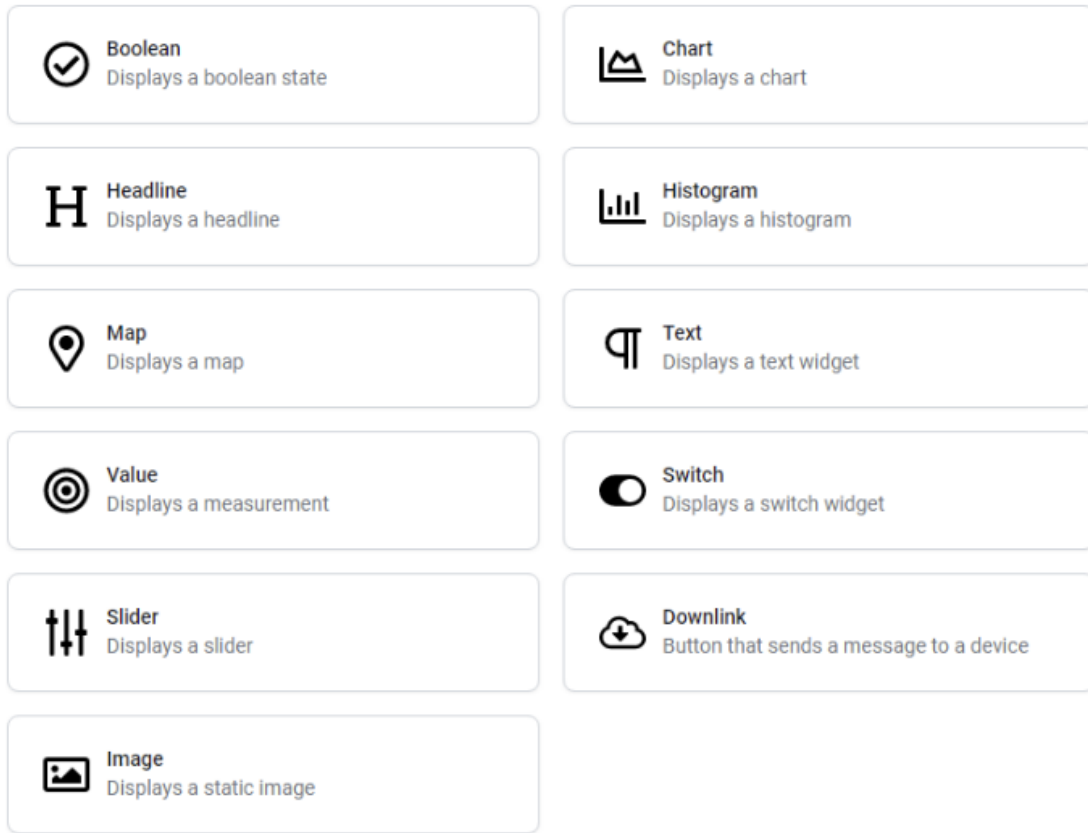


Figure 28: Datacake Widgets

For example, when creating a map, click on the **Map (Displays a map)** widget.

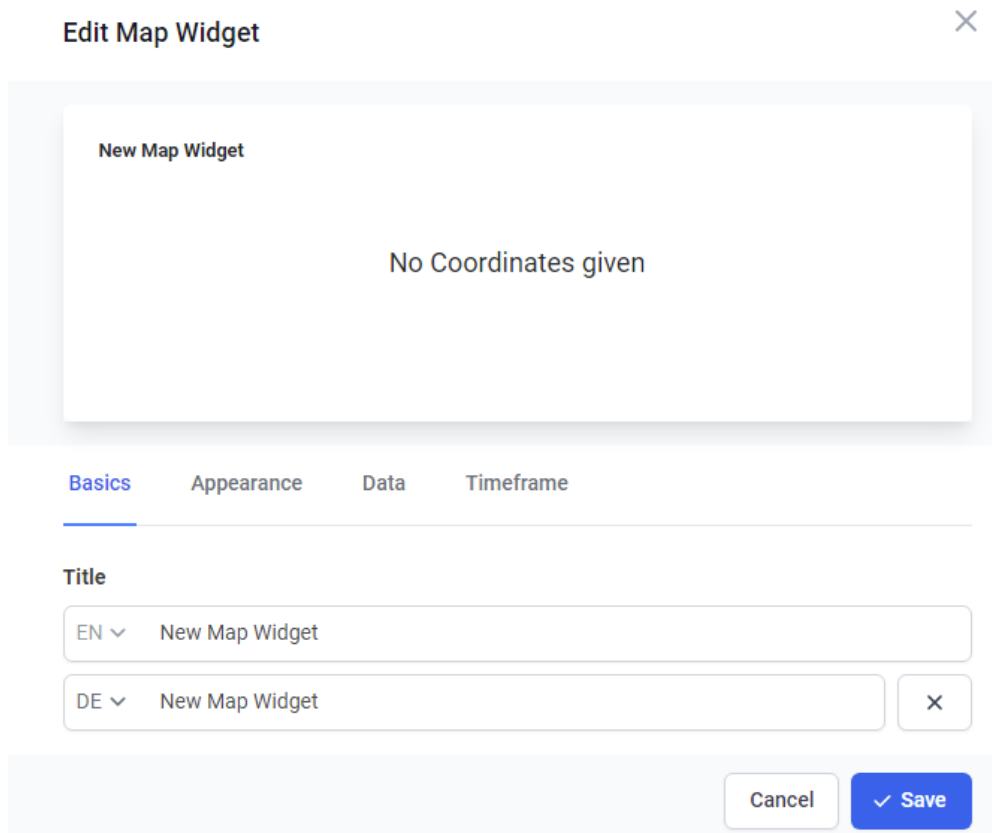


Figure 29: Map Widget

- In the **Basics** tab, you can type a name for the widget. In the **Appearance** tab, you can set a design to your widget (color, style). The **Data** tab is most important. Here you need to select the location field you have created above. Click on the **+ Add field** and select the **Location** field. In addition, in the **Timeframe** tab, you can enable whether the map will show historical data.

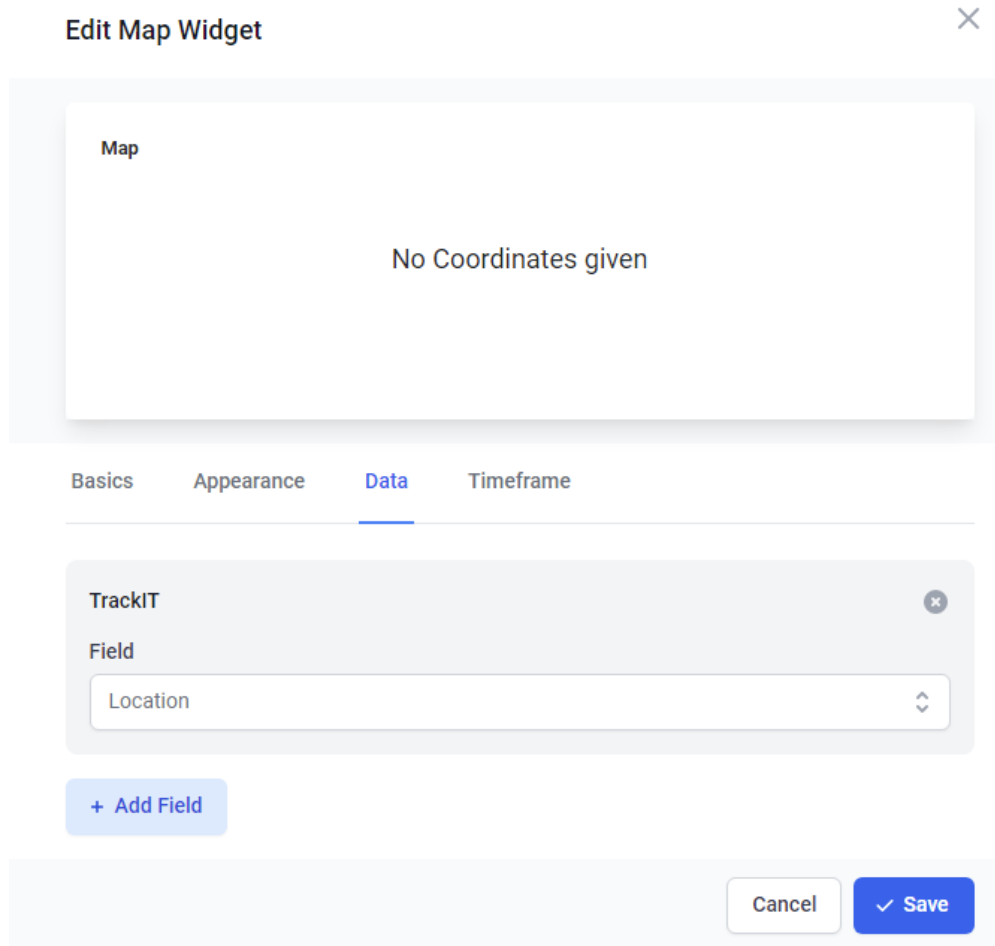


Figure 30: Map Field

11. Once, you have set the field, click on **Save** and you will see the map in the Dashboard.

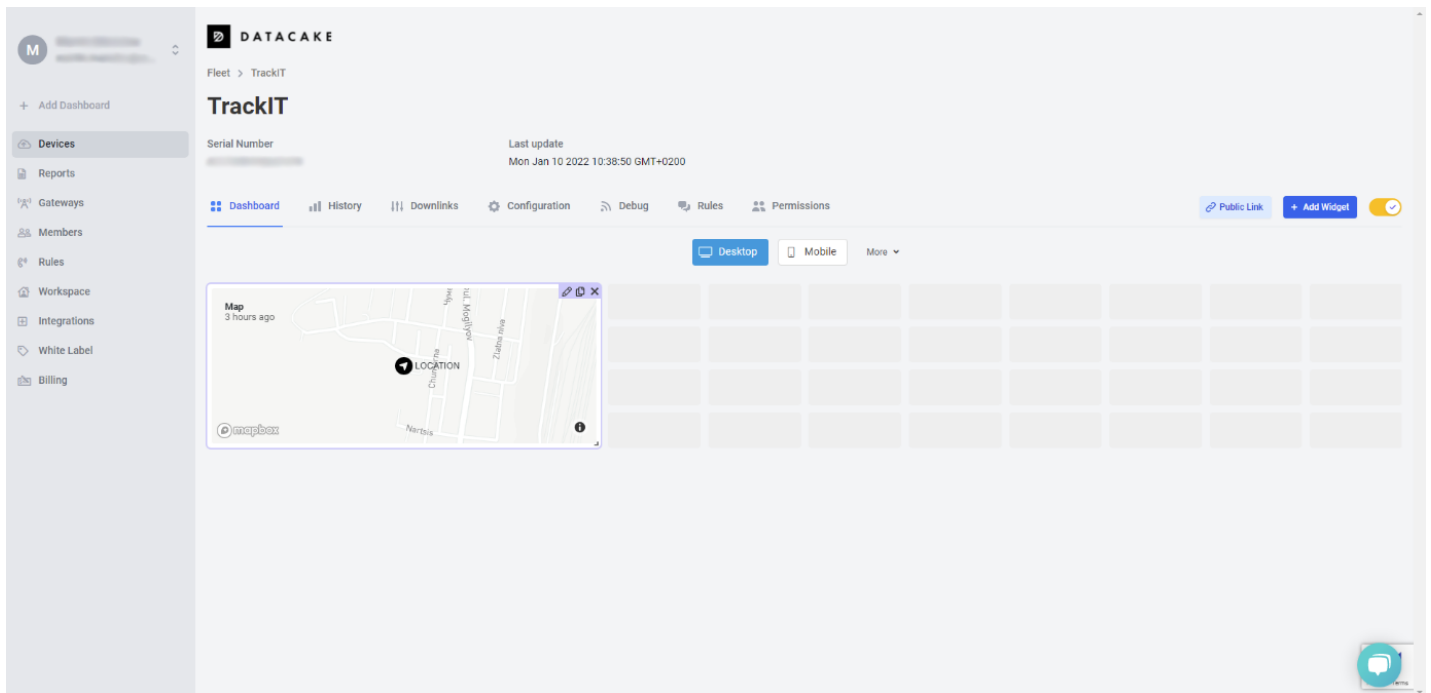


Figure 31: Datacake Map

In addition, if you want to add a widget to see the battery level of the device, click again on **+ Add Widget** and select **Value**. In the **Data** tab, select **Battery** for **Field**.

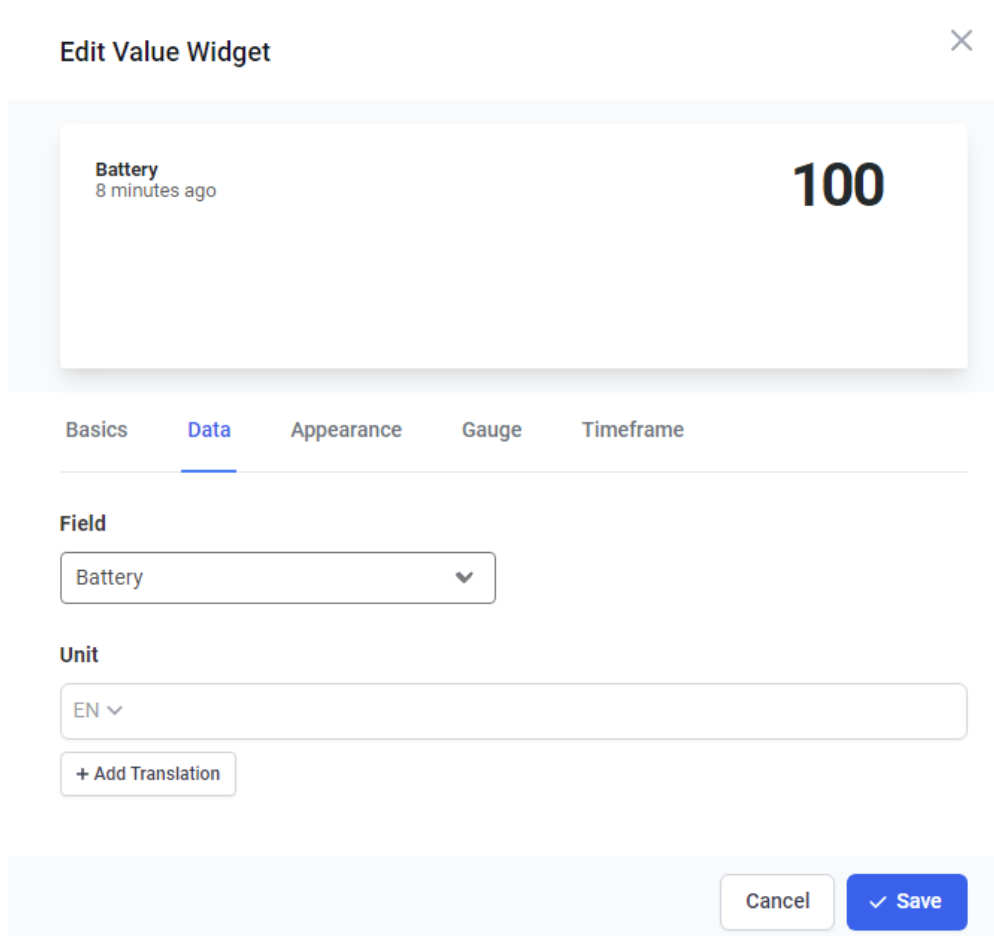


Figure 32: Add Battery Widget

Click **Save** to add the widget.

- If you want, you can create other widgets. For string/integer field types, choose **Value (Displays a measurement)** when creating the widget, for **Boolean** fields – **Boolean (Displays a boolean state)** widget, and so on.

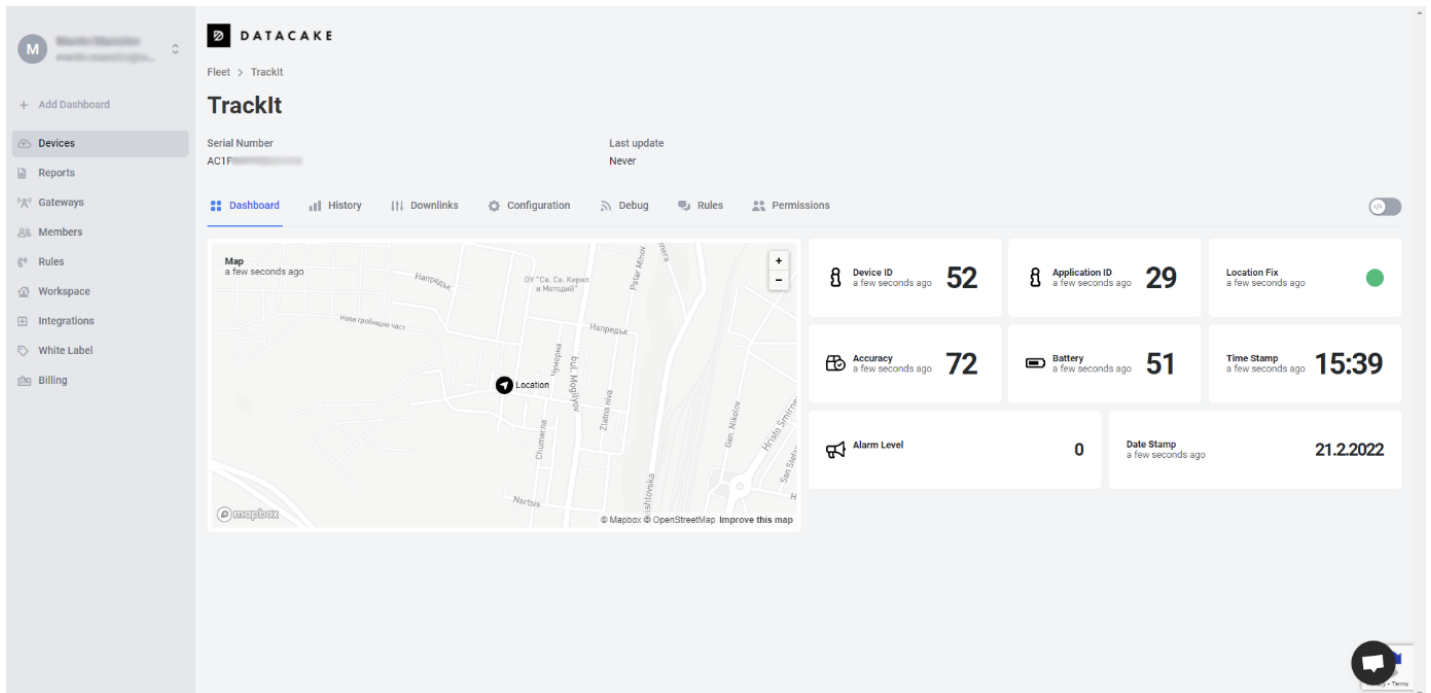


Figure 33: TrackIt Dashboard

- Once you add your widgets, you can customize your Dashboard depending on your needs. Remember, when you finish customizing your Dashboard, deactivate the **Edit mode** by clicking the **yellow switch** to save your changes.