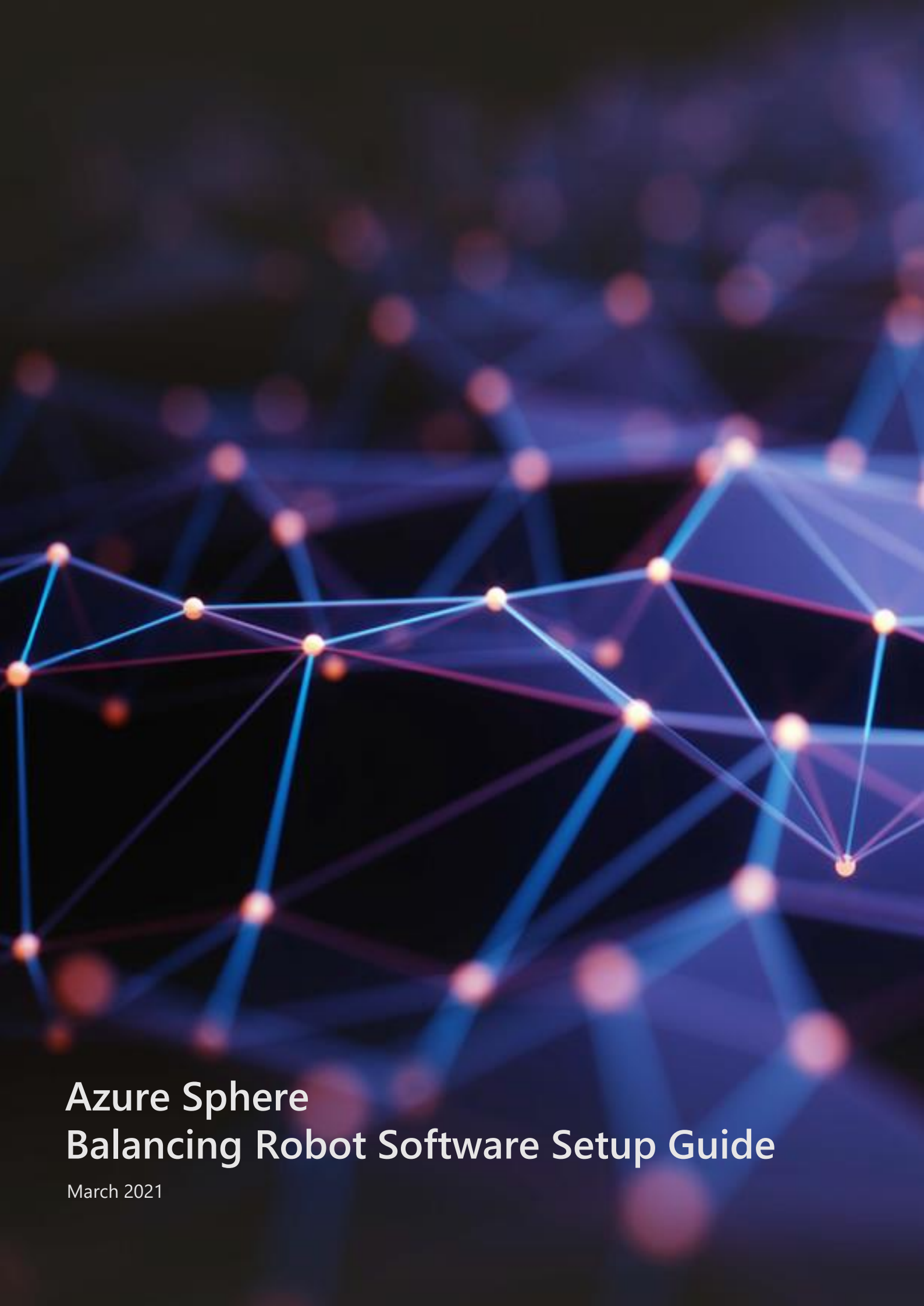


Azure Sphere
Balancing Robot Software Setup Guide

March 2021

Contents

Welcome to the Balancing Robot Software Setup Guide

Purpose & Scope

This document describes the software setup process for the Azure Sphere Balancing Robot demo device – this includes:

- *Creating and configuring an Azure IoT Central application*
- *Creating and configuring an Azure Sphere Tenant*
- *Building the Azure Sphere Robot application for Over The Air deployment*

Creating and Configuring an Azure IoT Central application

Creating an Azure IoT Central Application

The Robot sends telemetry to Azure IoT Central, this includes Battery Level, Current compass direction (number and text), and number of obstacles avoided, there's also the ability to give the Robot a desired direction to face.

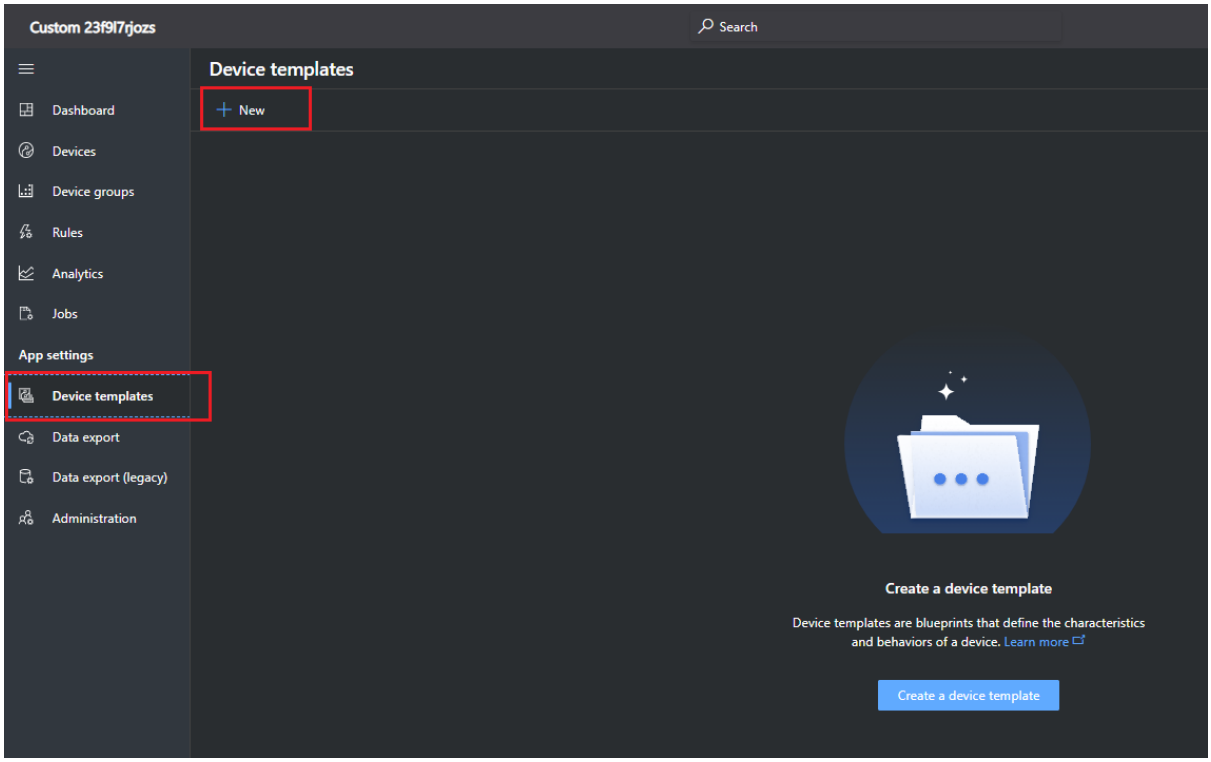
You need to create an Azure IoT Central application that your Robot will connect to, follow the Create an Application instructions, once created you will need to import the Robot Device Template, and configure the IoT Central View and Settings.

Import the Robot Device Template

Once your Azure IoT Central application is created you will need to either define a Device Template, or import an existing Device Template – The Device Template defines the data model for your device, we've pre-created a Device Template which you can import to your IoT Central application – the templace can be found in the Software\Azure_IoT_Central_Template folder, and is called **RealTimeRobotIoTCTemplate.json**.

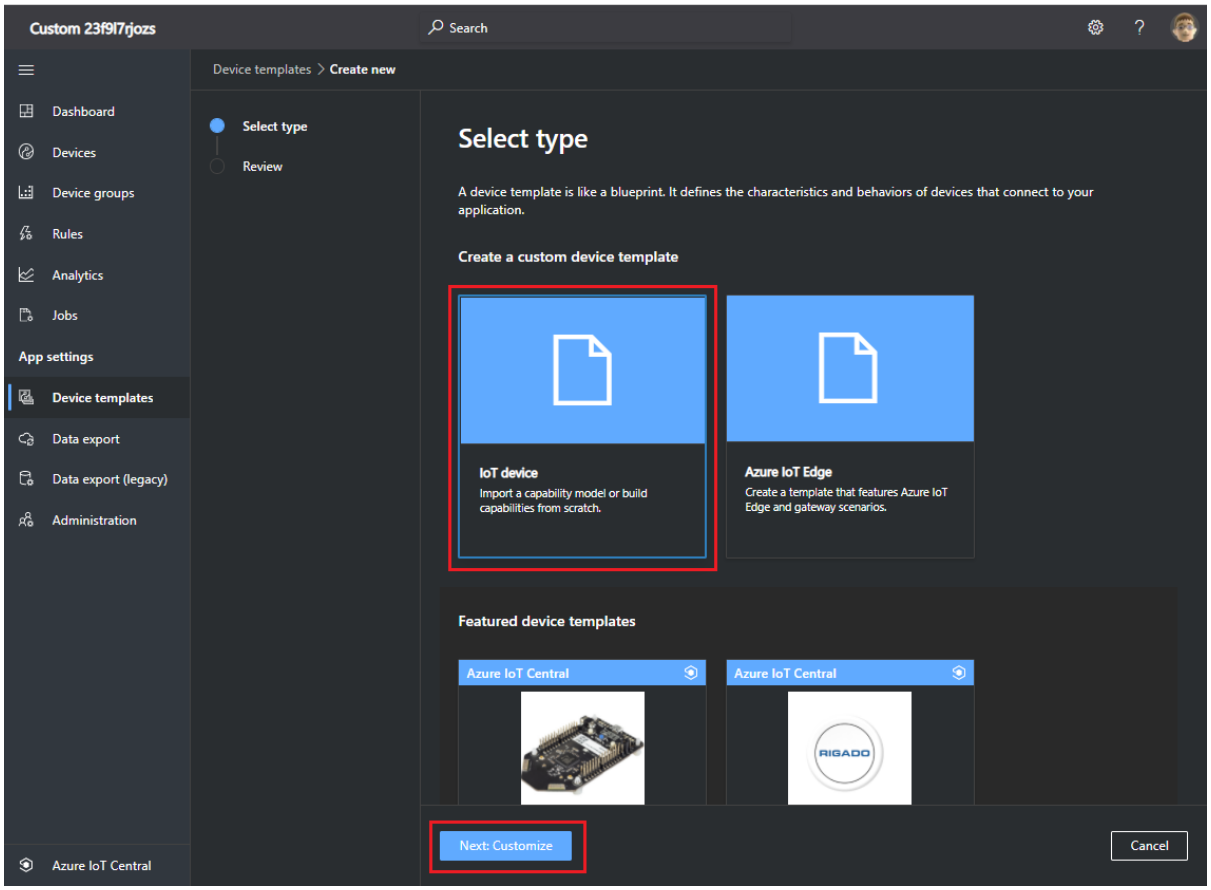
To import the Device Template:

- Select the **Device Templates** option on the left side of the IoT Central application
- Select **New**



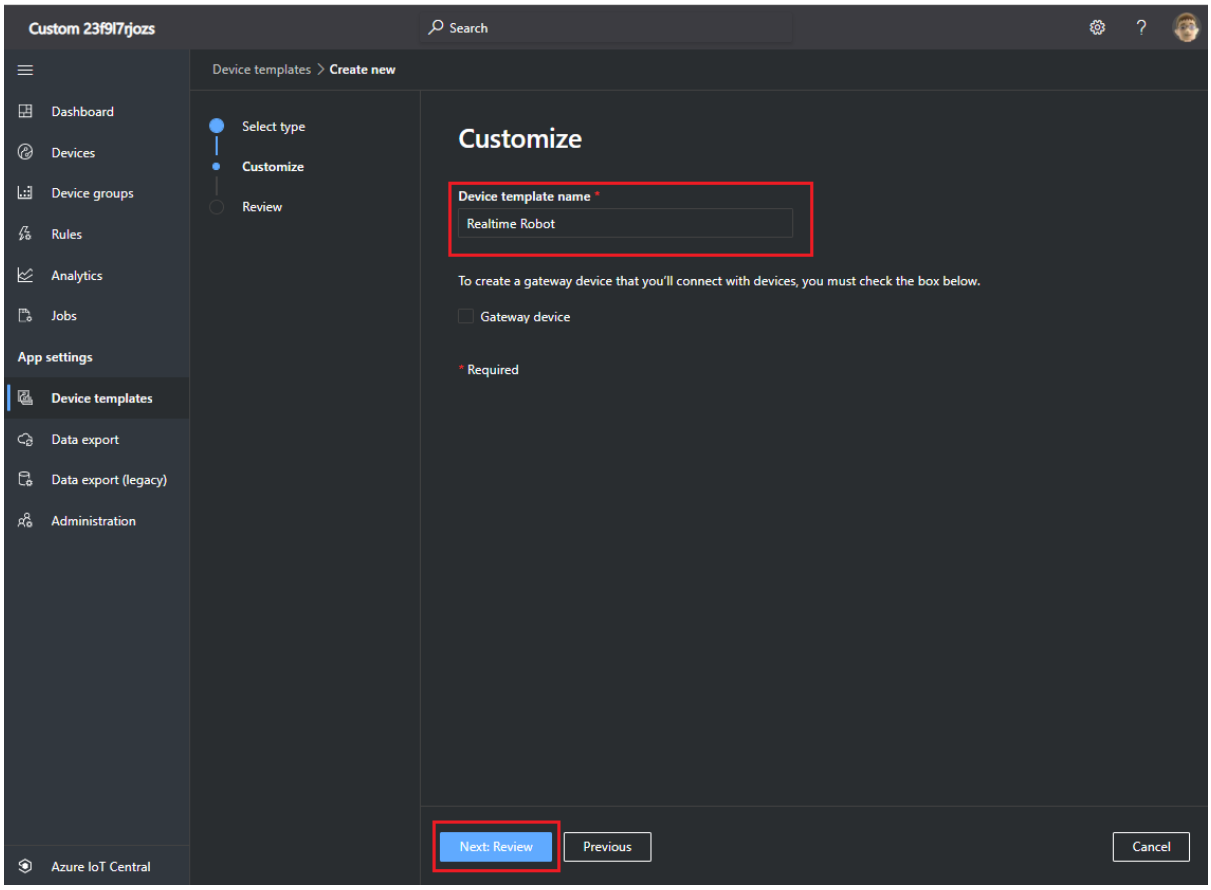
- Select **IoT device** on the 'Select Type' page

- Then click **Next: Customize**



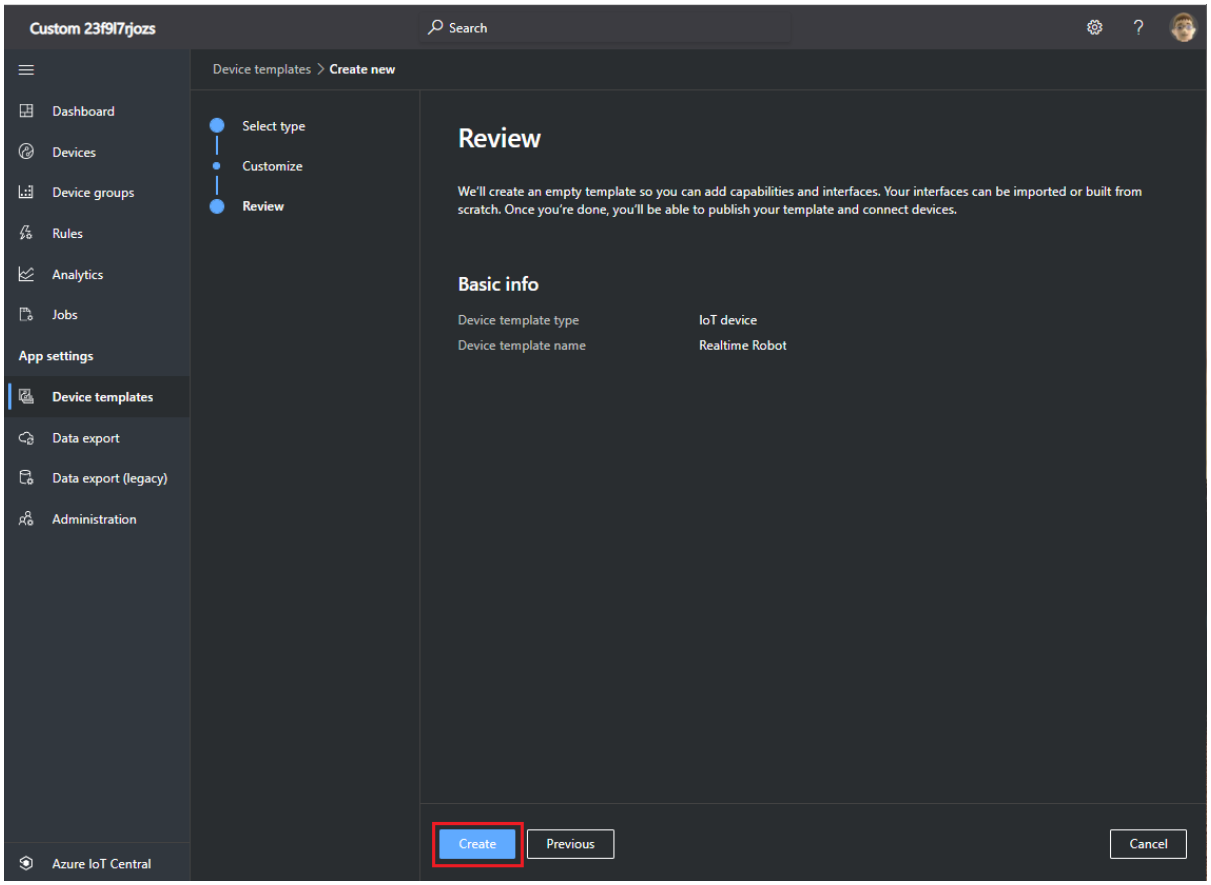
On the Customize page

- Enter a *Device template name*, example: **Realtime Robot**
- Click **Next: Review**



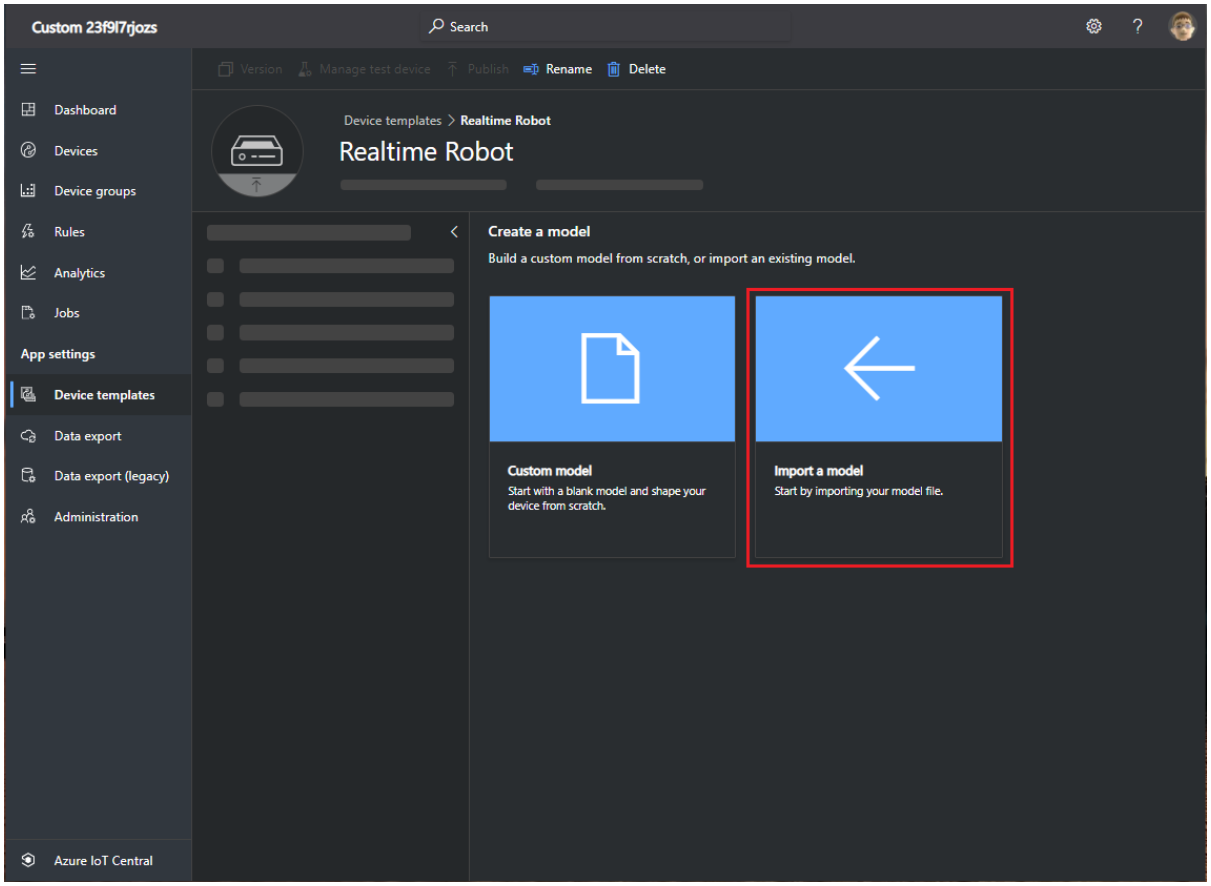
On the *Review* page

- Click **Create**



Now that the Device Template has been created you can now import the data model.

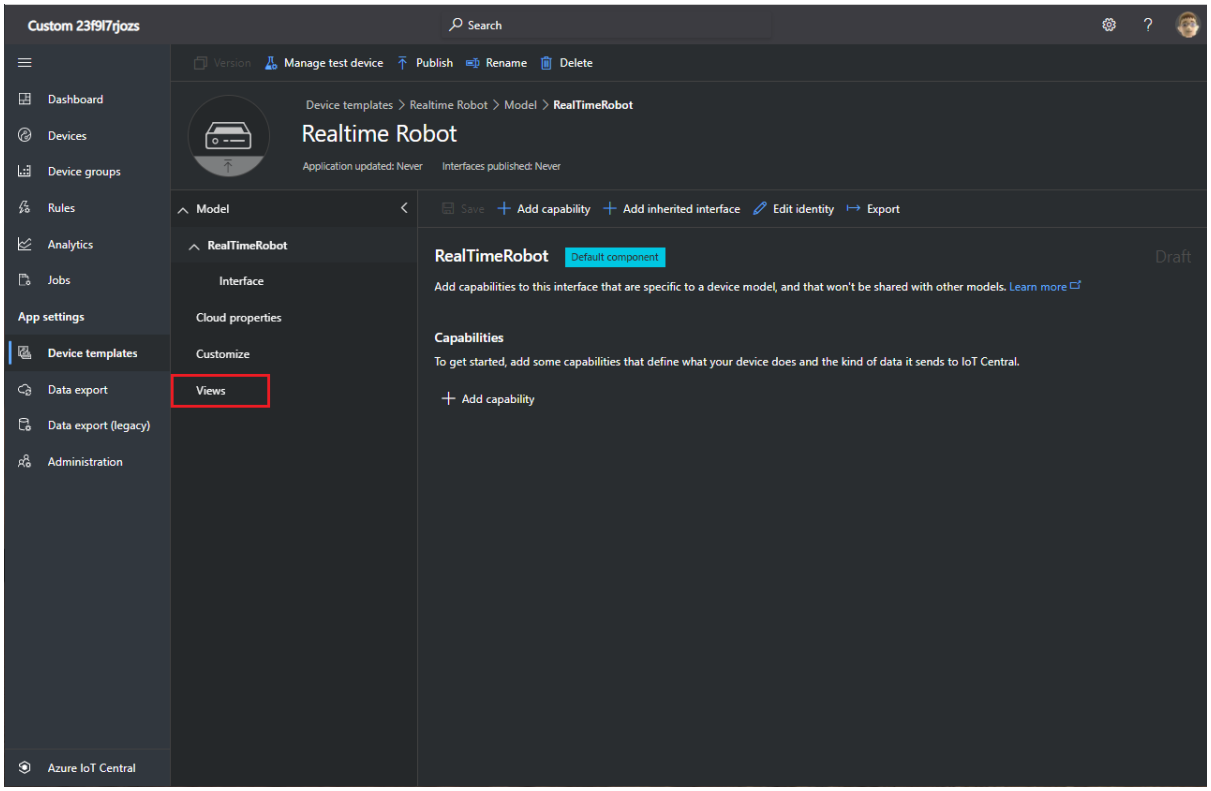
- Select **Import a model**



- Select the **RealTimeRobotIoTCTemplate.json** file from the BalancingRobot/Software folder

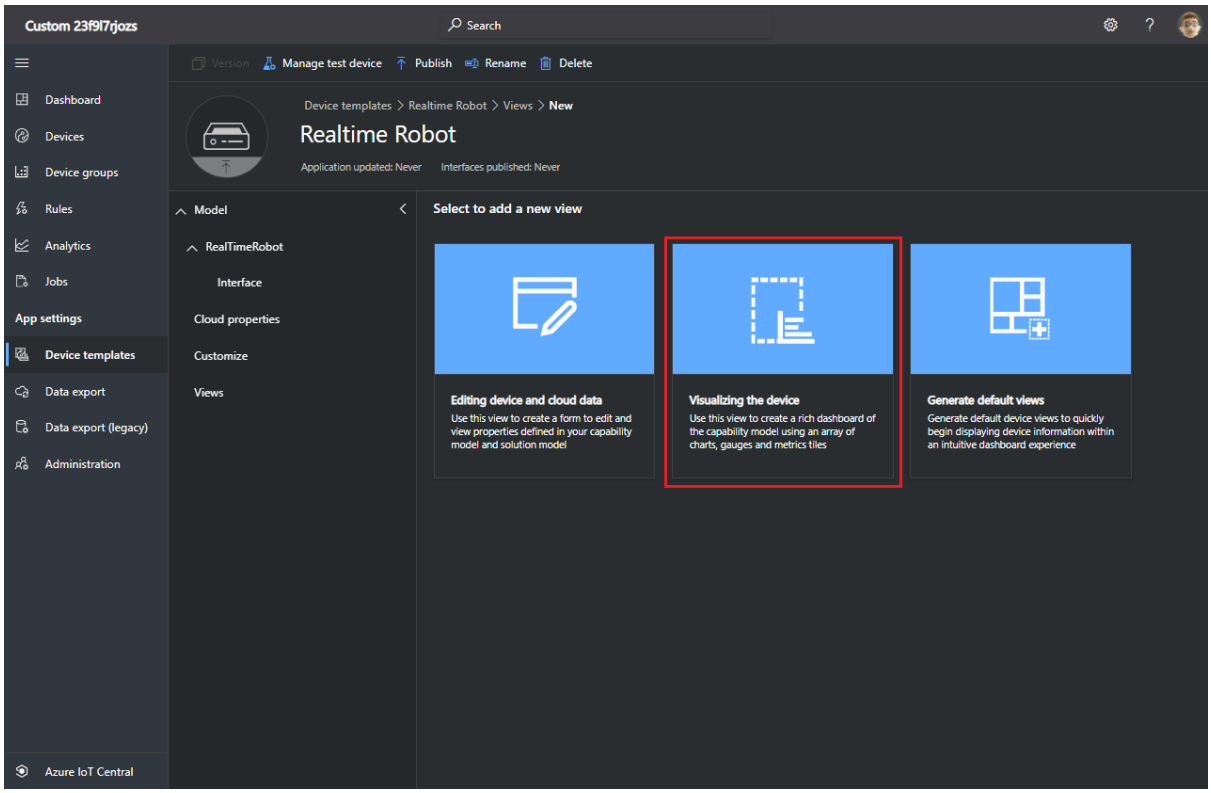
Once the model has been imported you will need to define a couple of views – one view will display telemetry from the Robot, the second will enable the ability to set the compass direction the Robot will face.

- Select **Views** in the Realtime Robot Model



You can now customize the views to show specific telemetry and properties.

- Select **Visualizing the device**

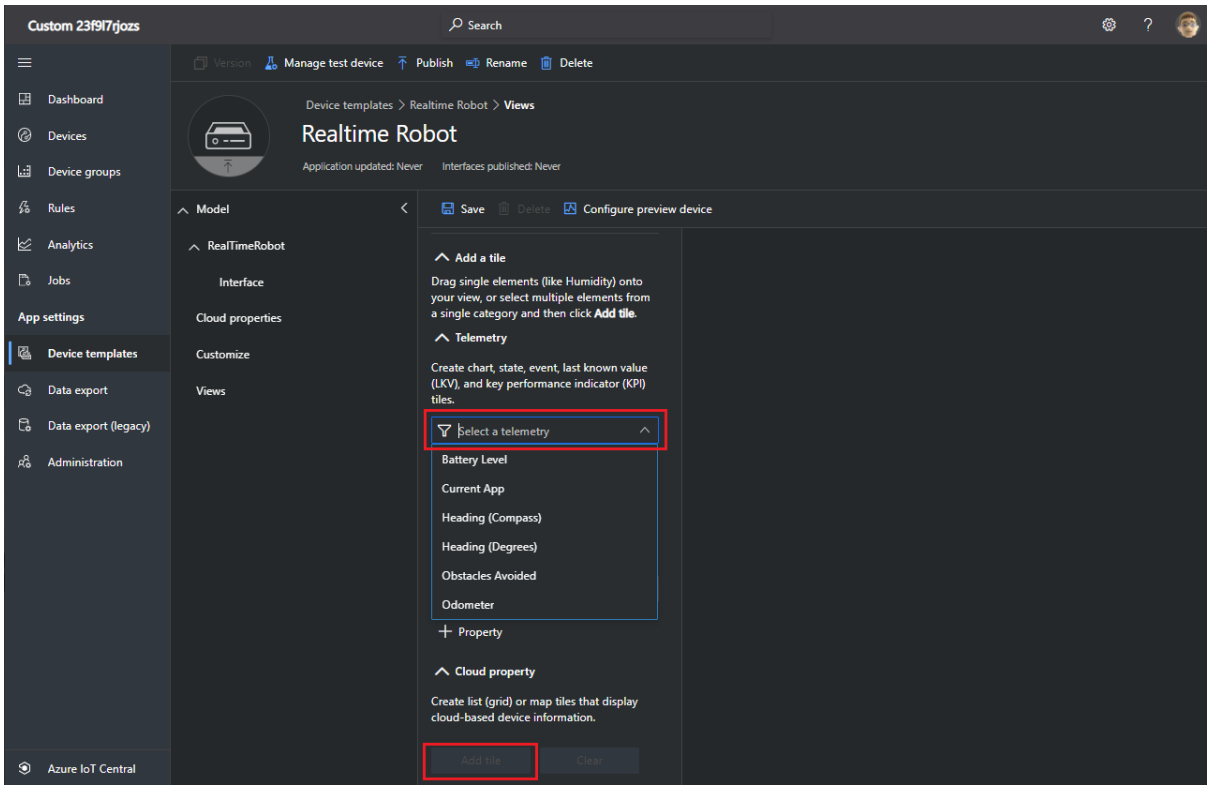


You will need to add 5 telemetry items into the view, these are:

- Battery Level

- Heading (Compass)
- Heading (Degrees)
- Obstacles Avoided
- Current App

- For each of the telemetry items, click **Select a telemetry**
- Choose the appropriate item from the list (see above)
- Click **Add Tile**

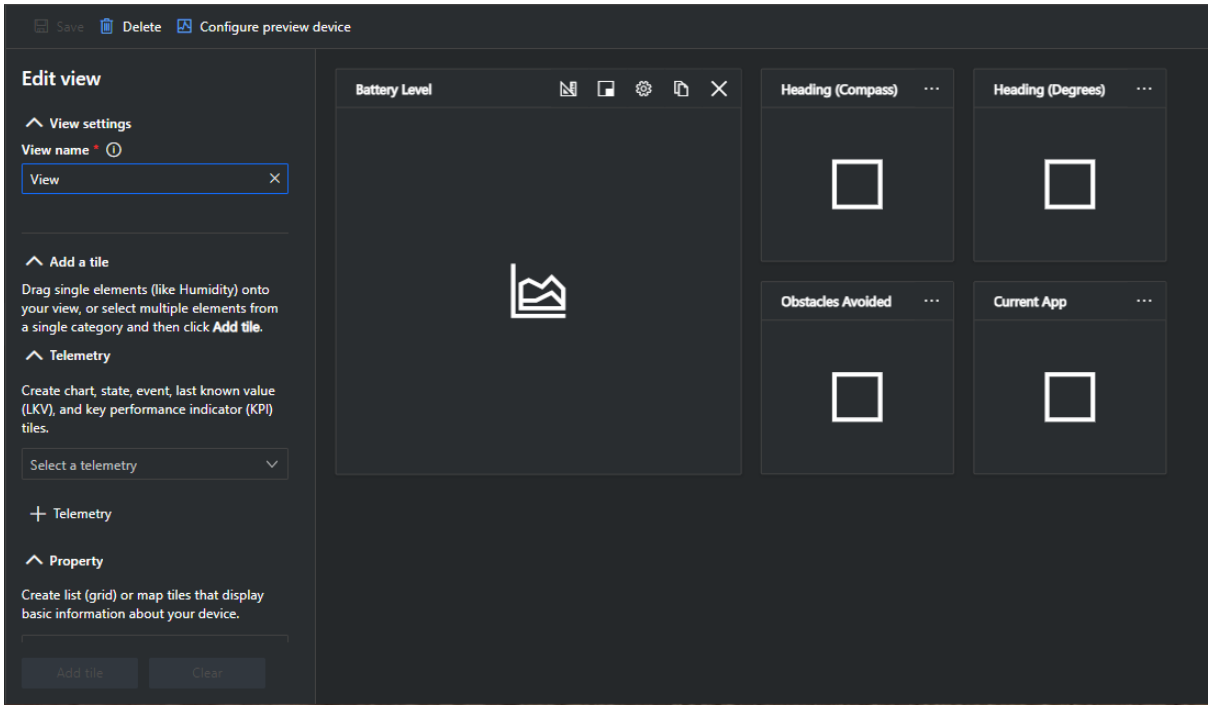


Configure the view with the following properties.

Item	Size	View Type
Battery Level	2x2	Line Graph
Heading (Compass)	1x1	Last Known Value
Heading (Degrees)	1x1	Last Known Value
Obstacles Avoided	1x1	Last Known Value
Current App	1x1	Last Known Value

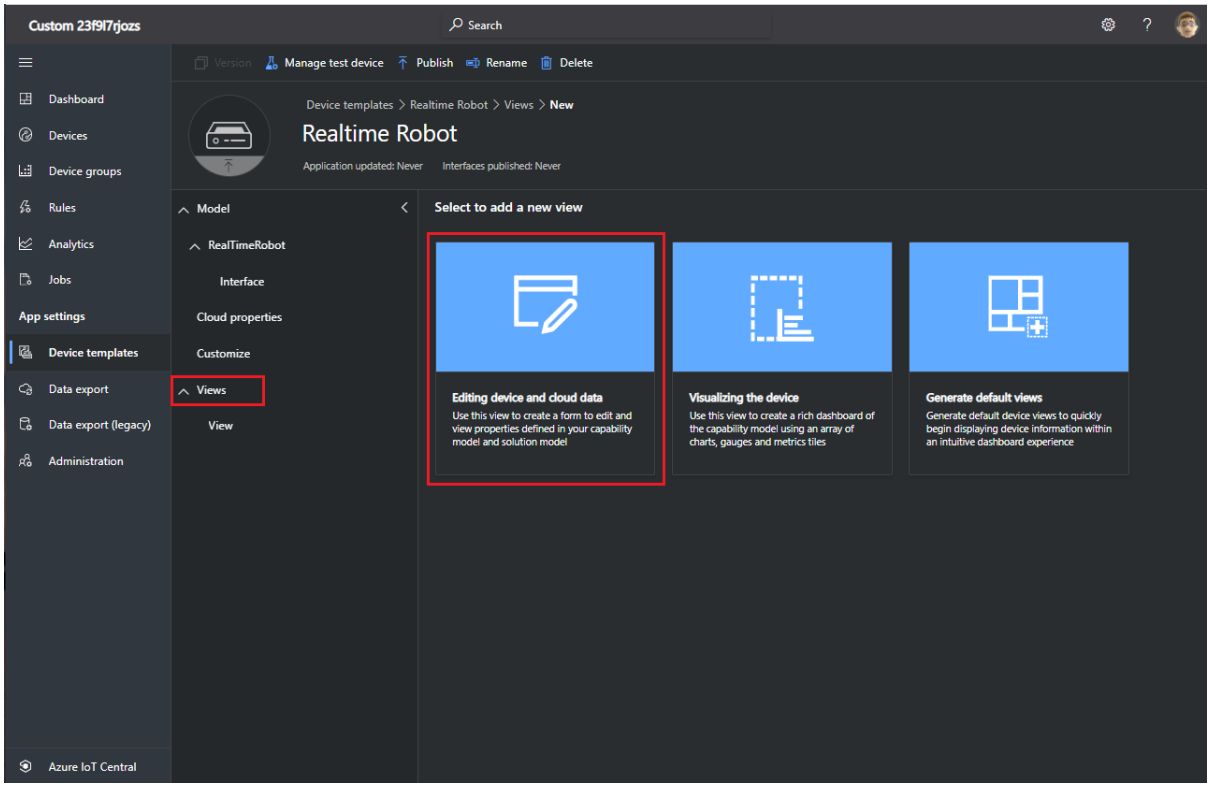
Note: Don't forget to click **Save** once you have finished configuring the view.

Your view should look like the following:

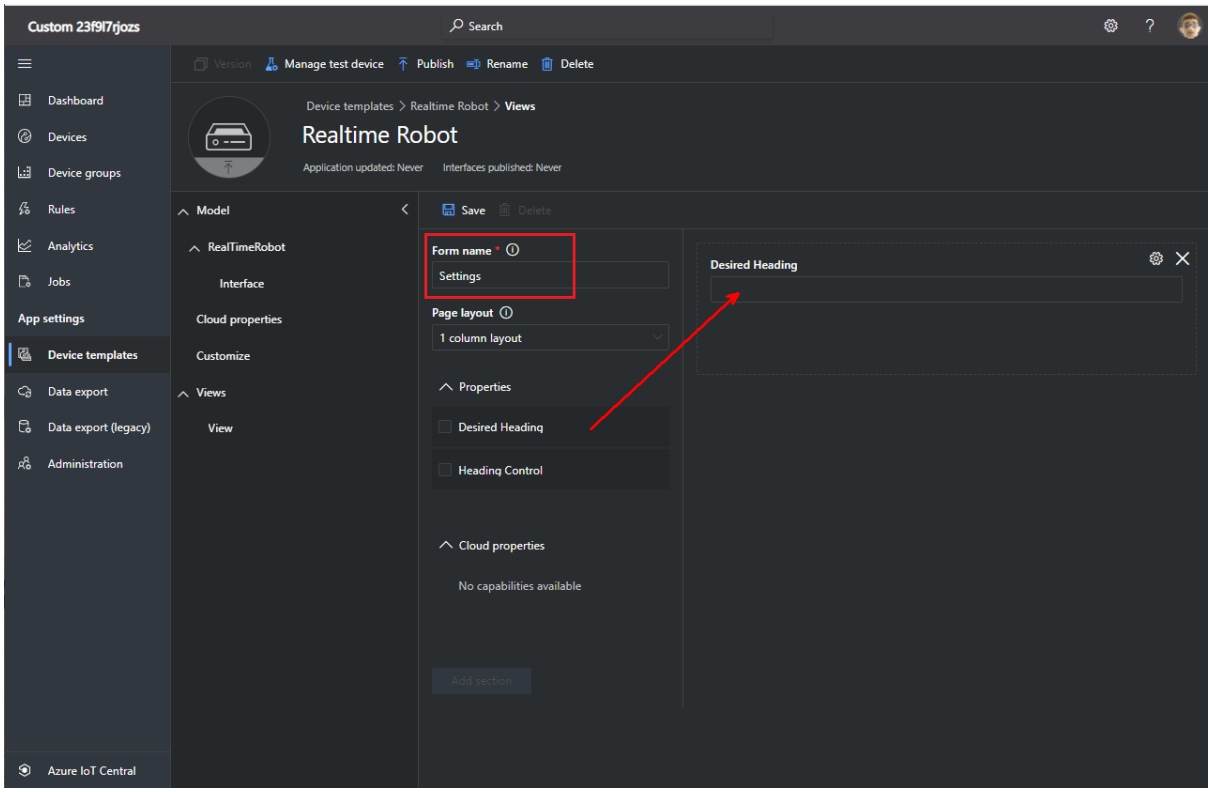


Next, add settings to enable configuring the robot to point in a specific direction.

- Select **Views**
- Select **Editing device and cloud data**



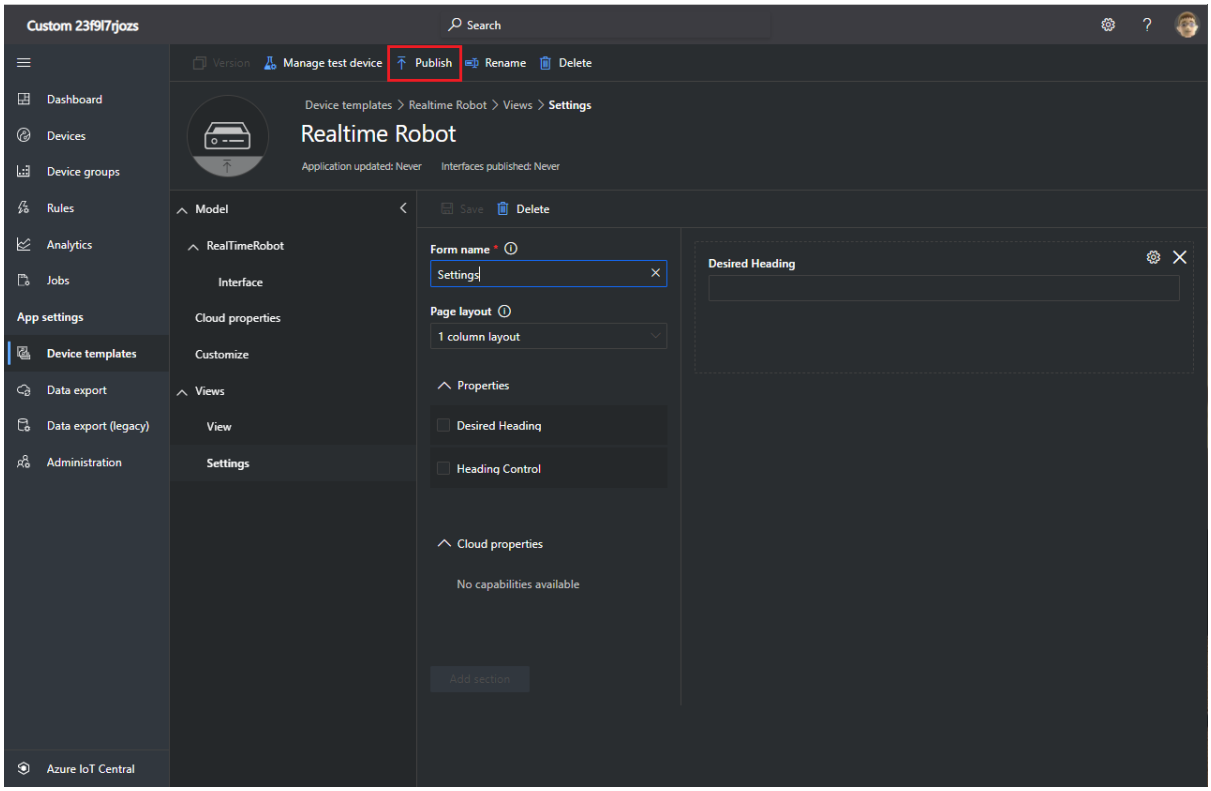
- Change the Form Name to **Settings**
- Drag the **Desired Heading** to the form preview area (see the arrow below)



- Click **Save**

The Device Template is now ready to publish.

- Click **Publish –** When prompted, click **Publish** to confirm the changes.



The Azure IoT Central app is now mostly configured (you still need to setup the trust relationship between your Azure Sphere Tenant and Azure IoT Central, this will be covered in the next section).

Setup The Azure Sphere Environment

The Balancing Robot demo supports Over The Air updates (here's an IoT Show that explains and demonstrates Over The Air updates for Azure Sphere). Supporting Over The Air updates requires that devices are claimed into an Azure Sphere tenant, are selected into an appropriate Product/Device Group, this is explained in the video linked above, and in the Tutorial: Create a cloud deployment.

Create an Azure Sphere Tenant

Instructions for creating an Azure Sphere tenant can be found in the Microsoft online documentation – Create an Azure Sphere Tenant

Note that you will need the SDK for Windows or Linux, and an unclaimed device – if you have an existing tenant you could reuse this for the Robot demo, you can create a new Product/Device Groups used by the demo within your existing tenant.

Linking your Azure Sphere tenant to Azure IoT Central

The devices in your tenant will need to authenticate to your Azure IoT Central application – the following document explains how to link your tenant to Azure IoT Central

Note that your Azure Sphere High-Level application app_manifest.json will need to be updated to contain the Azure IoT Central endpoints, this is documented in the Azure IoT Sample in the Azure Sphere Github samples repository.

Creating the Azure Sphere Product

You can confirm that your Azure Sphere developer command prompt is using the correct Tenant by typing:

```
azsphere tenant show-selected
```

You create a Product within the tenant using the following command:

```
azsphere product create --name MyProduct --description "My First Product"
```

The Demo instructions use a Product called 'Robot', the following command will create the Robot product

```
azsphere product create --name Robot --description "Robot Demo"
```

You can confirm that the Robot product has been created in your tenant by using the following command:

```
azsphere product list
```

Create the Device Groups

The Over the Air demo uses two device groups, one to deploy Robot 'AppA', and the other to deploy Robot 'AppB' – The Real-Time application is identical for both AppA and AppB, the High-Level application will display a different icon on the front panel depending on which application is being deployed.

The demo requires two Device Groups, these are 'AppA' and 'AppB' – create the two Device Groups as follows:

```
azsphere device-group create --name AppA --description "RobotAppA" --product Robot
```

```
azsphere device-group create --name AppB --description "RobotAppB" --product Robot
```

Enable cloud-based deployment for your device

You may have been testing your Robot code on a device by building and deploying the code directly from Visual Studio/Visual Studio Code – To support Over the Air updates your device needs to be in a Device Group that supports over the air update.

To disable development on your currently connected device, and to enable cloud deployment, type the following:

```
azsphere device enable-cloud-test --product "Robot" --device-group "AppA"
```

Deploying the Azure Sphere Applications

The Robot demo uses two applications, the Azure RTOS application runs on one of the M4 cores, the High-Level application runs on the A7 core. Once both applications are built you will have two .imagepackage files (one for the Real-Time app, and one for the High-Level app).

You can create a deployment to a specific Device Group using the following commands.

```
azsphere device-group deployment create --devicegroupname AppA -f --product Robot
--filepath Robot_HighLevel.imagepackage --filepath Robot_RealTime.imagepackage
```

```
azsphere device-group deployment create --devicegroupname AppB -f --product Robot
--filepath Robot_HighLevel.imagepackage --filepath Robot_RealTime.imagepackage
```

Note: The High-Level application will need to be rebuilt with the appropriate CmdArgs setting to identify the application as ‘AppA’ or ‘AppB’.

Building the Robot Software

Real-Time and High Level Apps

The Robot project contains two applications, the Real-Time code (based on Azure RTOS) runs on one of the M4 cores, the High-Level application runs on the A7 core.

There are two application CMakeLists.txt files, one for the High-Level application, and one for the Real-Time application, both need to be built/deployed when working with a development enabled board (deploy the real-time application first), for an Over The Air deployment you need to deploy both applications.

Please refer to the README.md files for both Real-Time and High Level apps for instructions on how to build each.