Smart Cities Hackathon Tier 1 Challenge 1

Tier 1 Challenge 1: Create a virtual sensor for the other challenges

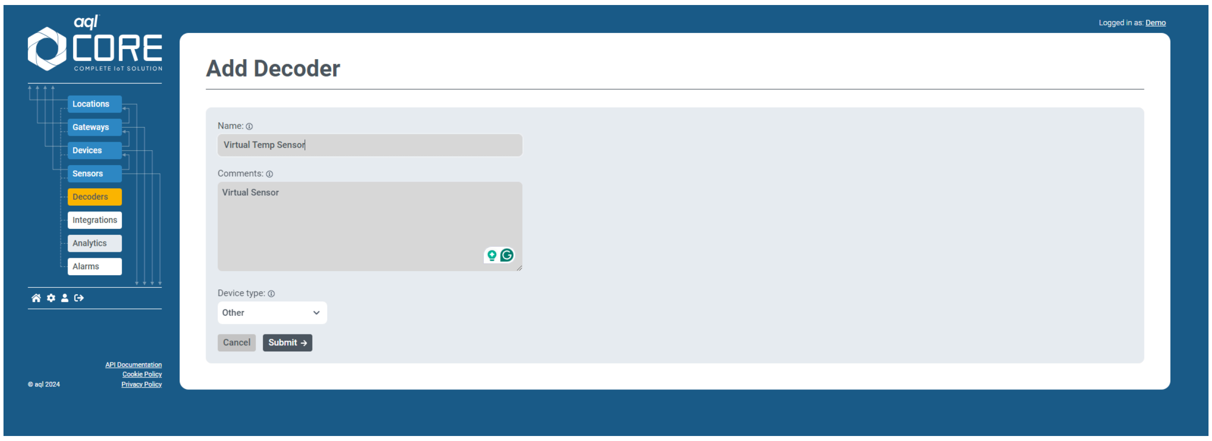
This challenge will see you create a virtual sensor for core.aql.com. This will allow you to upload readings via the API and via MQTT to simulate readings. We will use this sensor for other Tier 2 and Tier 3 Challenges. This challenge will allow you to learn about https://core.aql.com end to end, so you have the knowledge and skills to complete and build upon the other challenges. Good luck and enjoy. You will learn how to make a Curl Request to https://api.core.aql.com/ and write Javascript for your decoder

Step 1

Log into the aql Core IoT platform (https://core.aql.com/login) with your provided event credentials.

Step 2

Select the decoders option from the left-hand menu. Select “Add Decoder”. Enter a name for your virtual sensor using your team name and type of sensor. For example Team1-Temp-Virtual Sensor. Enter a comment and select the type as “Other”. Click “Submit”.



Add Decoder

Name: Virtual Temp Sensor

Comments: Virtual Sensor

Device type: Other

Cancel Submit →

Step 3

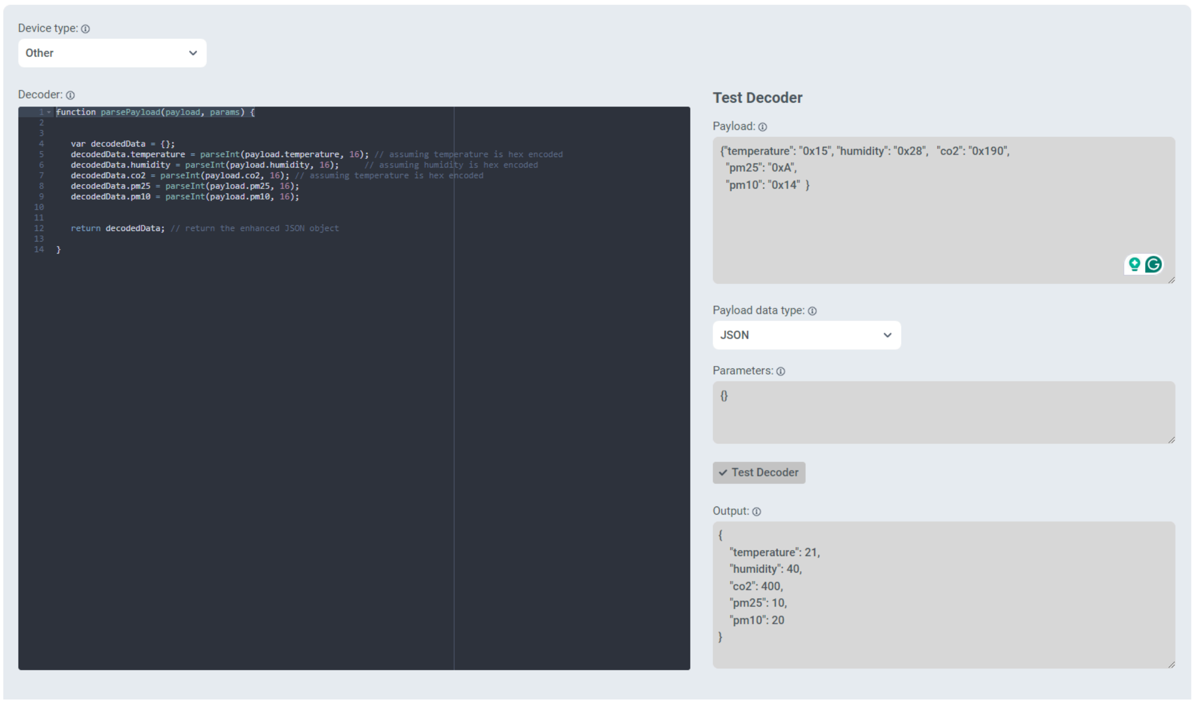
The decoder page allows you to create complex logic to parse sensor readings. Sensor readings are normally returned as a byte array. In this example, we are going to build a JSON object of any sensor reading. This is a good task for the AI Assistant, you can ask it to create a JSON object for a virtual sensor, below is an example of a starter JSON object, where the data being uploaded is in HEX. This JSON should be dropped into the “Payload” box on the configuration screen for your new decoder.

```
{
  "temperature": "0x15",
  "humidity": "0x28",
  "co2": "0x190",
  "pm25": "0xA",
  "pm10": "0x14"
}
```

Add the following Javascript decoder to the body of the Function `parsePayload(payload, params) {`

```
var decodedData = {};
decodedData.temperature = parseInt(payload.temperature, 16); // assuming
temperature is hex encoded
decodedData.humidity = parseInt(payload.humidity, 16);      // assuming
humidity is hex encoded
decodedData.co2 = parseInt(payload.co2, 16); // assuming temperature is hex
encoded
decodedData.pm25 = parseInt(payload.pm25, 16);
decodedData.pm10 = parseInt(payload.pm10, 16);


return decodedData; // return the enhanced JSON object
```



Device type: Other

Decoder:

```
function parsePayload(payload, params) {


   var decodedData = {};
   decodedData.temperature = parseInt(payload.temperature, 16); // assuming temperature is hex encoded
   decodedData.humidity = parseInt(payload.humidity, 16);      // assuming humidity is hex encoded
   decodedData.co2 = parseInt(payload.co2, 16); // assuming temperature is hex encoded
   decodedData.pm25 = parseInt(payload.pm25, 16);
   decodedData.pm10 = parseInt(payload.pm10, 16);


   return decodedData; // return the enhanced JSON object

}
```

Test Decoder

Payload:

```
{"temperature": "0x15", "humidity": "0x28",  "co2": "0x190",
 "pm25": "0xA",
 "pm10": "0x14" }
```

Payload data type: JSON

Parameters:

```
{}
```

Output:

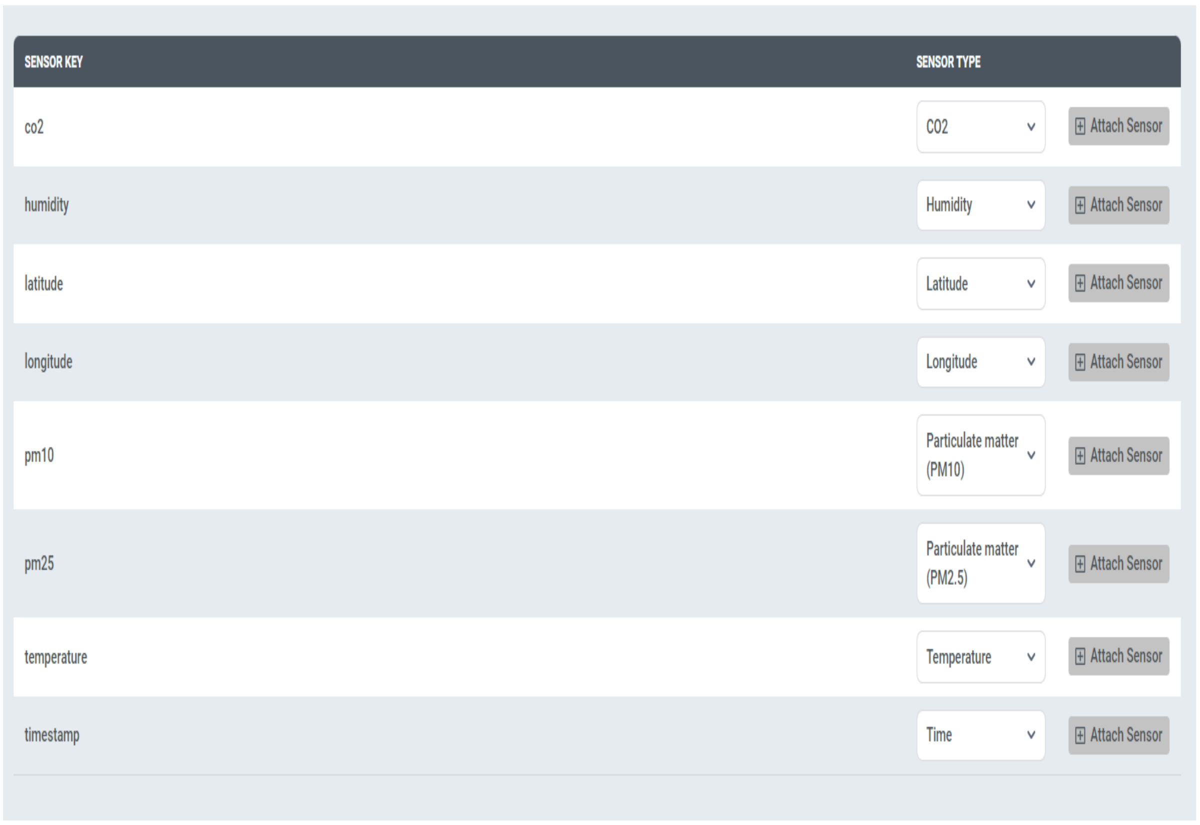
```
{
  "temperature": 21,
  "humidity": 40,
  "co2": 400,
  "pm25": 10,
  "pm10": 20
}
```

Step 4

Select “Test Decoder” and the output should match the JSON ingest, but the values should now be displayed as Int values instead of the hex values provided. You could extend this to be a series of bytes instead of a hex value.

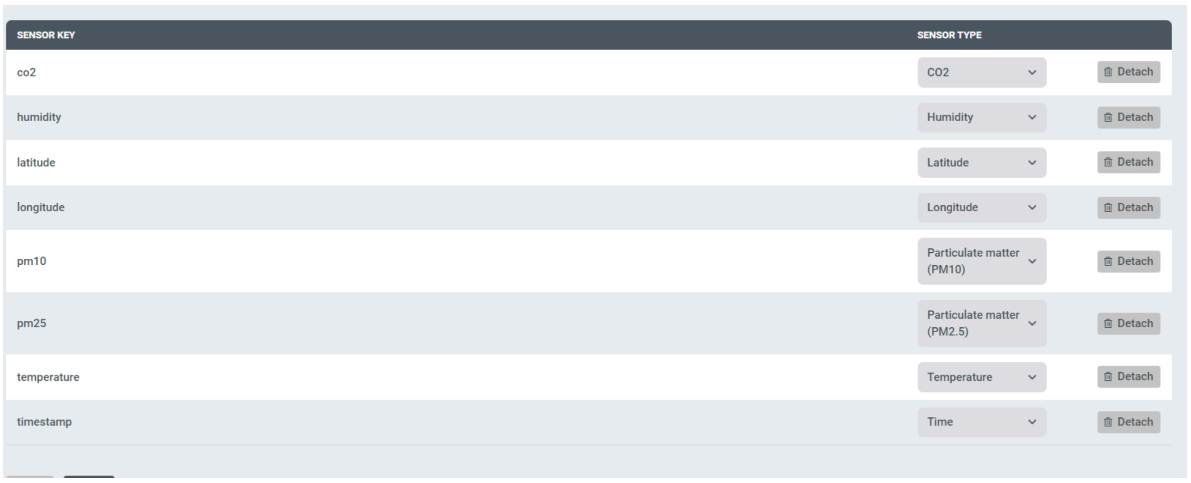
Step 5

Scroll down the page, and you will see the “Sensor Mapping” element of the page. This is where you map the JSON output to the list of supported sensors for core.aql.com. Core.aql.com will attempt to automatically match up each sensor key to the correct sensor type. If any look incorrect, click on the sensor dropdown and select the correct sensor.



SENSOR KEY	SENSOR TYPE	
co2	CO2	Attach Sensor
humidity	Humidity	Attach Sensor
latitude	Latitude	Attach Sensor
longitude	Longitude	Attach Sensor
pm10	Particulate matter (PM10)	Attach Sensor
pm25	Particulate matter (PM2.5)	Attach Sensor
temperature	Temperature	Attach Sensor
timestamp	Time	Attach Sensor

Once you are happy with the key and the sensor type select “Attach Sensor”.



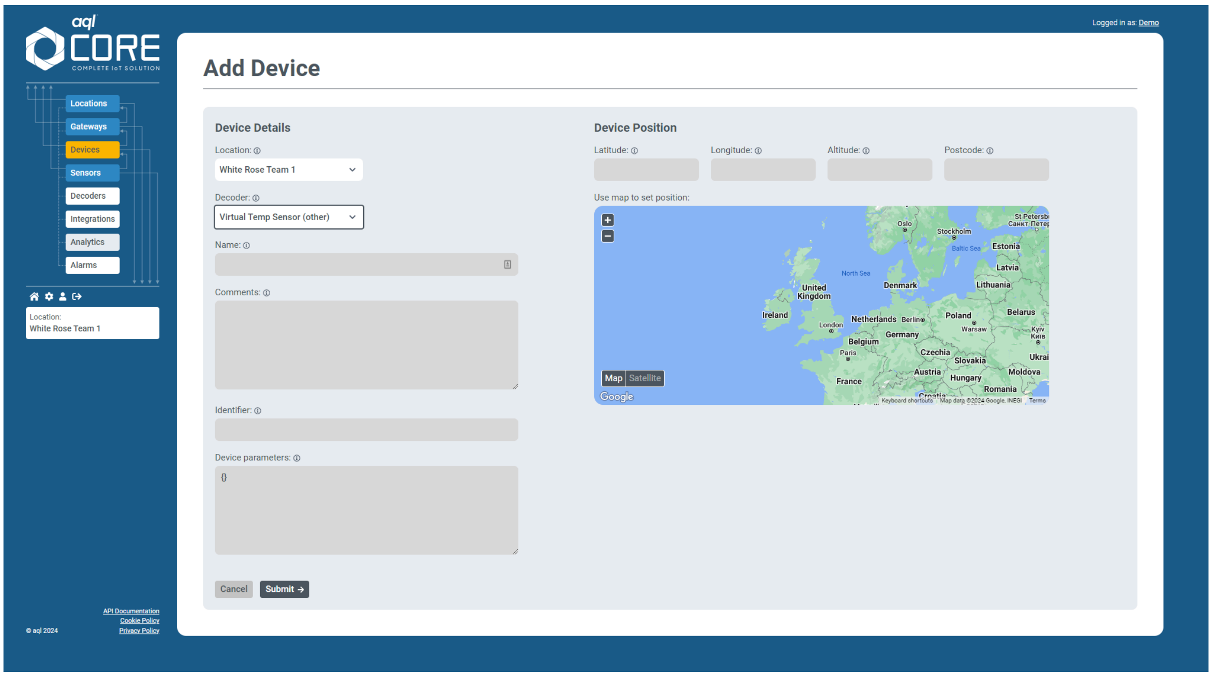
SENSOR KEY	SENSOR TYPE	
co2	CO2	Detach
humidity	Humidity	Detach
latitude	Latitude	Detach
longitude	Longitude	Detach
pm10	Particulate matter (PM10)	Detach
pm25	Particulate matter (PM2.5)	Detach
temperature	Temperature	Detach
timestamp	Time	Detach

Once you have attached all the sensors finally hit the SAVE button.

Step 6

Next, select devices from the left-hand menu. Select your team from the location dropdown, and select “Add Device”.

Step 7



Location:
White Rose Team 1

Add Device

Device Details

Location: White Rose Team 1

Decoder: Virtual Temp Sensor (other)

Name:

Comments:

Identifier:

Device parameters: {}

Device Position

Latitude:

Longitude:

Altitude:

Postcode:

Use map to set position:

Step 7

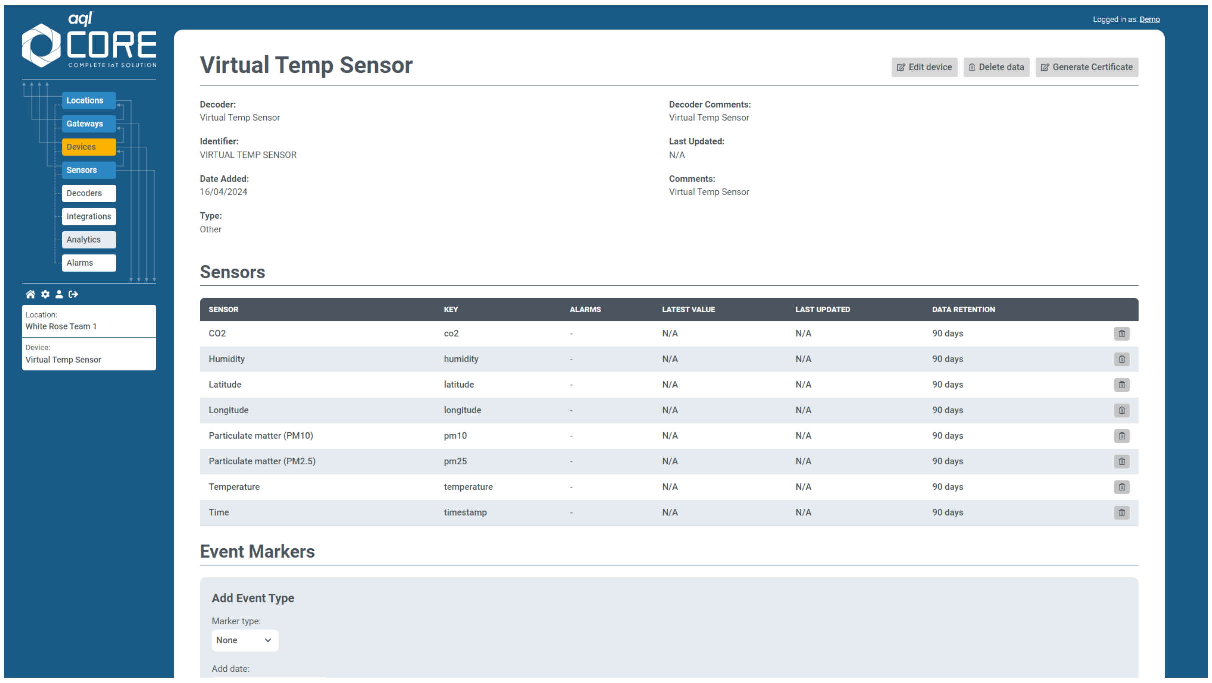
Select your newly created decoder from the dropdown menu, then enter a name and identifier, plus a comment that describes your virtual device.

Enter a unique device name. You can then assign some device parameters; these are JSON objects that you can use within the decoder that have device specific parameters that the sensor doesn't return. For example,

```
{
  "tag1" : 1,
  "tag2" :2
}
```

Next enter the GPS locations for today's event. If you aren't sure you can ask your AI Assistant for these details.

Finally, select Save and your new device is ready to test.



Location:
White Rose Team 1

Device:
Virtual Temp Sensor

Virtual Temp Sensor

Decoder:
Virtual Temp Sensor

Decoder Comments:
Virtual Temp Sensor

Identifier:
VIRTUAL TEMP SENSOR

Last Updated:
N/A

Date Added:
16/04/2024

Comments:
Virtual Temp Sensor

Type:
Other

Sensors

SENSOR	KEY	ALARMS	LATEST VALUE	LAST UPDATED	DATA RETENTION
CO2	co2	-	N/A	N/A	90 days
Humidity	humidity	-	N/A	N/A	90 days
Latitude	latitude	-	N/A	N/A	90 days
Longitude	longitude	-	N/A	N/A	90 days
Particulate matter (PM10)	pm10	-	N/A	N/A	90 days
Particulate matter (PM2.5)	pm25	-	N/A	N/A	90 days
Temperature	temperature	-	N/A	N/A	90 days
Time	timestamp	-	N/A	N/A	90 days

Event Markers

Add Event Type

Marker type:
None

Add date:

Step 8

Record the device ID for later by checking the core URL for the device, which is in the format; https://core.aql.com/devices/YourDeviceID

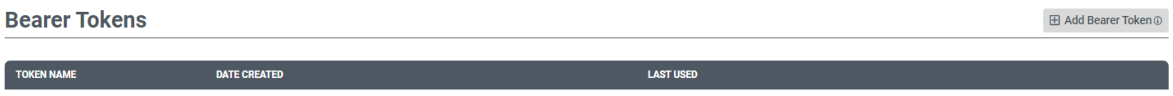
You will now see your new virtual device, with all the sensors we created in the earlier decoder.

If you add further sensors to your decoder, these will appear automatically when the initial reading is uploaded.

Step 9

We are now going to create a bearer token so we can upload an initial reading. Select the settings option in the bottom left of the nav bar.

Create a new bearer token and allocate a name for this token.



Bearer Tokens

TOKEN NAME	DATE CREATED	LAST USED

Copy the token as you will need this for the JavaScript later.

Step 10

Install Postman or curl.

Curl:

https://curl.se/windows/

Postman:
https://www.postman.com/

Step 11

We are going to upload a sensor reading for the newly created device. We can use this step for future challenges. The request will send the same sensor data we provided to test the decoder. This will take the supplied JSON object and run your newly created decoder to translate the reading into a JSON object.

Note that, because we have attached the hex-decoder, this device will ONLY correctly receive data in hexadecimal number format - e.g. 0x14A

The documentation for this can be found at https://api.core.aql.com/doc/

Using Curl:
Replace the device ID and your bearer token from earlier in the snippet below.

Unset
```
curl --location
'https://api.core.aql.com/v1/devices/external-data-webhook/other/YOURDEVICEI
D \
--header 'Authorization: Bearer YOUR_BEARER_TOKEN' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data '{
  "temperature": "0x15",
  "humidity": "0x28",
  "co2": "0x190",
  "pm25": "0xA",
  "pm10": "0x14"
}'
```

On some platforms, successful submission will return a 201. On other platforms, no return will be generated for a submission, but will be absent of error messages.

Using Postman:

Tier 1: Challenge 1

Open Postman and click on “New” in the upper left corner
Choose the HTTPS option, selecting “POST” from the request type dropdown

Then enter the request URL, replacing the device ID;
https://api.core.aql.com/v1/devices/external-data-webhook/other/YOUR_DEVICE_ID

Select the “Headers” tab and click “bulk edit” to add the following, replacing the token with your bearer from earlier

Unset
```
Authorization: Bearer YOUR_BEARER_TOKEN
Accept: application/json
Content-Type: application/json
```

Select the “Body” tab and click the “raw” option, ensuring the content type is set to “JSON”
Add the following JSON data;

Unset
```
{
  "temperature": "0x15",
  "humidity": "0x28",
  "co2": "0x190",
  "pm25": "0xA",
  "pm10": "0x14"
}
```

Hit “SEND” - in the event of a successful submission, a 201 will be returned.

Step 12

Navigate back to your device http://core.aql.com/devices/YOURDEVICE_ID

You will see the data frames in the logging section and the latest reading value will be displayed.

Step 13

Tier 1: Challenge 1

Attempt to upload different readings by adjusting the values or number of lines in the JSON input, then resubmit.

Navigate back to your device to confirm this new data frame has been logged.

You have now created your virtual sensor and can upload readings. Well done! You can now move on to the next challenge.