CONNECTING SENQUIP DEVICES TO CUMULOCITY IOT

1. Introduction

This application note details how to integrate Senquip telemetry devices with Cumulocity IoT from Software AG using MQTT static templates. Senquip devices and Cumulocity IoT also support HTTP and other MQTT protocols, but these will not be considered in this application note.

Cumulocity IoT enables full IoT solutions to be created in minutes by anyone who can use everyday office productivity apps. This allows your business to adopt an extremely efficient, low-effort, rapid-refinement, and agile approach to IoT solution creation.

This application note is also available as a video on YouTube.

Connect: Connect any "thing" using any protocol over any network using a "plug-and-play" approach.

Analyse: Get instant insights from a range of analytics techniques and visualize these in real-time interactive dashboards.

Integrate: Embed IoT data into your business applications and orchestrate workflows between them.

MQTT stands for Message Queuing Telemetry Transport and is a lightweight, publish-subscribe network protocol that transports messages between devices. To ease device integration Cumulocity IoT already supports several static templates that can be used by any client to send standard messages to Cumulocity IoT. This application note will make use of static templates that will be implemented using a simple script running on the Senquip device.

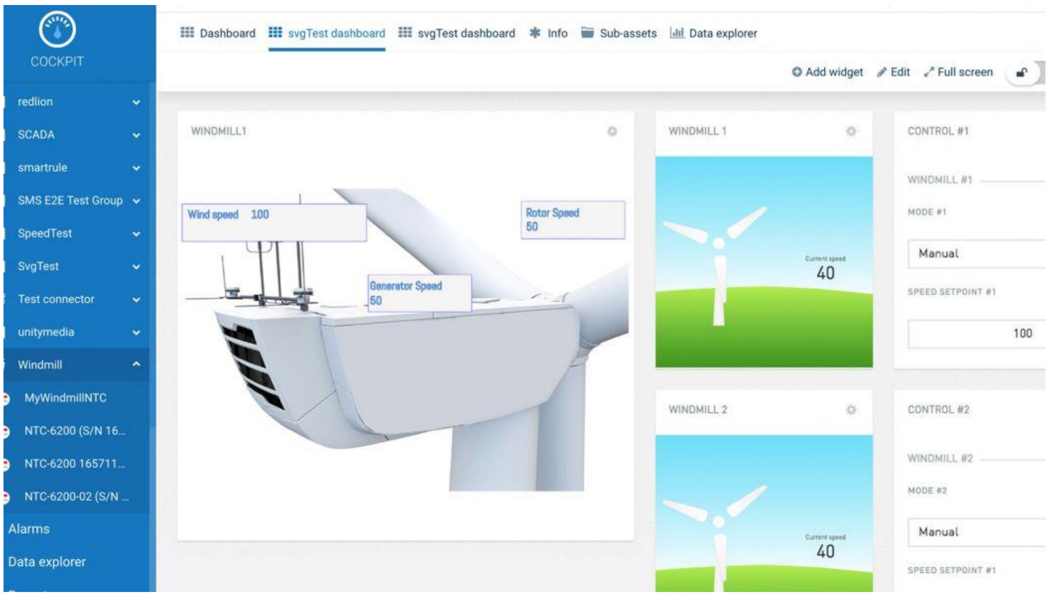


Figure 1 - Cumulocity IoT Dashboard

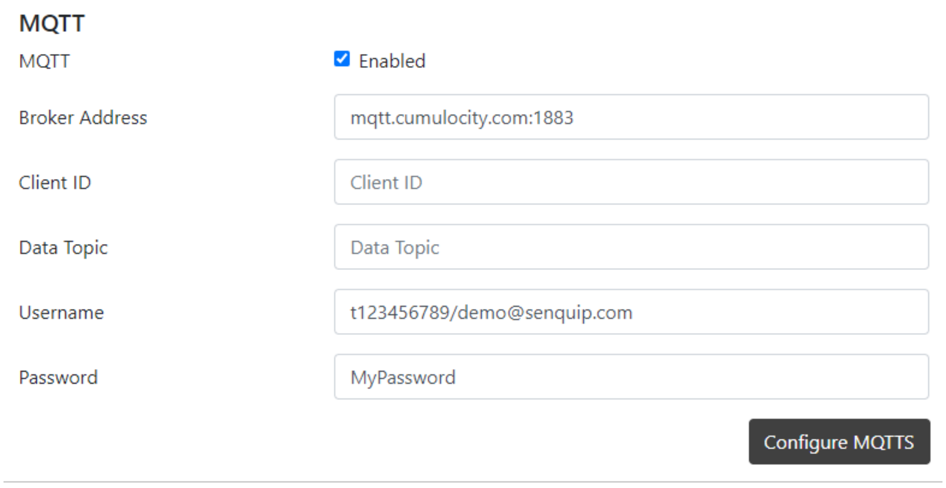
2. Device and Cumulocity IoT Configuration

This section outlines how to configure a Senquip device to communicate with Cumulocity IoT over MQTT. It is assumed that the user has created an account on the Senquip Portal to allow device configuration and has an account on Cumulocity IoT. It is always a good idea to upgrade to the latest Senquip device firmware.

Static templates allow devices to send configuration data to specify the capability of the attached device, properties such as hardware and firmware revisions, measurements including position, alarms, and events. Each static template has a defined structure that is documented on the Cumulocity website.

Static templates support automatic creation of devices if they do not exist. When you send data, Cumulocity IoT will automatically create a device for the MQTT Client ID. If you want to create the device on your own, your first message must be the device creation. In this case Cumulocity IoT will create the device from the template.

To setup the MQTT endpoint details, using the Senquip Portal, go to *Settings>Endpoint* and fill in the MQTT section.


MQTT

MQTT	✓ Enabled
Broker Address	mqtt.cumulocity.com:1883
Client ID	Client ID
Data Topic	Data Topic
Username	t123456789/demo@senquip.com
Password	MyPassword

Figure 2 - Example MQTT configuration for Cumulocity IoT

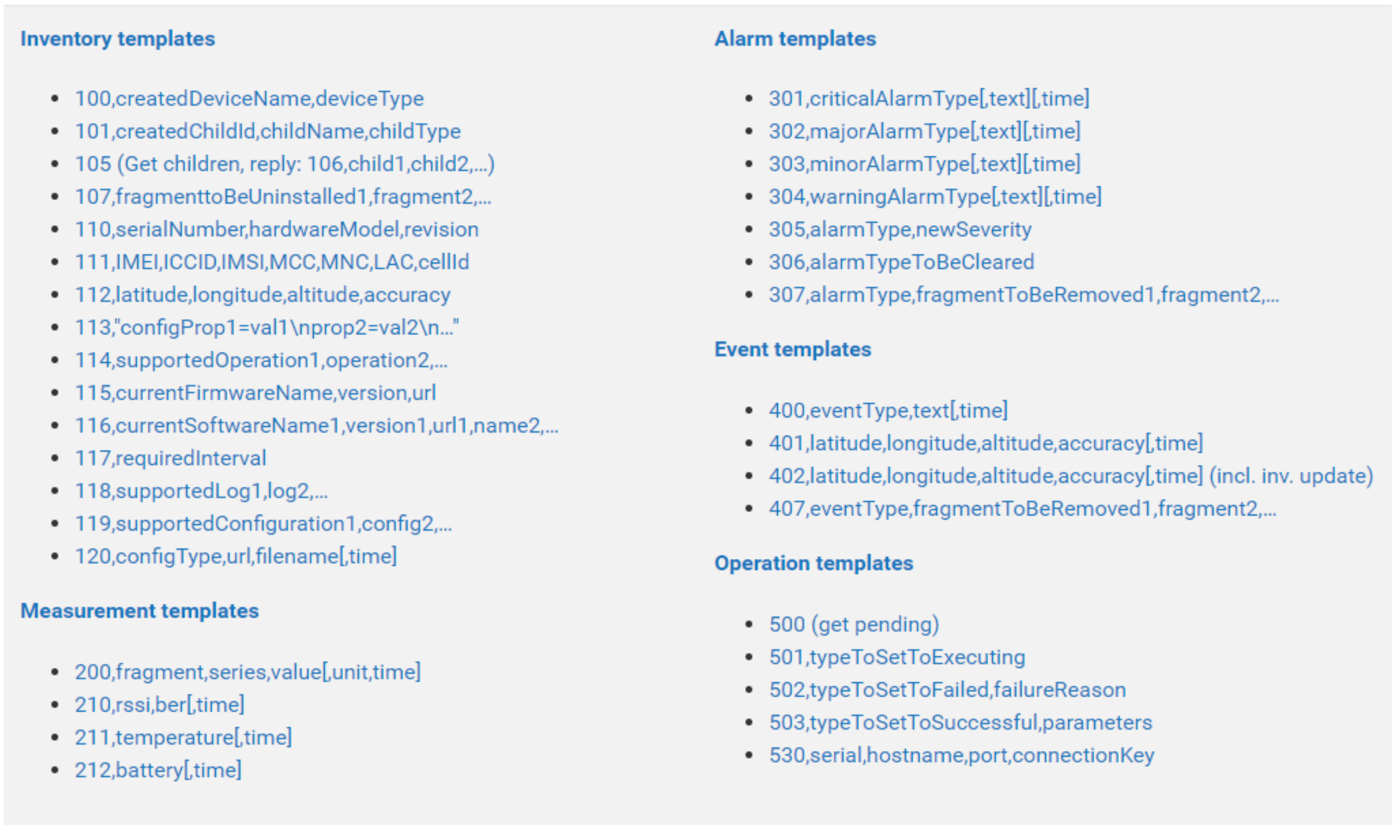
The Broker Address is of the format mqtt.cumulocity.com and will depend on your region and the type of account. The Port is 1883 and so the Broker Address becomes “mqtt.cumulocity.com:1883”. If left blank, the Client ID will default to the Senquip device ID. To use static templates, you need to publish the MQTT messages to the Topic “s/us”. In this case, we will be publishing MQTT messages within the script and so you can leave the Data Topic blank.

When you register for an account on Cumulocity IoT, you will receive a tenant ID. Your username will be formed from the tenant ID and username in the format <tenantID/username>. In this case the tenant ID is “t123456789” and the username is “demo@senquip.com” and so the username is “t123456789/demo@senquip.com”. The Password is the password for you Cumulocity IoT account.

There is no configuration required in the Cumulocity IoT platform. Devices will be created as they first communicate.

3. Creating the Static Templates to Create a Device

To ease device integration, Cumulocity IoT already supports a number of static templates that can be used by any client without the need to create your own templates. These templates focus on the most commonly used messages for device management purposes. These templates will be generated using a script that runs on the Senquip device. A list of static templates is shown in Figure 3.



Inventory templates

- 100,createdDeviceName,deviceType
- 101,createdChildId,childName,childType
- 105 (Get children, reply: 106,child1,child2,...)
- 107,fragmenttoBeUninstalled1,fragment2,...
- 110,serialNumber,hardwareModel,revision
- 111,IMEI,ICCID,IMSI,MCC,MNC,LAC,cellId
- 112,latitude,longitude,altitude,accuracy
- 113,"configProp1=val1\nprop2=val2\n..."
- 114,supportedOperation1,operation2,...
- 115,currentFirmwareName,version,url
- 116,currentSoftwareName1,version1,url1,name2,...
- 117,requiredInterval
- 118,supportedLog1,log2,...
- 119,supportedConfiguration1,config2,...
- 120,configType,url,filename[,time]

Measurement templates

- 200,fragment,series,value[,unit,time]
- 210,rssi,ber[,time]
- 211,temperature[,time]
- 212,battery[,time]

Alarm templates

- 301,criticalAlarmType[,text][,time]
- 302,majorAlarmType[,text][,time]
- 303,minorAlarmType[,text][,time]
- 304,warningAlarmType[,text][,time]
- 305,alarmType,newSeverity
- 306,alarmTypeToBeCleared
- 307,alarmType,fragmentToBeRemoved1,fragment2,...

Event templates

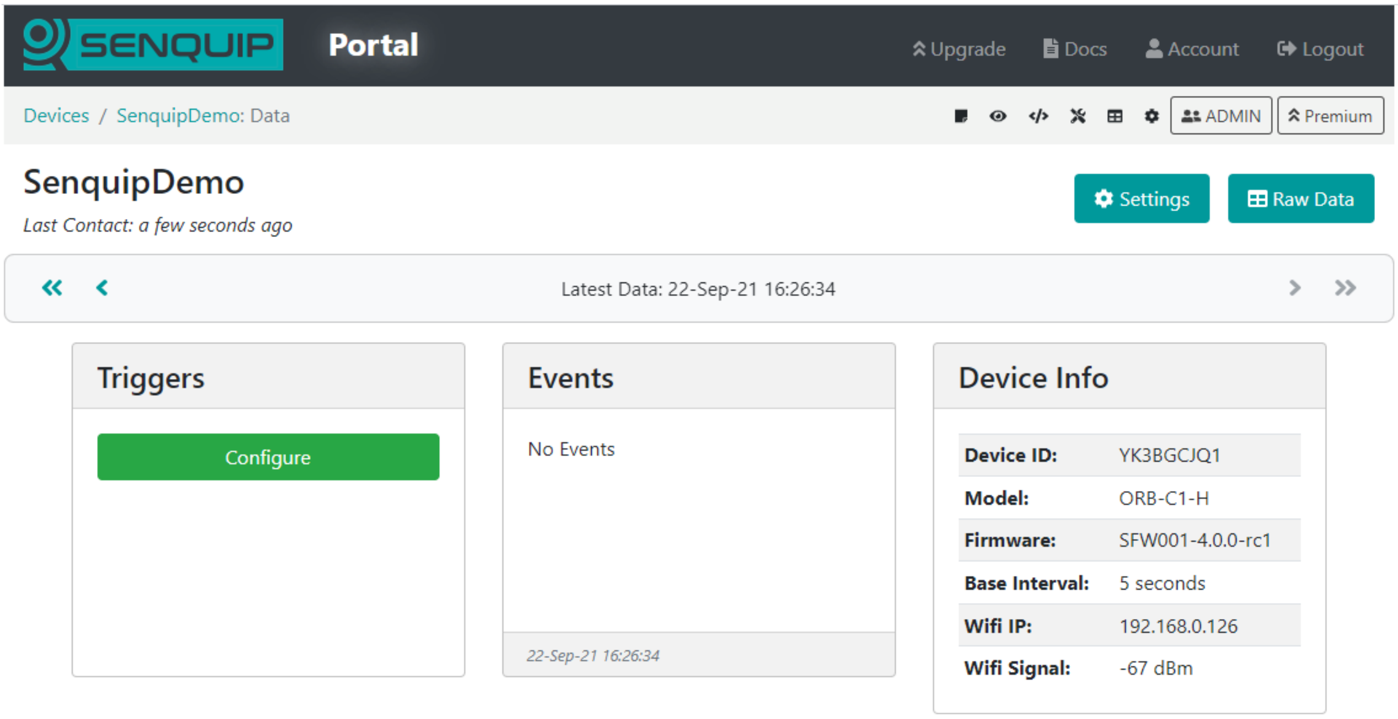
- 400,eventType,text[,time]
- 401,latitude,longitude,altitude,accuracy[,time]
- 402,latitude,longitude,altitude,accuracy[,time] (incl. inv. update)
- 407,eventType,fragmentToBeRemoved1,fragment2,...

Operation templates

- 500 (get pending)
- 501,typeToSetToExecuting
- 502,typeToSetToFailed,failureReason
- 503,typeToSetToSuccessful,parameters
- 530,serial,hostname,port,connectionKey

Figure 3 - Available Static Templates.

Messages that are used for device setup are known as Inventory Templates and are all in the 100's range. These messages set up the device and specify hardware and firmware settings. They only need to be sent once or if there is a change. In the example script, the setup messages are programmed to be sent when the user presses a trigger button on the Senquip Portal, causing a setup function in the script to execute. Device details such as the device name and firmware revision can be read from within a script using the Cfg.get() instruction. For further details on setting up trigger buttons on the Senquip Portal and other scripting fratures, see the Senquip Scripting Guide.



SenquipDemo

Last Contact: a few seconds ago

Latest Data: 22-Sep-21 16:26:34

No Events

22-Sep-21 16:26:34

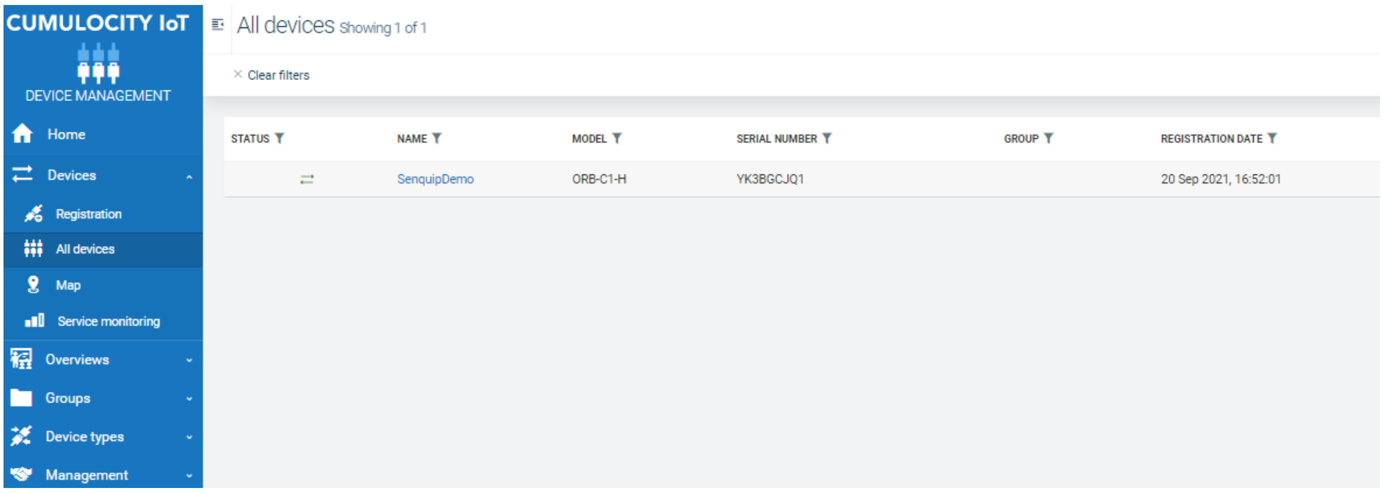
Device ID:	YK3BGCJQ1
Model:	ORB-C1-H
Firmware:	SFW001-4.0.0-rc1
Base Interval:	5 seconds
Wifi IP:	192.168.0.126
Wifi Signal:	-67 dBm

Figure 4 - Trigger button to execute device setup

To create a new device on Cumulocity IoT, we use the 100 template. The format of this message is “100,device name,device type”. The line in the example script to send the 100 template looks like this:

```
MQTT.pub("s/us","100,"+Cfg.get('device.name')+",c8y_MQTTdevice”;
```

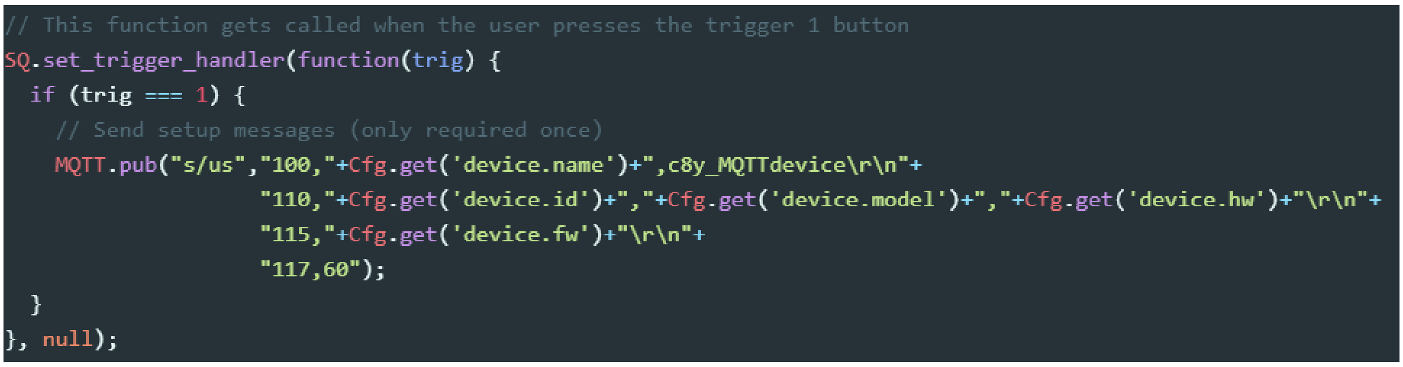
In this line of code, MQTT.pub publishes the given message to topic “s/us”. Once this message is published, a device will be created on your Cumulocity IoT dashboard.



STATUS	NAME	MODEL	SERIAL NUMBER	GROUP	REGISTRATION DATE
	SenquipDemo	ORB-C1-H	YK3BGCJQ1		20 Sep 2021, 16:52:01

Figure 5 - Device Created on Cumulocity IoT

We send further configuration messages of type 110, 115, and 117 to send the device ID, model, hardware revision, and firmware revision. Multiple static templates can be published at once by putting them on new lines (\r\n). The entire function, that is executed when the trigger button is pressed is shown in Figure 6.

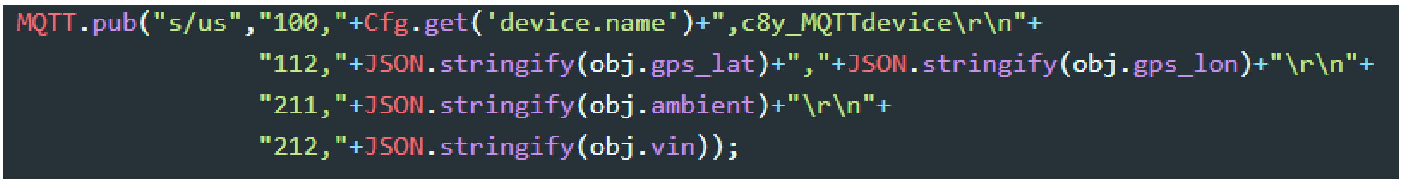


```
// This function gets called when the user presses the trigger 1 button
SQ.set_trigger_handler(function(trig) {
  if (trig === 1) {
    // Send setup messages (only required once)
    MQTT.pub("s/us","100,"+Cfg.get('device.name')+",c8y_MQTTdevice\r\n"+
                    "110,"+Cfg.get('device.id')+","+Cfg.get('device.model')+","+Cfg.get('device.hw')+"\r\n"+
                    "115,"+Cfg.get('device.fw')+"\r\n"+
                    "117,60");
  }
}, null);
```

Figure 6 - Script to Configure a Device

4. Creating the Static Templates to Send Data and Create Alarms

In this example, we have turned on the GPS and ambient temperature sensors on the Senquip device and are going to send these along with other data to our Cumulocity IoT dashboard. Again we use static templates to send data. The Measurement Templates are all in the 200's range. In the snippet of code in Figure 7, we are sending location using template 112, temperature using 211, and battery voltage using 212.

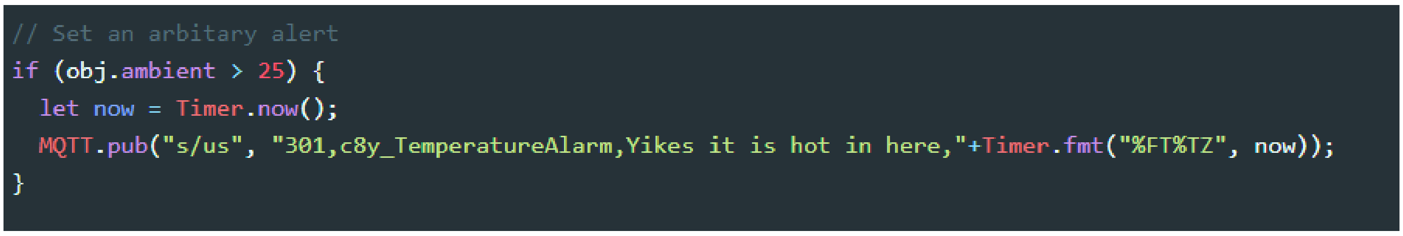


```
MQTT.pub("s/us","100,"+Cfg.get('device.name')+",c8y_MQTTdevice\r\n"+
                "112,"+JSON.stringify(obj.gps_lat)+","+JSON.stringify(obj.gps_lon)+"\r\n"+
                "211,"+JSON.stringify(obj.ambient)+"\r\n"+
                "212,"+JSON.stringify(obj.vin));
```

Figure 7 - Sending Data with a Script

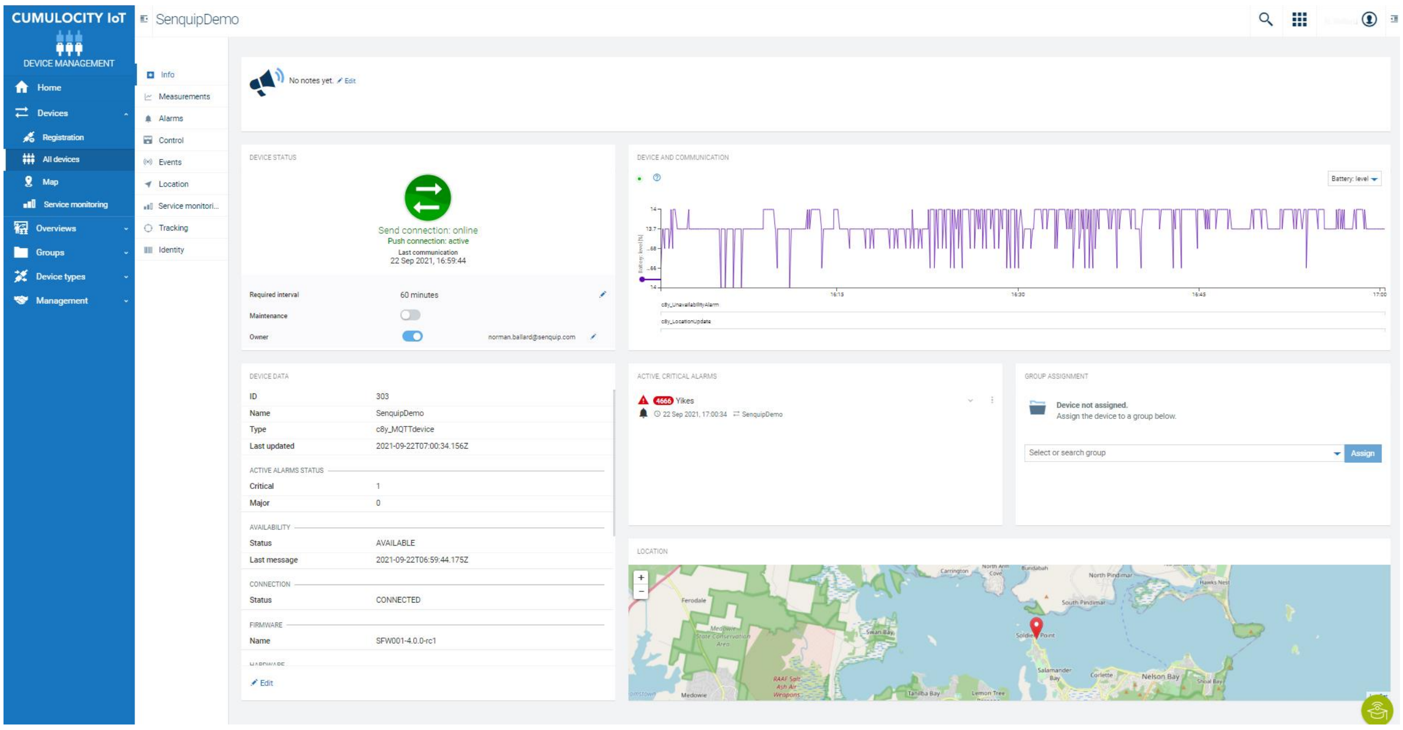
We have also created an alarm if the ambient temperature increases above 25 degrees. Alarms are created using Alarm Templates in the 300's range.

The script to send the measurements and process alerts is in a function that runs after each device measurement cycle and so data will be sent at the Senquip device base interval.



```
// Set an arbitary alert
if (obj.ambient > 25) {
  let now = Timer.now();
  MQTT.pub("s/us", "301,c8y_TemperatureAlarm,Yikes it is hot in here,"+Timer.fmt("%FT%TZ", now));
}
```

In the Device Manager on Cumulocity IoT we can see the location, battery voltage, an alarm and other information is being shown on a dashboard.



No notes yet.

DEVICE STATUS

Send connection: online
Push connection: active
Last communication
22 Sep 2021, 16:59:44

Required interval	60 minutes
Maintenance	
Owner	norman.ballard@senquip.com

DEVICE AND COMMUNICATION

DEVICE DATA

ID	303
Name	SenquipDemo
Type	c8y_MQTTdevice
Last updated	2021-09-22T07:00:34.156Z
ACTIVE ALARMS STATUS	
Critical	1
Major	0
AVAILABILITY	
Status	AVAILABLE
Last message	2021-09-22T06:59:44.175Z
CONNECTION	
Status	CONNECTED
FIRMWARE	
Name	SFW001-4.0.0-rc1

ACTIVE, CRITICAL ALARMS

4666 Yikes
22 Sep 2021, 17:00:34 SenquipDemo

GROUP ASSIGNMENT

Device not assigned.
Assign the device to a group below.

Figure 8 - Data Arriving on Cumulocity IoT

The dashboard shown in Figure 8 has been achieved with no setup of Cumulocity IoT at all. The next step is to use the Cockpit function to manage your device, create dashboards, and explore your data.

5. Conclusion

Sending data to Cumulocity IoT is as simple as configuring the Endpoint settings on the Senquip Device and writing a simple script. Almost no configuration is required on Cumulocity IoT as devices are created automatically.

Appendix 1: Source Code

```
load('senquip.js');
load('api_config.js');
load('api_endpoint.js');
load('api_timer.js');

// This function gets called every time a measurement cycle finishes
SQ.set_data_handler(function(data) {
  let obj = JSON.parse(data);

  // Send some data
  if ((typeof obj.gps_lat === 'number') && (typeof obj.gps_lon === 'number')) {
   MQTT.pub("s/us","100,"+Cfg.get('device.name')+",c8y_MQTTdevice\r\n"+
                   "112,"+JSON.stringify(obj.gps_lat)+","+JSON.stringify(obj.gps_lon)+"\r\n"+
                   "211,"+JSON.stringify(obj.ambient)+"\r\n"+
                  "212,"+JSON.stringify(obj.vin));
  }

  // If running off cellular, send details
  if (typeof obj.gsm_imei === 'string') {
    MQTT.pub("s/us", "111,"+JSON.stringify(obj.gsm_imei)+","+JSON.stringify(obj.gsm_iccid));
  }

  // Set an arbitary alert
  if (obj.ambient > 25) {
    let now = Timer.now();
    MQTT.pub("s/us", "301,c8y_TemperatureAlarm,Yikes it is hot,"+Timer.fmt("%FT%TZ", now));
  }

}, null);


// This function gets called when the user presses the trigger 1 button
SQ.set_trigger_handler(function(trig) {
  if (trig === 1) {
    // Send setup messages (only required once)
    MQTT.pub("s/us","100,"+Cfg.get('device.name')+",c8y_MQTTdevice\r\n"+
"110,"+Cfg.get('device.id')+","+Cfg.get('device.model')+","+Cfg.get('device.hw')+"\r\n"+
                    "115,"+Cfg.get('device.fw')+"\r\n"+
                   "117,60");
  }
}, null);
```