# IoT with the MQTT Plugin

In this project you will create an app displays data from an IoT device using the Mendix MQTT plugin.

### Prerequisites

* This case was prepared using version 7.21.0 of the desktop Mendix Modeler.
* You must have access to either a MQTT capable device or a simulator

## MQTT

MQTT is a standard

MQTT Broker

MQTT Publisher

MQTT Subscriber

## Create the App

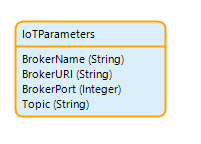
Create an app using the Blank template. Name the app according to the format provided by your instructor.

## Create the Domain Model

In this section we’ll lay the groundwork before connecting to the MQTT broker.

### Create the IoTParameters Entity

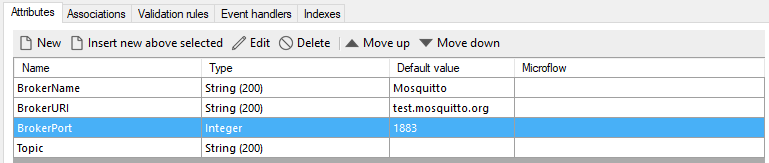
Create the following entity in the MyFirstModule domain model.



In this case we’ll be using the Mosquitto.org public broker. You can include the parameters values as default values to make it easier to work on the app.

**Attribute Value**

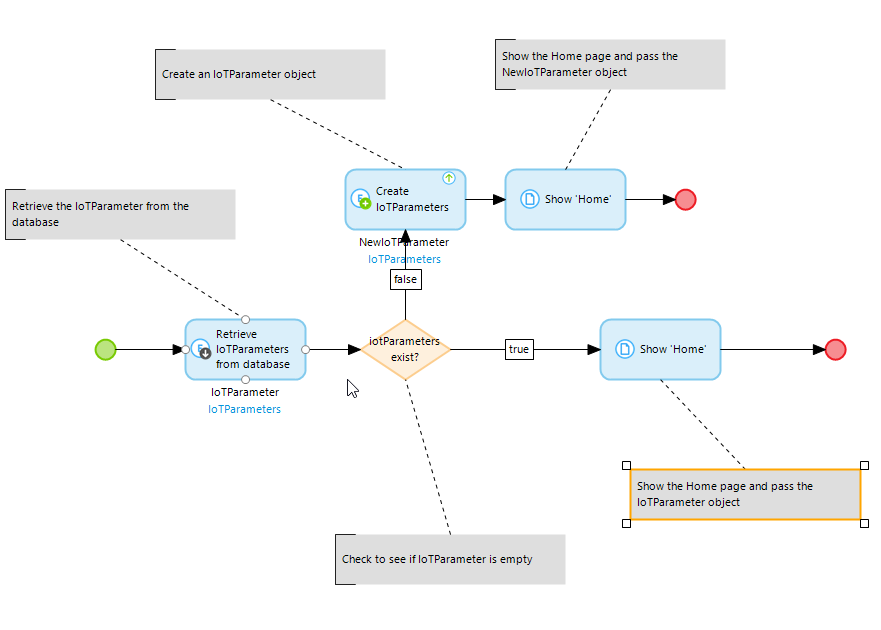
BrokerName Mosquitto  
BrokerURI test.mosquitto.org  
Port 1883



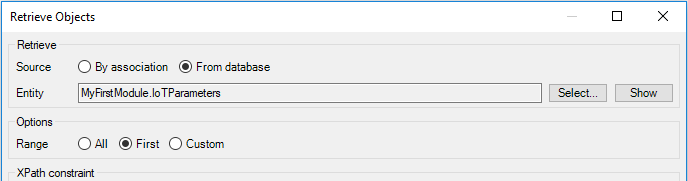
### Create a Microflow to Open the Home Page

Create a folder called Microflows in MyFirstModule and place all microflows in the case in this folder.

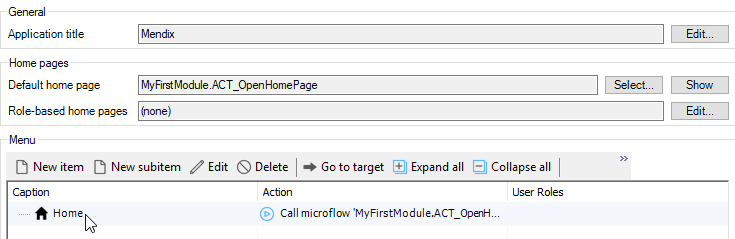
An IoTParameters object will be the context for the Home page so we must create a microflow to open the home page that will retrieve the IoTParameter object from the database or create it if it doesn’t exist. The final microflow looks like this.



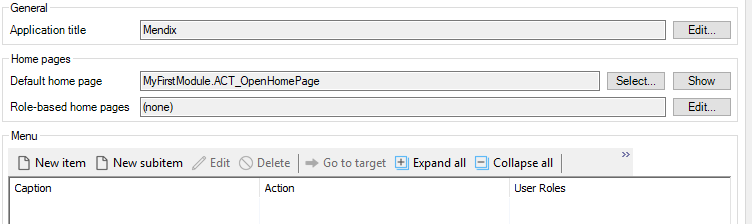
Most of this is straight forward. The only detail that may not be obvious is that you have to configure the first activity to retrieve only the First object.



Now you have to configure the Navigation to use the microflow rather than simply opening the page. Make sure you update the navigation in both places (Default home page and Menu) on the Responsive navigation profile.



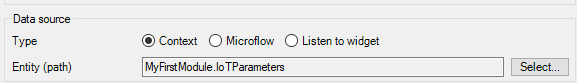
Update the Default home page on the Hybrid phone app online navigation profile as well.



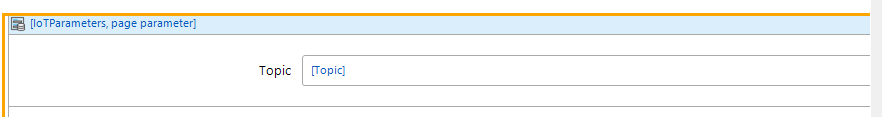
### Edit the Home Page

Create a folder called Pages in MyFirstModule and place all pages in the case in that folder. You can drag the Home page into the Pages folder.

Add a Data view widget to the body of the Home page and configure its datasource to be the IoTParameter entity.

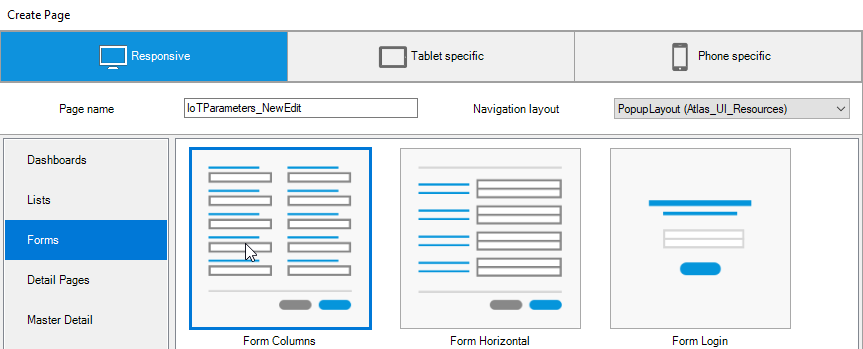


We want the Topic attribute widget on this page so you can either allow Mendix to automatically fill the contents of the Data view then delete everything except the Topic attribute or choose not to allow Mendix to fill the contents and drag the Topic attribute from the Connector tab.

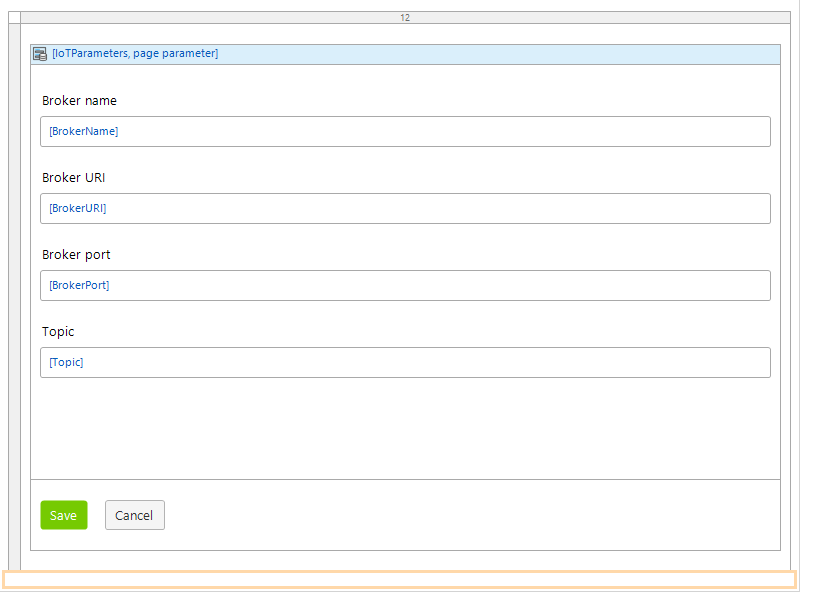


### Create a Page to Edit the IoT Parameters

We’ll use the Save and Cancel buttons later but for now add a third button next to the Save and Cancel buttons and configure it to open a page called IoTParameters\_NewEdit. Configure the page as shown



The page should have a Data view bound to IoTParameters and look like this:



## Configure the MQTT Plugin

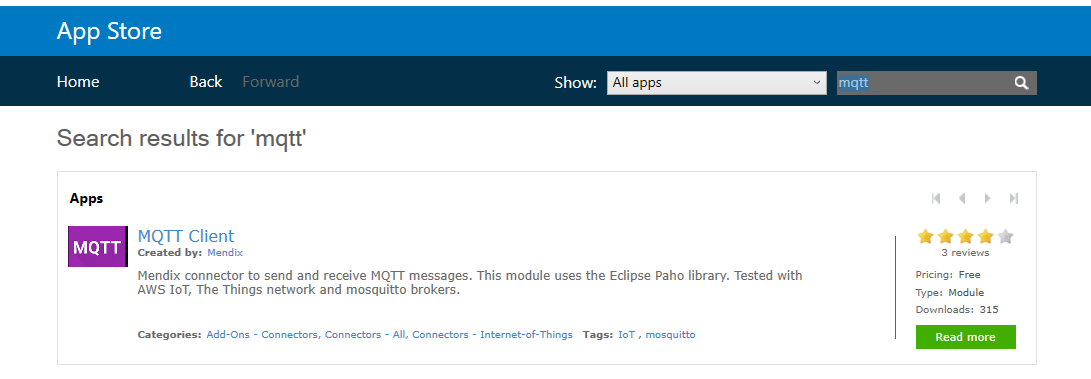
Now that we have some basic out of the way we can begin the configuration of the MQTT plugin.

### Add the MQTT Plugin

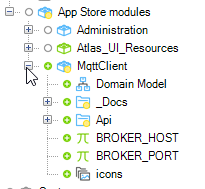
The MQTT Plugin can be found on the Mendix App Store. Click the shopping cart icon on the top right.



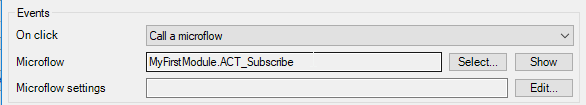
Search for MQTT.



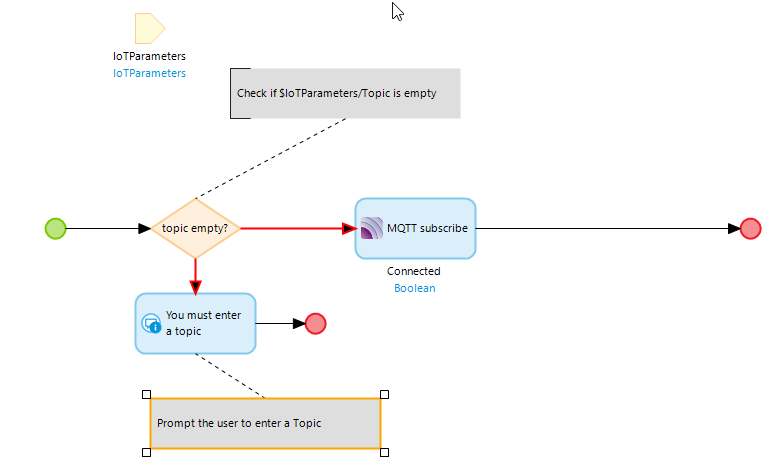
Click MQTT Client and then click Download. When prompted choose to import the module into your project as a new module. The module is added and can be viewed in the Project Explorer.



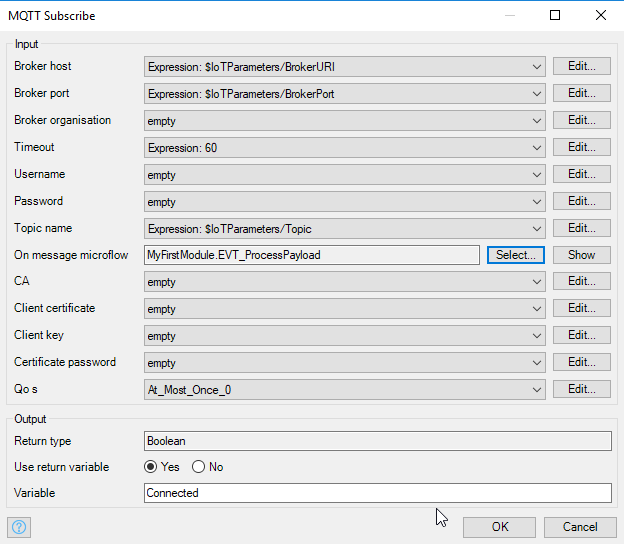
### Create a Microflow to Subscribe to a Topic

Edit the Save button on the Home page and change its Caption property to “Subscribe” and configure the Events section as shown in the image below. Create the ACT\_Subscribe microflow in the Microflows folder.

The ACT\_Subscribe microflow looks like the image below. The first thing we do is use an Exclusive split to check to make sure that Topic isn’t empty. If it is, we show the user a message and end.



The MQTT subscribe activity is configured as shown below. The EVT\_ProcessPayload microflow will be called each time a message is received from the broker. The Connected variable will be true if we were able to connect successfully.

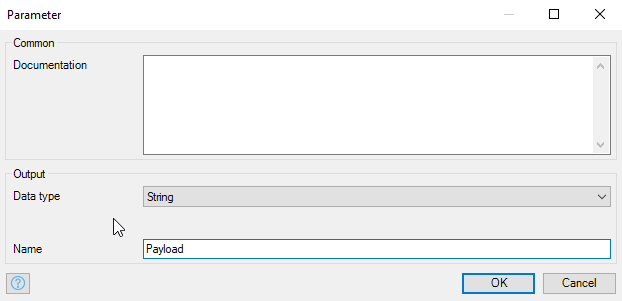


### Create the EVT\_ProcessLayload Microflow

The microflow that processes messages from the broker requires two parameters: Payload and Topic. These are both string variables.



The configuration of Payload looks like this for example.

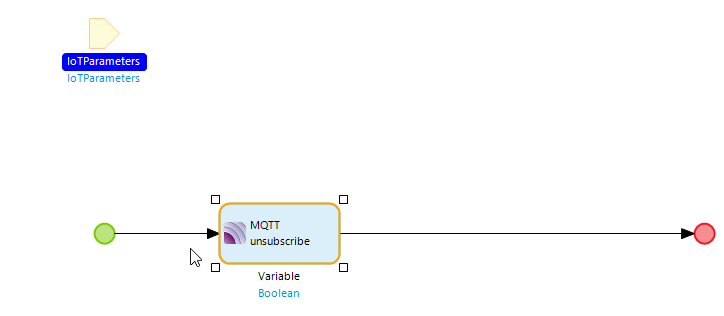


We have to add a lot more to this later but for now, this will work.

### Create a Microflow to Unsubscribe

Configure the Cancel button on the Home screen to run a microflow called ACT\_Unsubscribe.

The microflow is shown below.

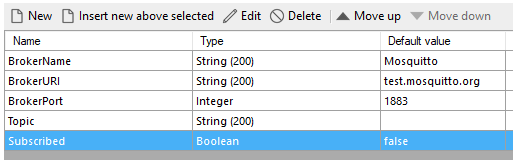


The Home page buttons look like this now.

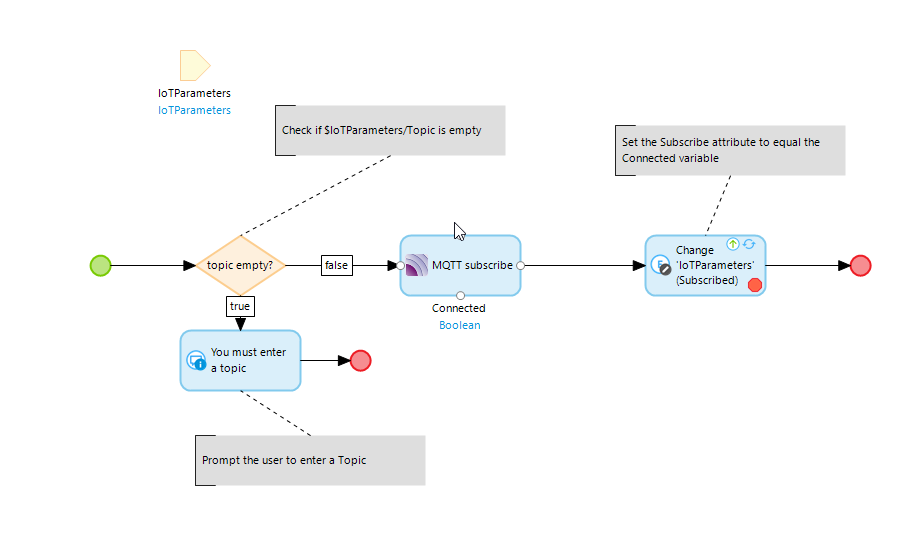
### Add an Attribute to Manage the Visibility of the Subscribe and Unsubscribe Buttons



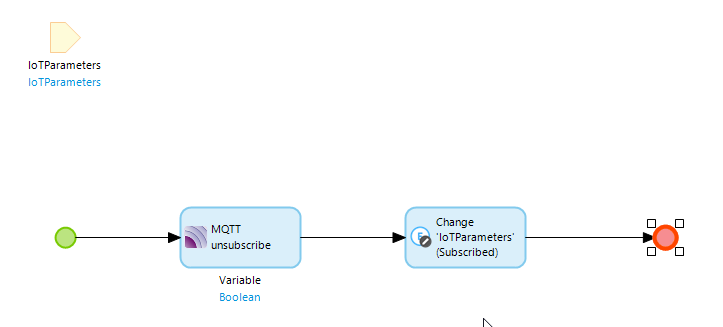
It doesn’t make much sense to show both the subscribe and unsubscribe buttons at the same time. The subscribe button should show only if we have not subscribed yet. Add an attribute to the IoTParameters called subscribed.



Update the ACT\_Subscribe microflow by adding a Change object activity at the end that sets the Subscribed attribute of the IoTParamters object equal to the Connected variable. Remember to commit the object and refresh in client.



Add a similar activity to the ACT\_Unsubscribe microflow to set the Subscribed attribute to false.

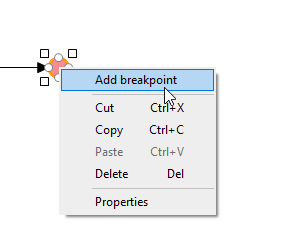


Configure the visibility on the subscribe and unsubscribe buttons to depend on the Subscribed attribute.

### Copy the Payload

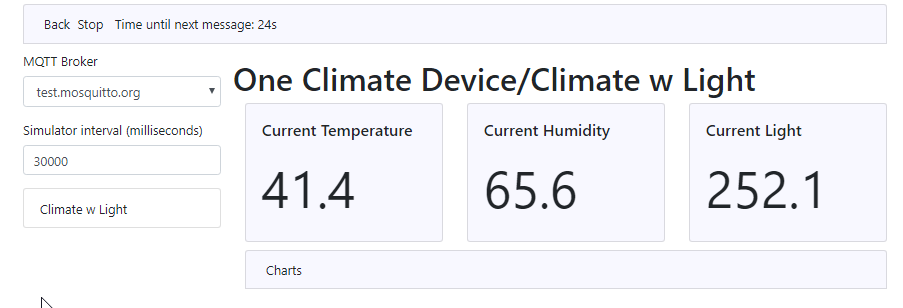
To test the app you must have either a physical sensor capable of connecting to an MQTT broker or a simulator. The remainder of this case uses a simulation of a climate sensor.

Open the EVT\_ProcessPayload microflow and create a breakpoint on the End event.

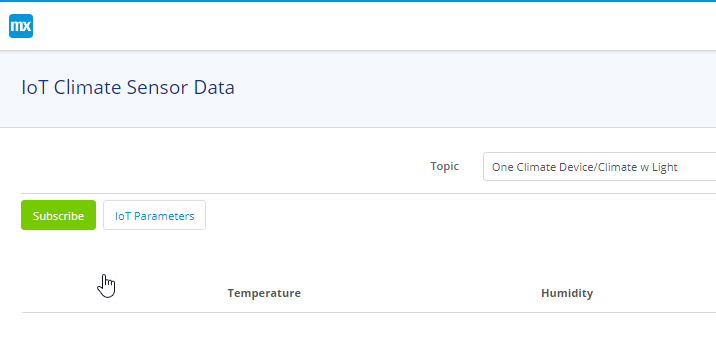


This will cause the app to pause when execution reaches this point. We will use this to copy the payload so we can use it to create the import structures required to process payloads.

The image below shows a simulator that simulates a climate sensor.



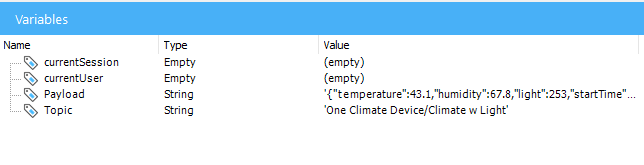
Run the app and, once the device or simulator has published the topic to the MQTT broker, enter the topic in the topic field and click the Subscribe.



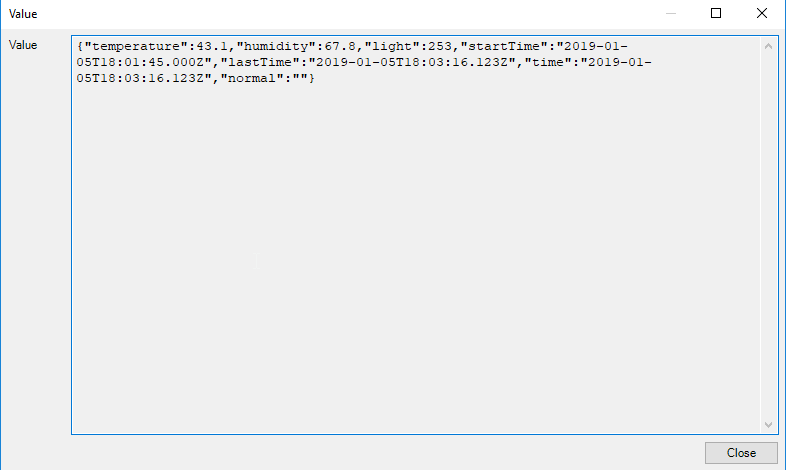
When a message arrives, the execution will pause (you’ll see a red border around the End event).



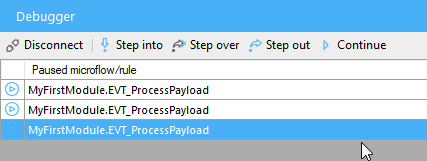
Open the Variables tab at the bottom of the modeler.



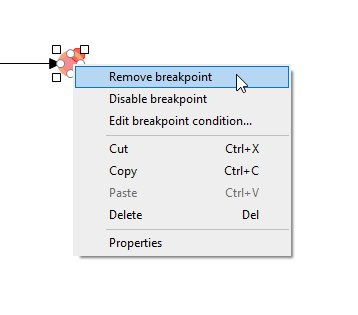
Double-click the Payload variable and copy the Value and save it somewhere.



When you are ready, open the Debugger tab and click Continue to continue the execution.



Remove the breakpoint and you can unsubscribe.

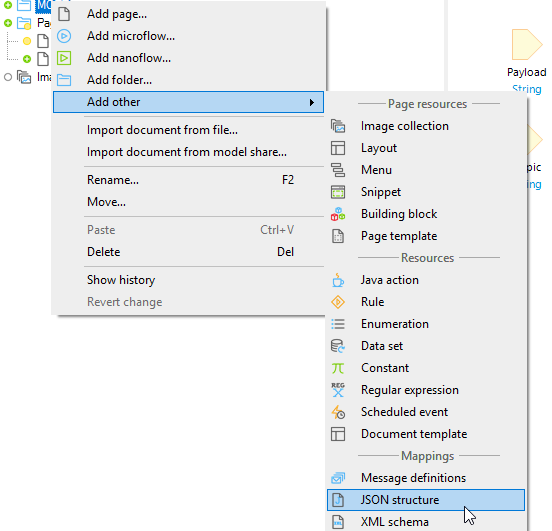


## Process the MQTT Payload

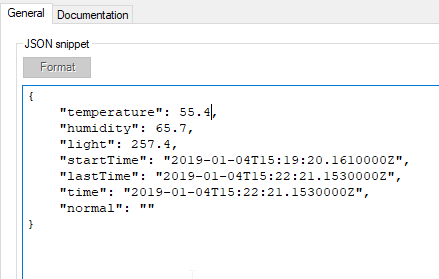
The payload from the MQTT broker is the message sent by the publisher (the device or simulator) and it arrives as a string as you saw in the previous section. To parse the string and insert it into the database, we have to create an import mapping.

### Create the JSON Structure

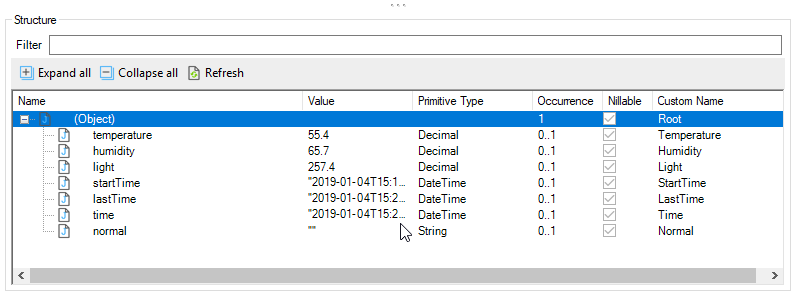
Create a new folder in the MyFirstModule moduled called MQTT and then create a JSON Structure in the folder.



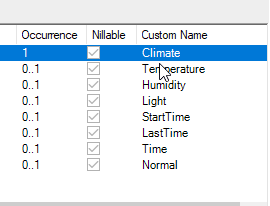
Paste the JSON string you copied in the previous section as the JSON snippet and click Format. Mendix will infer the data types from the values in the snippet so make sure they are the types you want. For example, the temperature in the snippet was an integer and was edited to be a decimal.



Click Refresh to interpret the JSON.

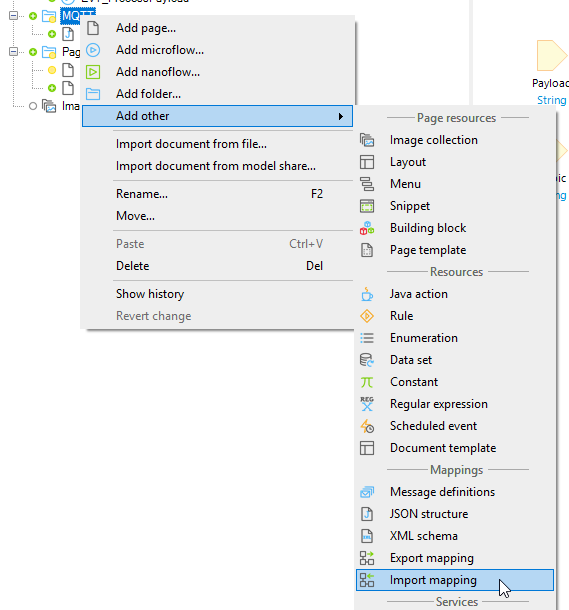


Note the (Object) name is Root and this will be the name of the domain model entity. Root isn’t very descriptive so edit the name to reflect the type of data you will store.

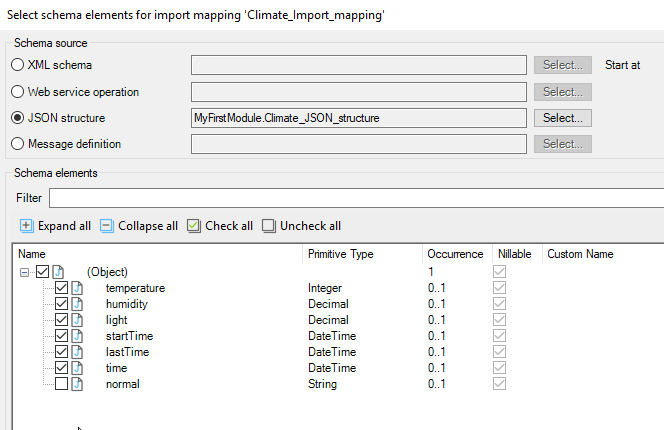


### Create the Import Mapping

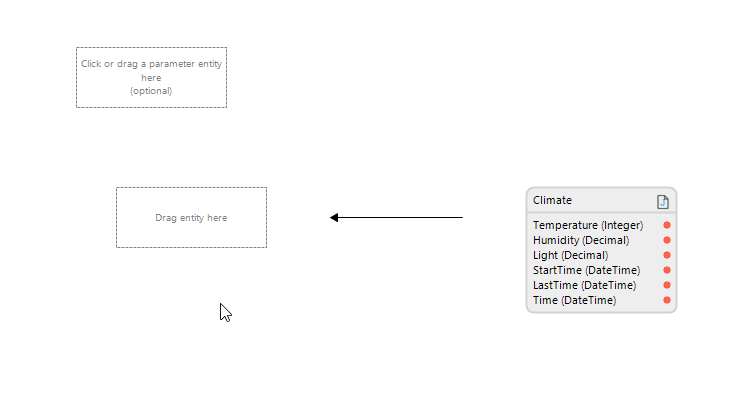
Now create an Import mapping in the MQTT folder.



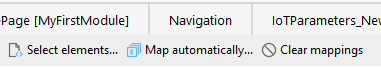
Select the JSON structure you just created and select all the attributes you want to store.



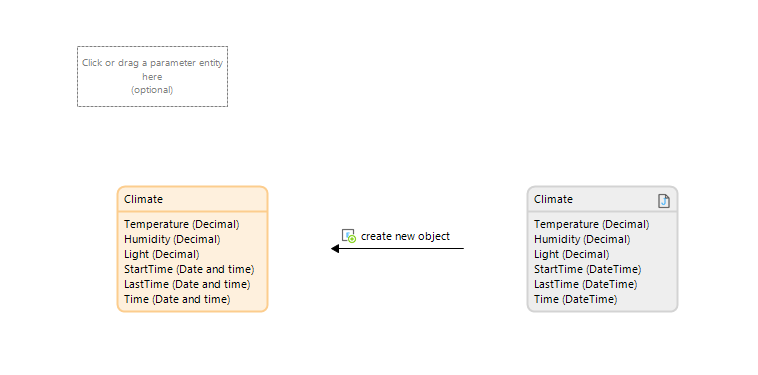
When you click OK, the Import mapping will look like this.



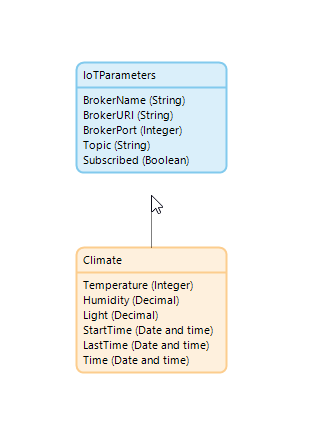
Click Map automatically at the top of the editor to create the domain model entity.



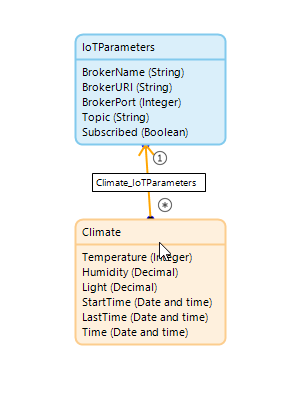
The entity is on the left.



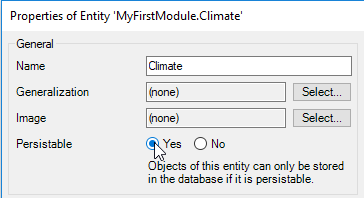
You can also find it in the domain model. It’s yellow because it is created as a nonpersistable entity (the data will not be stored in the database). Create an association between Climate and IoTParameters by clicking on the border of Climate and dragging-and-dropping onto the border of IoTParameters.



This makes a one-to-many association.

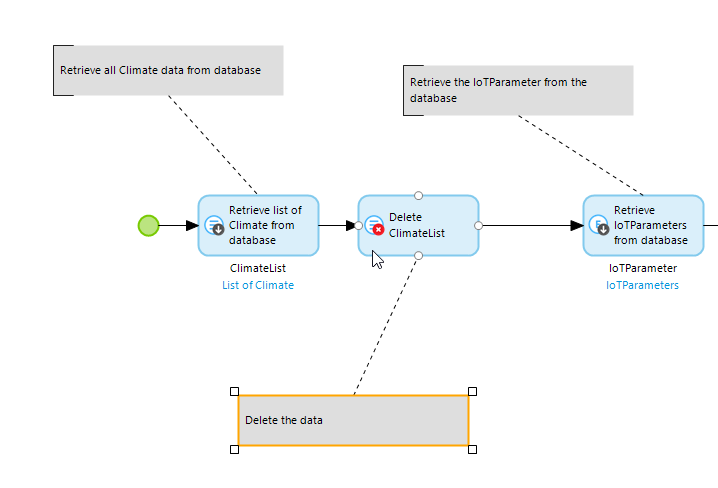


Finally, open the properties of the Climate entity and make it persistable.

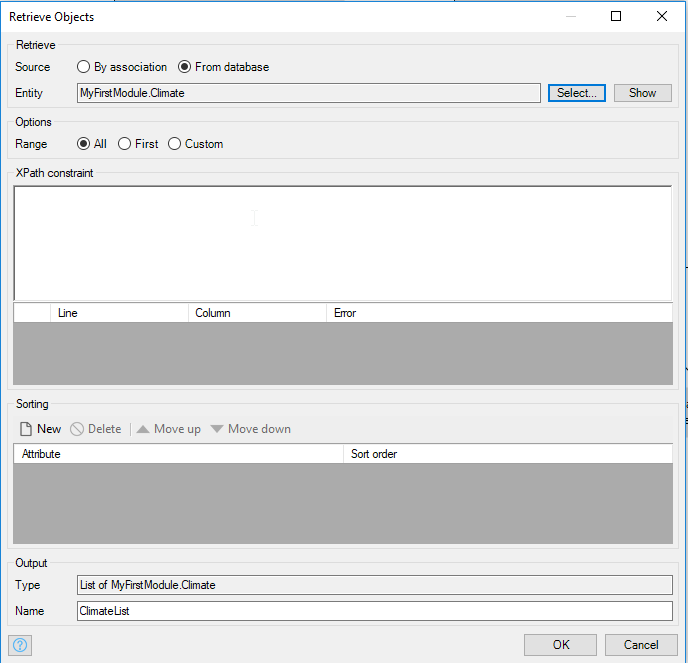


### Update the ACT\_OpenHomePage Microflow

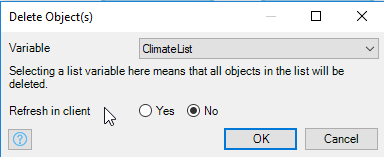
Because we have to make the Climate entity persistable, we need to make sure that we delete all the data store in the entity when the app starts. Open the ACT\_OpenHomePage microflow and add two activities to the beginning.



First retrieve all the Climate data….

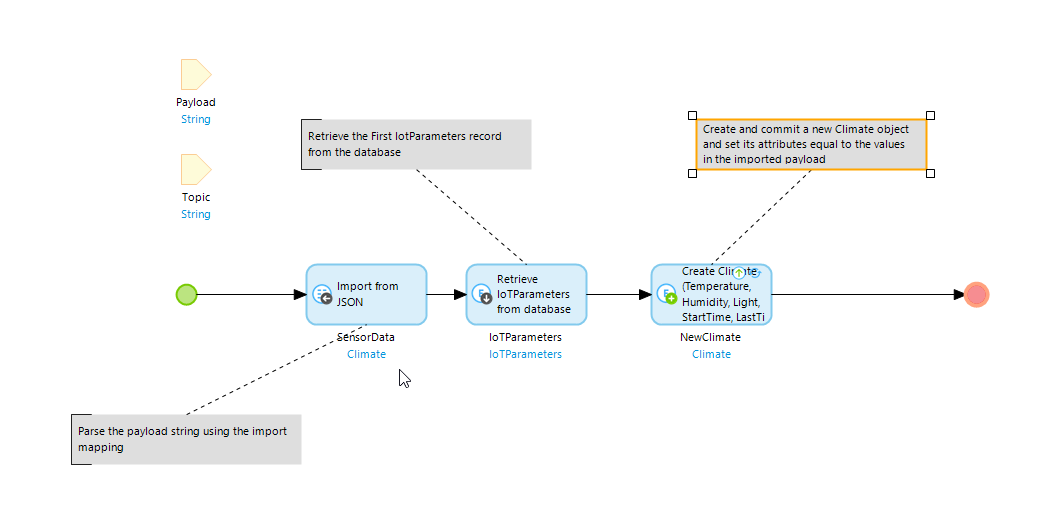


Then delete it.

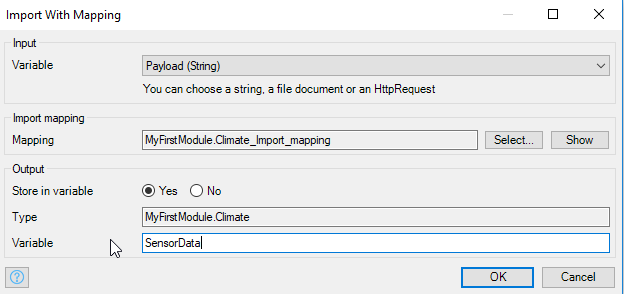


### Complete the EVT\_ProcessPayload Microflow

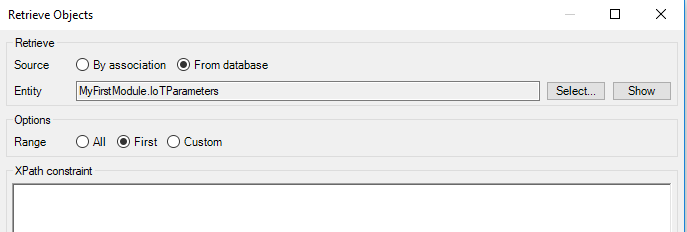
Now we can complete the EVT\_ProcessPayload microflow.



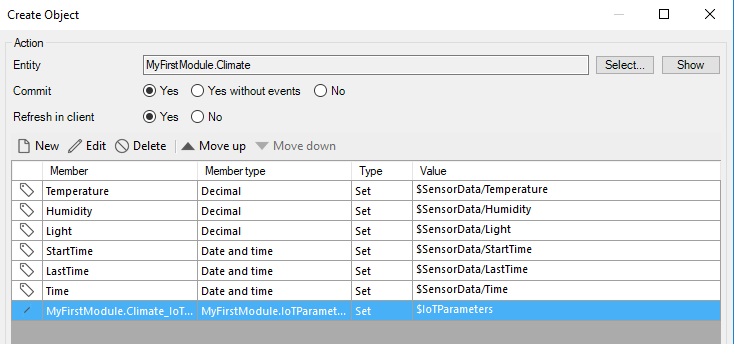
The first activity is an Import from JSON activity configured as shown below.



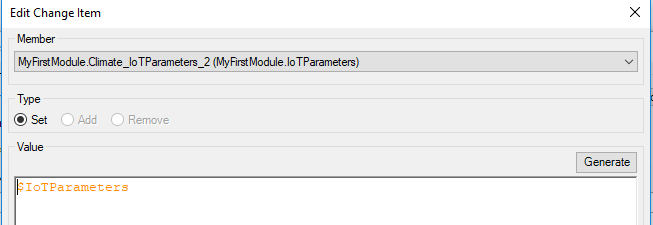
Next, retrieve the IoTParameters object.



Finally, create a Climate object. Remember to Commit it.

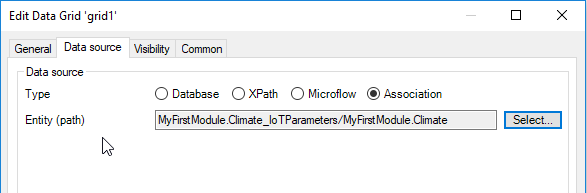


Set the association equal to the IoTParameter object.

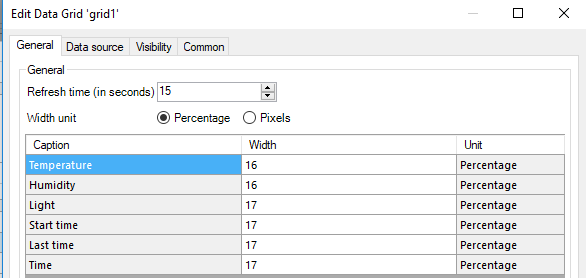


### Edit the Home Page

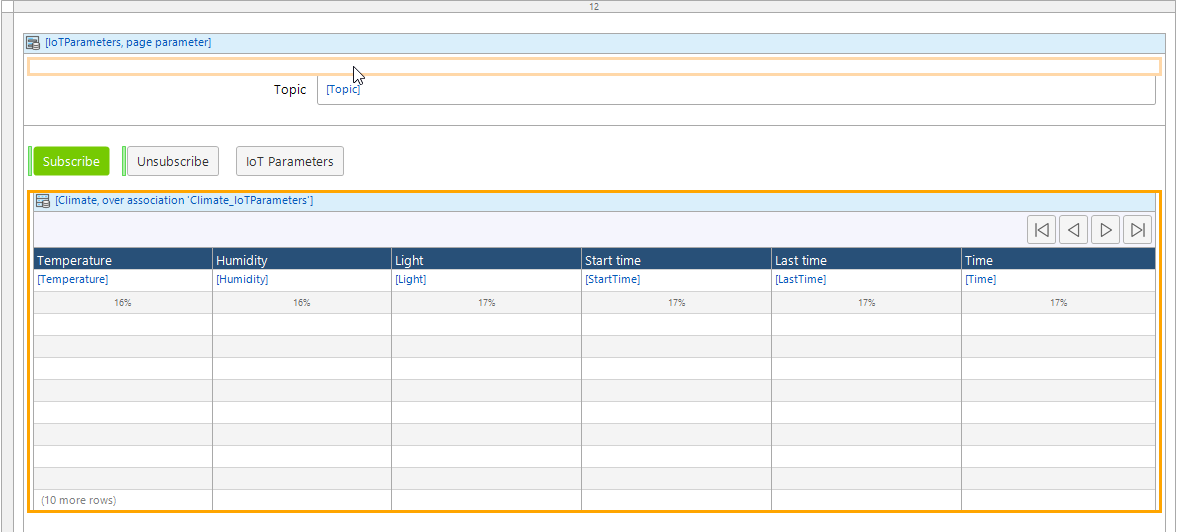
Add a Data grid to the Home page below the buttons but inside the Data view. Configure the data source for the Data grid using the association between IoTParameters and Climate.



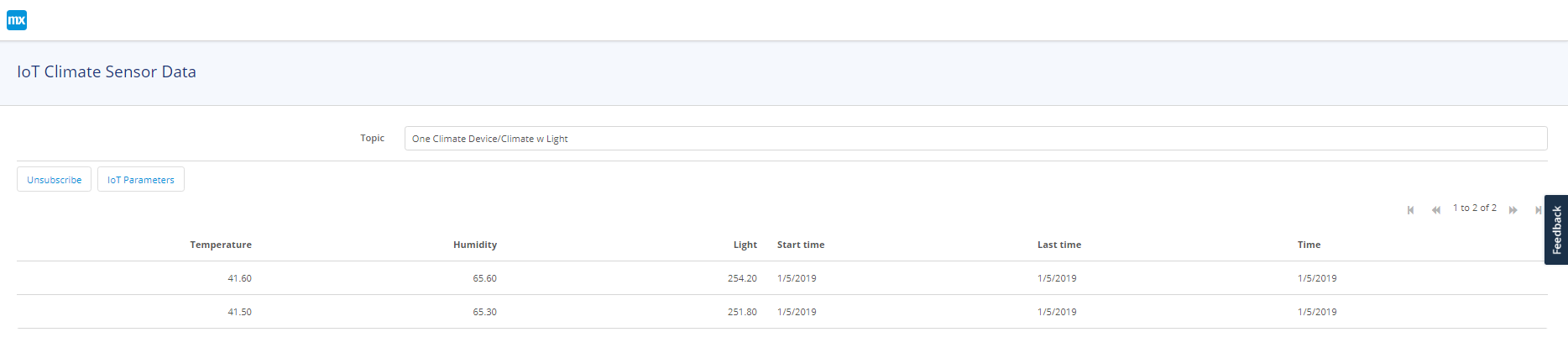
Set the Refresh time to a value greater than 0.



Allow Mendix to fill the contents of the Data grid automatically.



Now the data will appear in the grid.

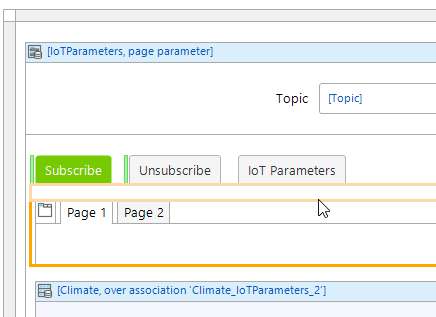


## Add Graphs

A table isn’t a particularly interesting way to view the data so let’s add some graphs.

### Add a Tab Widget

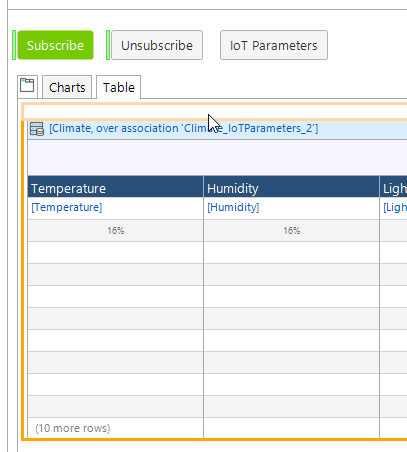
We’ll keep the table but put in a separate tab. Drag a Tab container widget from the Toolbox tab onto the home page, just below the buttons.



Change the Captions for the tabs to Charts and Table.

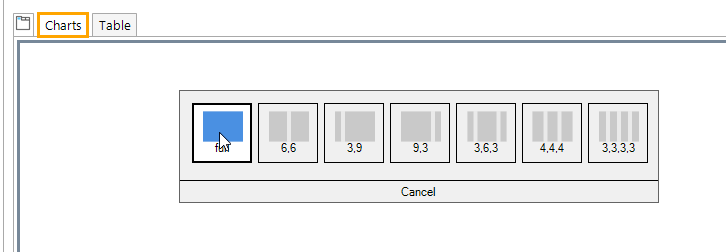


Select the Table tab and drag the Data grid onto it.

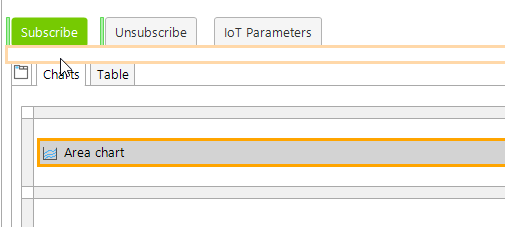


### Add a Temperature Chart

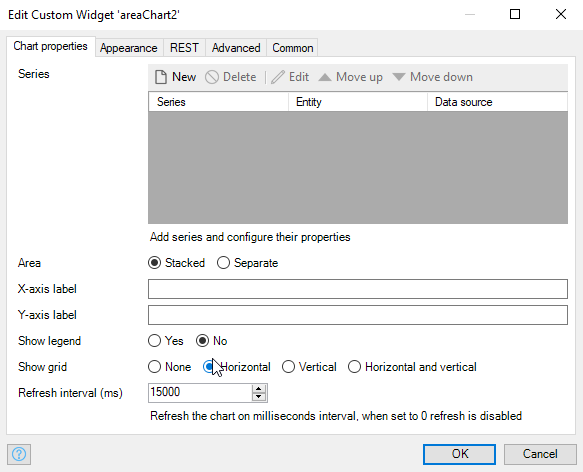
Drag a Layout grid onto the Charts tab and select the full layout.



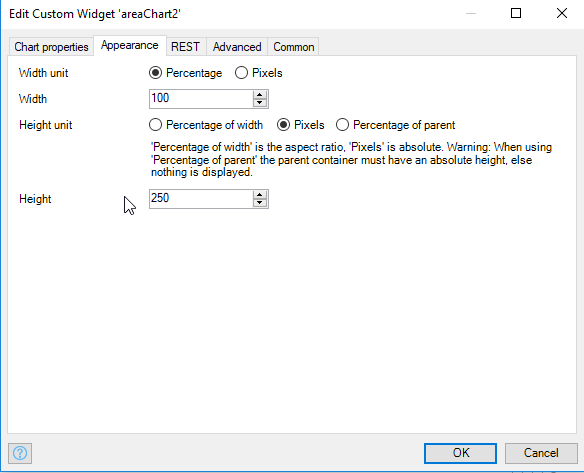
Drag a Area Chart into the Layout grid.



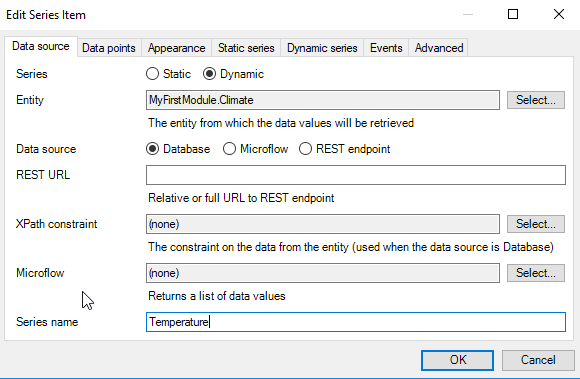
Open the Area Chart properties and configure the Refresh interval to 15000.



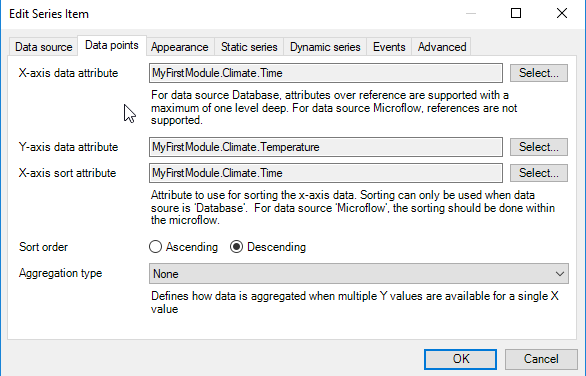
Configure the Appearance tab as shown below.



On the Chart properties tab click New to add a Series. Configure the Data source tab as shown below.

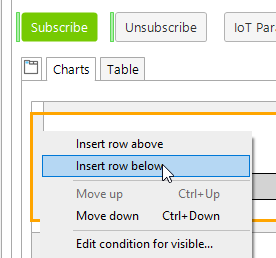


Configure the Data points tab as shown.



### Add Additional Charts

Add a row to the Layout grid by right-clicking the border and selecting Insert row below.



Do this twice and create charts for humidity and temperature.

