

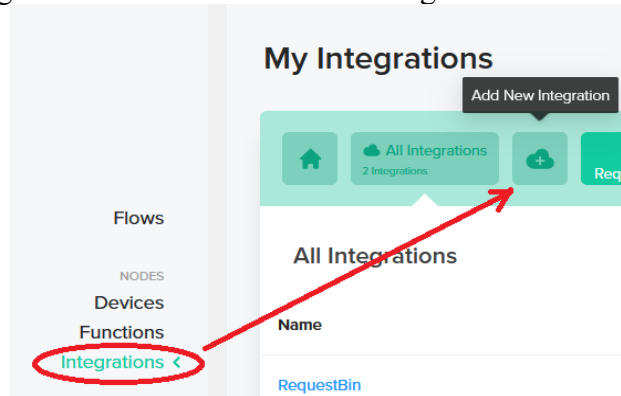
Follow the steps in the Light-Tracker Adding Device on Helium Console but instead of proceeding to the [Cayenne myDevices Integration with Helium Console](#) (at the bottom of the page) follow the next steps.

Sondehub Integration with Helium Console

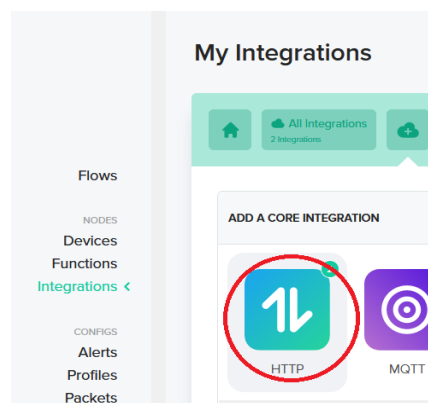
In a previous tutorial [Adding Device on Helium Console](#) you have successfully added a new device and seen your data flowing. The next step is to get that data in the correct format to SondeHub. The following steps show how.

Integrations

- 1) click on the “Integrations and then “Add New Integration” button.



- 2) select HTTP



- 3) set up the SondeHub endpoint details: - see the Endpoint URL to: `https://api.v2.sondehub.org/helium` and the endpoint type to “POST”

STEP 2 - ENDPOINT DETAILS

POST GET PUT PATCH

Endpoint URL (Required)
`https://api.v2.sondehub.org/helium`

HTTP Headers (Optional usage for payload interpolation)

Key	Value
+ Add Header	

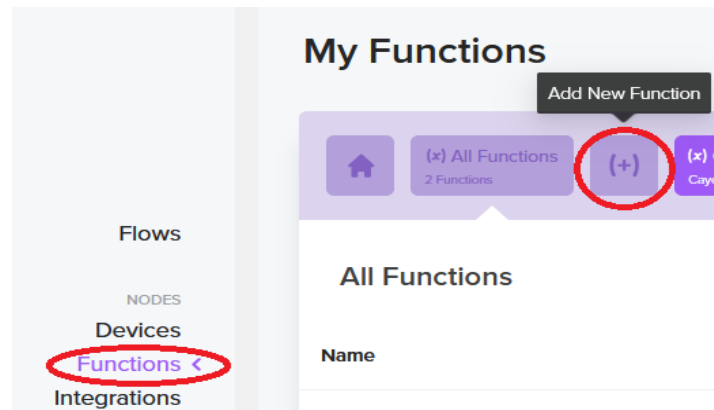
URL Params (Optional usage for payload interpolation)

+ Add Param

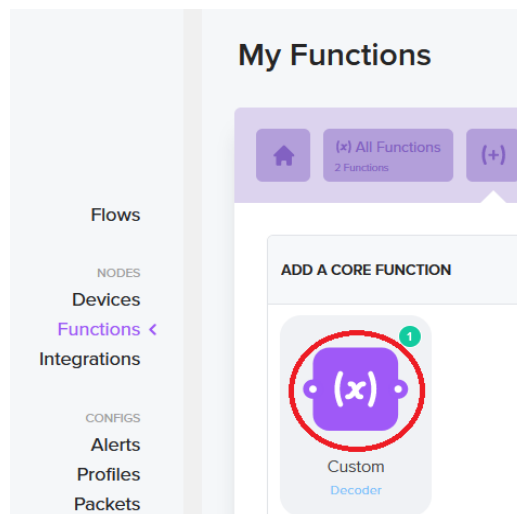
At the bottom of the page give the integration a name (such as “SondeHub”) and add it.

Functions

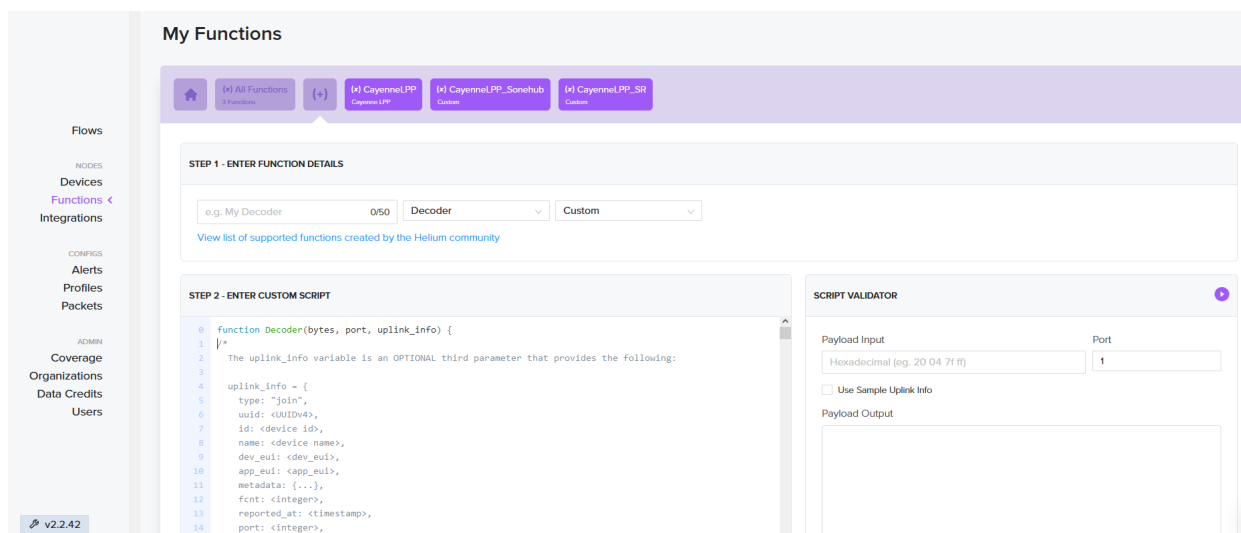
- 4) The next step is to create a Function – this is used to convert the Cayenne LPP data collected by Helium into the format required by SondeHub. Click on “Functions” and “Add New Function”.



Then click on “Custom”.



Once inside the Custom Decoder page there are a couple of steps to complete.



1: Give the Function a name such as “CayenneLPP_SondeHub”

2: Download Decoder.js from here:

<https://github.com/SteveRan/Helium-Sonedhub-Cayenne-decoder/blob/main/Decoder.js>

Helium-Sonedhub-Cayenne-decoder / Decoder.js

SteveRan Create Decoder.js

124 lines (114 loc) • 4.02 KB Code 55% faster with GitHub Copilot

CodeBlame

RawCopyDownloadEditDropdownCode

```
1  function Decoder(bytes, port) {
2
3      function Unsigned1Byte(index) {
4          return bytes[index];
5      }
6
7      function Unsigned2Byte(index) {
8          return (bytes[index] << 8) + bytes[index + 1];
9      }
10
11     function Unsigned3Byte(index) {
12         return (bytes[index] << 16) + (bytes[index + 1] << 8) + bytes[index + 2];
13     }
14 }
```

Delete the prototype custom script and replace with the downloaded script.

STEP 1 - ENTER FUNCTION DETAILS

CayenneLPP_Sonehub18/50DecoderCustom

[View list of supported functions created by the Helium community](#)

STEP 2 - ENTER CUSTOM SCRIPT

```
1  function Decoder(bytes, port) {
2
3      function Unsigned1Byte(index) {
4          return bytes[index];
5      }
6
7      function Unsigned2Byte(index) {
8          return (bytes[index] << 8) + bytes[index + 1];
9      }
10
11     function Unsigned3Byte(index) {
12         return (bytes[index] << 16) + (bytes[index + 1] << 8) + bytes[index + 2];
13     }
14
15     function Signed1Byte(index) {
16         num = Unsigned1Byte(index);
17         if (num > 0x7F) {
18             num = num - 0x100;
19         }
20         return num;
21     }
22
23     function Signed2Byte(index) {
24         num = Unsigned2Byte(index);
25         if (num > 0x7FFF) {
26             num = num - 0x10000;
27         }
28         return num;
29     }
30 }
```

SCRIPT VALIDATOR

Payload InputHexadecimal (eg. 20 04 7f ff)

Port1

☐ Use Sample Uplink Info

Payload Output

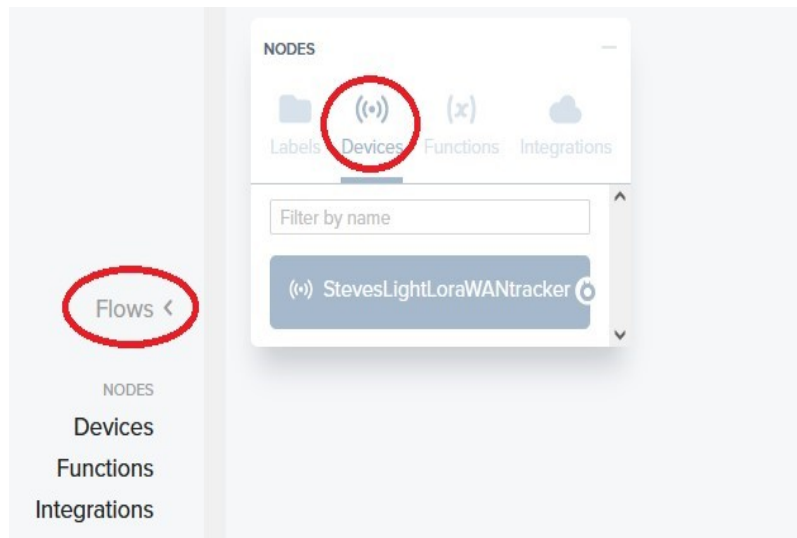
Save the Function (Bottom Right).

The script converts the Cayenne LPP formatted raw data into the JSON format required by SondeHub – giving it the correct field names. This Script is specific to the Light-Tracker – if your using a different Cayenne LPP tracker you will probably have to modify the code.

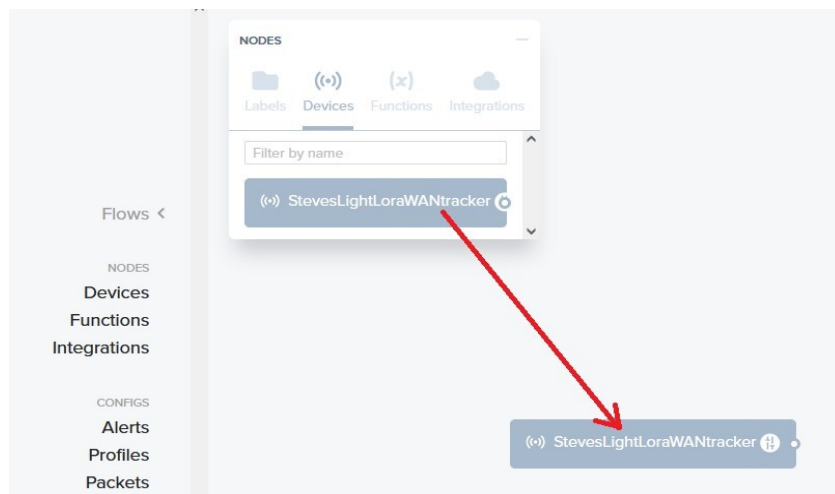
The final step is to connect your Device through the Function to your Integration. To do this use “Flows”.

Flows

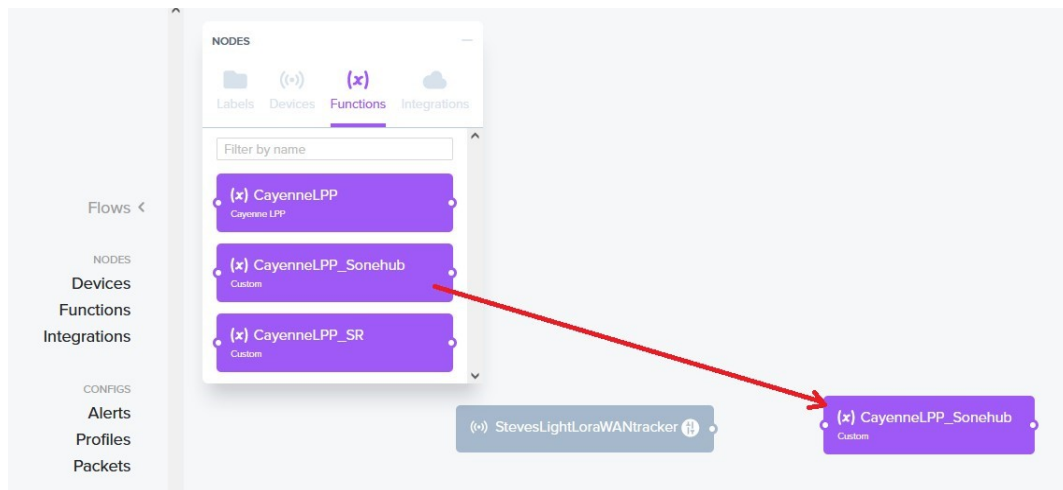
5) In the “Flows” page select “Devices”



Drag your device into the flows page:



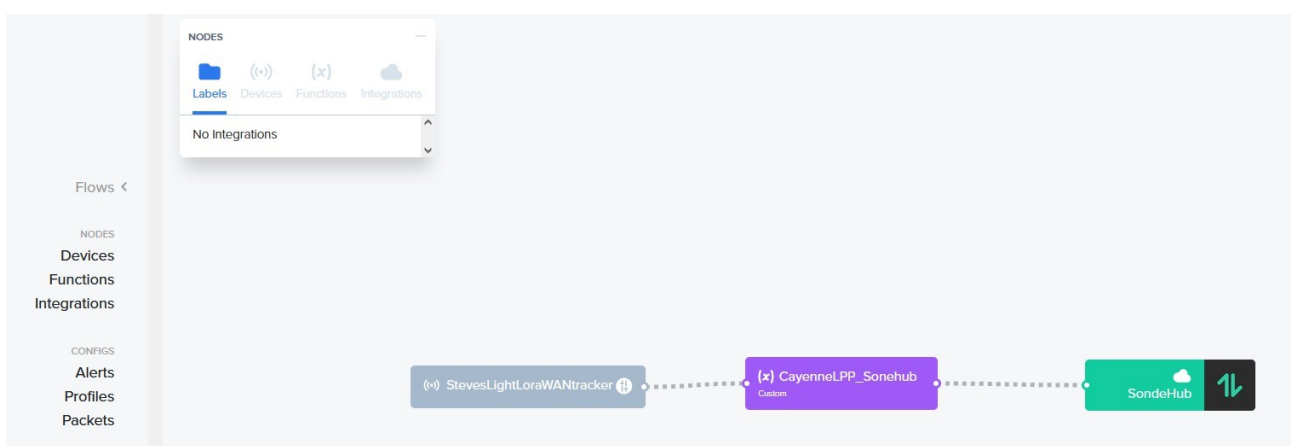
Then drag your Function:



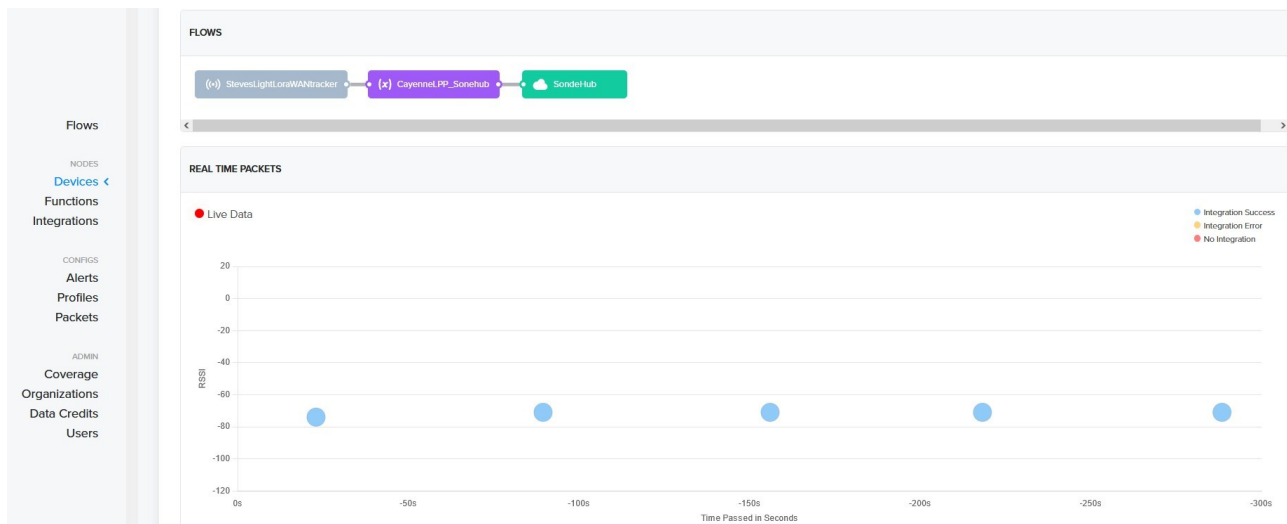
Then your Integration:



Finally connect your device to the Function and the Function to the Integration. This is achieved by clicking on a circle, holding and dragging to another. When the pointer is over circle it will change to a '+'.

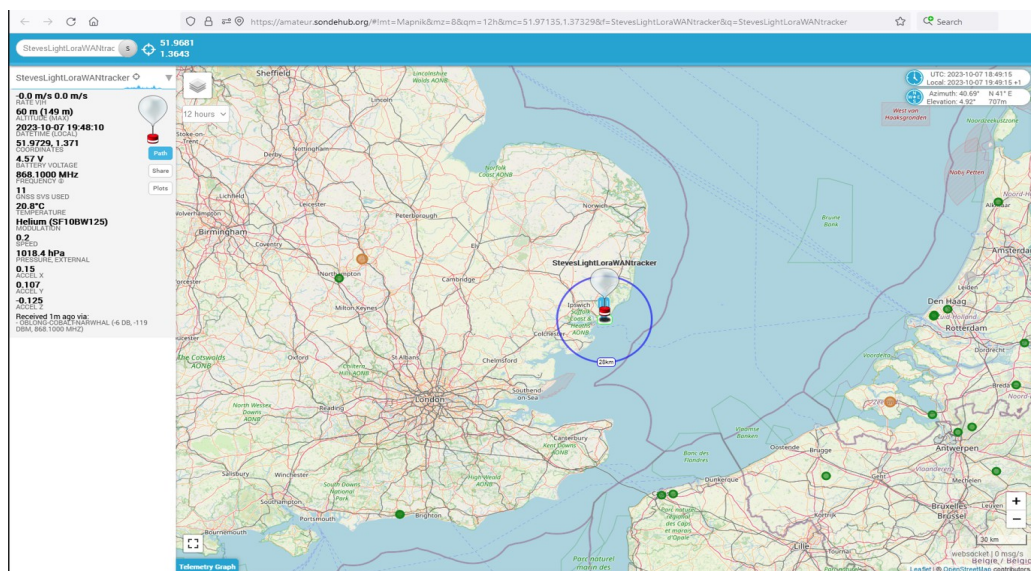


Once completed data should be flowing from your device to Sondehub. To check this go back to the “Devices” page.



The tracker is only operational when light blue dots appear in the Event log screen.

Finally check data is appearing on [SondeHub](https://matteus.sondehub.org/):



Click on “Plots” to see graphed data.

Further Information

Flows: <https://www.youtube.com/watch?v=XrbN1CHApBI>

Functions: <https://www.youtube.com/watch?v=UNUOLbIKXww>

Integrations: <https://www.youtube.com/watch?v=lnERw1f7TGE>

<https://docs.helium.com/console>

<https://github.com/projecthorus/sondehub-infra/wiki/Helium-Network-Gateway>

Further Help

You can check the format of your data is correct for Sondehub by using a RequestBin integration:

<https://public.requestbin.com> - its simplest to create a public bin.

The following is useful if you want to convert payload data:

<https://www.rapidtables.com/web/tools/base64-decode.html>

To check your Function is working correctly, using RequestBin take the base64 data from a Cayenne message:

"payload": "AYgH7jEANY4AGPQCZwDSAwIByQQADgUCAEcGACgHcyfICHEAS//i/5AJAgAG"

and pass it through the base 64 decoder to get the the equivalent Hex.

Base64 Decode

Base64 Decode Base64 Encode Image to Base64 Base64 to Image

Base64 decode to text string/image/hex/binary:

Open File

Or paste/drop base64 data here

AYgH7jEANY4AGPQCZwDSAwIByQQADgUCAEcGACgHcyfICHEAS//i/5AJAgAG

Output type

Text string Image file **Hex** Binary

Output numbers delimiter

Space , Comma **x None**

Decode x Reset Swap

Hex dump output

01 88 07 EE 31 00 35 8E 00 18 F4 02 67 00 D2 03 02 01 C9 04 00 0E 05
02 00 47 06 00 28 07 73 27 C8 08 71 00 4B FF E2 FF 90 09 02 00 06

Finally paste in the Hex into the Flows script validator and press the purple play button. Check the decoded payload output and compare with the SondeHub specification.

CUSTOM SCRIPT

```
0 function Decoder(bytes, port) {  
1  
2   function Unsigned1Byte(index) {  
3     return bytes[index];  
4   }  
5  
6   function Unsigned2Byte(index) {  
7     return (bytes[index] << 8) + bytes[index + 1];  
8   }  
9  
10  function Unsigned3Byte(index) {  
11    return (bytes[index] << 16) + (bytes[index + 1] << 8) + bytes[index + 2];  
12  }  
13  
14  function Signed1Byte(index) {  
15    num = Unsigned1Byte(index);  
16    if (num > 0x7F) {  
17      num = num - 0x100;  
18    }  
19    return num;  
20  }  
21  function Signed2Byte(index) {
```

SCRIPT VALIDATOR

Payload Input Port

07 73 27 C8 08 71 00 4B FF E2 FF 90 09 02 06 1

☐ Use Sample Uplink Info

Payload Output

```
{  
  "latitude": 51.9729,  
  "longitude": 1.371,  
  "altitude": 63.88,  
  "temp": 21,  
  "battery": 4.57,  
  "sats": 14,  
  "speed": 0.71,  
  "heading": 40,  
  "ext_pressure": 1018.4,  
  "accel_x": 0.075,  
  "accel_y": -0.03,  
  "accel_z": -0.112
```