IBM Integration Bus

MQTT using IoT Cloud

June 2015
Hands-on lab built at product
Version 10.0.0.0

Contents

1. Introduction

This lab guide provides a “hands on” introduction to the IIB support of MQTT and how IIB V10 can be integrated with IBM Internet of Things (IoT) Foundation. The IoT Foundation hosts a MQTT server on the internet and provides a quick start demonstration service. This lab showcases a simple flow that connects to that demonstration service and receives messages published by the device simulator (a temperature sensor), provided along with the quick start service, over the MQTT protocol.

1.1 MQTT support in IIB V10

MQ Telemetry Transport (MQTT) is a lightweight publish/subscribe messaging protocol. IBM Integration Bus provides built-in input and output nodes for processing MQTT messages.

The MQTT messaging protocol provides robust messaging features for communicating with remote systems and devices, and also minimizes network bandwidth and device resource requirements.

The protocol is designed for devices in constrained environments, such as embedded systems, cell phones, and sensors with limited processing ability and memory, and for systems that are connected to unreliable networks.

MQTT uses the publish/subscribe style of messaging that enables the information provider (publisher) to be decoupled from the consumer of the information (subscriber). Consequently, the MQTT protocol is ideal for the machine-to-machine, or Internet of Things, world of communication.

IBM Integration Bus V10 provides support for MQTT through two MQTT nodes:

- MQTTPublish:

  Provides an (outbound) interface for publishing data to MQTT endpoints (topic strings)

- MQTTSubscribe:

  Provides an (inbound) subscription interface to obtain data from MQTT endpoints.

1.2 Lab Guide overview

This lab guide provides an outline of how IIB V10 can be used to receive messages from devices, using the MQTT Subscribe node.

You will:

- Run the device simulator (temperature sensor) provided along with the quick start service to publish data on a particular topic.
- Create a simple flow with an MQTTSubscribe node. Configure the node to subscribe to the same topic where the messages are being published by the device simulator.
- Check if the temperature of the device is more than 20. If it is, then send a notification message to an MQ queue. If not, then write the messages to a file.

2. Lab preparation

2.1 Start the device simulator

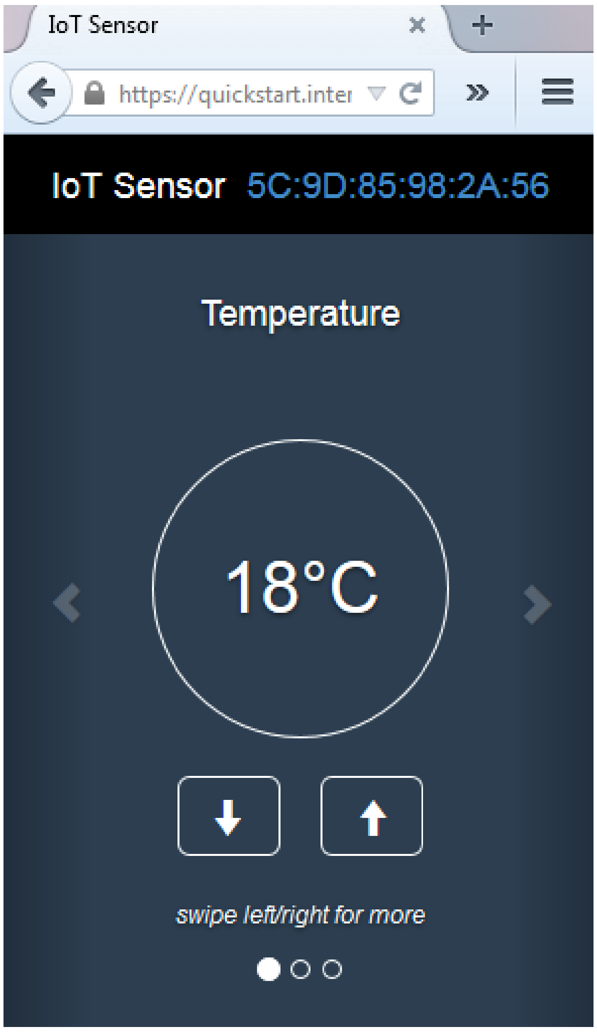
1. In a Firefox browser, open the link http://quickstart.internetofthings.ibmcloud.com/iotsensor

   This shortcut is shown in the IOT shortcut folder, named "IOT Sensor".

   This opens up the temperature monitor device simulator.

2. Note down the MAC address available on top right of the simulator. For example if the MAC address is 5C:9D:85:98:2A:56 then remove the colons, change the upper case letters into lower case and keep a note of the MAC address below, in the following form:

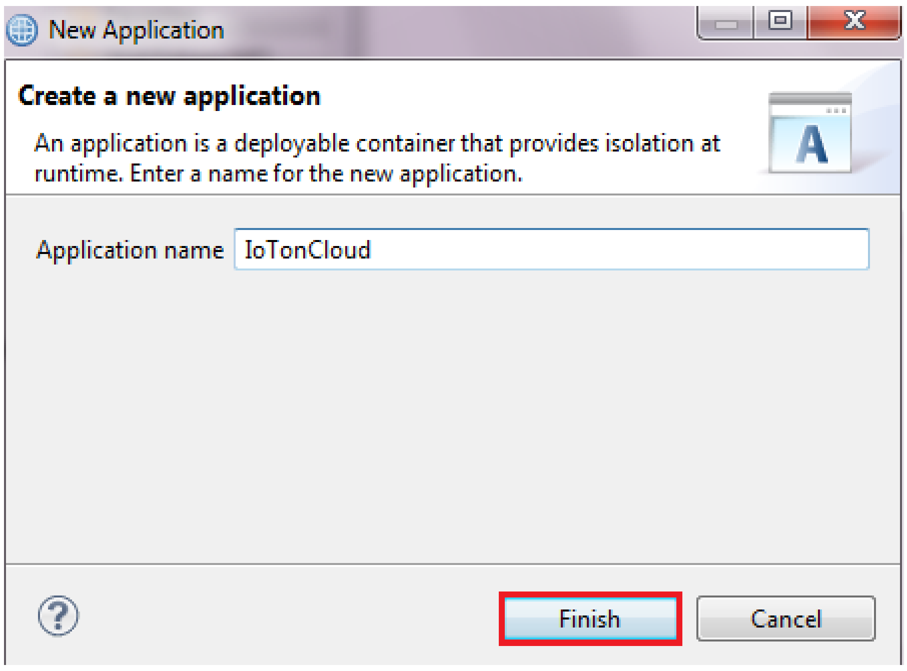
```
5c9d85982a56
```



3. Do not close this browser window or refresh, as the MAC address will change. Later in the lab, this MAC address will be used as part of the topic name.

2.2 Configure a basic MQTT application in IIB

1. In the Integration Toolkit, switch to a new workspace.
   Use “File > Switch Workspace > Other”. Name the new workspace IoTCloud.
2. Click File > New > Application to create a new application called ‘IoTonCloud’. Click Finish.

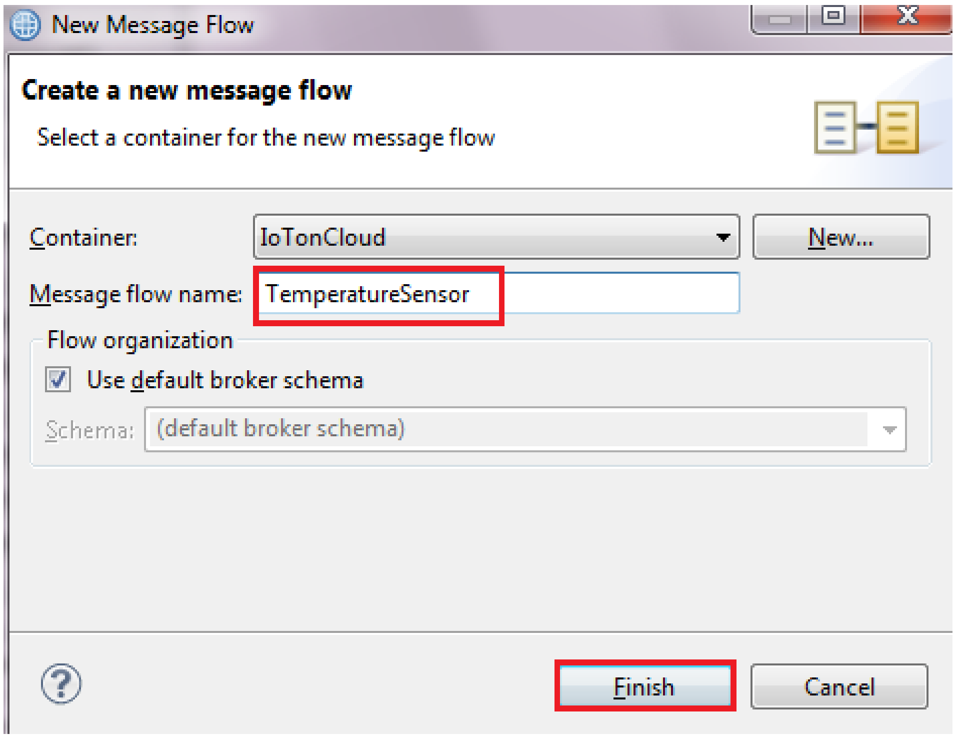

New Application

Create a new application

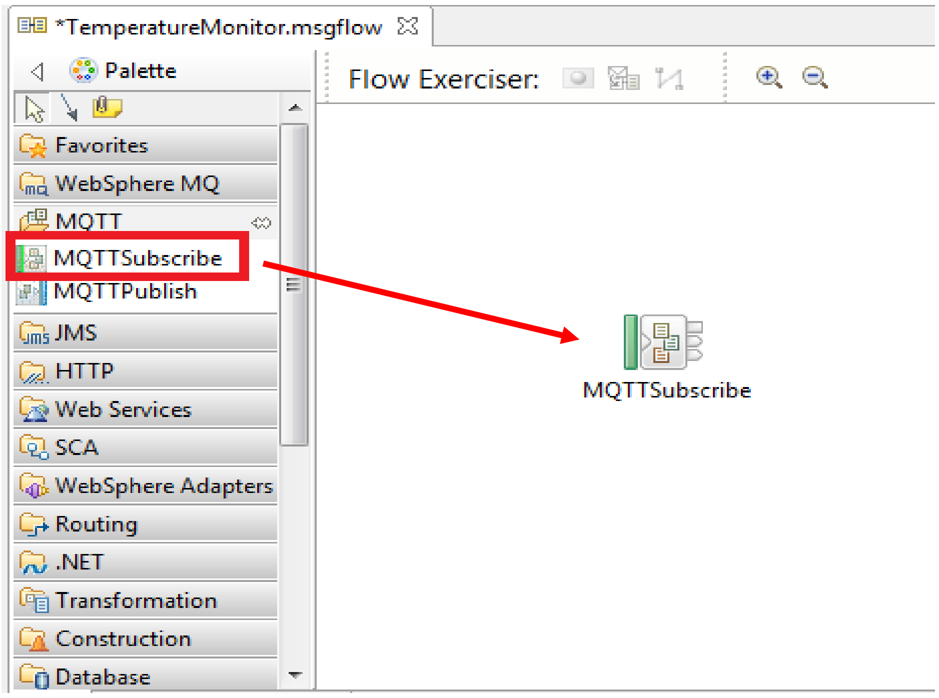
An application is a deployable container that provides isolation at runtime. Enter a name for the new application.

Application name: IoTonCloud

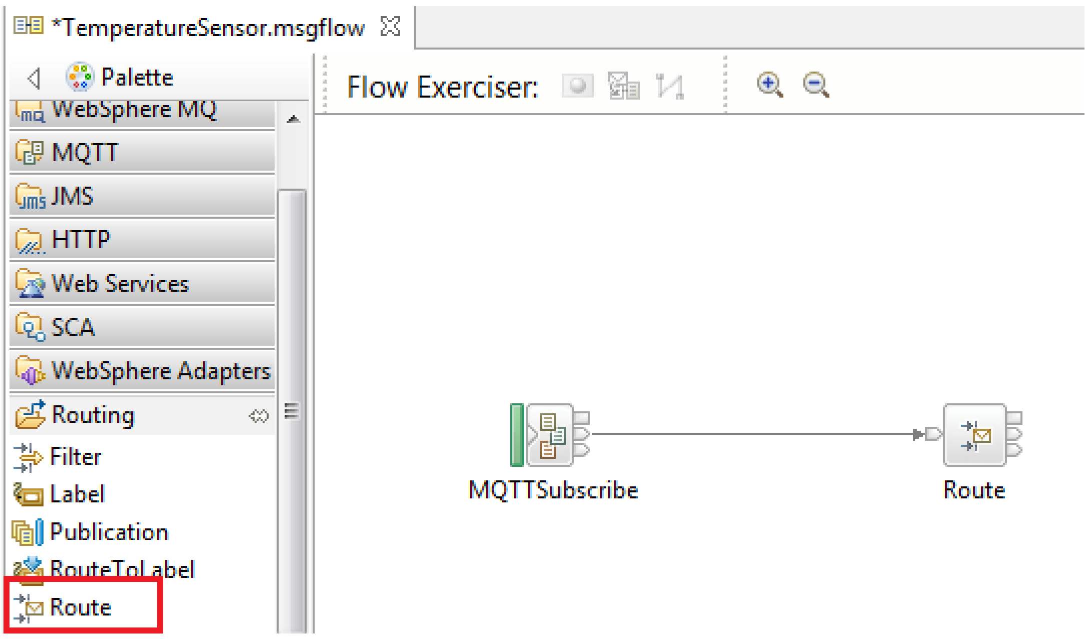
3. Right click on the application and select New → Message Flow to create a new message flow by name 'TemperatureSensor'



4. In the message flow editor, drop an MQTTSubscribe node from the palette.

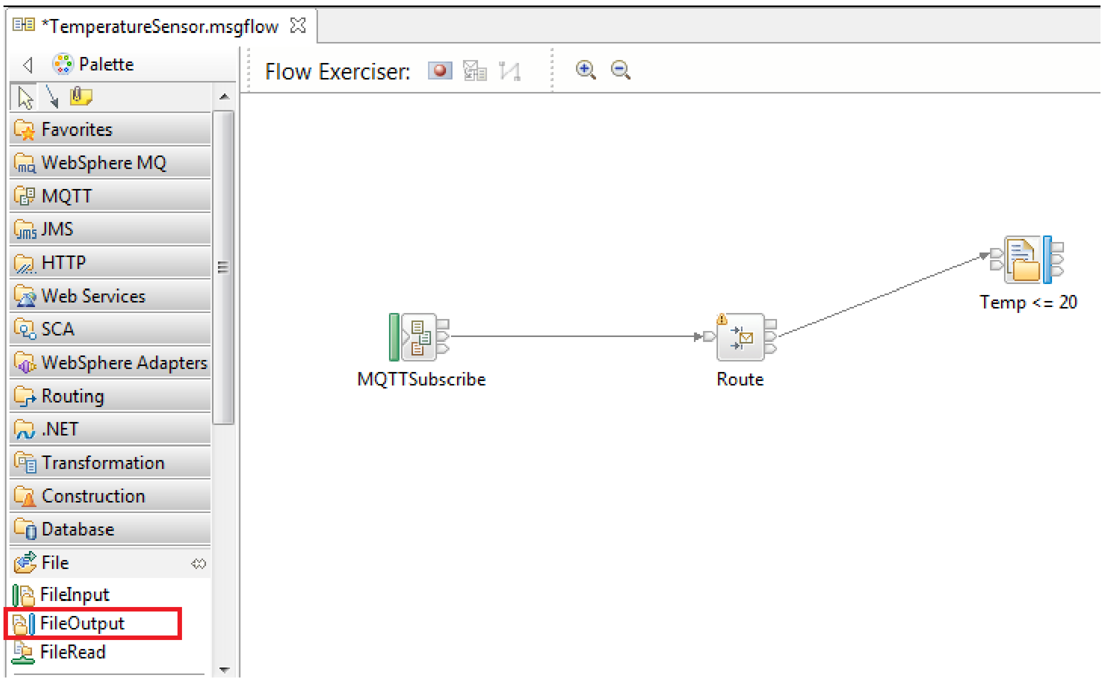


5. Drop a Route node and connect the ‘Out’ Terminal of the MQTTSubscribe node to the ‘In’ Terminal of the Route node.



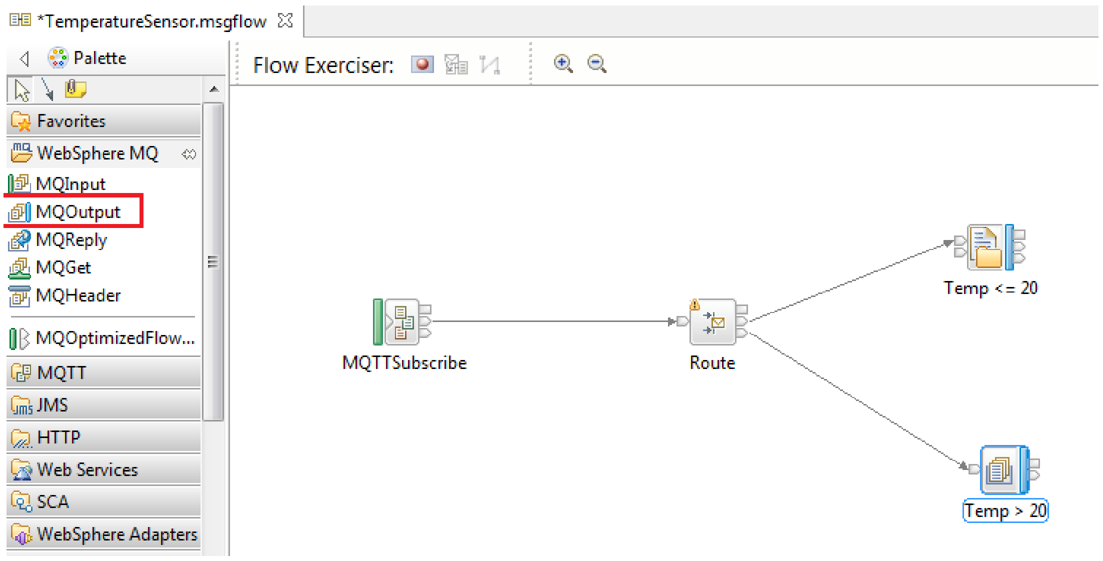
6. Drop a FileOutput node. Rename the node as ‘Temp <= 20’.

Connect the 'Default' terminal of Route node to the 'In' terminal of FileOutput node.



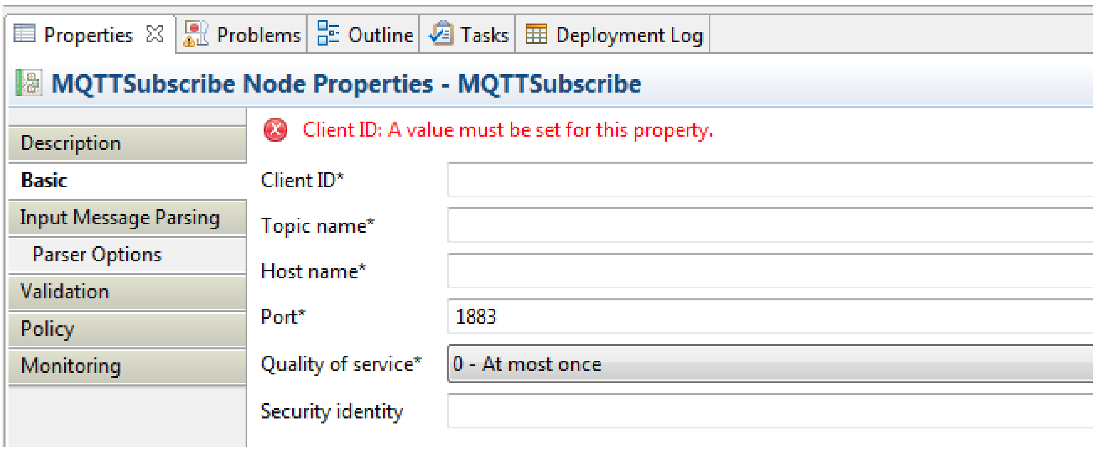
7. Drop an MQOutput node. Name the node as ‘Temp > 20’.

   Connect the ‘Match’ terminal of the Route node to the ‘In’ terminal of the MQOutput node.



8. Now, start configuring the nodes.

   Select the MQTTSubscribe node and look at the Properties view.



MQTTSubscribe Node Properties - MQTTSubscribe

Client ID: A value must be set for this property.

Client ID*	
Topic name*	
Host name*	
Port*	1883
Quality of service*	0 - At most once
Security identity	

9. Fill in the mandatory values.

- **ClientID** – Supply an MQTT client ID of the form a:<org-id>:<unique-name>
  Every MQTT connection must use a unique client ID. Reuse of the same client ID will cause the earlier connection to be closed by the server.

  To subscribe to device messages, you can connect as an application, by using an 'a' at the start of the client ID parameter.
  Enter "quickstart" as your 'org-id'.
  Enter your name followed by your date of birth as the unique-name.

  For example, the ClientID could be "a:quickstart:dave121280"

  NOTE: The ClientID must not be more than 23 characters long. If it is, the application will deploy successfully, but will be unable to connect to the MQTT server.

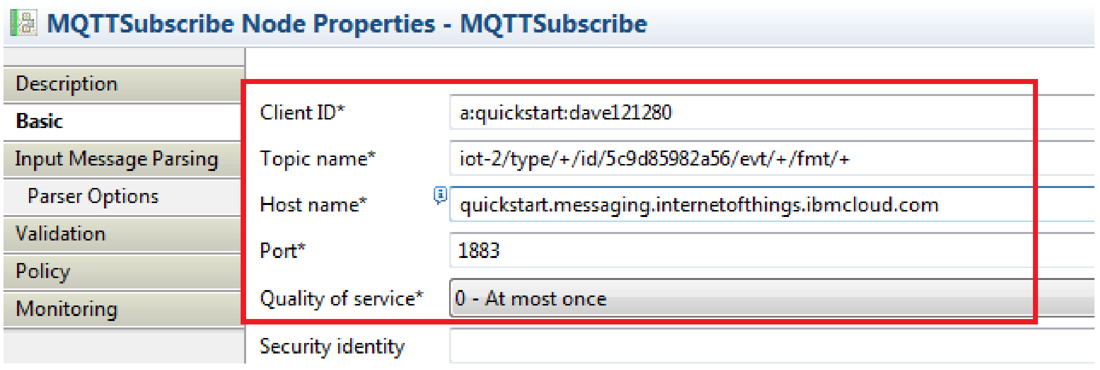
- **Topic name** - For the quick start service, subscribe to the following topic to receive all events from your device: iot-2/type/+/id/<device-id>/evt/+/fmt/+

  where device id is the MAC address that you noted in step 3 in section 2.1 above. For example

  iot-2/type/+/id/5c9d85982a56/evt/+/fmt/+

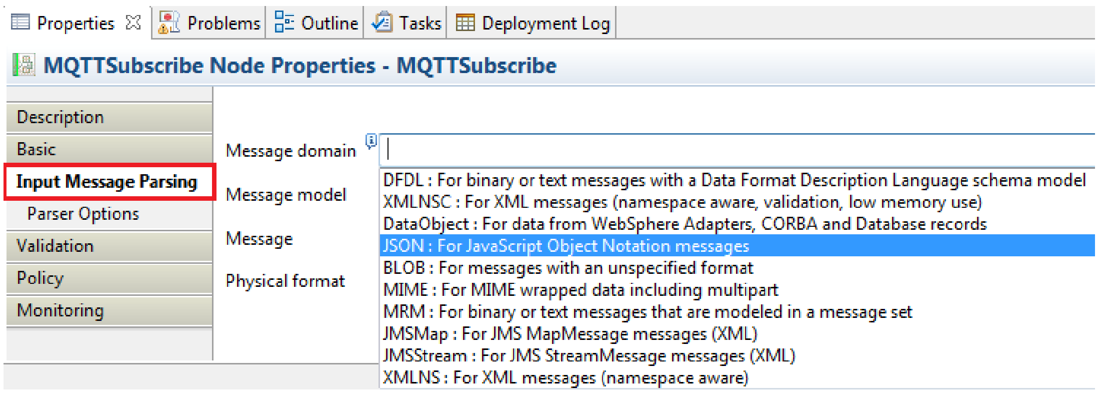
- **Host name** – quickstart.messaging.internetofthings.ibmcloud.com

  Port – 1883


MQTTSubscribe Node Properties - MQTTSubscribe

Client ID*	a:quickstart:dave121280
Topic name*	iot-2/type/+/id/5c9d85982a56/evt/+/fmt/+
Host name*	quickstart.messaging.internetofthings.ibmcloud.com
Port*	1883
Quality of service*	0 - At most once
Security identity	

10. The MQTTSubscribe node will receive data from the Temperature Sensor simulator in JSON format. Because the Route node will be configured to reference the JSON element “temp” by using an XPath statement, the MQTTSubscribe node (being the input node) has to parse the message to enable this. So, on the ‘Input Message Parsing’ tab set the Message Domain to ‘JSON’

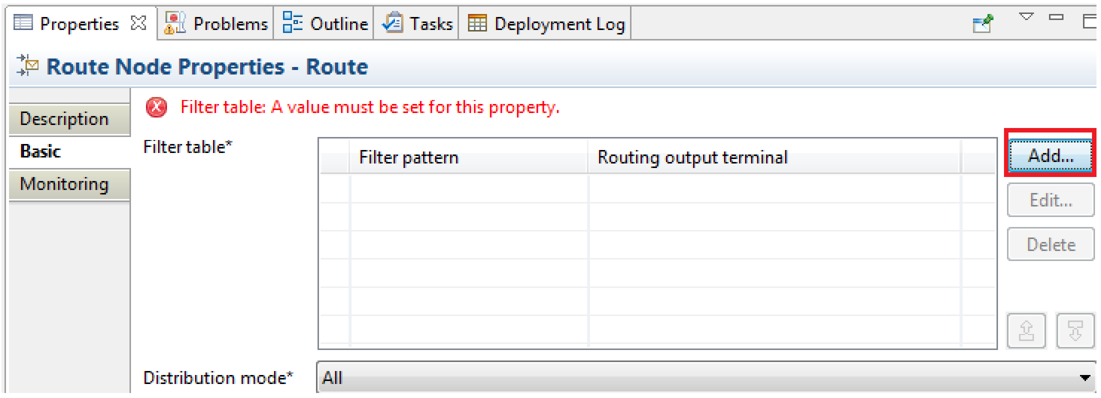


MQTTSubscribe Node Properties - MQTTSubscribe

Message domain
Message model
Message
Physical format

DFDL : For binary or text messages with a Data Format Description Language schema model
XMLNSC : For XML messages (namespace aware, validation, low memory use)
DataObject : For data from WebSphere Adapters, CORBA and Database records
JSON : For JavaScript Object Notation messages
BLOB : For messages with an unspecified format
MIME : For MIME wrapped data including multipart
MRM : For binary or text messages that are modeled in a message set
JMSMap : For JMS MapMessage messages (XML)
JMSStream : For JMS StreamMessage messages (XML)
XMLNS : For XML messages (namespace aware)

11. Now select the Route node to view its properties. Click on ‘Add’ to set the Filter pattern.



Route Node Properties - Route

Filter table: A value must be set for this property.

Filter table*

Filter pattern	Routing output terminal

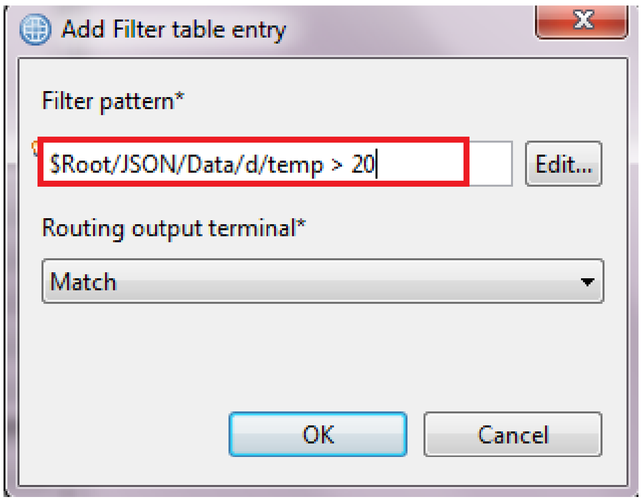
Distribution mode* All

12. Enter the Filter Pattern as

`$Root/JSON/Data/d/temp > 20`

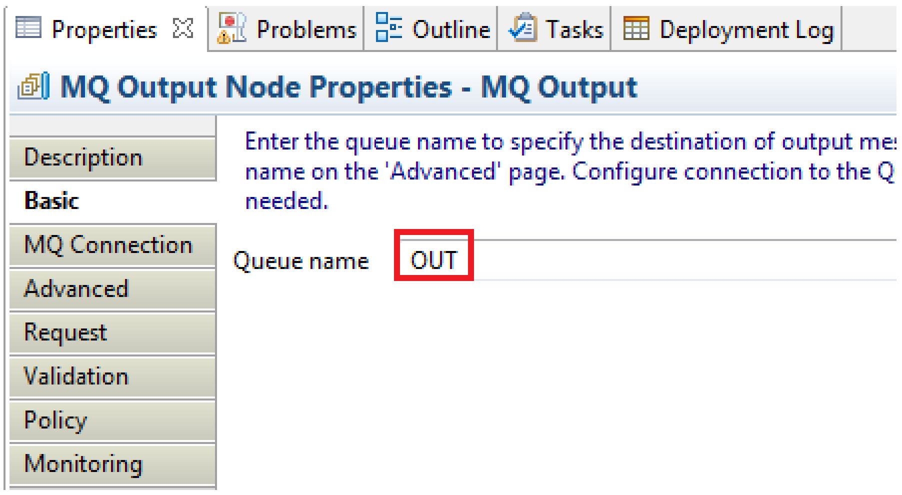
This means when the temperature value is more than 20 degrees, the message will be sent through the 'Match' terminal. If not, message will be sent through the 'Default' terminal.

Click OK.



Note: Please ignore the warning message that appears. The message indicates that the specified element is not available in any schema.

13. Continue with configuring the nodes. Select the MQOutput node to view its properties. On the Basic tab, set the queue name to 'OUT'.

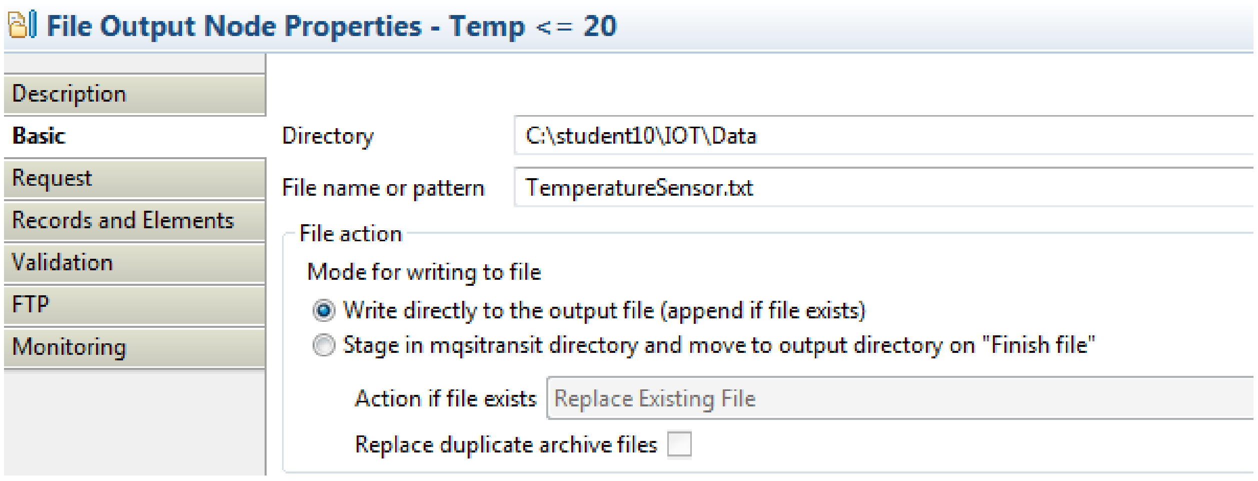


Enter the queue name to specify the destination of output me
name on the 'Advanced' page. Configure connection to the Q
needed.

Queue name OUT

14. Now select the FileOutput node and enter the details of the file location where the subscription messages have to be written to.

- Directory name – C:\student10\IOT\Data
- File name or Pattern – TemperatureSensor.txt
- Mode for writing to file – Select “Write directly to the output file(append if file exists)”


File Output Node Properties - Temp <= 20

Directory	C:\student10\IOT\Data
File name or pattern	TemperatureSensor.txt

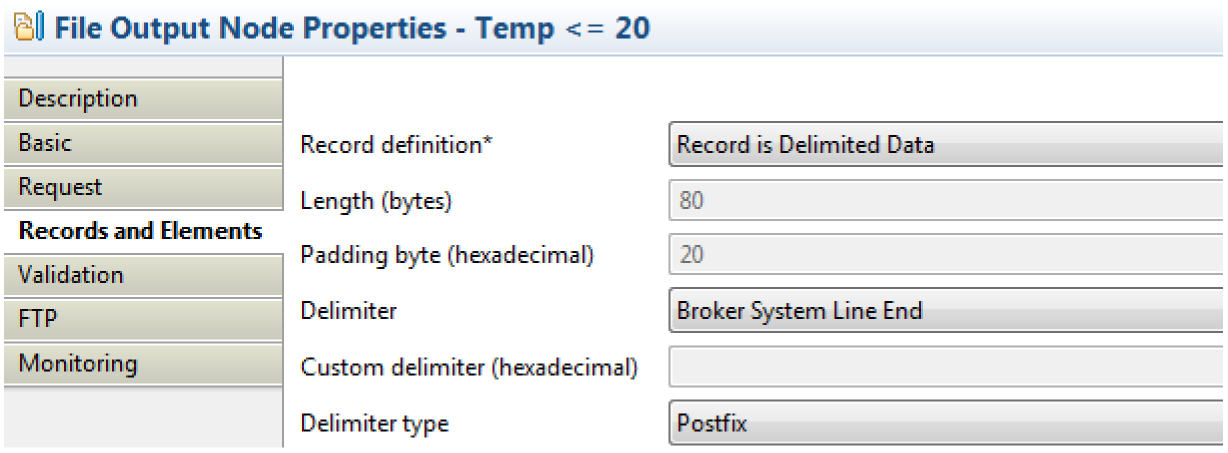
File action

Mode for writing to file

- [✓ Write directly to the output file (append if file exists)]
- Stage in mqsitransit directory and move to output directory on "Finish file"

Action if file exists	Replace Existing File
Replace duplicate archive files	☐

On Records and Elements, set Record Definition to "Record is Delimited Data"

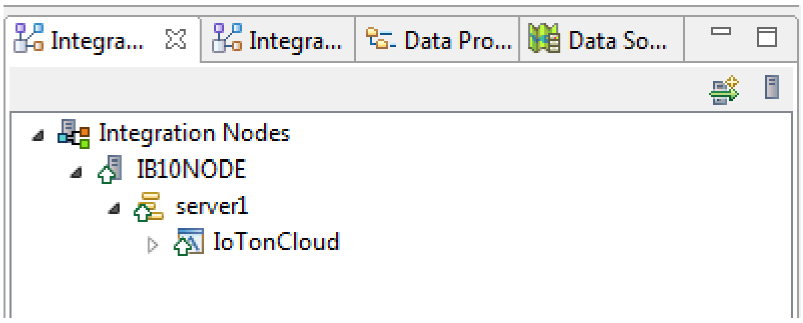

File Output Node Properties - Temp <= 20

Record definition*	Record is Delimited Data
Length (bytes)	80
Padding byte (hexadecimal)	20
Delimiter	Broker System Line End
Custom delimiter (hexadecimal)	
Delimiter type	Postfix

15. Save the flow Ctrl+S.

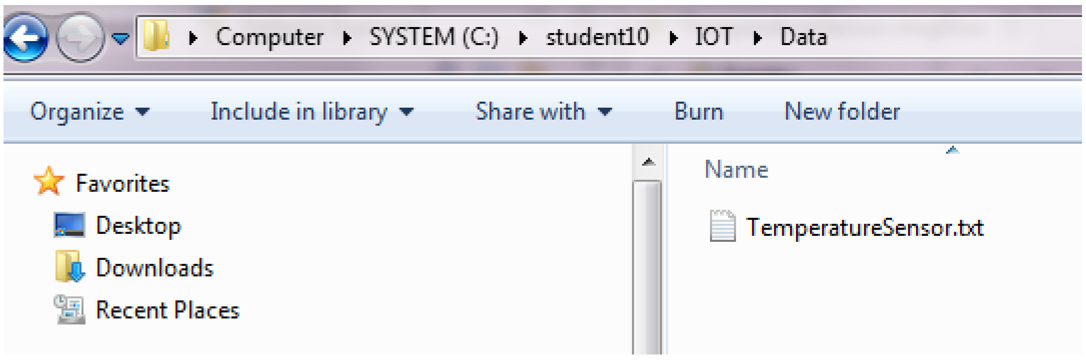
2.3 Deploy and test the MQTT application

1. Drag and drop the application 'IoTonCloud' on the 'server1' execution group under IB10NODE.

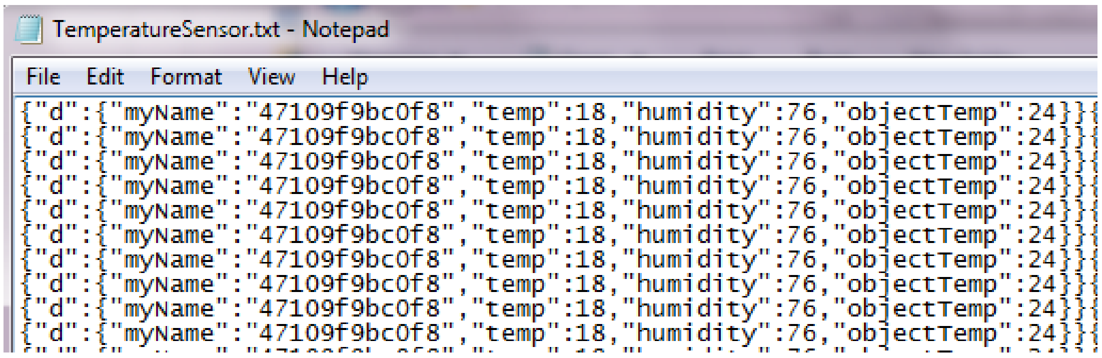


2. Once the flow is deployed successfully, the MQTTSubscribe node starts receiving the messages from the device simulator which was started in step 1 above.

3. If the published temperature is below 20, the subscription messages will be written to the file configured in the FileOutput node of the flow. You can check the directory C:\student10\IOT\Data for the file TemperatureSensor.txt.

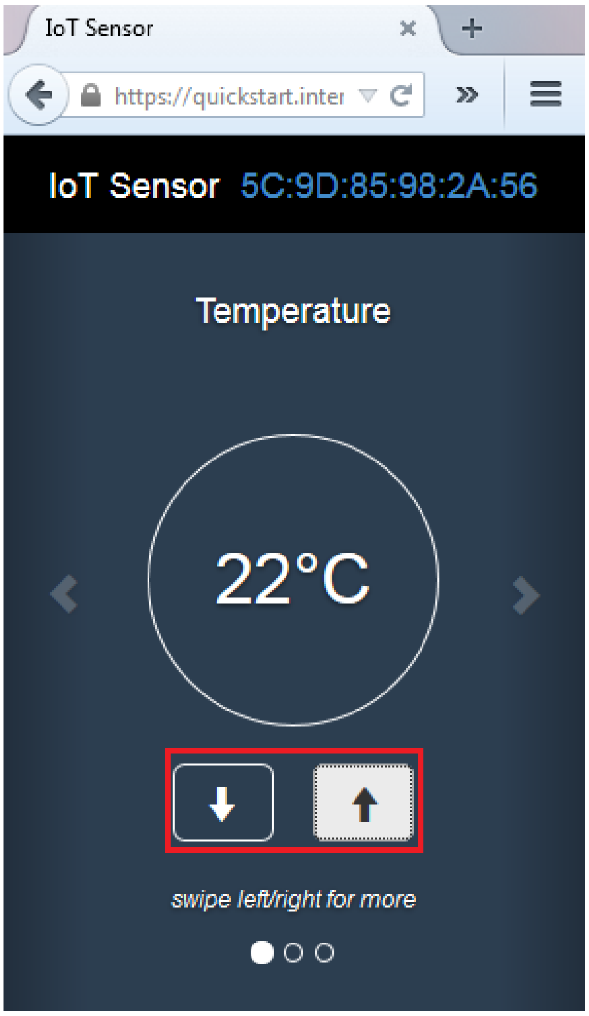


4. Open the file to view the received messages. The subscription messages are in JSON format



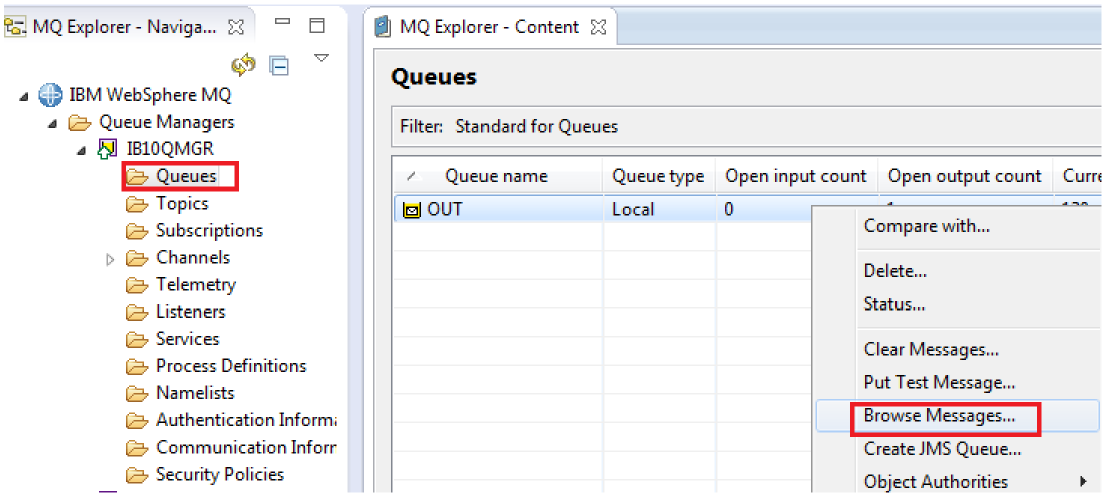
```
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
{"d":{"myName":"47109f9bc0f8","temp":18,"humidity":76,"objectTemp":24}}{
```

5. Now increase the temperature in the device simulator using the arrows available on the device



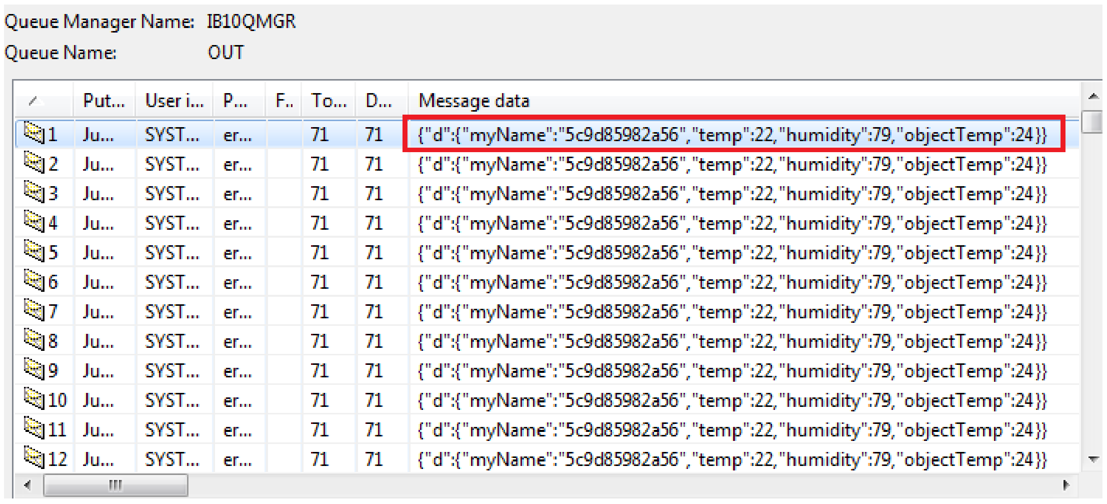
This action publishes messages to the MQTT server. As IIB subscribed to these messages using the MQTTSubscribe node, you can see the latest subscriptions written to the file.

As you increase the temperature to above 20 the messages are routed to MQ queue OUT. Open the MQExplorer. Select the IIB10QMGR -> Queues. Right click on OUT queue and select 'Browse messages'



Queue name	Queue type	Open input count	Open output count
OUT	Local	0	

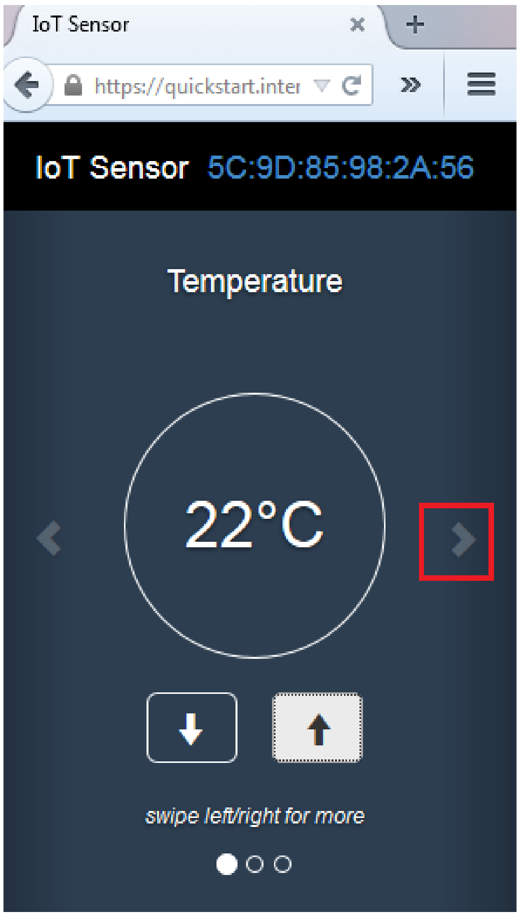
You can see the message data as shown below



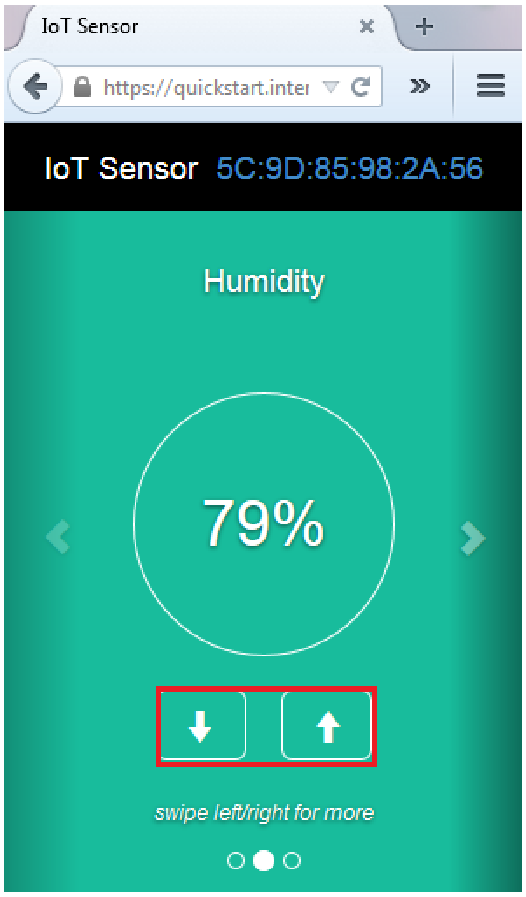
Queue Manager Name: IB10QMGR
Queue Name: OUT

	Put...	User i...	P...	F..	To...	D...	Message data
1	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
2	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
3	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
4	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
5	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
6	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
7	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
8	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
9	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
10	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
11	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}
12	Ju...	SYST...	er...		71	71	{"d":{"myName":"5c9d85982a56","temp":22,"humidity":79,"objectTemp":24}}

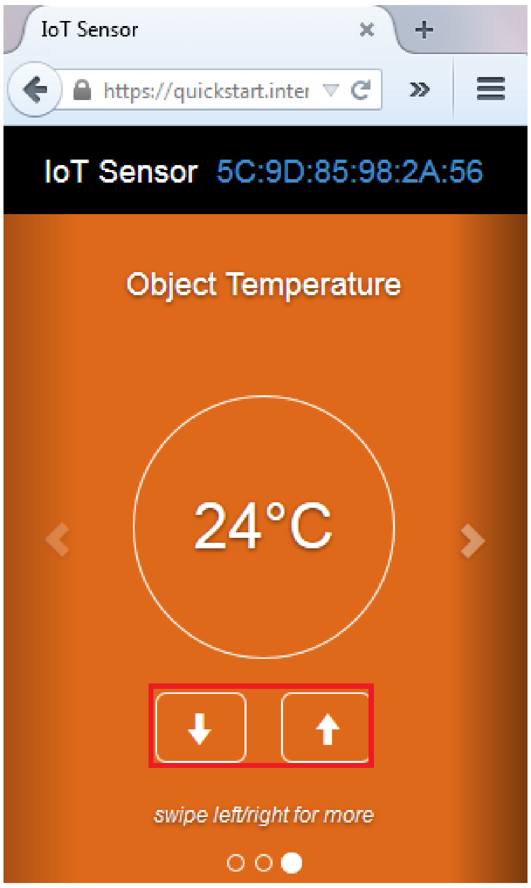
6. In the device simulator move to the 'Humidity' screen by clicking on the right arrow



7. You can modify the humidity values in here and check the file to see if you received the latest subscriptions



8. Click on the right arrow to move to the 'Object Temperature' screen and adjust the temperature values



9. Check the output file as you change the values of Humidity, Object Temperature. Stop the application when you have finished the testing.

   In Summary, the changes in any device (MQTT client) are published as messages to the MQTT server instance in the cloud and message flow nodes in IIB can subscribe to those messages and use them for any further processing or backend integration.

END OF LAB GUIDE