## NetSim Interfacing with Node Red and IBM Watson

**Software:** NetSim Standard v13.0 (64 bit), Node-RED both on server and client machine.

**Project Download Link:**

https://github.com/NetSim-TETCOS/IoT-Interfacing-with-Nodered-and-IBM-Watson_v13.0/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and setup the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

**Objective:** Interfacing NetSim with IBM node red and IBM Watson IoT cloud platform.

**About Node-RED and IBM- Watson QuickStart**

a. **Node-RED** is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.
   Node-RED is built on Node.js, taking full advantage of its event-driven, nonblocking model. This makes it ideal to run at the edge of the network on low-cost hardware such as the Raspberry Pi as well as in the cloud.

b. **Quickstart** is a way of getting a single device sending data to Watson IoT Platform that you can visualize in your browser. No sign up is required, you can see how easy it is to connect your device and, if you want to continue after QuickStart, you can sign up, add security to your device and you are ready to go.

## High Level Connectivity Diagram

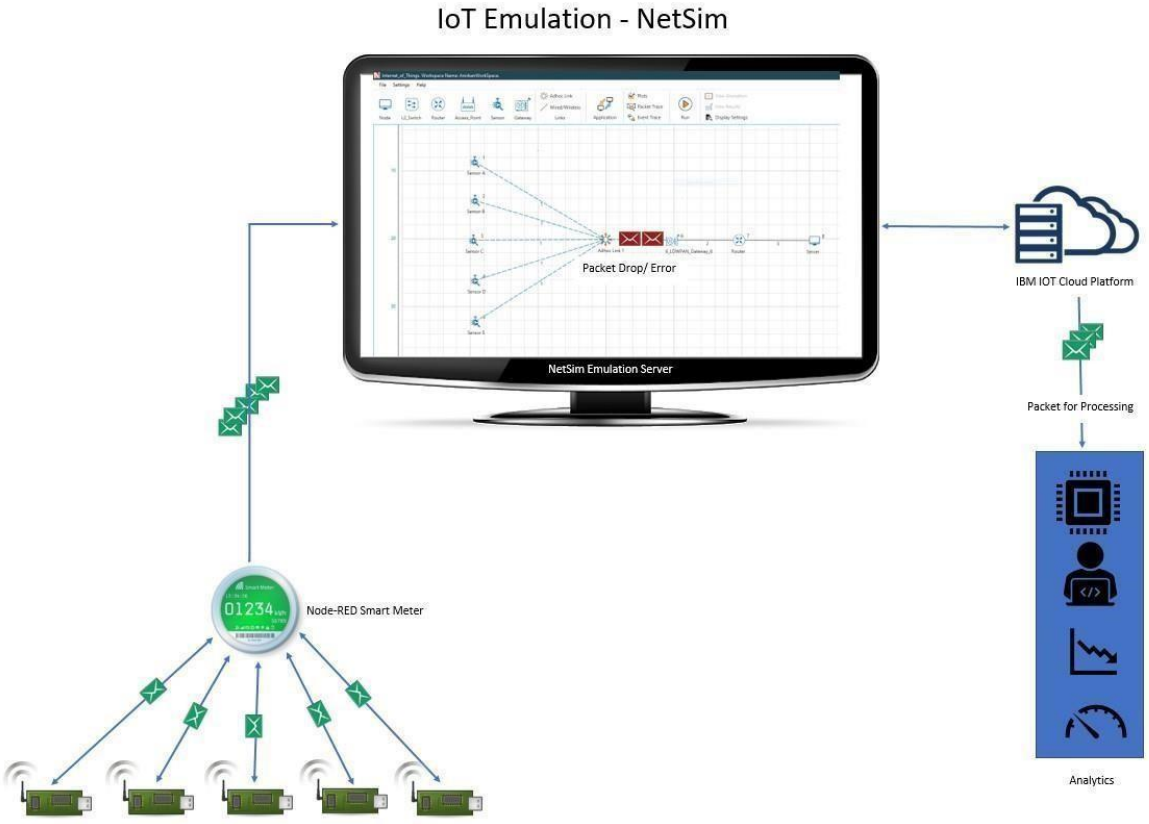


**Figure 1**: IOT Emulation -NetSim

- IBM node red 'virtual sensors' are mapped to sensors in NetSim.
- The traffic sent by the IBM Node-RED virtual sensor is routed inside NetSim and sent from the sensors to the sink in NetSim's virtual network.
- From the sink it is sent to IBM Watson IoT platform cloud server.

**Step by step implementation.**

**System 1 Setup: Sensor server and NetSim Emulation Server Installation Node-RED Installation and its working**

- Installation of Node Red on Windows: Follow the steps on how to install Nodered in windows, at https://nodered.org/docs/getting-started/windows
- Running Node-RED and configuring Sensor server:
- Run Node-RED through cmd window
- Open browser http://127.0.0.1:1880/?#flow/
- Install Node-red packages - commands
  i. npm Install Node-RED-contrib-scx-ibmiotapp
  ii. npm Install Node-RED-dashboard package
  iii. npm install node-redcontrib-ibm-watson-iot

**Setting up Sensor - Server**

- Import Server.json file i.e Menu > Import > Browse > <Select server.json> from from the downloaded project folder if you want to have sensor server
- Configure the device present in the flow graph in **Figure 2**.

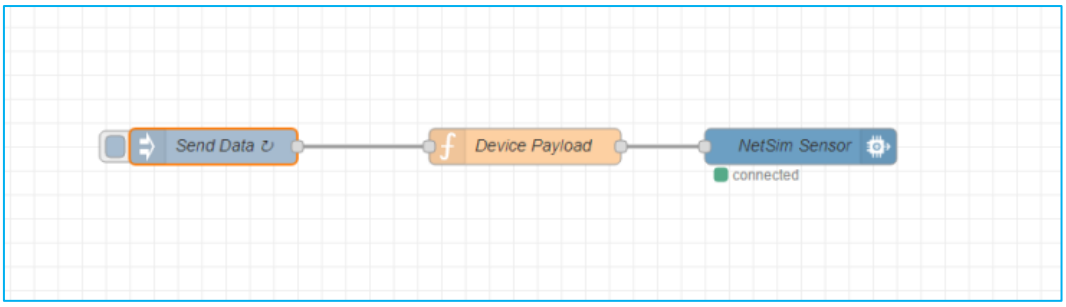


**Figure 2:** Flow design for Multiple sensors

Users can also use a smart meter to send the data. Which accepts the following as input and here is a format to give file as input as shown in **Figure 3**.

a. HTTP
b. USB
c. FILE

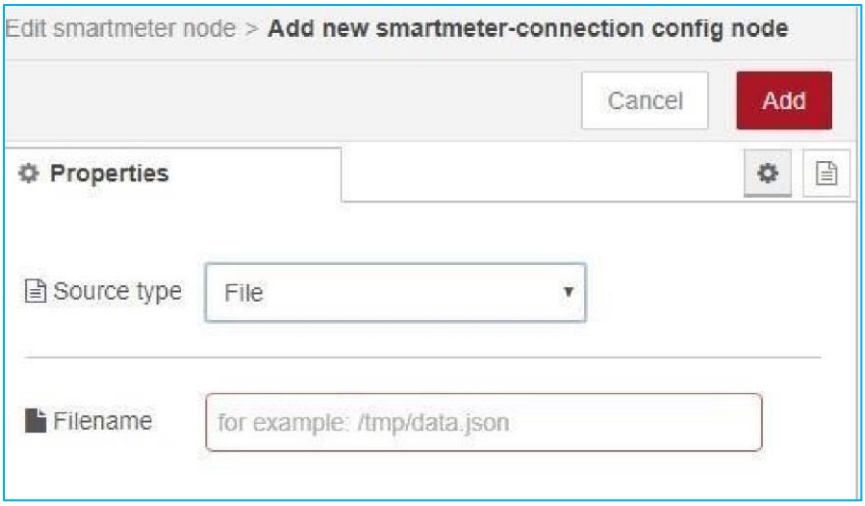


**Figure 3:** Smart meter to send the data

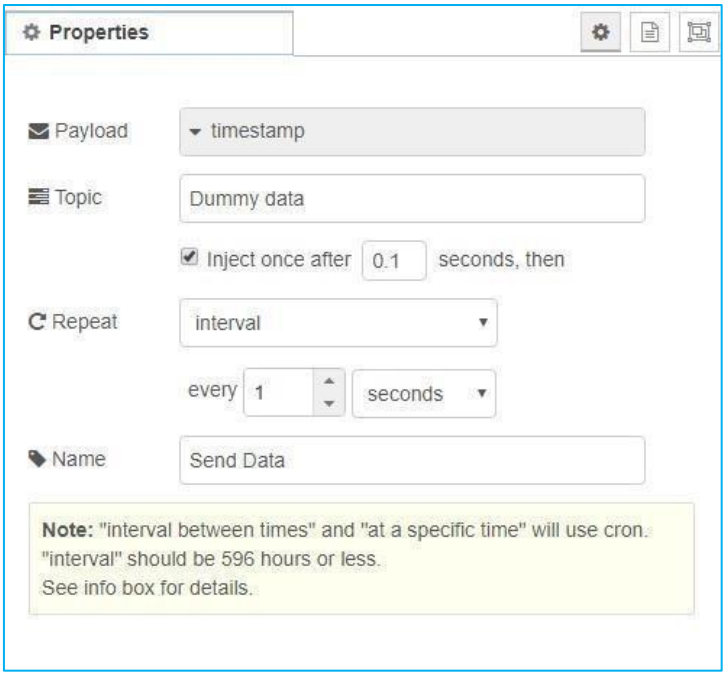


**Figure 4:** Sensor Data: Generating 1Packet/second which accepts decimal values

- Since we are now configuring a virtual sensor, the device payload is set by writing a function in Device Payload in flow chart as **Figure 5** Sensor Payload.

Function:

```
var counter1 = context.get('counter1')||0;
counter1 = counter1+1; if(counter1
> 1000) counter1 = 0;  context.set('counter1',counter1);

// Create MQTT message in JSON  msg = {payload: JSON.stringify({d:{"Dummy_Data" :
counter1}})};  return msg;
```

When having multiple sensors in the scenario, users can generate Payload by specifying port number for each sensor.

Users need to add ***msg.port=<port number>;*** *in the* payload function


**Figure 5**: Sensor Payload

### IOT Emulation using virtual Smart-meter using IBM Node-RED

Users can also use Node-RED Smart-meter to generate payload instead of writing virtual sensor function.
Node-RED requires additional installation of packages related to smart-meter.

Node-RED Node, that reads and parses the data from smartmeter devices. Supports for example Hager eHz Energy Meter, EMH Energy Meter, EFR SmartGridHub, Siemens 2WR5, Elster AS1440, Iskraemeco MT174, Itron EM214 Typ 720 etc.

For additional information on how to install smart-meter and use that as payload generator look into https://flows.nodered.org/node/node-red-contrib-smartmeter.

### Sending data from real sensor

Users can also use real sensor instead of virtual sensors. i.e., users can connect real sensors to Raspberry Pi and then install Node- red on Raspberry Pi and configure the Node-flow.

While adding function payload users can directly retrieve the real sensor values and send them to cloud.

Here is an example on how users can connect real sensors to IBM Watson IoT Platform. https://www.hackster.io/madoyon/connect-sensors-to-ibm-watson-iot-platform-083973

### Encryption and Decryption of sensor data

The packet payload can also be encrypted if required. No encryption is applied to the payload by default. Node-RED nodes uses CryptoJS to encrypt and decrypt messages.

For detail on how to encrypt and decrypt the message users can refer to the examples at https://flows.nodered.org/node/node-red-contrib-crypto-js

- Connect NetSim Senor node to IBM Watson QuickStart IoT Platform

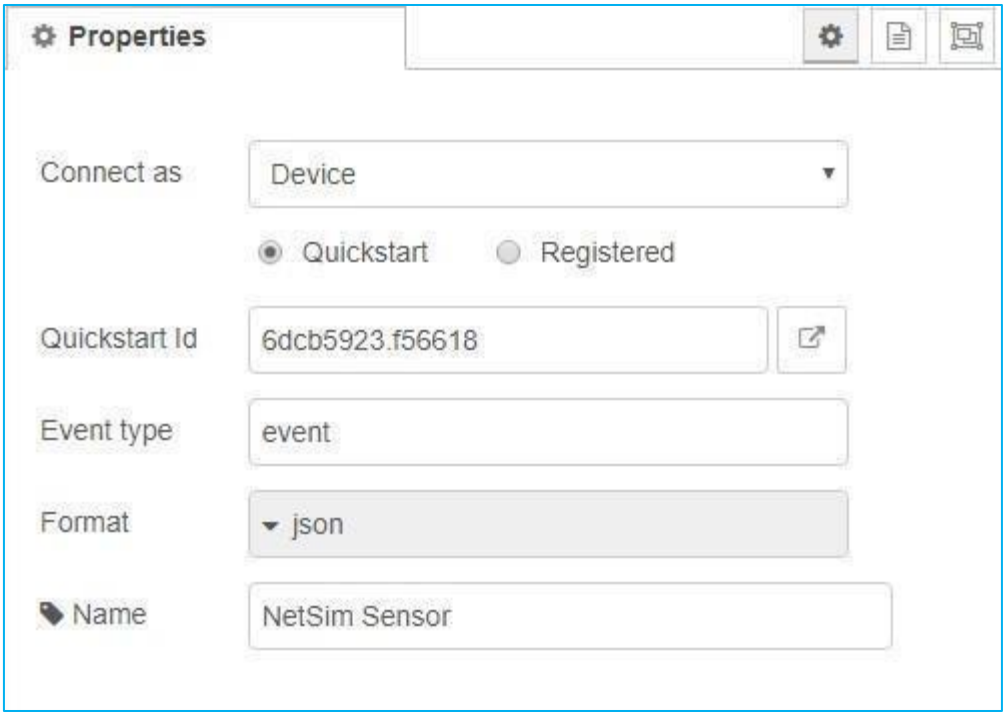


**Figure 6:** NetSim Senor node to IBM Watson QuickStart IoT Platform

- Deploy the Node-RED Flow
- Open the Link icon present in **Figure 6** to view live sensor data the payload that you set.

**Installation of NetSim Emulator and setting up emulation server**

- Install NetSim as per NetSim Installation Guide and Start NetSim as administrator (Make sure that Emulator licenses are available)
- Open IoT network module and create a simple scenario with one sensor connecting to cloud as shown in **Figure 7**.
- Create Emulation application between Sensor and Cloud (Also users can specify port if known) as shown in **Figure 8**.
    I. Source Real IP: 192.168.0.20 (System running Node Red)
    II. Destination Real IP:0.0.0.0 (Mapping traffic to IBM Watson cloud)

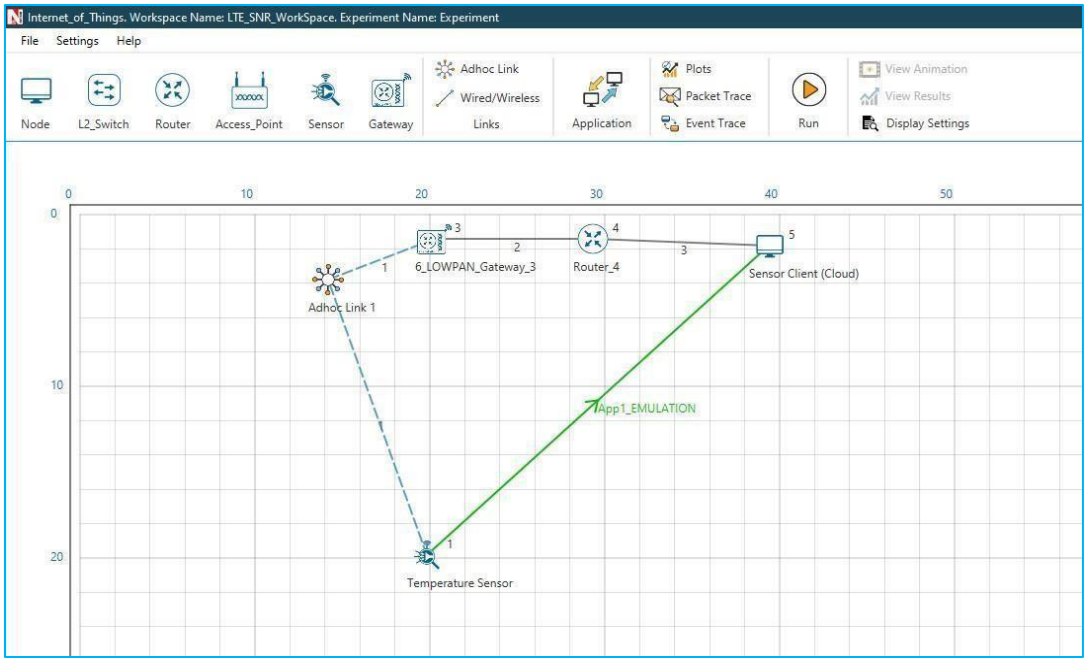


**Figure 7:** Network designed in NetSim

- Also place the sensor and the gateway at suitable distances for pathloss to have an impact on simulation performed.
- Now Run the Simulation for 100s

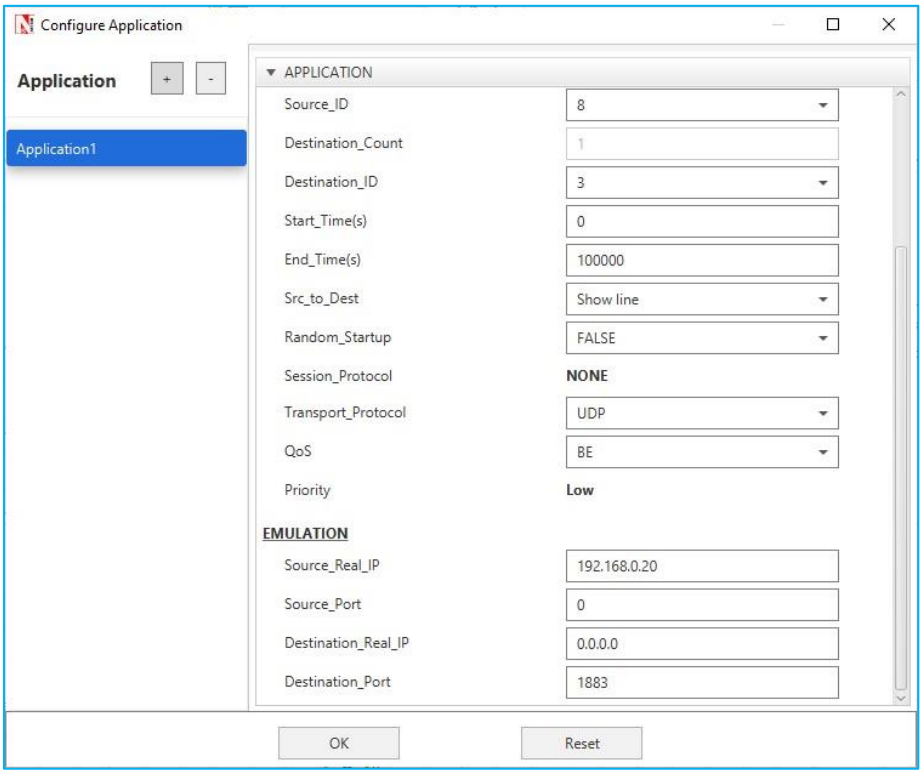


**Figure 8:** NetSim Application Panel

## System 2 Setup: Sensor Client

a. Install Node-RED and perform its running as it was explained above for Sensor client.
b. Design the flow graph as shown in **Figure 9**.
c. Configure the device present in the flow graph in **Figure 10**, **Figure 11** and **Figure 12**.

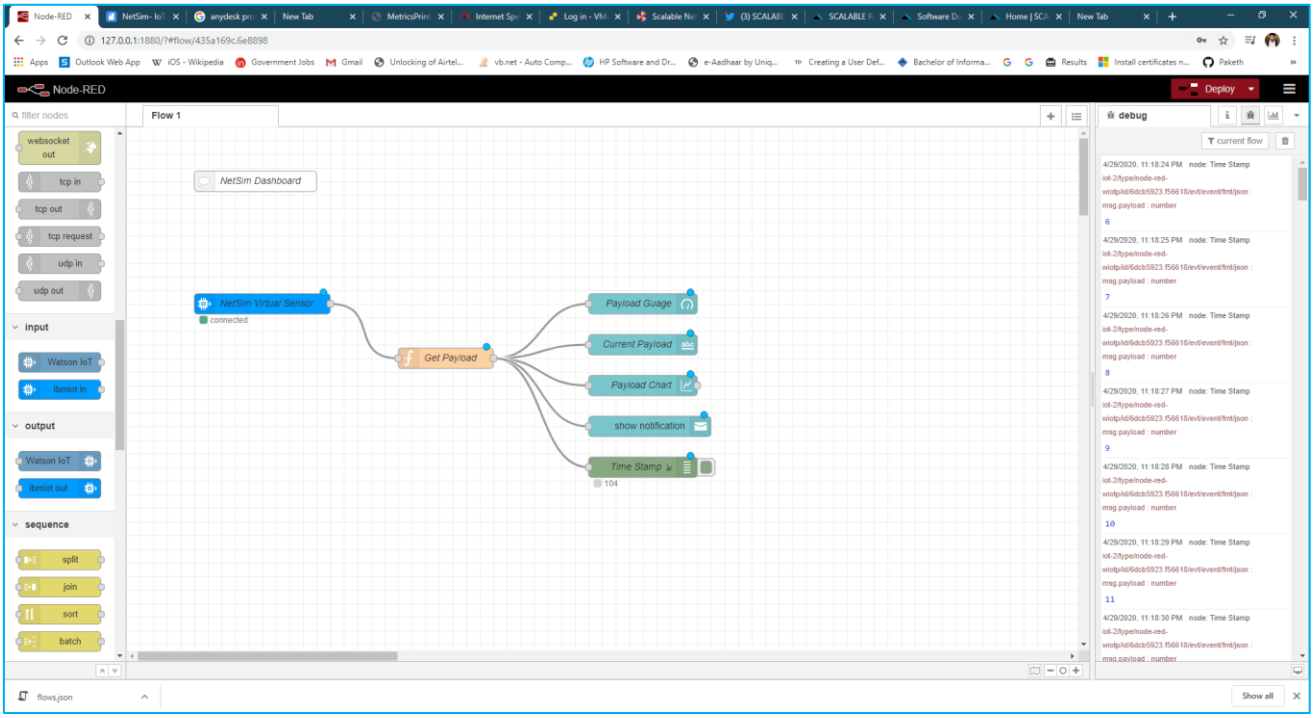


**Figure 9**:Design the flow graph

d. Configure NetSim Virtual Senor Node as shown in **Figure 10**.

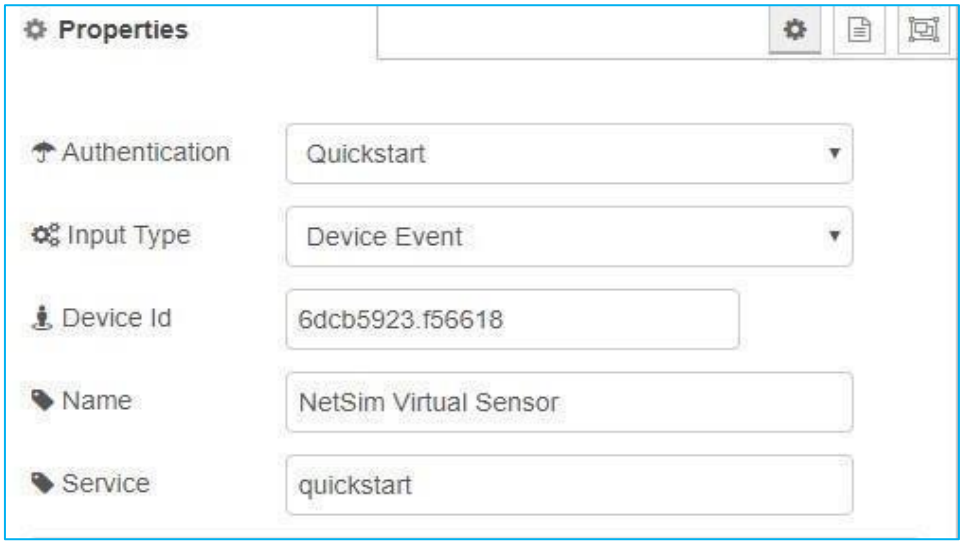


**Figure 10:** Virtual Senor Node Properties

e. To access the payload that we generated in Sensor cloud use the Get payload Function as shown in Figure1.11:
msg.payload=msg.payload.d.dummy_data
return msg;

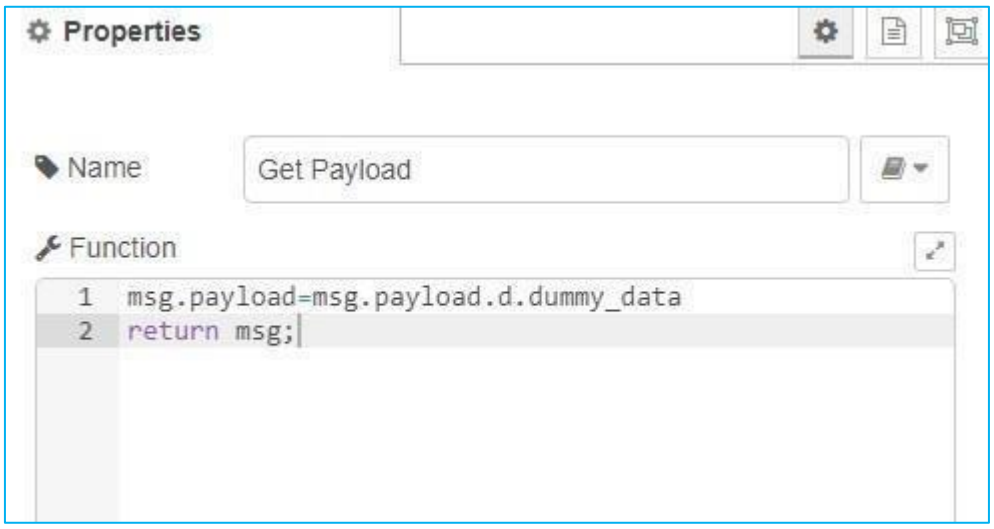


**Figure 11:** Get payload Function

f. Enable Debug window using Debug node to access payload to file as shown in Figure 1.12. If you need that payload to be written to a file, then while running Node-RED command to the beginning you can start writing it to a file as

***Node-RED >getpayload.txt***

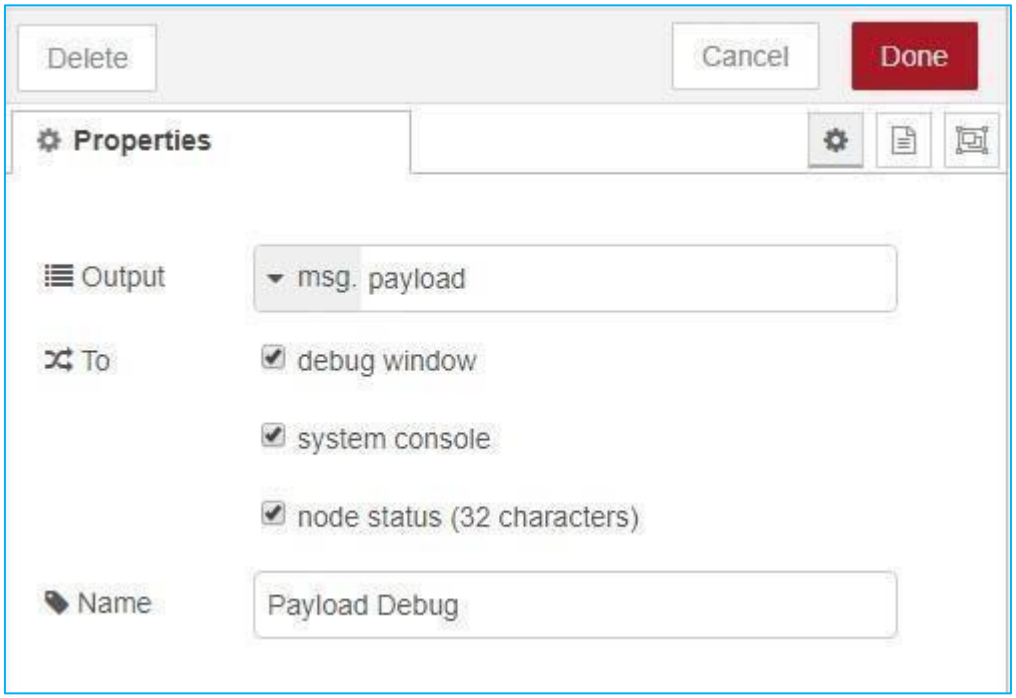


**Figure 12:** Access payload to file

g. Deploy the Node-RED Flow

h. Make sure that you start receiving the data through the console as shown in **Figure 13**.

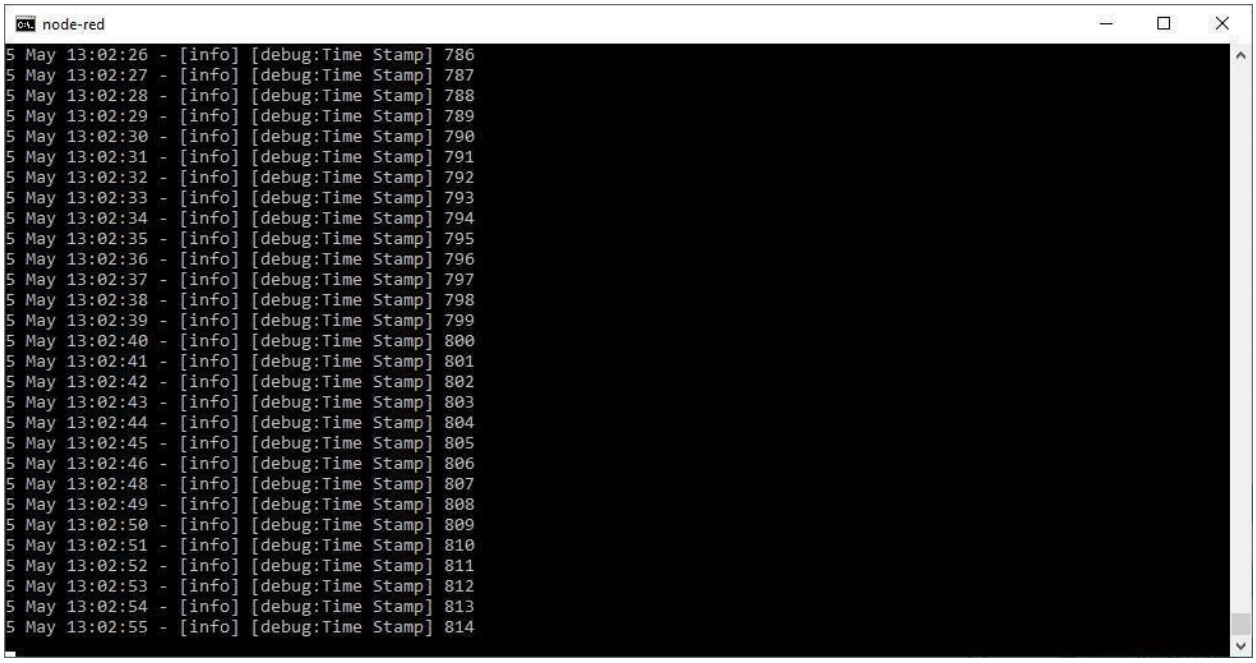


**Figure 13:** you start receiving the data through the console

### Routing Traffic Through NetSim (Emulation)

Once both the systems are setup and the client starts receiving sensor data that was set in the server, we are ready to perform NetSim Emulation.

Run NetSim Emulation for all the cases mentioned below and create a log of the debug window when receiving the sensor data.

### Example: Impact of variation of channel conditions and sensor count on packet delivery ratio

**Note:** Since we are performing Emulation the results will vary count of received packets.

Once the emulation is started the Node-red Client CLI starts receiving the emulated packet.

Open Client CLI > Count the number of packets received for every Emulation varying the Pathloss-Exponent.

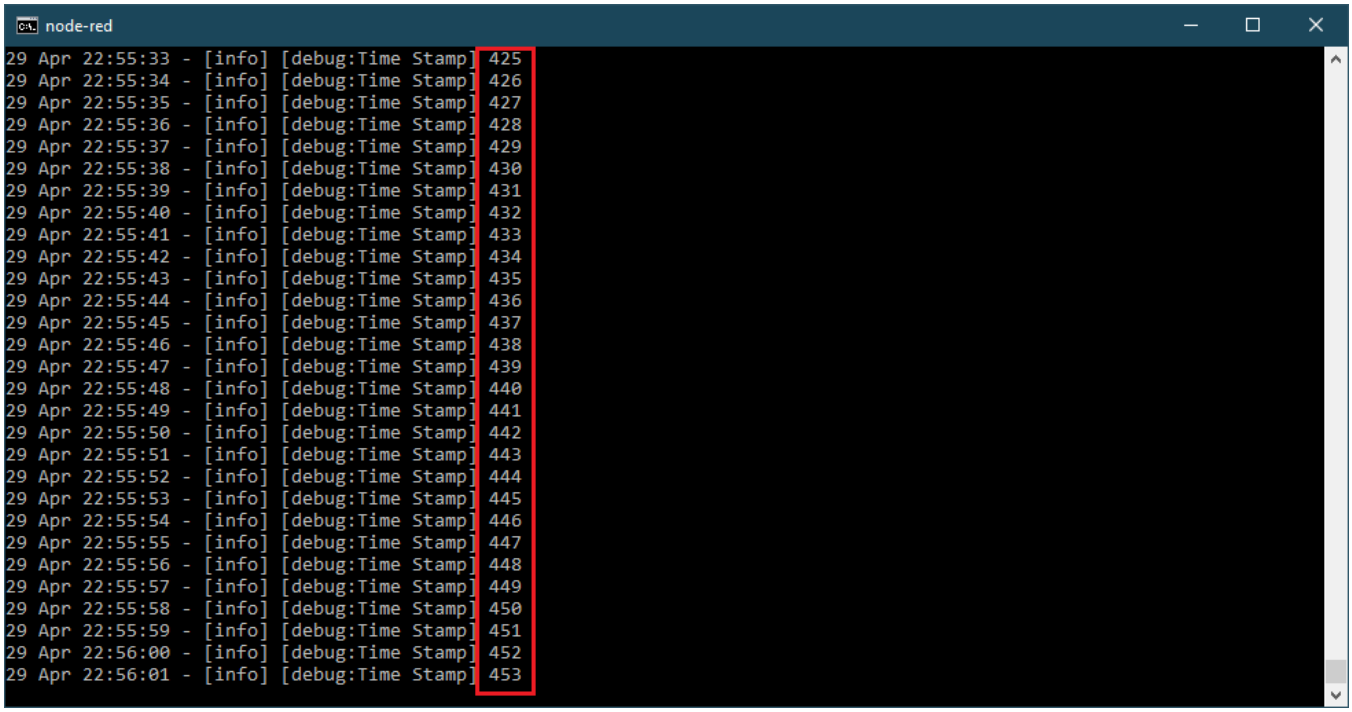


**Figure 14:** the number of packets received for every Emulation varying the Pathloss- Exponent.

Here are approximate results noted, which may vary for different system and multiple emulation.

### Case 1: Sensor count 1

| Channel Characteristics | Emulation Application 1 Packet received out of 100 |
|---|---|
| **No Pathloss** | 89 |
| **Pathloss - 2** | 70 |
| **Pathloss - 3** | 58 |
| **Pathloss - 3.5** | 28 |

**Table 1:** Results for One Sensor with different Channel Characteristics

### Case 2: Sensor count 2

| Channel Characteristics | Emulation Application 1 | Emulation Application 2 |
|---|---|---|
| **No Pathloss** | 75 | 80 |
| **Pathloss - 2** | 73 | 78 |
| **Pathloss - 3** | 77 | 63 |
| **Pathloss - 3.5** | 32 | 27 |

**Table 2:** Results for two Sensor with different Channel Characteristics

### Case 3: Sensor count 5

| Channel Characteristics | App 1 | App 2 | App 3 | App 4 | App 5 |
|---|---|---|---|---|---|
| **No Pathloss** | 63 | 59 | 68 | 44 | 61 |
| **Pathloss - 2** | 55 | 50 | 52 | 41 | 51 |
| **Pathloss - 3** | 31 | 35 | 28 | 26 | 49 |
| **Pathloss - 3.5** | 19 | 8 | 12 | 6 | 13 |

**Table 3:** Results for five Sensor with different Channel Characteristics

### Case 4: Sensor count 10

| Channel Characteristics | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **No Pathloss** | 35 | 32 | 39 | 39 | 46 | 43 | 38 | 34 | 45 | 48 |
| **Pathloss - 2** | 30 | 28 | 31 | 34 | 39 | 41 | 33 | 20 | 41 | 18 |
| **Pathloss - 3** | 9 | 5 | 10 | 8 | 3 | 5 | 2 | 15 | 11 | 5 |
| **Pathloss - 3.5** | 5 | 0 | 6 | 2 | 0 | 0 | 0 | 8 | 5 | 0 |

**Table 4:** Results for ten Sensor with different Channel Characteristics