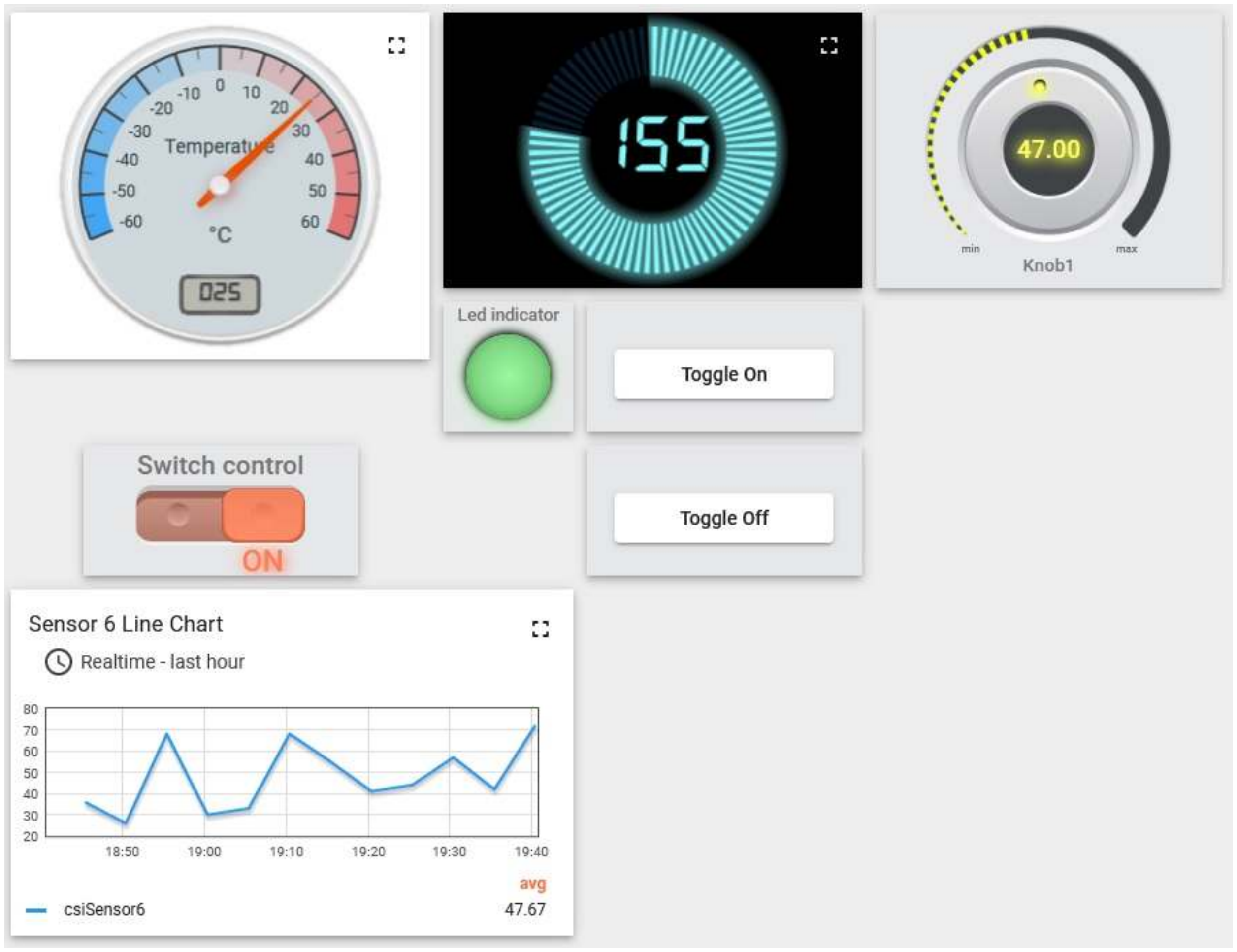
MQTT User Guide Addendum - Thingsboard.io

(Updated 7-Nov-2022)

Control Solutions Babel Buster IoT gateways now support Thingsboard.io MQTT services. Thingsboard provides a number of capabilities including interactive real time dashboards. Here is an example of a demo dashboard we recently built for test purposes.



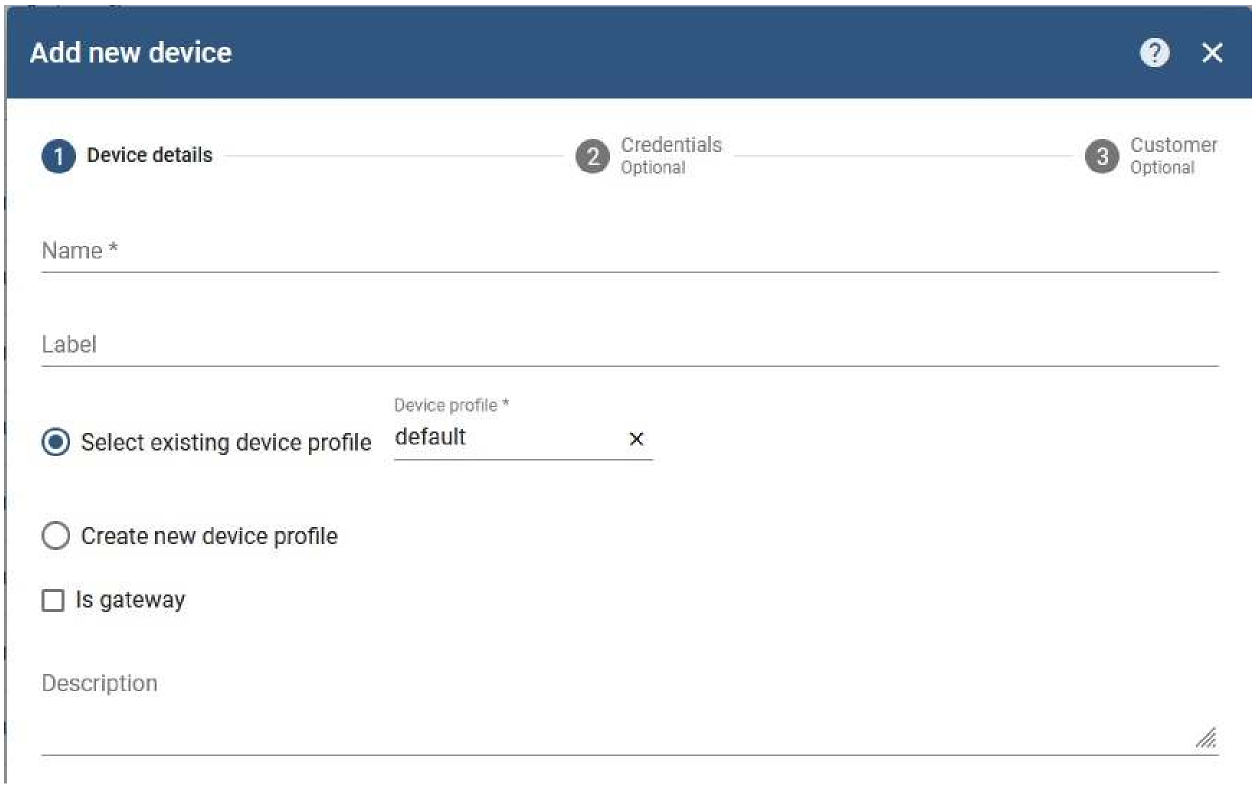
We refer to "MQTT device" throughout the discussion that follows because the discussion is generic to any of the Control Solutions products that have MQTT capability such as the Babel Buster IoT gateways like MQ-61 or MQ-73.

Start by signing up here: https://demo.thingsboard.io/signup

Once you have an account, you can log in here: https://demo.thingsboard.io/login

The home page contains a number of links to tutorials and documentation.

To add a new device, select the Devices page from the list on the left. Then click "+" in upper right corner and "Add new device".


Add new device

1 Device details — 2 Credentials Optional — 3 Customer Optional

Name *

Label

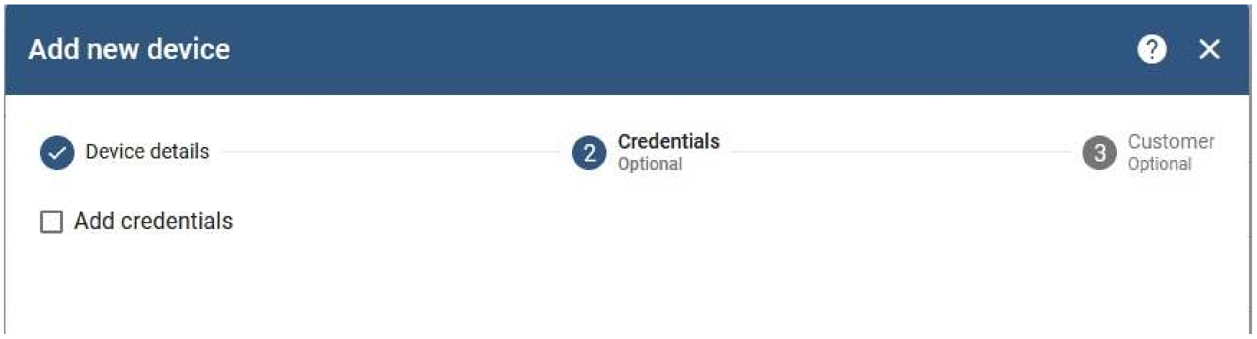
Device profile *: default

[✓ Select existing device profile] Create new device profile

☐ Is gateway

Description

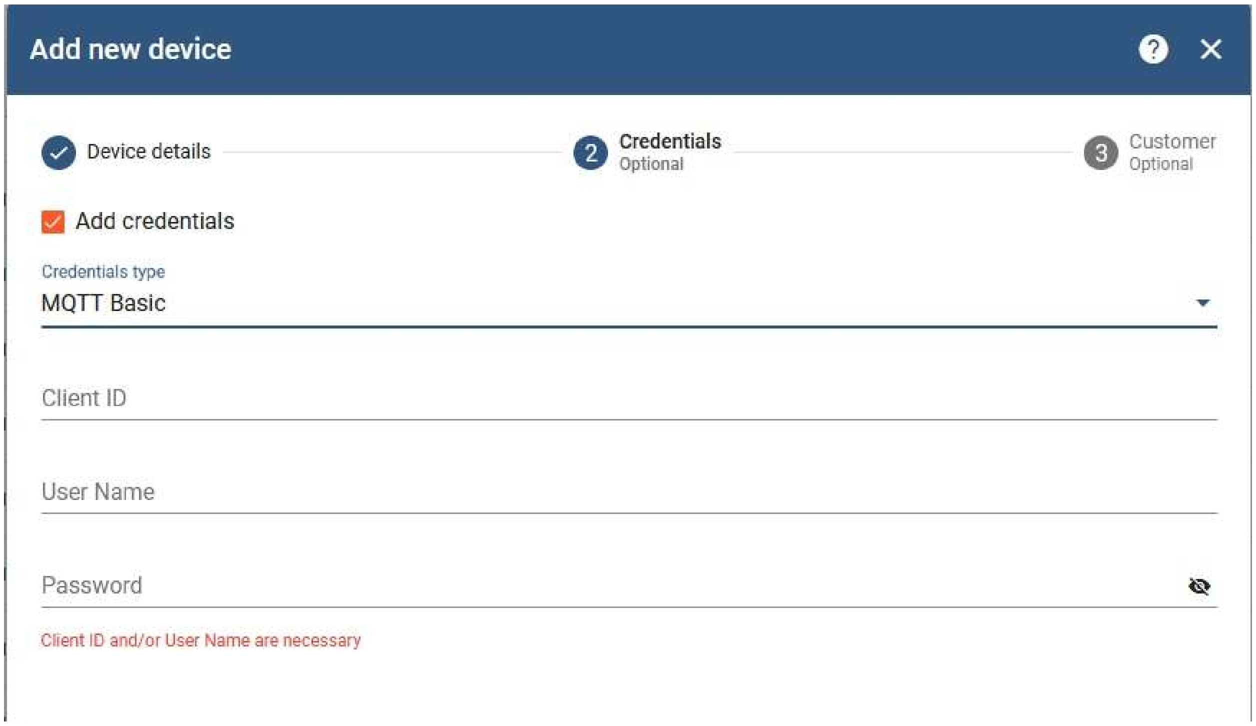
Give the device a name, and select the default profile. Click "Next: Credentials" in the lower right.


Add new device

✓ Device details — 2 Credentials Optional — 3 Customer Optional

☐ Add credentials

Select "Add credentials".



☑ Add credentials

Credentials type: MQTT Basic

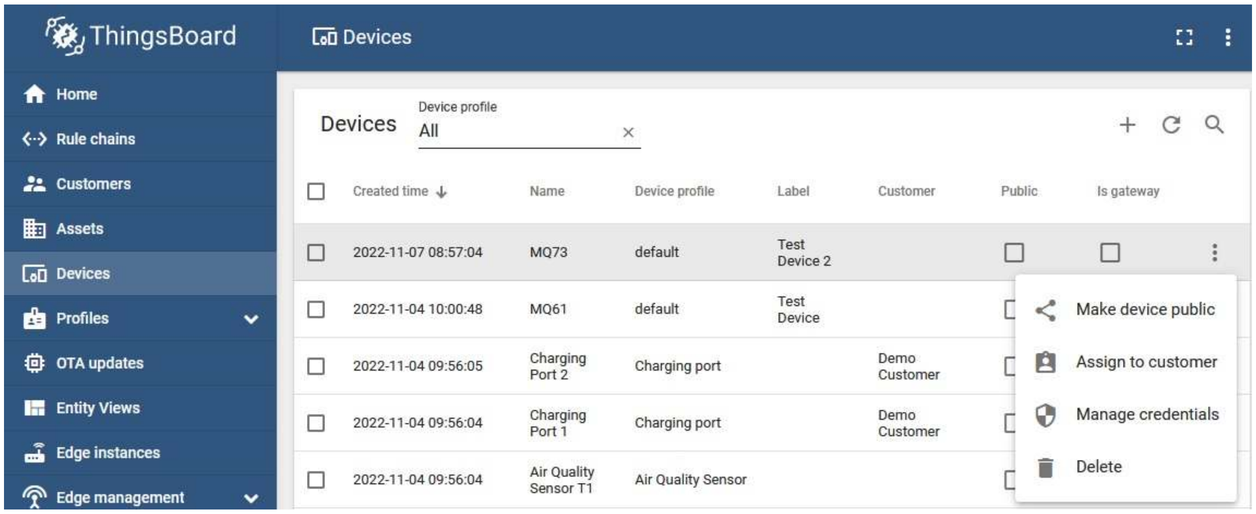
Client ID

User Name

Password

Client ID and/or User Name are necessary

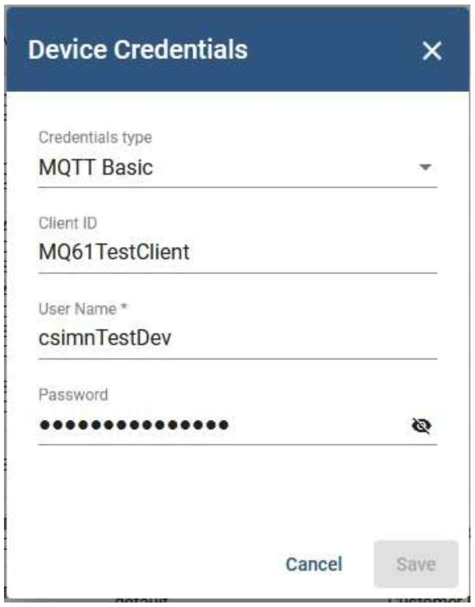
Then select "MQTT Basic" from Credentials type. Now provide a Client ID, User Name, and Password. If you clicked Add before setting credentials or you want to change them later (or just check them), simply click the Manage Credentials icon on that device's line on the list of devices on the Devices page. If the Devices page width is reduced, the icons turn into a drop-down menu.



Devices — Device profile: All

Created time	Name	Device profile	Label	Customer	Public	Is gateway
2022-11-07 08:57:04	MQ73	default	Test Device 2		☐	☐
2022-11-04 10:00:48	MQ61	default	Test Device		☐	
2022-11-04 09:56:05	Charging Port 2	Charging port		Demo Customer	☐	
2022-11-04 09:56:04	Charging Port 1	Charging port		Demo Customer	☐	
2022-11-04 09:56:04	Air Quality Sensor T1	Air Quality Sensor			☐	

You can view or retrieve the device credentials at any time.



Device Credentials

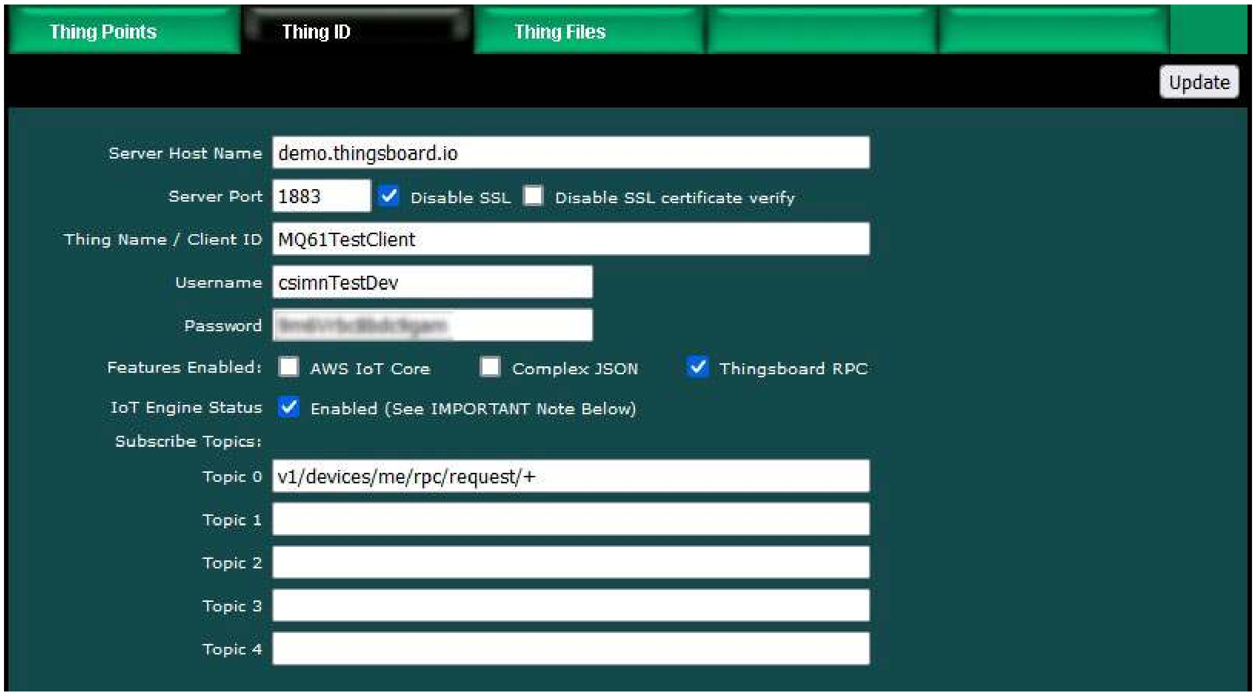
Credentials type
MQTT Basic

Client ID
MQ61TestClient

User Name *
csimnTestDev

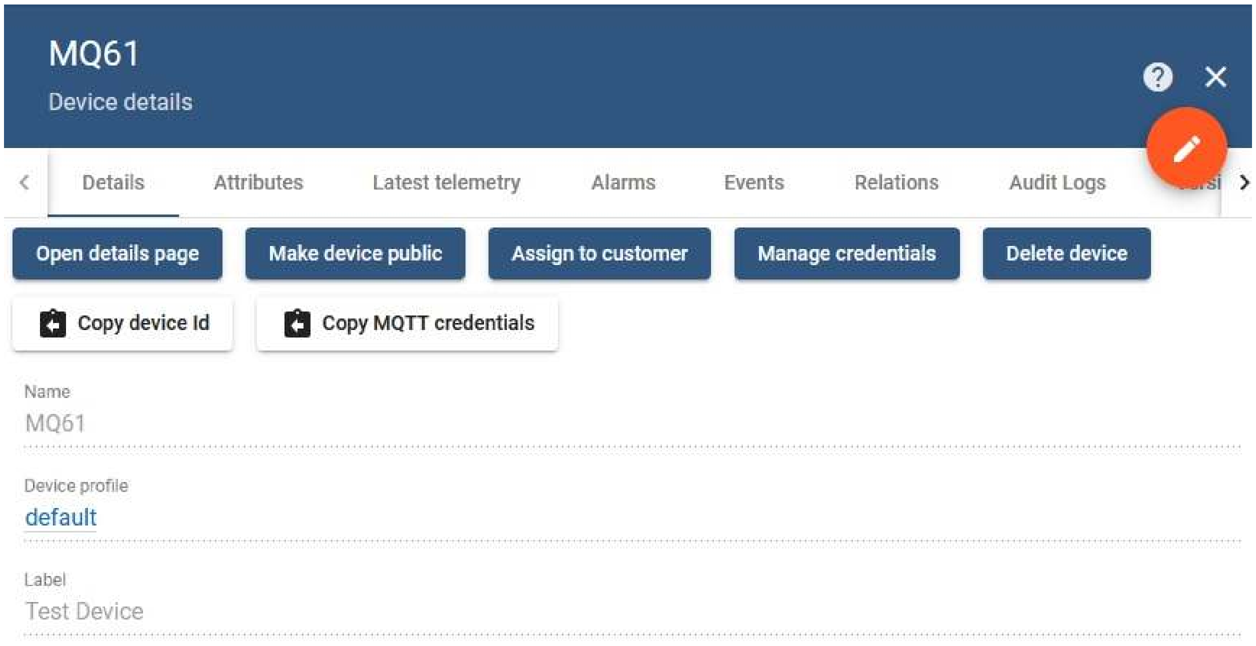
Password
•••••••••••••••

Once you have created the device on the Thingsboard server, set up your MQTT device (e.g. MQ-61) as illustrated here. *Be sure that the Features Enabled line shows AWS IoT Core NOT enabled, Complex JSON NOT enabled, and Thingsboard RPC enabled.*



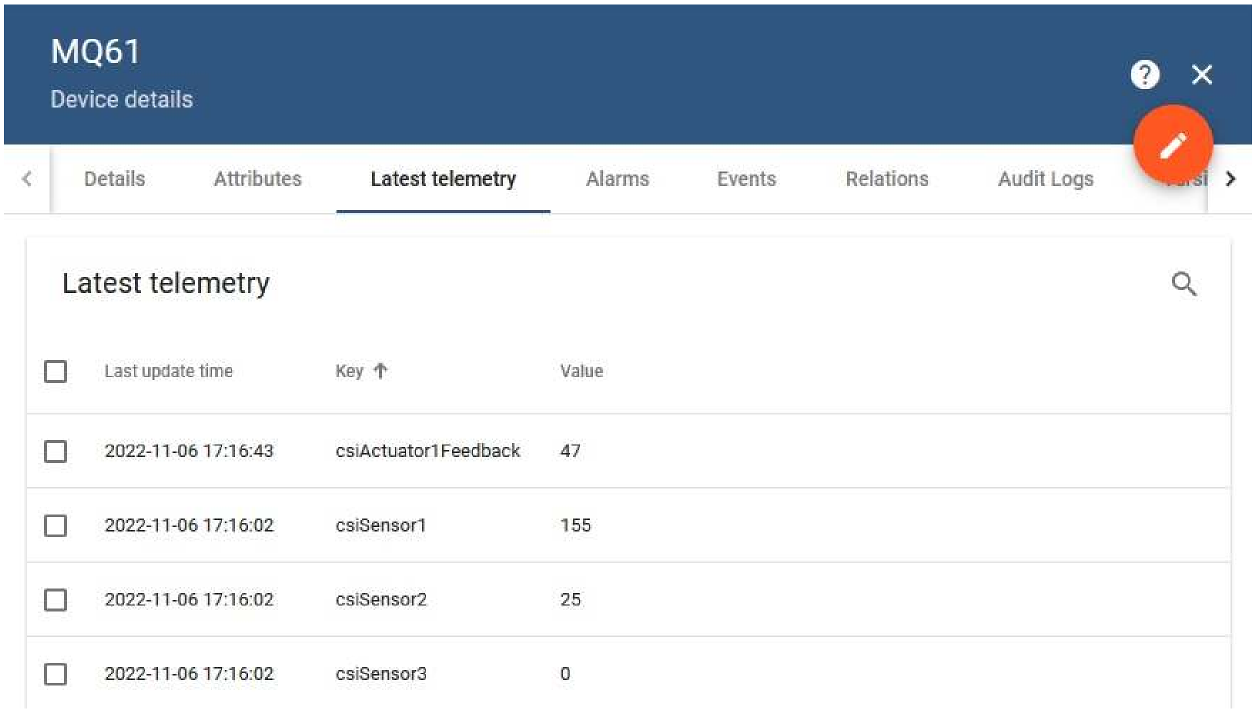
Server Host Name	demo.thingsboard.io
Server Port	1883 ✓ Disable SSL ☐ Disable SSL certificate verify
Thing Name / Client ID	MQ61TestClient
Username	csimnTestDev
Password	[redacted]
Features Enabled:	☐ AWS IoT Core ☐ Complex JSON ✓ Thingsboard RPC
IoT Engine Status	✓ Enabled (See IMPORTANT Note Below)
Subscribe Topics:	
Topic 0	v1/devices/me/rpc/request/+
Topic 1	
Topic 2	
Topic 3	
Topic 4	

Once data starts flowing to the device, you can look at the most recently received data by clicking on the device on the Devices page. You will then see a popup dialog that looks like this.



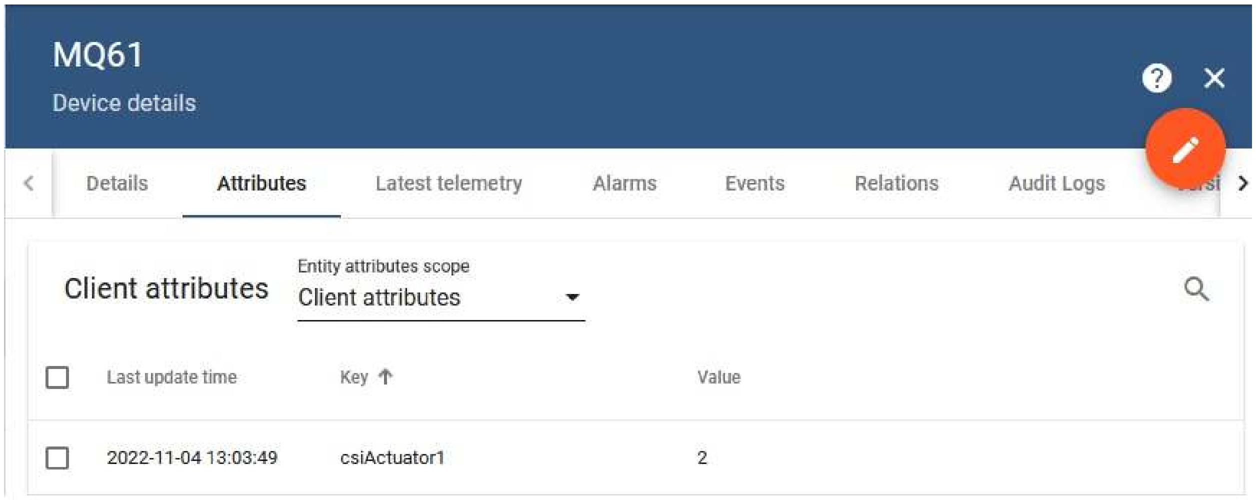
Name	MQ61
Device profile	default
Label	Test Device

Data can be sent to Thingsboard as either telemetry or attributes. Click on the Latest telemetry tab to see the most recent data points sent as telemetry.



Last update time	Key	Value
2022-11-06 17:16:43	csiActuator1Feedback	47
2022-11-06 17:16:02	csiSensor1	155
2022-11-06 17:16:02	csiSensor2	25
2022-11-06 17:16:02	csiSensor3	0

Click on the Attributes tab to see data most recently sent as attributes.

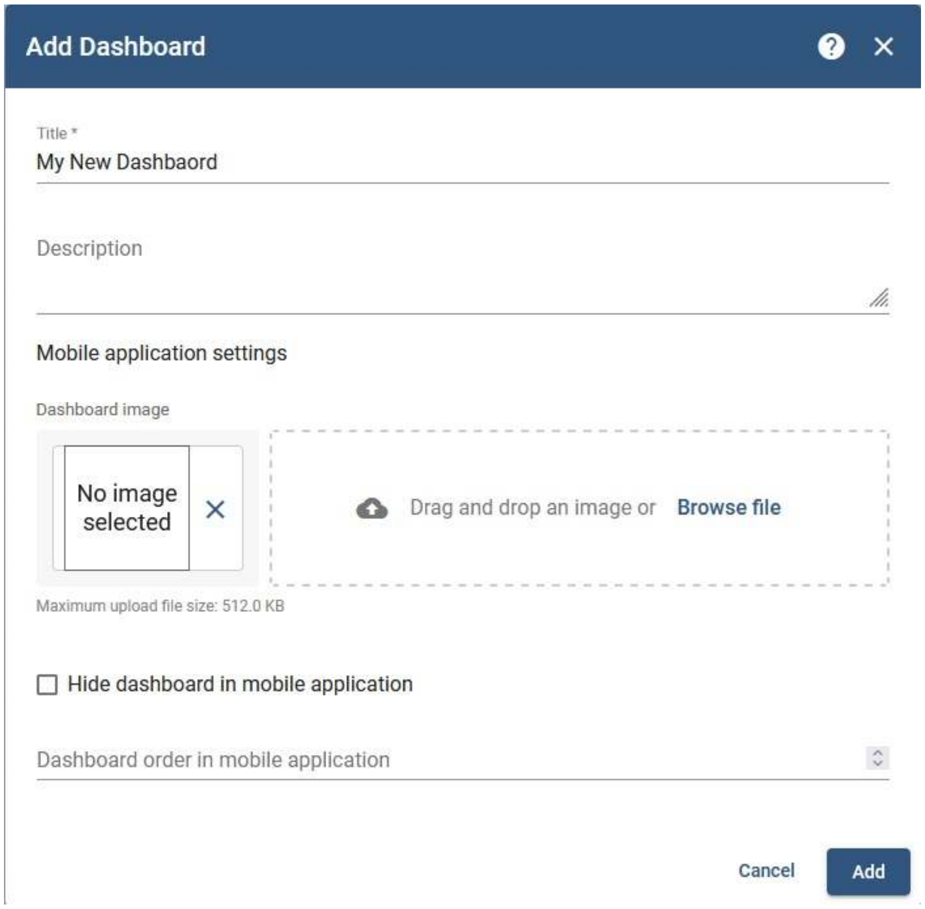


Client attributes

Entity attributes scope: Client attributes

Last update time	Key ↑	Value
2022-11-04 13:03:49	csiActuator1	2

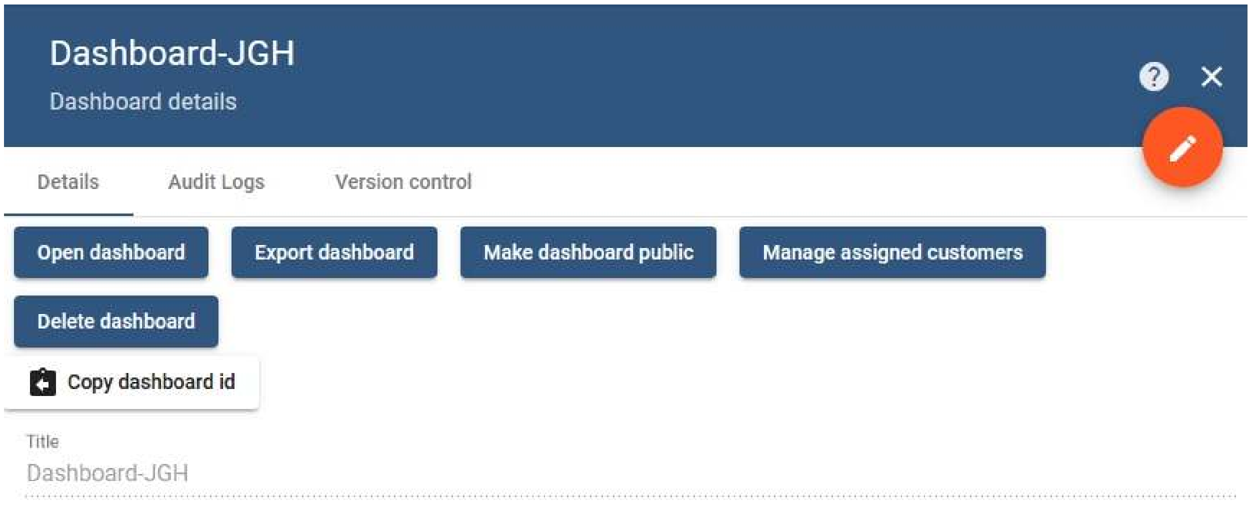
To create a new dashboard, go to the Dashboards page and click the "+" icon in the upper right corner. Give the dashboard a name. You don't need to do anything else at this point other than click Add.


Add Dashboard
Description
Mobile application settings
Dashboard image
No image selected
Drag and drop an image or Browse file
Maximum upload file size: 512.0 KB
Dashboard order in mobile application
Cancel
Add

Title *: My New Dashbaord

☐ Hide dashboard in mobile application

When you click on your dashboard name on the Dashboard list, you will get a popup dialog that looks like this. Click "Open dashboard".

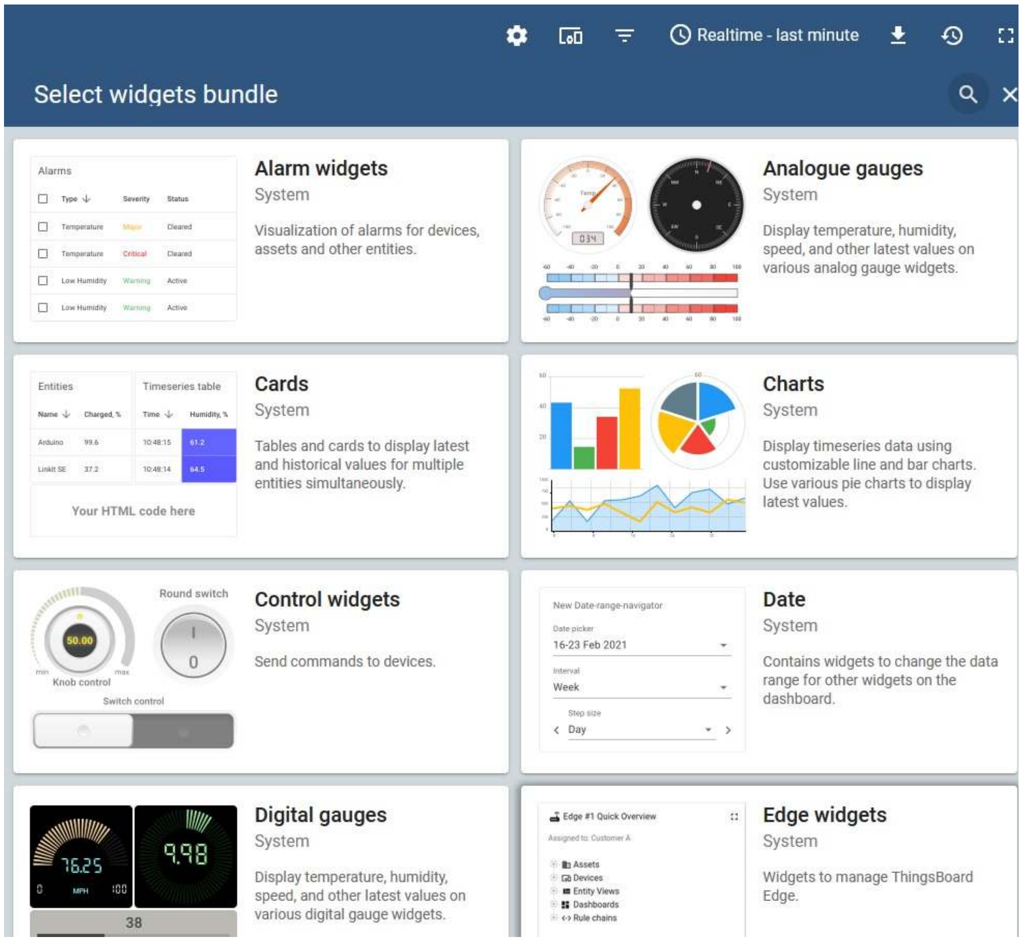

Dashboard-JGH

Dashboard details

Title

Dashboard-JGH

The dashboard will open, showing any dashboard widgets you previously placed. Click the pencil icon in the lower right to modify or add widgets. Click the "+" button lower right, then "Create new widget" icon, to bring up the list of widgets to pick from. Here is a screen shot showing part of the list. Click through to find the widget you wish to place and select it.



Select widgets bundle

Alarm widgets
System
Visualization of alarms for devices, assets and other entities.

Analogue gauges
System
Display temperature, humidity, speed, and other latest values on various analog gauge widgets.

Cards
System
Tables and cards to display latest and historical values for multiple entities simultaneously.

Charts
System
Display timeseries data using customizable line and bar charts. Use various pie charts to display latest values.

Control widgets
System
Send commands to devices.

Date
System
Contains widgets to change the data range for other widgets on the dashboard.

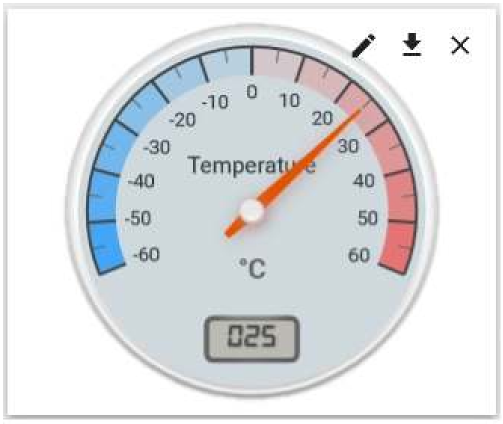
Digital gauges
System
Display temperature, humidity, speed, and other latest values on various digital gauge widgets.

Edge widgets
System
Widgets to manage ThingsBoard Edge.

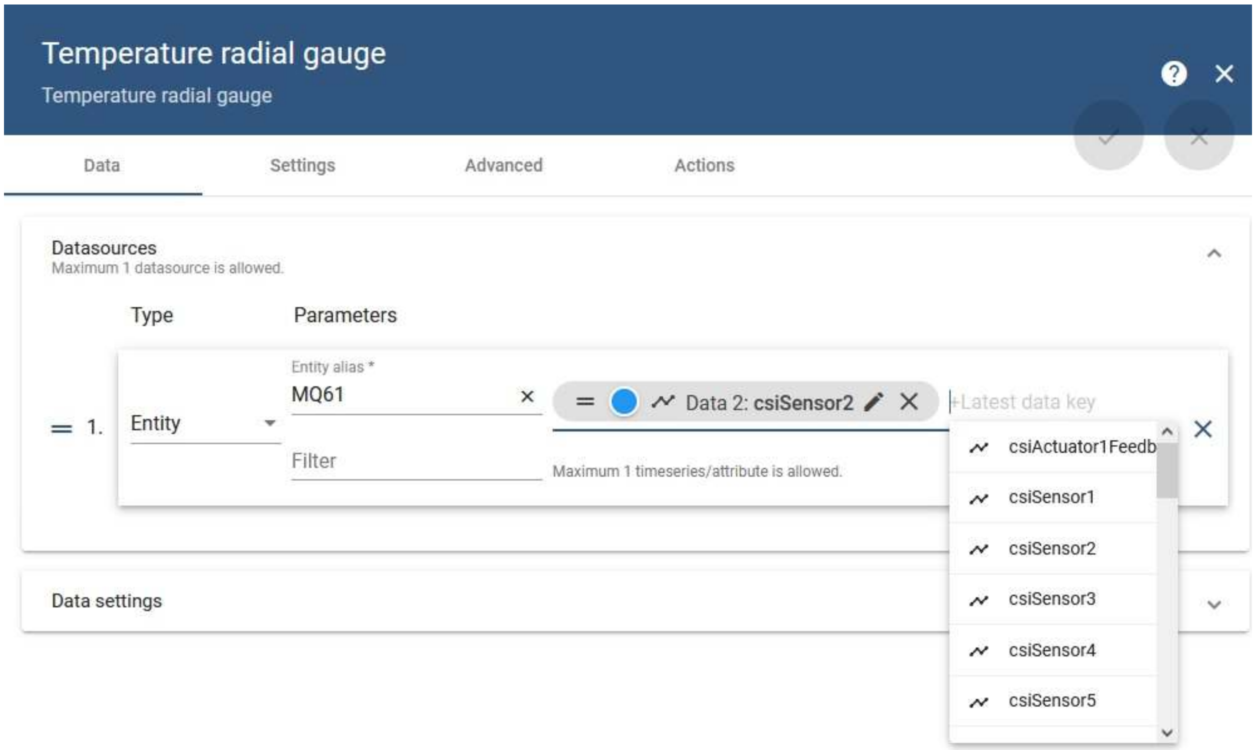
When the dashboard is open for editing, all of the previously placed widgets will have icons in the upper right corner. You can remove a widget by clicking the X. To edit the widget, click the pencil icon.

Whether adding a new widget or editing an existing widget, the templates will look the same for a given widget, but the templates are somewhat different for each widget type.

Before adding gauges to display data, it is important to configure your MQTT gateway to publish at least one set of data. This allows Thingsboard to know what your data names are.



To configure a gauge to display data from your MQTT device, select data source Entity, select the device name you gave your MQTT device on the Devices page, and then select a data point from tat device. Notice that holding the mouse over the data field area displays a list of all known telemetry points. In this case, we have selected csiSensor2 for display on this gauge.



Temperature radial gauge

Temperature radial gauge

Data | Settings | Advanced | Actions

Datasources
Maximum 1 datasource is allowed.

Type | Parameters

1. Entity | Entity alias * MQ61 | Data 2: csiSensor2 | +Latest data key

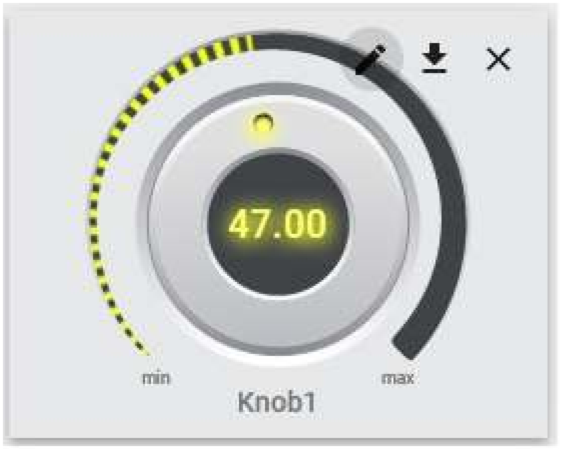
Filter

Maximum 1 timeseries/attribute is allowed.

csiActuator1Feedb
csiSensor1
csiSensor2
csiSensor3
csiSensor4
csiSensor5

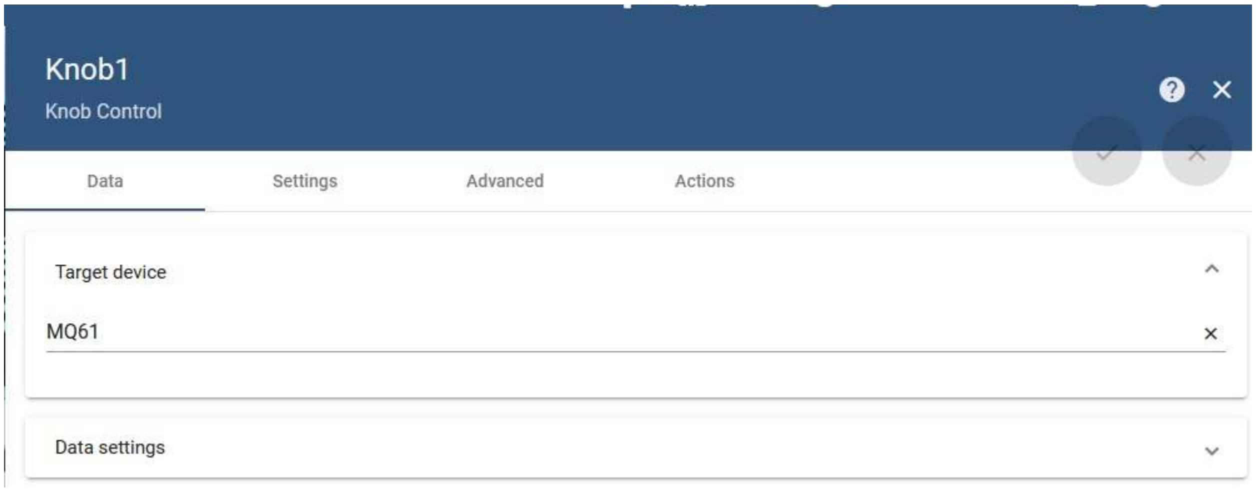
Data settings

To use the gauge in its default form, you do not need to make any additional changes on the Settings or Advanced page for the gauge. Try the gauge out as is and then come back and tweak the appearance later.



The knob widget looks somewhat like a gauge, but is a dial that you can turn on your browser page and its setting will be sent back to the MQTT device.

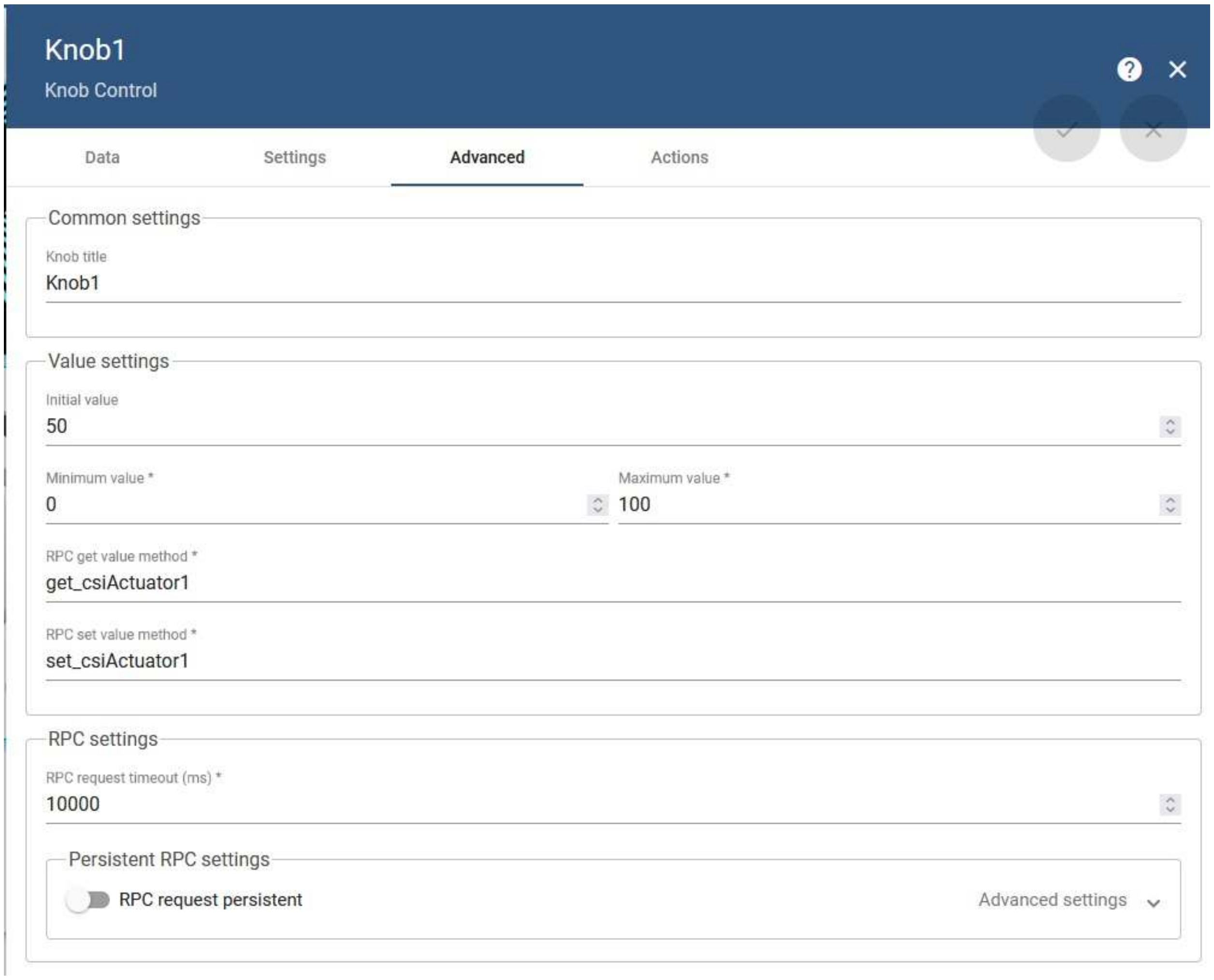
When you add or edit a knob widget, the first tab simply selects the target device. Enter the device name from the Devices page. You do not need to do anything with Data settings on this tab.



Target device

MQ61

The Settings tab for the knob sets visual appearance. Leave the default settings initially. You do not need to do anything on the Actions tab. But there are a couple of very important things you need to do on the Advanced tab. Enter "get_XXX" for RPC get value method and "set_XXX" for RPC set value method where XXX is the attribute name assigned in your MQTT device. Here is an example.


Knob1

Knob Control

Data | Settings | Advanced | Actions

Common settings

Knob title: Knob1

Value settings

Initial value: 50

Minimum value *: 0

Maximum value *: 100

RPC get value method *: get_csiActuator1

RPC set value method *: set_csiActuator1

RPC settings

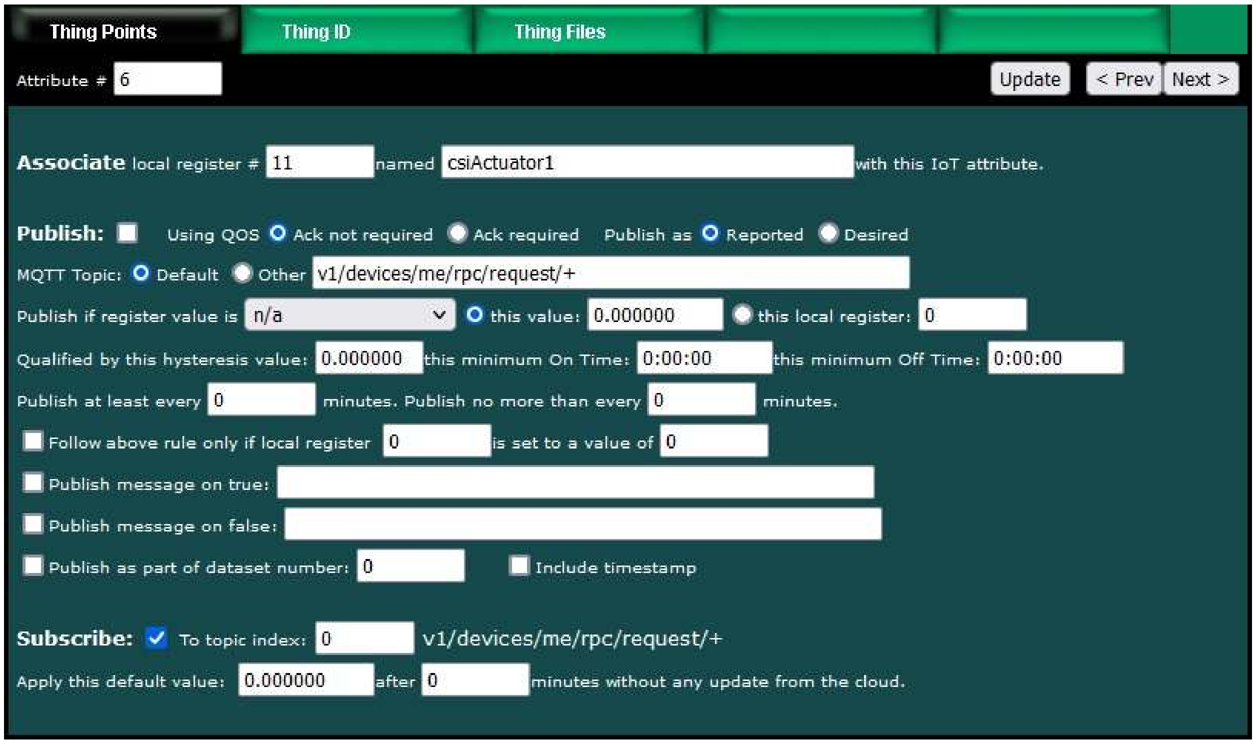
RPC request timeout (ms) *: 10000

Persistent RPC settings

RPC request persistent

Advanced settings

Here is the configuration in the MQTT device (e.g. MQ-61) for the point that will end up receiving changes to the knob configured above. Not that this point subscribes, and is not published. Most importantly, note the topic. In order to receive data from Thingsboard.io, the topic must be "v1/devices/me/rpc/request/+".



Attribute # 6 | Update | < Prev | Next >

Associate local register # 11 named csiActuator1 with this IoT attribute.

Publish: ☐ Using QOS [✓ Ack not required] Ack required Publish as [✓ Reported] Desired

MQTT Topic: [✓ Default] Other v1/devices/me/rpc/request/+

Publish if register value is n/a [✓ this value:] 0.000000 this local register: 0

Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Publish at least every 0 minutes. Publish no more than every 0 minutes.

☐ Follow above rule only if local register 0 is set to a value of 0

☐ Publish message on true:

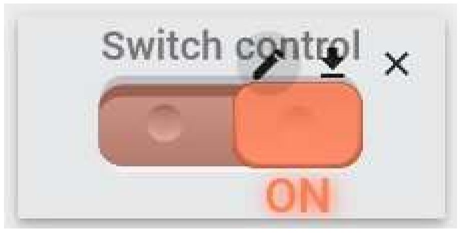
☐ Publish message on false:

☐ Publish as part of dataset number: 0 ☐ Include timestamp

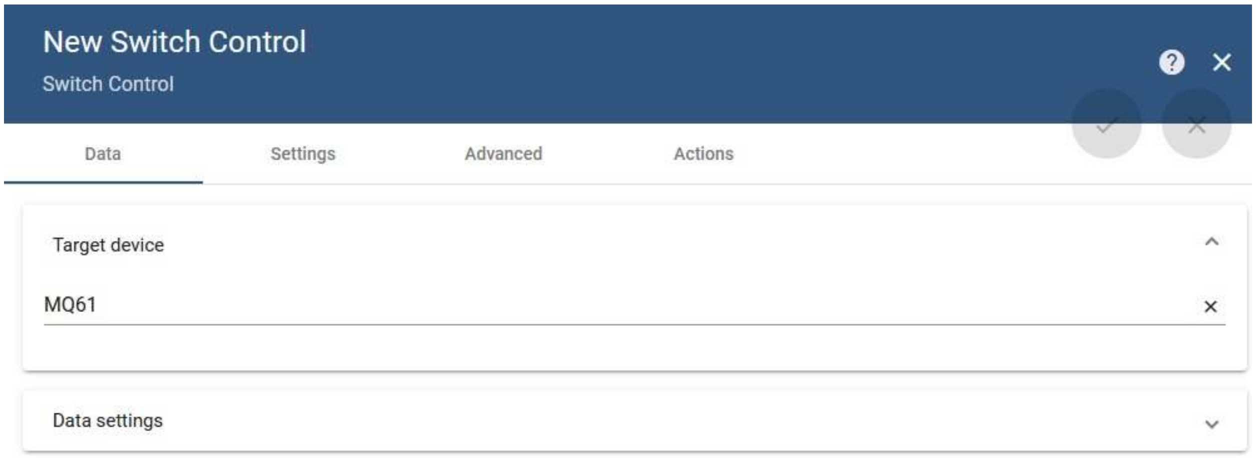
Subscribe: ✓ To topic index: 0 v1/devices/me/rpc/request/+

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Configuring a switch control is similar to setting up a knob.



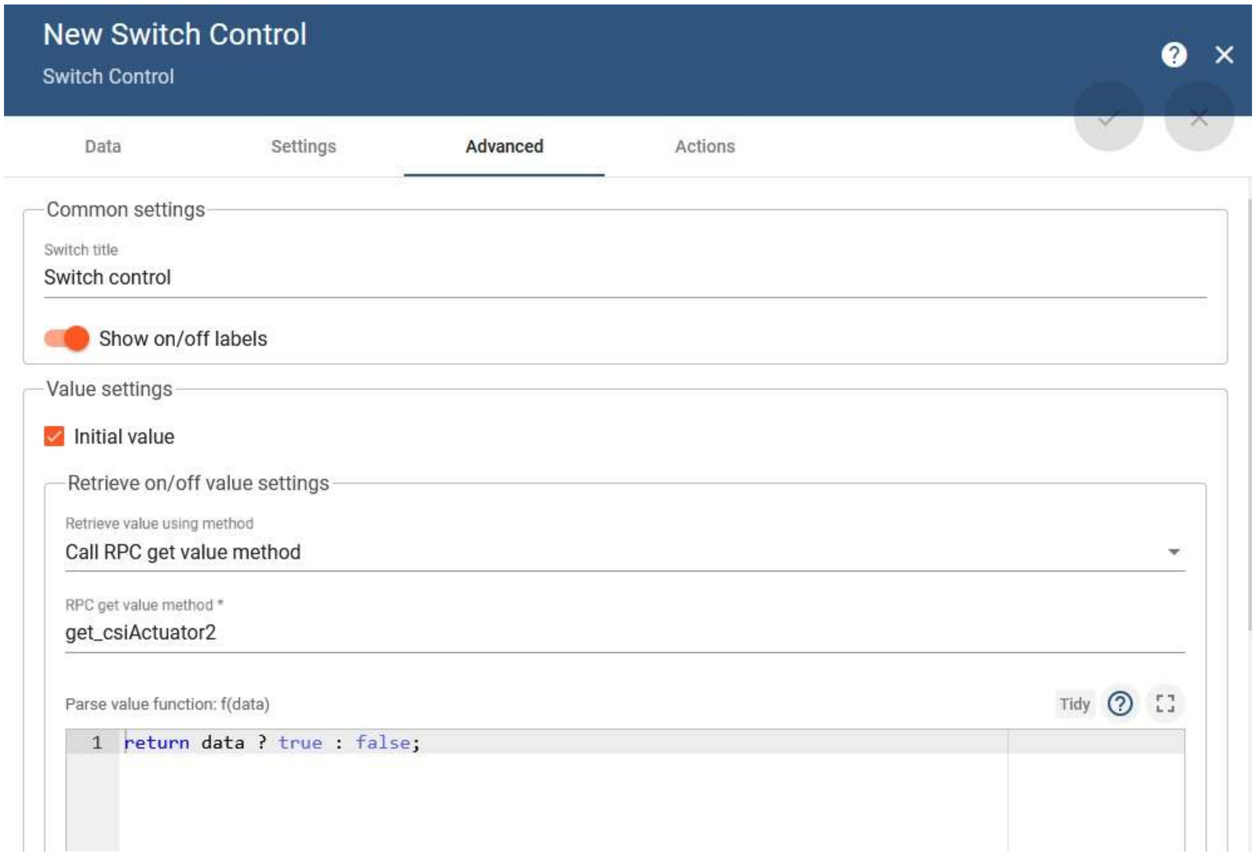
Set the target device on the Data tab, and don't do anything with Data Settings for now. Again the Settings tab is all about visual settings and you don't need to do anything here to get the switch functioning. You don't need to do anything on the Actions tab either.



Target device

MQ61

Once again, the Advanced tab has a couple of important things to take care of. The Initial Value will default to not being selected. Check this box so that the switch will reflect the current state in the MQTT device upon opening the dashboard. The RPC get value method should be "get_XXX" where XXX is the attribute or data point name in the MQTT device (e.g. MQ-61).



Common settings

Switch title: Switch control

Show on/off labels: on

Value settings

Initial value: ✓

Retrieve on/off value settings

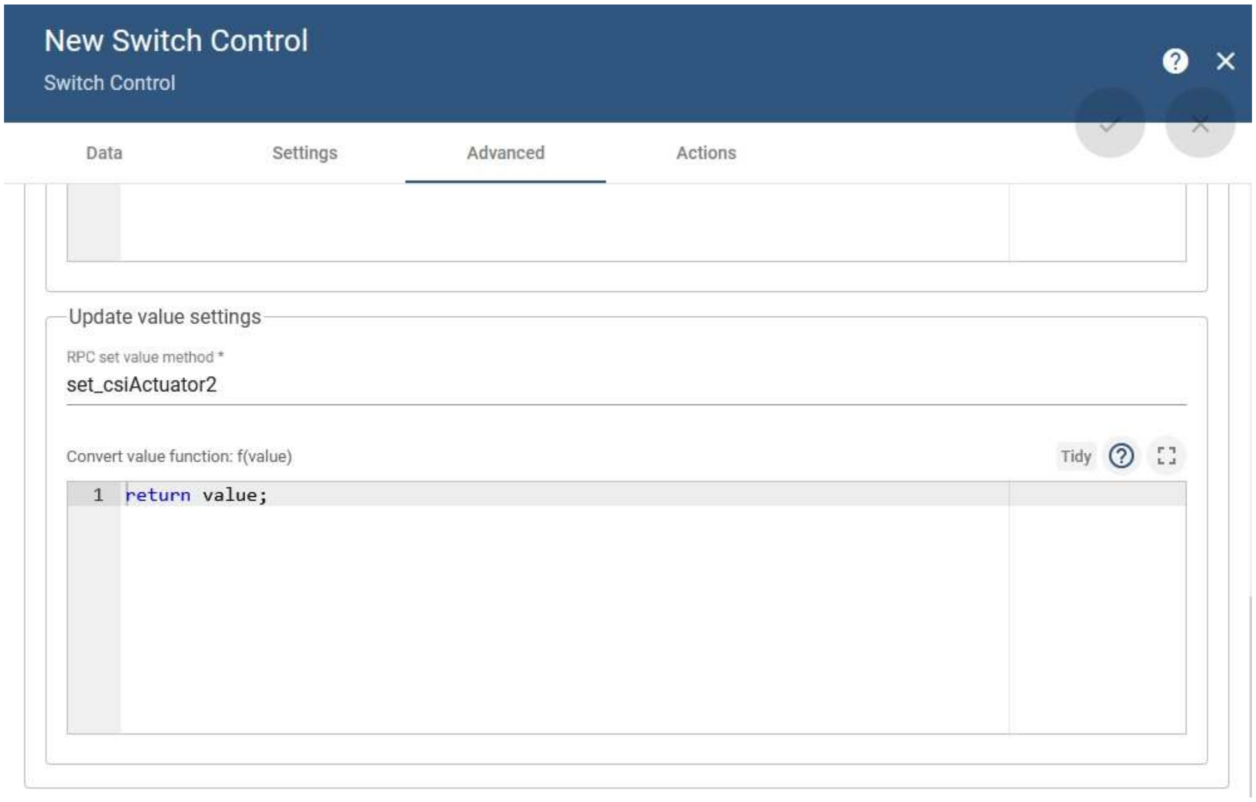
Retrieve value using method: Call RPC get value method

RPC get value method *: get_csiActuator2

Parse value function: f(data)

```
return data ? true : false;
```

Scroll down in the Advanced tab and set the RPC set value method to "set_XXX" where XXX is the attribute name.


New Switch Control

Switch Control

Update value settings

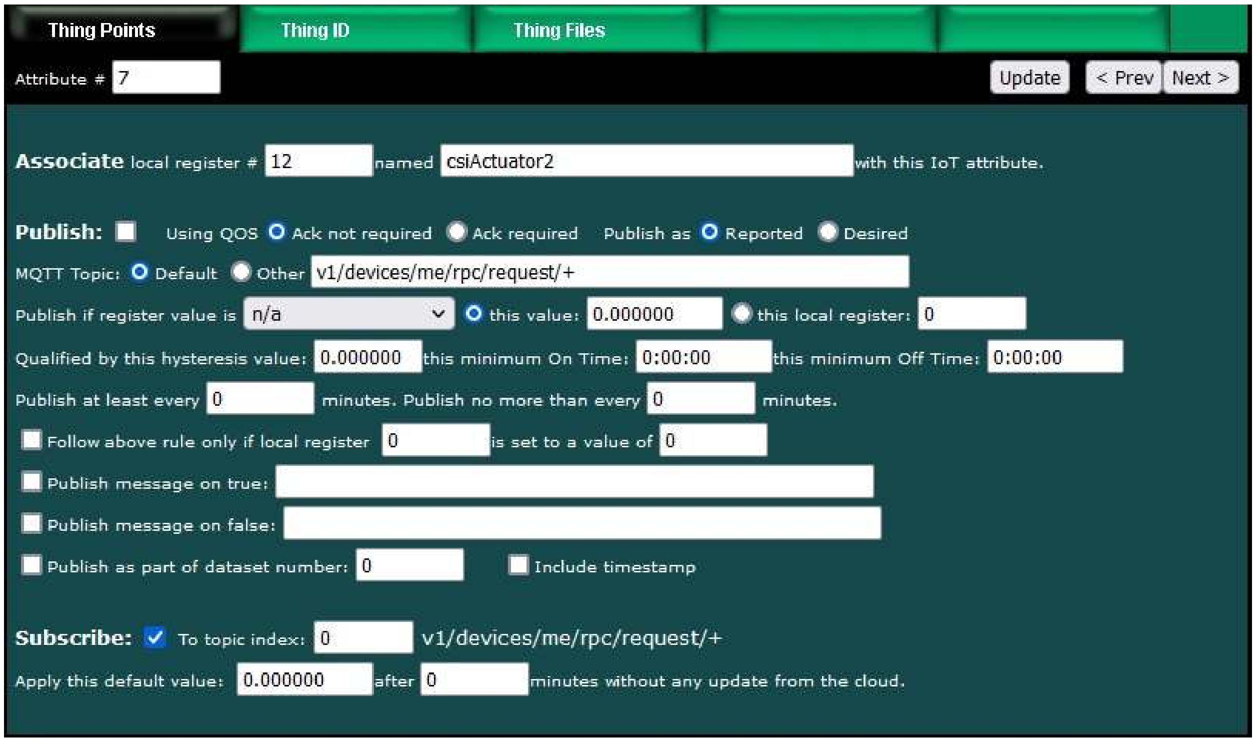
RPC set value method *

set_csiActuator2

Convert value function: f(value)

```
return value;
```

Here is the attribute or data point configuration in the MQTT device that corresponds with the above switch. Note again that the point must subscribe to "v1/devices/me/rpc/request/+".



Attribute # 7

Associate local register # 12 named csiActuator2 with this IoT attribute.

Publish: ☐ Using QOS [✓ Ack not required] Ack required Publish as [✓ Reported] Desired

MQTT Topic: [✓ Default] Other v1/devices/me/rpc/request/+

Publish if register value is n/a [✓ this value: 0.000000] this local register: 0

Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Publish at least every 0 minutes. Publish no more than every 0 minutes.

☐ Follow above rule only if local register 0 is set to a value of 0

☐ Publish message on true:

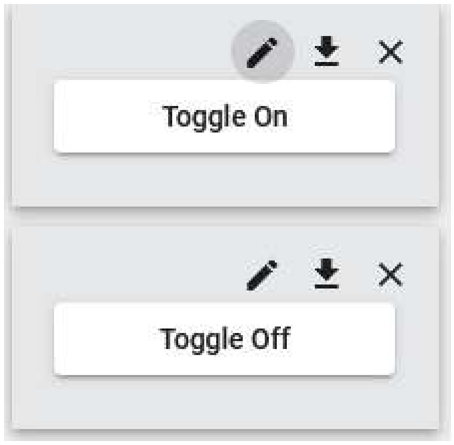
☐ Publish message on false:

☐ Publish as part of dataset number: 0 ☐ Include timestamp

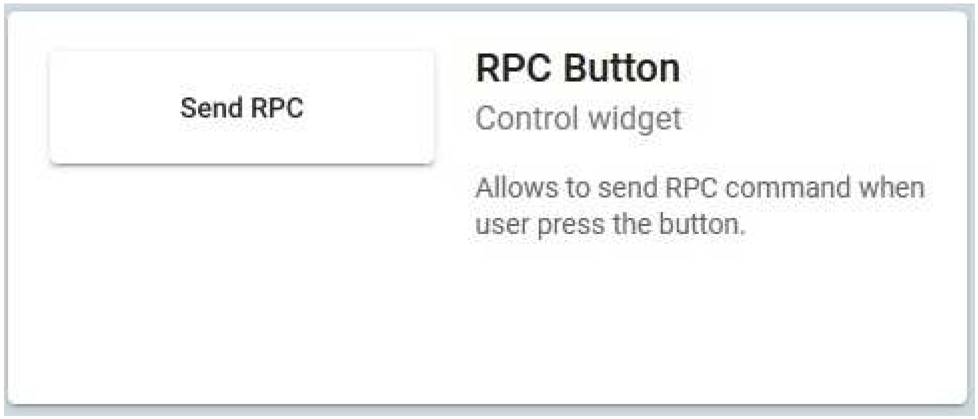
Subscribe: ☑ To topic index: 0 v1/devices/me/rpc/request/+

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

The "Toggle On" and "Toggle Off" buttons in our example are RPC buttons.



When adding a new RPC button, this is the icon to select from the Widget bundle.

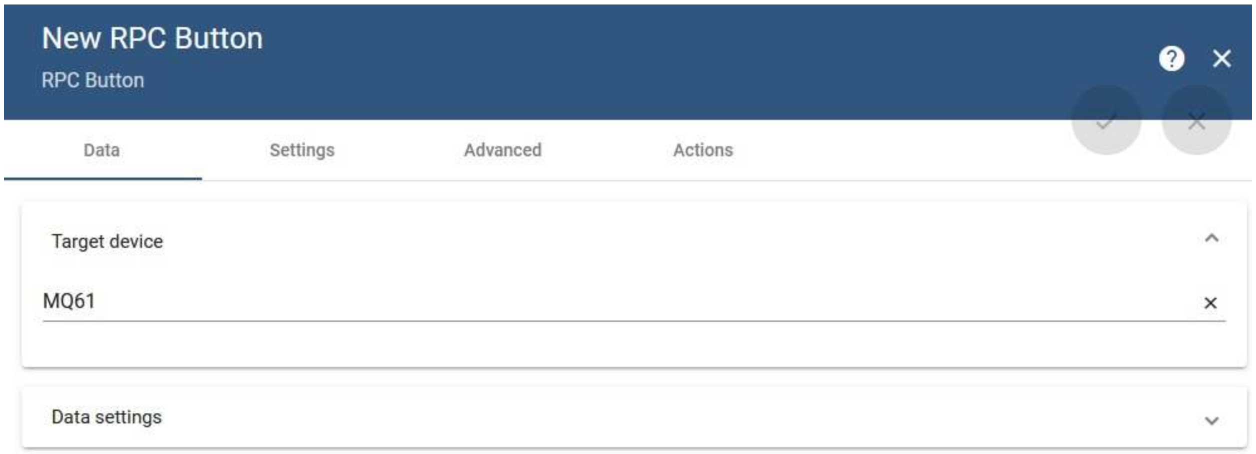


RPC Button

Control widget

Allows to send RPC command when user press the button.

Assign the target device. The Settings tab is all visual settings that don't need changing initially. You don't need to do anything on the Actions tab.



New RPC Button

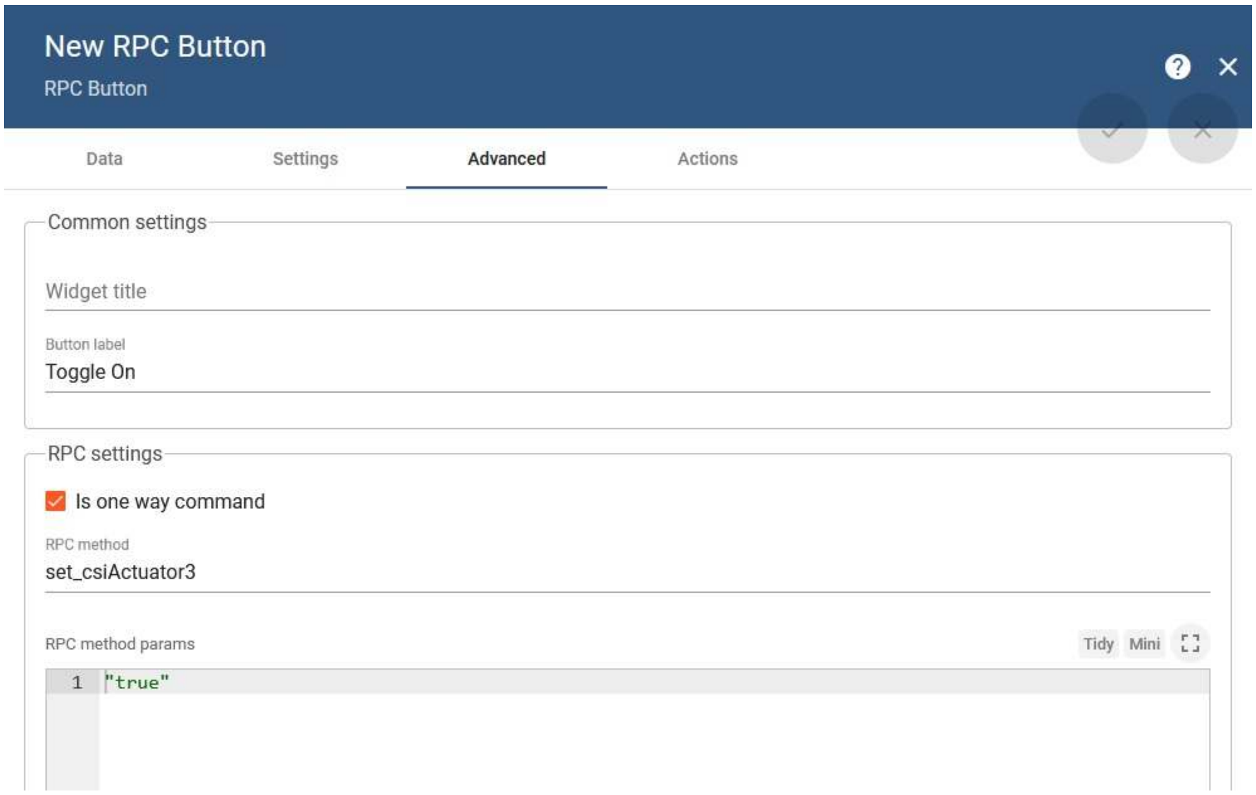
RPC Button

Target device

MQ61

Data settings

Here is what you need to do on the Advanced tab. Provide a button label. Select "Is one way command" and enter "set_XXX" for RPC method where XXX is the attribute name or data point name in the MQTT device. Enter the RPC method params. For the Toggle On button, enter "true" as illustrated here.


New RPC Button

RPC Button

Data | Settings | Advanced | Actions

Common settings

Widget title

Button label
Toggle On

RPC settings

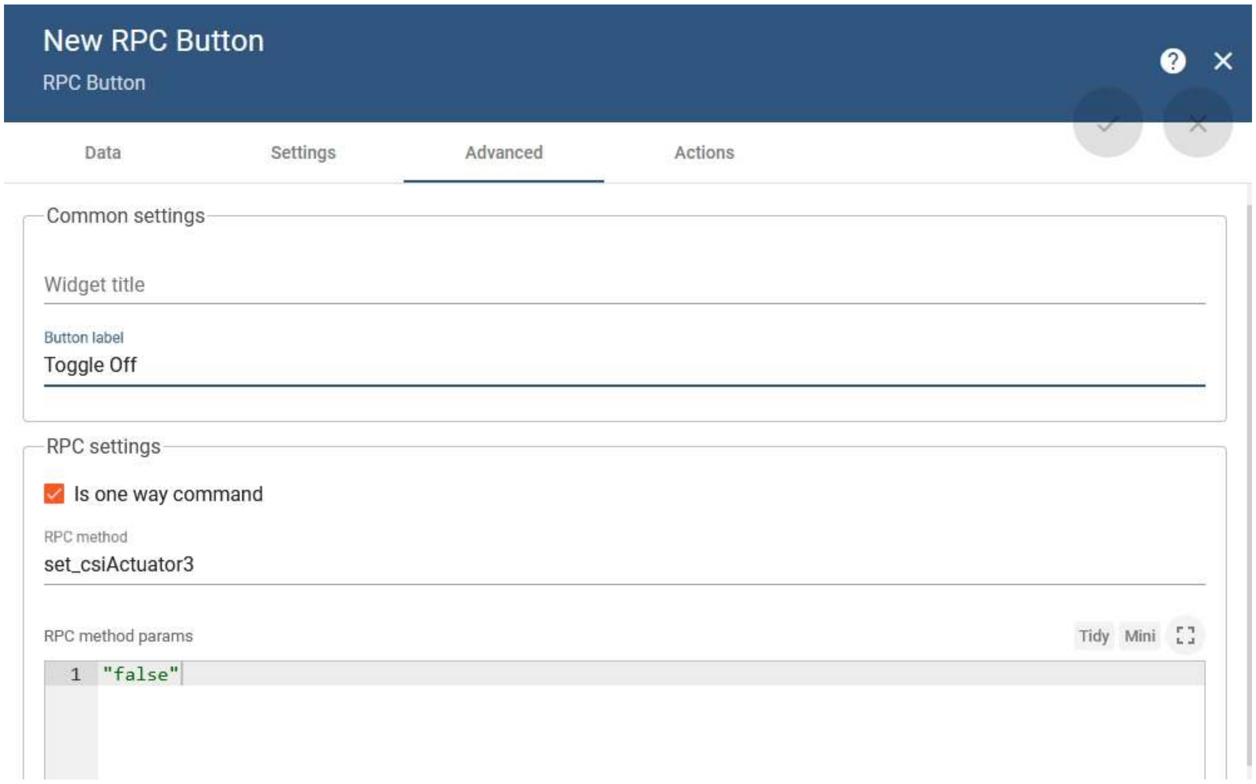
☑ Is one way command

RPC method
set_csiActuator3

RPC method params

```
"true"
```

The Toggle Off button is configured the same way, except its RPC method params is "false".


New RPC Button

RPC Button

Data | Settings | Advanced | Actions

Common settings

Widget title

Button label
Toggle Off

RPC settings

☑ Is one way command

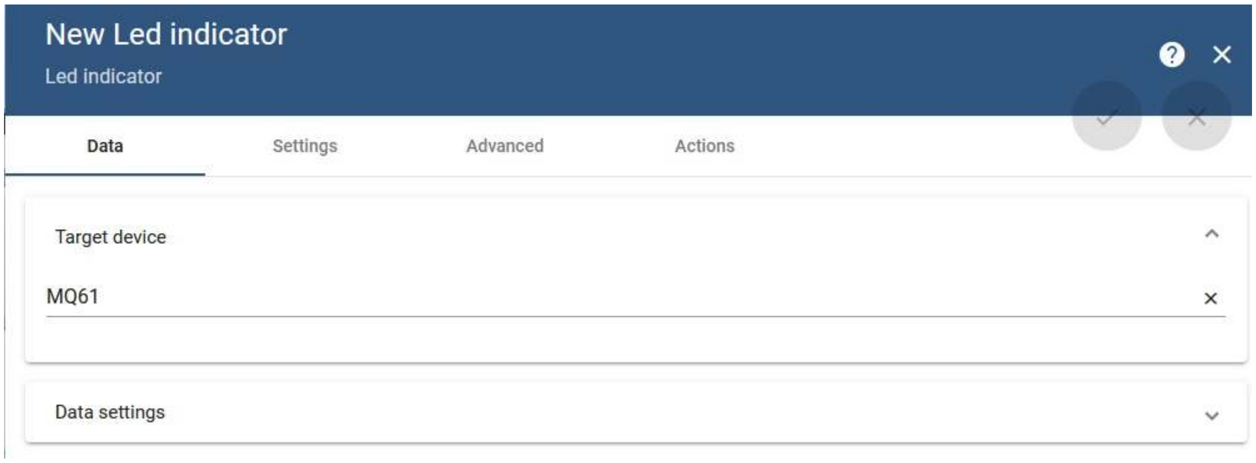
RPC method
set_csiActuator3

RPC method params

```
"false"
```

We have chosen to make the Toggle On and Toggle Off buttons turn the LED widget on and off.

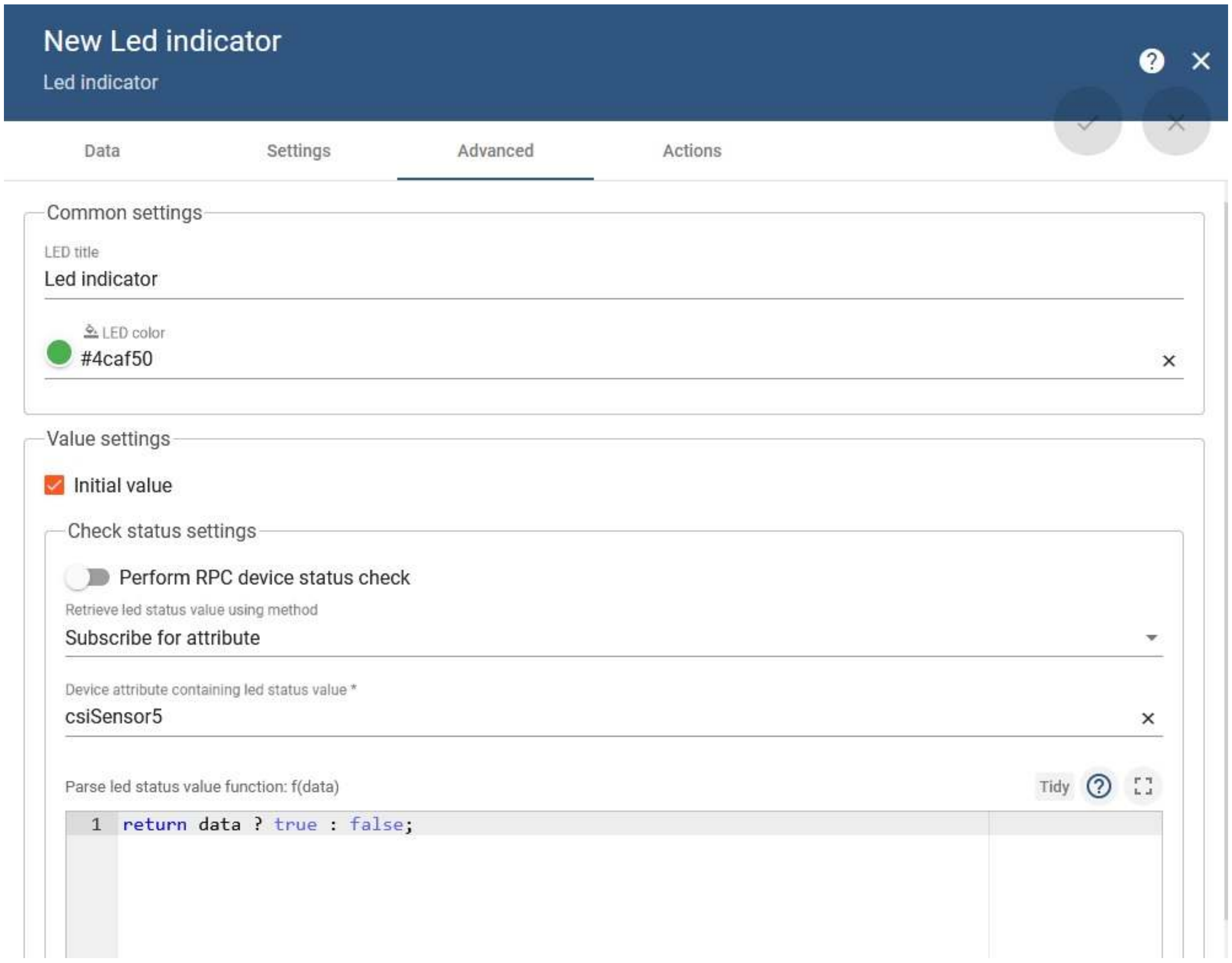
To configure the LED, once again start by assigning the target device. Settings are visual, you can come back to those later. You don't need to do anything on the Actions tab.



Target device

MQ61

On the Advanced tab for the LED, check initial Value, select "Subscribe for attribute" and provide the attribute name.

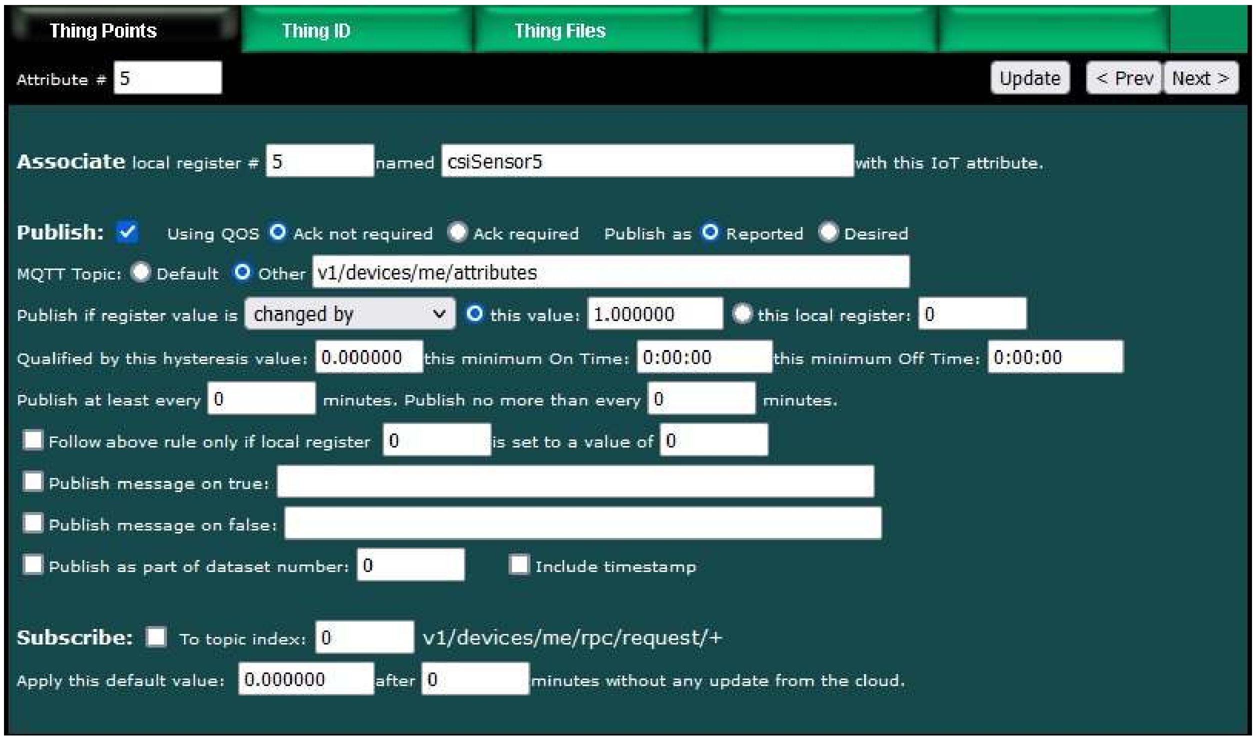

New Led indicator
Led indicator
Data Settings Advanced Actions

Common settings
LED title
Led indicator
LED color
#4caf50

Value settings
[✓] Initial value
Check status settings
Perform RPC device status check
Retrieve led status value using method
Subscribe for attribute
Device attribute containing led status value *
csiSensor5
Parse led status value function: f(data)

```
return data ? true : false;
```

Here is the attribute configuration in the MQTT device (e.g. MQ-61) for the LED. Notice that in this case we publish to attributes rather than telemetry.



Attribute # 5 Update < Prev Next >

Associate local register # 5 named csiSensor5 with this IoT attribute.

Publish: ✓ Using QOS [✓ Ack not required] Ack required Publish as [✓ Reported] Desired

MQTT Topic: Default [✓ Other] v1/devices/me/attributes

Publish if register value is changed by [✓ this value:] 1.000000 this local register: 0

Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Publish at least every 0 minutes. Publish no more than every 0 minutes.

☐ Follow above rule only if local register 0 is set to a value of 0

☐ Publish message on true:

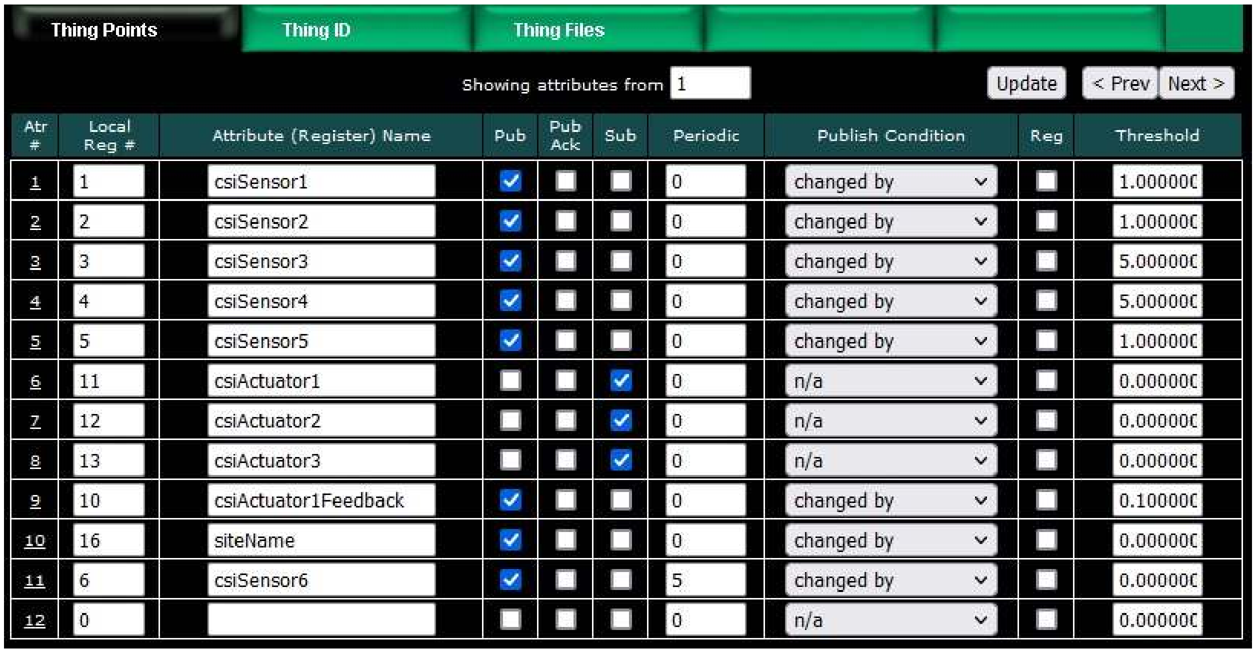
☐ Publish message on false:

☐ Publish as part of dataset number: 0 ☐ Include timestamp

Subscribe: ☐ To topic index: 0 v1/devices/me/rpc/request/+

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

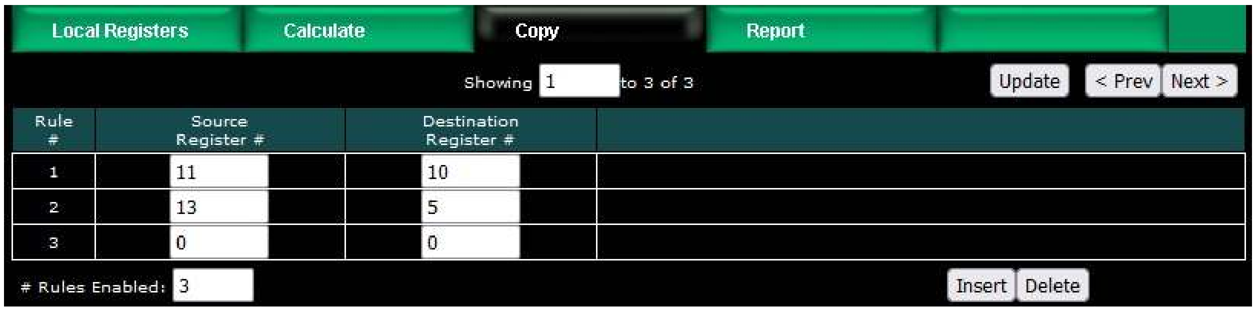
We need to do one more thing on the MQTT device side to connect the buttons to the LED. Our point list looks like this. We want to connect csiActuator3 (the buttons) to csiSensor5 (the LED).



Showing attributes from 1 Update < Prev Next >

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	✓	☐	☐	0	changed by	☐	1.00000
2	2	csiSensor2	✓	☐	☐	0	changed by	☐	1.00000
3	3	csiSensor3	✓	☐	☐	0	changed by	☐	5.00000
4	4	csiSensor4	✓	☐	☐	0	changed by	☐	5.00000
5	5	csiSensor5	✓	☐	☐	0	changed by	☐	1.00000
6	11	csiActuator1	☐	☐	✓	0	n/a	☐	0.00000
7	12	csiActuator2	☐	☐	✓	0	n/a	☐	0.00000
8	13	csiActuator3	☐	☐	✓	0	n/a	☐	0.00000
9	10	csiActuator1Feedback	✓	☐	☐	0	changed by	☐	0.10000
10	16	siteName	✓	☐	☐	0	changed by	☐	0.00000
11	6	csiSensor6	✓	☐	☐	5	changed by	☐	0.00000
12	0		☐	☐	☐	0	n/a	☐	0.00000

The connection from csiActuator3 to csiSensor5 is made with a simple copy rule. Rule #2 in our configuration illustrated below makes this connection. Now any time a new state comes in on csiActuator3, it is published back to the dashboard as csiSensor5.

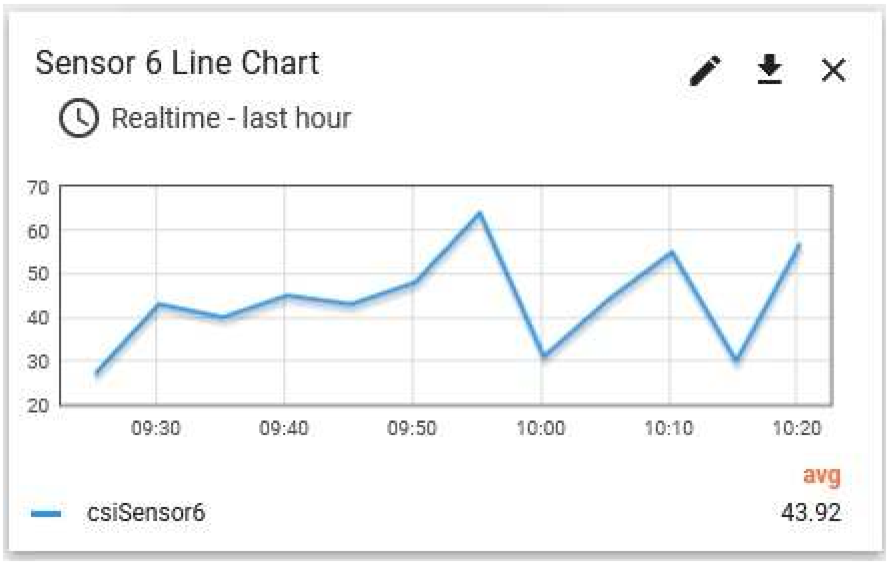


Showing 1 to 3 of 3

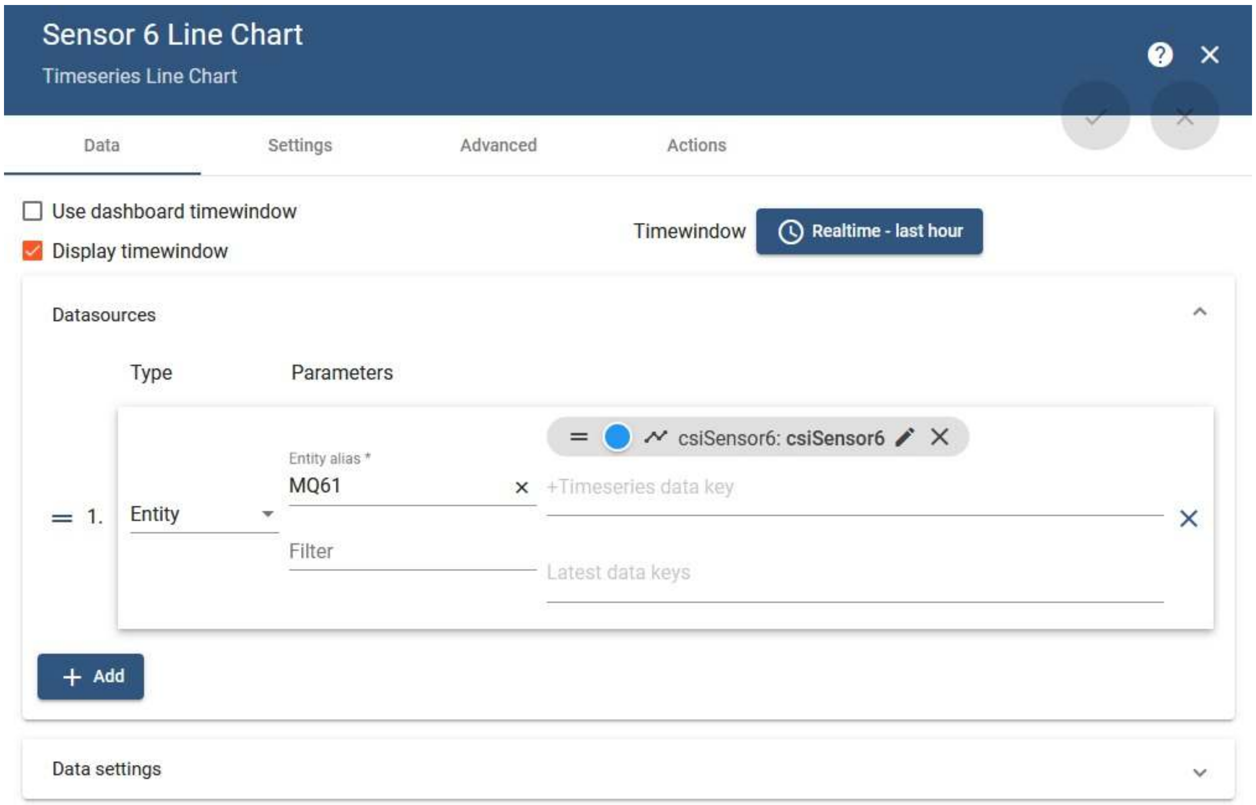
Rule #	Source Register #	Destination Register #
1	11	10
2	13	5
3	0	0

Rules Enabled: 3

The dashboard includes several available widget types for display of data as charts or graphs. Our example uses a simple chart.



Assigning a data point to display is done in the same manner as selecting a data point for a gauge.

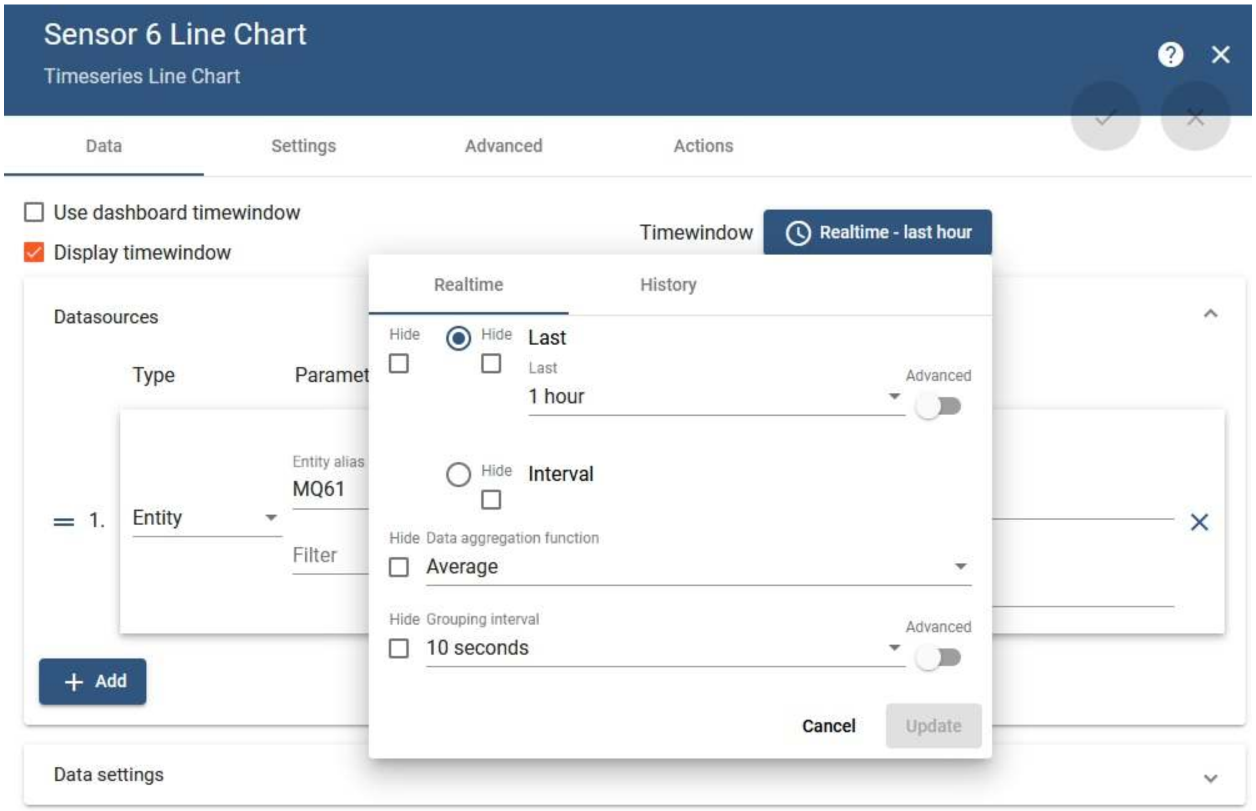

Sensor 6 Line Chart
Timeseries Line Chart

☐ Use dashboard timewindow
☑ Display timewindow
Timewindow: Realtime - last hour

Datasources

Type	Parameters	
1. Entity	Entity alias *: MQ61	csiSensor6: csiSensor6
	Filter	Latest data keys

Turn off "Use dashboard timewindow" and click on the Timewindow icon to the right to select your own time window.



Sensor 6 Line Chart

Timeseries Line Chart

Data | Settings | Advanced | Actions

☐ Use dashboard timewindow

☑ Display timewindow

Timewindow: Realtime - last hour

Datasources

Type	Parameters
1. Entity	Entity alias: MQ61

Realtime | History

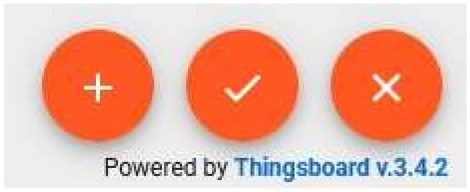
[✓ Last] Last: 1 hour

Interval

Data aggregation function: Average

Grouping interval: 10 seconds

Once you are done making changes to widgets, click the check icon in the lower right to save changes and exit edit mode.


Powered by Thingsboard v.3.4.2