IoT Central - Hands-on Lab

Scenario

Suppose you run a company that operates a fleet of refrigerated trucks. You have many customers within a city, and you operate from a base. You command each truck to deliver its contents to a customer.

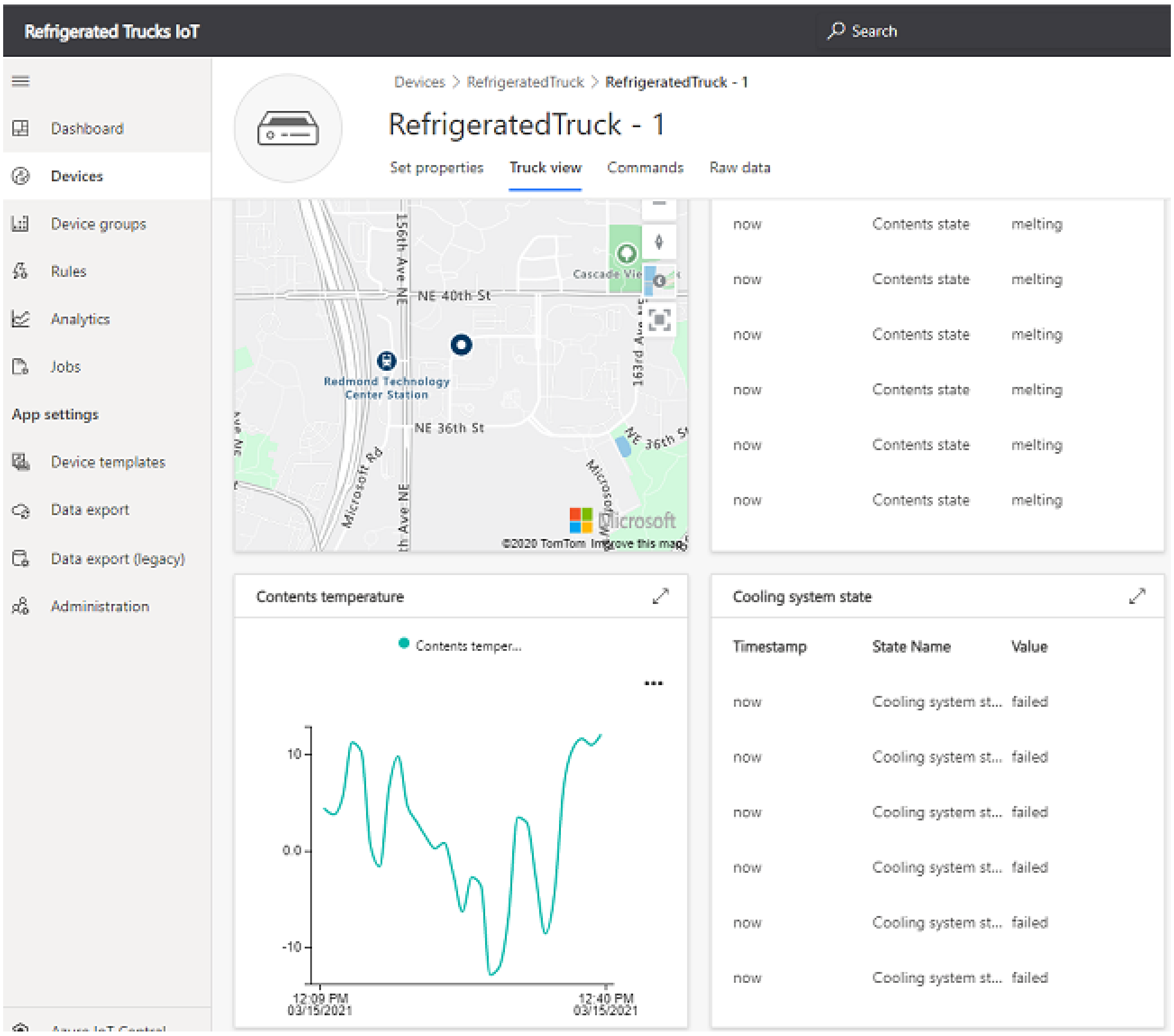
If the cooling system fails on a truck and the contents start to melt, you need to instruct the truck to return to base and unload the contents. Or you can instead deliver the contents to a customer who's nearby when the cooling system fails.

To make these decisions, you need an up-to-date picture of all that happens with your trucks. You need to know the location of each truck on a map, the status of the cooling system, and the status of the contents.

IoT Central provides all you need to handle this scenario. In the following image, for example, the colored circles show the location of a truck on its way to a customer.

Final Application

At the end of this hands=on lab your newly created application should look like the below one.



now	Contents state	melting
now	Contents state	melting
now	Contents state	melting
now	Contents state	melting
now	Contents state	melting
now	Contents state	melting

Cooling system state

Timestamp	State Name	Value
now	Cooling system st...	failed
now	Cooling system st...	failed
now	Cooling system st...	failed
now	Cooling system st...	failed
now	Cooling system st...	failed
now	Cooling system st...	failed

IoT Central Theory

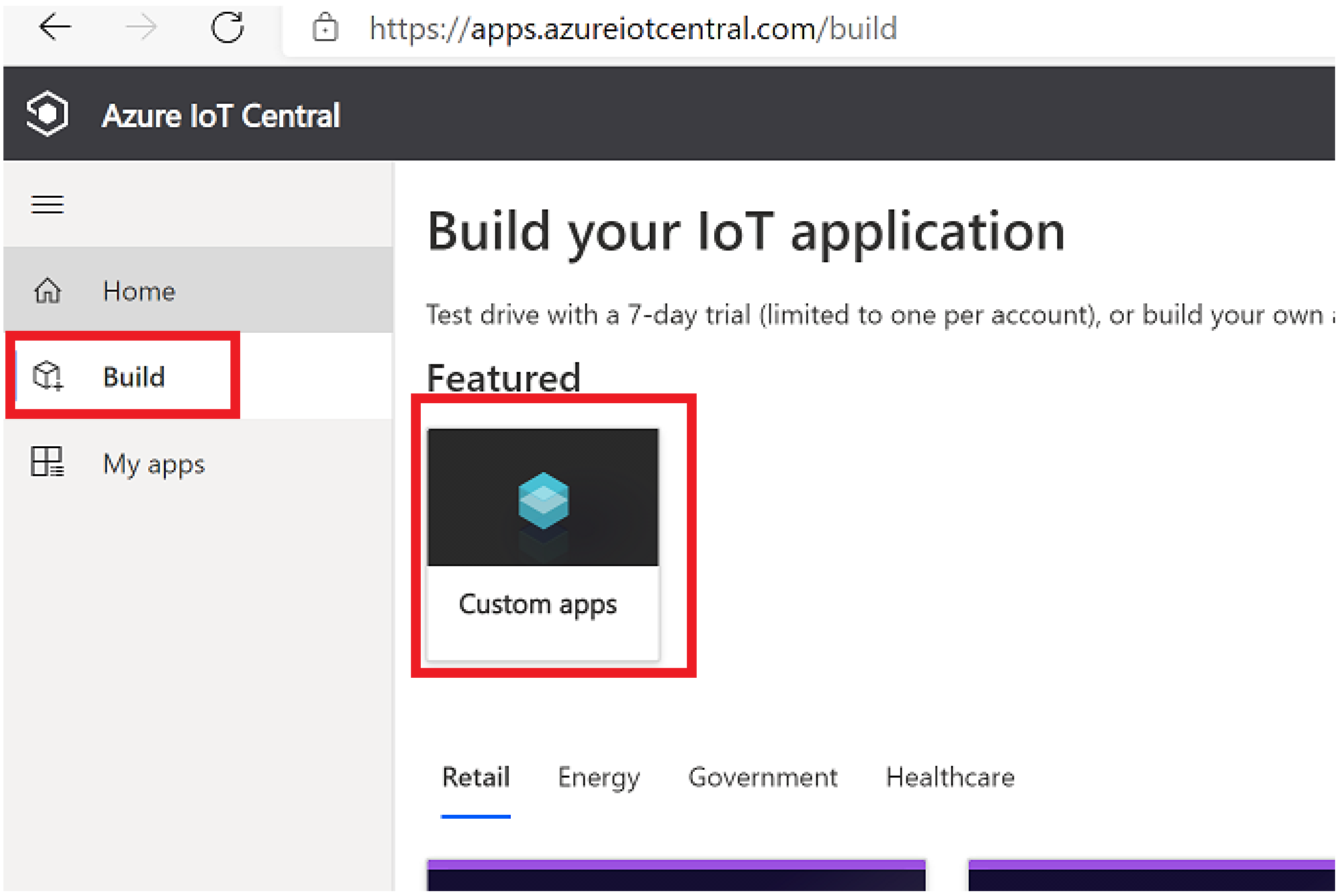
The deck presented in this section it is available in the pdf files folder.

Content - Hands-on Lab:

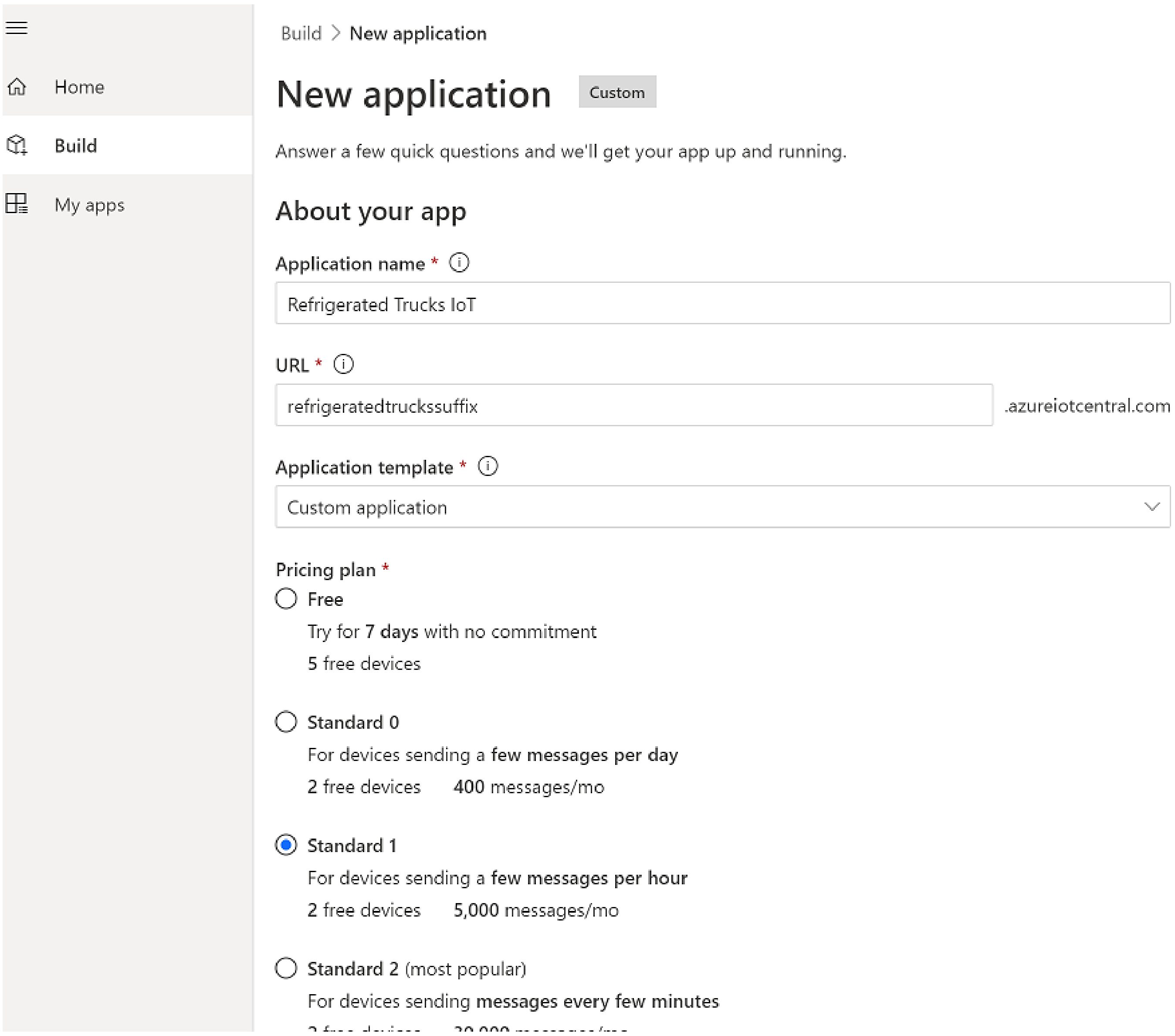
Exercise 1: Create a Custom IoT Central app

Task 1: Creating an Application

In the browser open Azure IoT Central: https://apps.azureiotcentral.com/


https://apps.azureiotcentral.com/build
Azure IoT Central
Home
Build
My apps
Build your IoT application
Test drive with a 7-day trial (limited to one per account), or build your own
Featured
Custom apps
Retail
Energy
Government
Healthcare

After selecting Custom apps, you should fill the fields in the Application Form:



New application Custom

Answer a few quick questions and we'll get your app up and running.

About your app

Application name * Refrigerated Trucks IoT

URL * refrigeratedtruckssuffix .azureiotcentral.com

Application template * Custom application

Pricing plan *

Free
Try for 7 days with no commitment
5 free devices

Standard 0
For devices sending a few messages per day
2 free devices 400 messages/mo

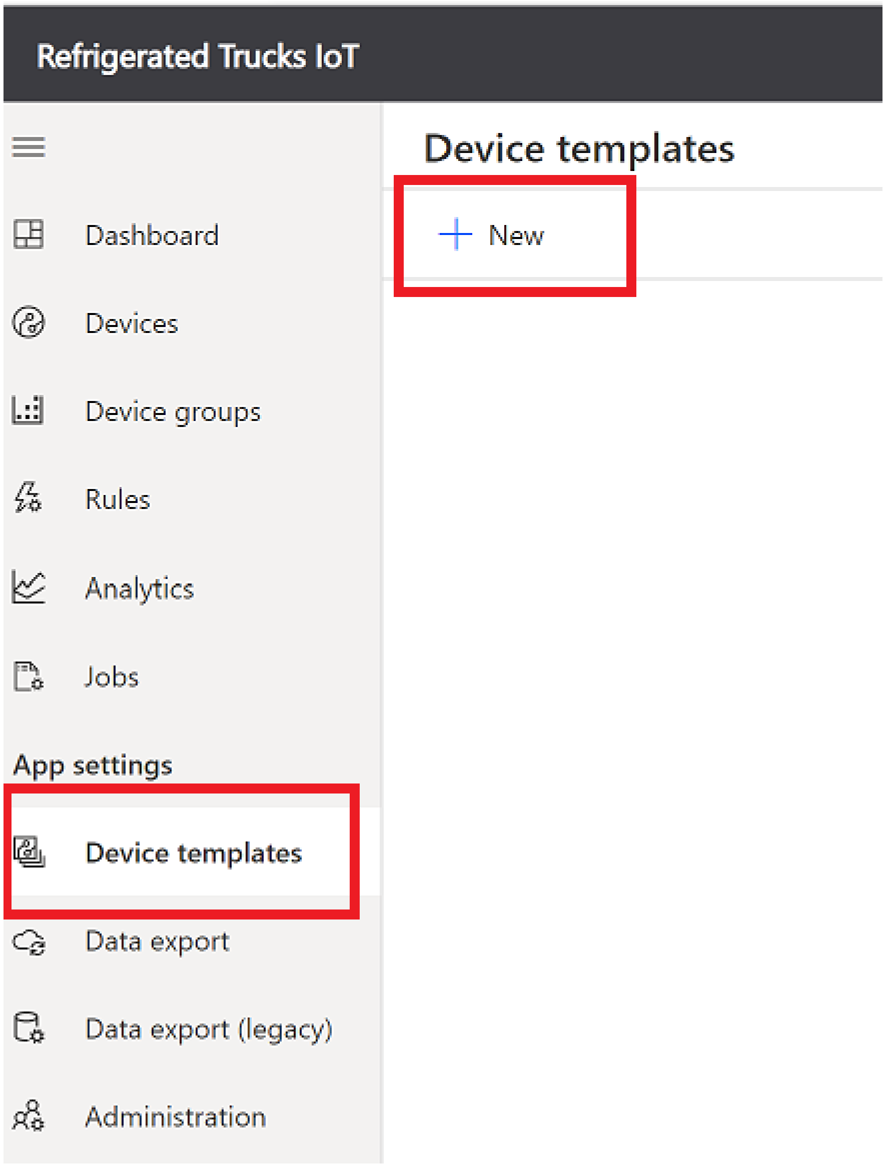
[✓ Standard 1]
For devices sending a few messages per hour
2 free devices 5,000 messages/mo

Standard 2 (most popular)
For devices sending messages every few minutes

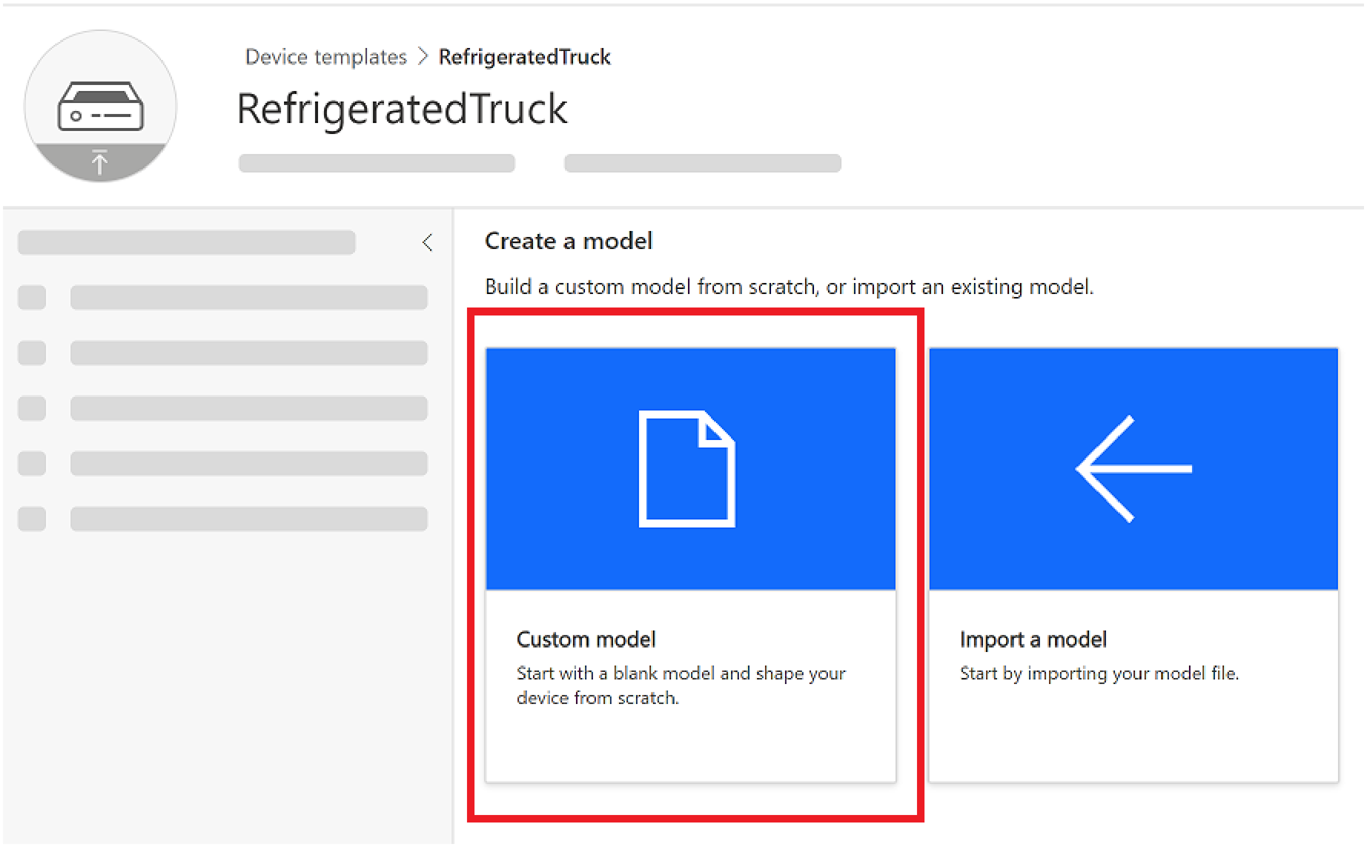
- **Application Name:** Refrigerated Trucks
- **URL:** refrigerated-trucks-SUFFIX must be a unique URL
- **Application Template:** Custom application, default.
- **Pricing Plan:** Standard 1
- **Directory:** Your current company
- **Azure Subscription:** Your current subscription
- **Location** Select the region you are using for this training.

Then select **Create**

Once your Application is available the next step will be to **Create a device template**. On your left menu click on **Device Templates** and then in **New**



1. Select **IoT Device** then **Next: Customize**
2. In the customize screen assign a **Device Template name** RefrigeratedTruck
3. Don't select **Gateway device** box
4. Select **Next: Review**. Then select **Create**.
5. In the Create a model area, select Custom model. Your view should now look similar to the following image.


Device templates > RefrigeratedTruck

RefrigeratedTruck

Create a model

Build a custom model from scratch, or import an existing model.

Custom model

Start with a blank model and shape your device from scratch.

Import a model

Start by importing your model file.

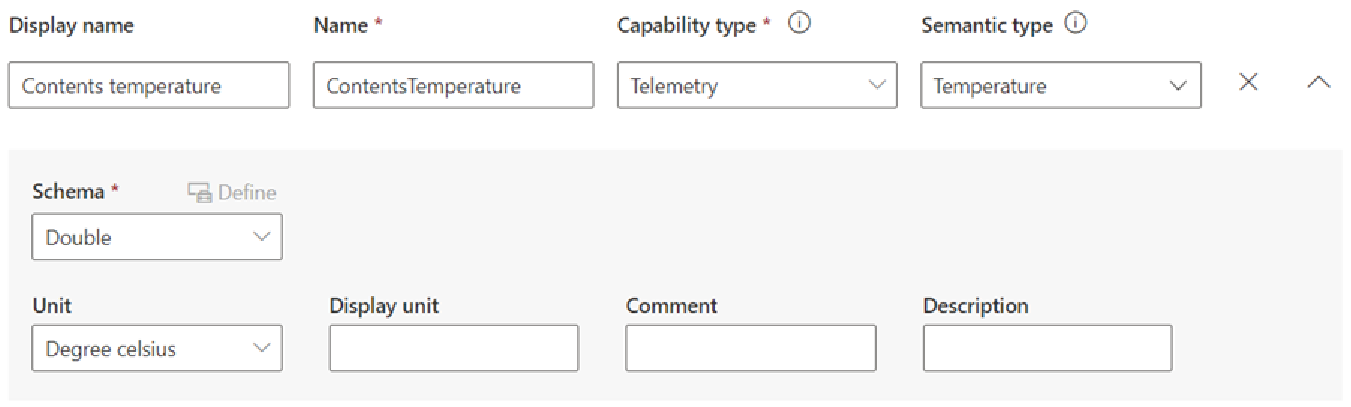
6. Select **Add an inherited interface**.
7. Then select **Custom** to start building from a blank interface

Task 2: Add Capabilities - Telemetry

1. To get started, select **Add capability**. Then enter the values in the following table.

Entry Summary	Value				
Display Name	Contents temperature				
Name	ContentsTemperature				
Capability type	Telemetry				
Semantic type	Temperature	Schema	Double	Unit	C

Your window should now look like the following image:



Display name	Name *	Capability type *	Semantic type
Contents temperature	ContentsTemperature	Telemetry	Temperature

Schema * Define
Double

Unit	Display unit	Comment	Description
Degree celsius			

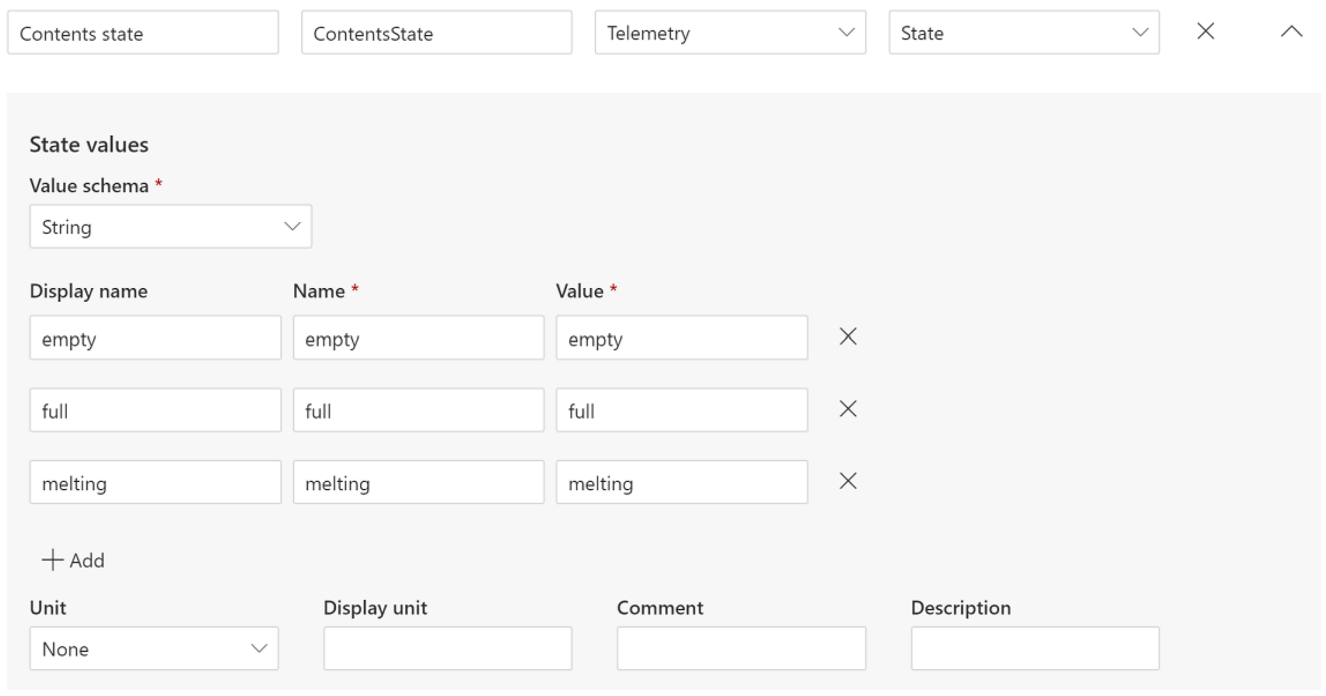
Note: The interface names must be entered exactly as shown in this unit. The names and entries must exactly match in the code you'll add later in this module.

2. States are important. They let the operator know what's happening. A state in IoT Central is a name associated with a range of values. Later you'll choose a color to associate with each value.

Use the **Add capability** control to add a state for the truck's refrigerated contents: **empty**, **full**, or **melting**.

Entry Summary	Value		
Display Name	Contents state		
Name	ContentsState		
Capability type	Telemetry		
Semantic type	State	Schema	String

Select **Add**. For Display name and Value, enter empty. The Name field should be populated automatically with empty. So all three fields are identical, containing **empty**. Add two more state values: **full** and **melting**. Again, the same text should appear in the fields for Display name, Name, and Value.



Contents state	ContentsState	Telemetry	State

State values
Value schema *
String

Display name	Name *	Value *
empty	empty	empty
full	full	full
melting	melting	melting

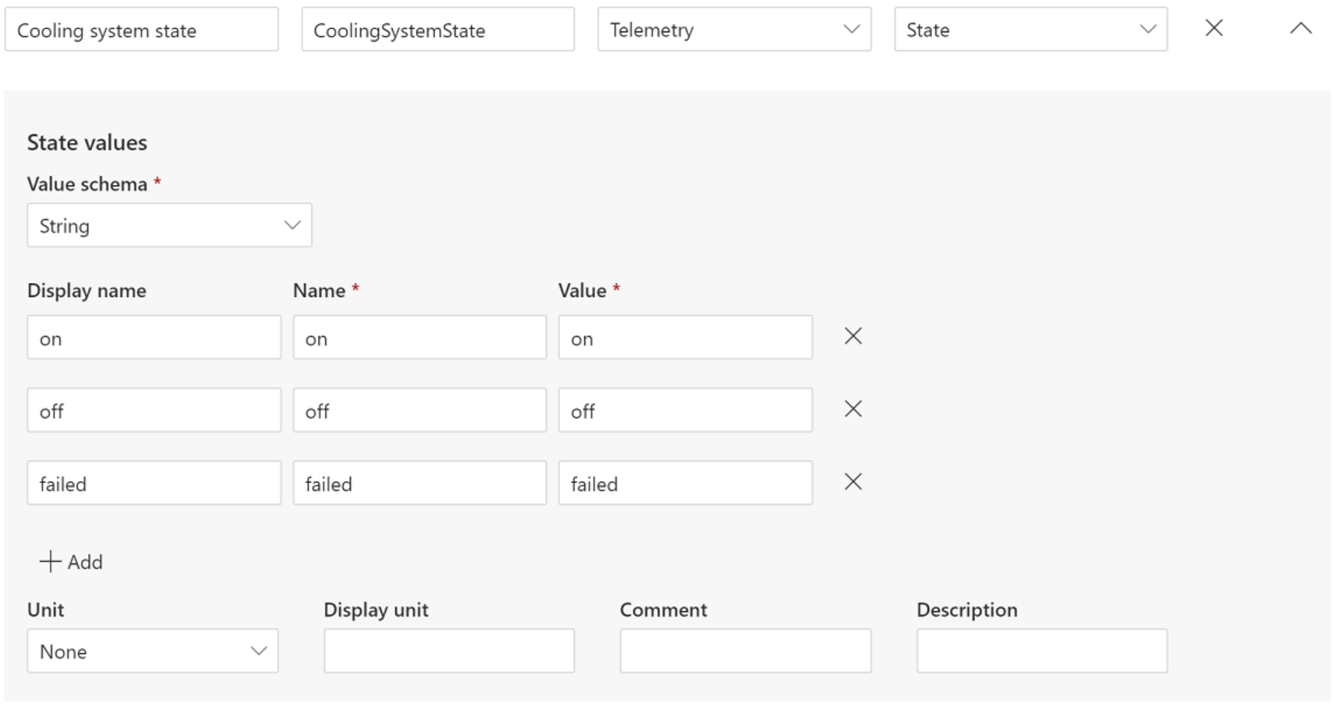
+ Add

Unit	Display unit	Comment	Description
None			

3. If the cooling system fails, as you'll see in the following units, the chances of the contents melting increase considerably.

Entry Summary	Value		
Display Name	Cooling system state		
Name	CoolingSystemState		
Capability type	Telemetry		
Semantic type	State	Value schema	String

Add **on**, **off**, and **failed** entries for the cooling system. Start by selecting Add capability. Then add another state:


Cooling system state | CoolingSystemState | Telemetry | State

State values

Value schema *: String

Display name	Name *	Value *
on	on	on
off	off	off
failed	failed	failed

Unit: None

Display unit

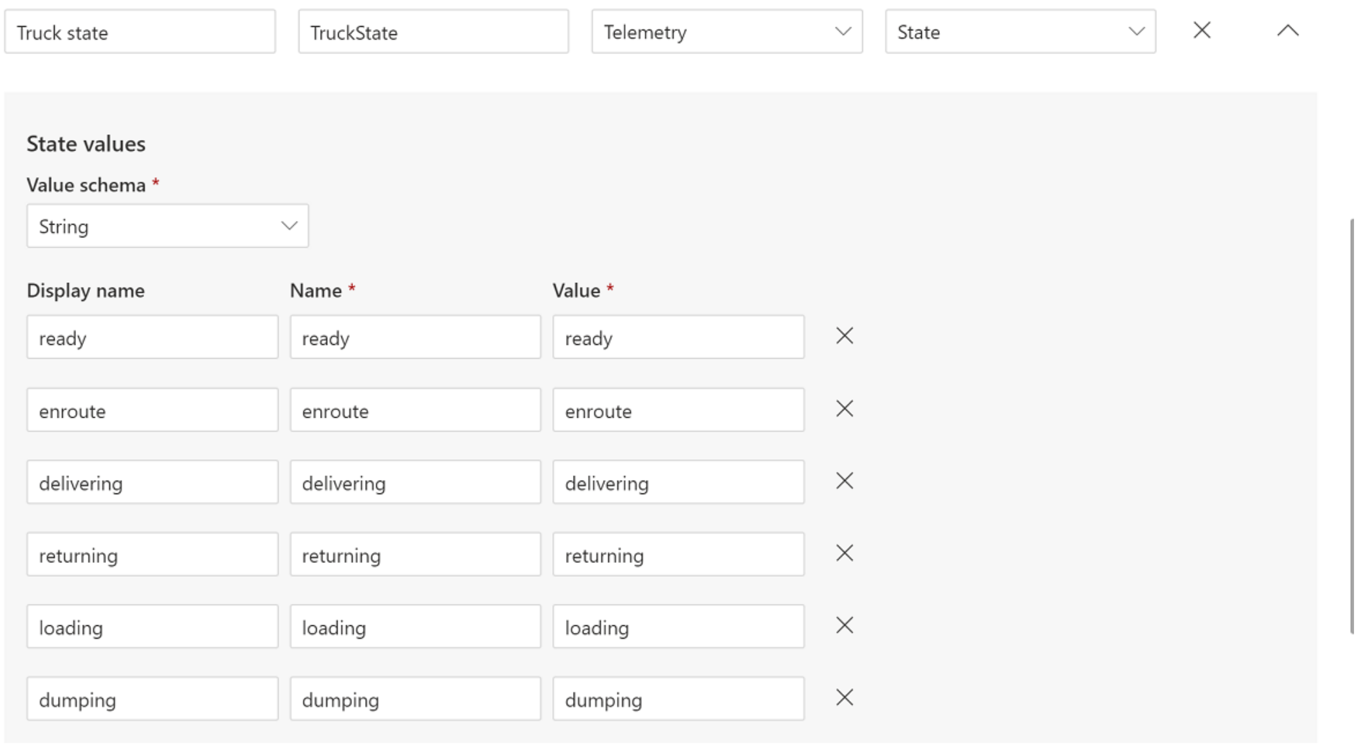
Comment

Description

4. A more complex state is the state of the truck itself. If all goes well, a truck's normal routing might be ready, enroute, delivering, returning, loading, and back to ready again. Also add the dumping state to account for the disposal of melted contents! To create the new state, use the same process as for the last two steps.

Entry Summary	Value		
Display Name	Truck state		
Name	TruckState		
Capability type	Telemetry		
Semantic type	State	Value schema	String

Now add: **ready**, **enroute**, **delivering**, **returning**, **loading**, and **dumping** as shown below:



Truck state	TruckState	Telemetry	State

State values

Value schema *

String

Display name	Name *	Value *
ready	ready	ready
enroute	enroute	enroute
delivering	delivering	delivering
returning	returning	returning
loading	loading	loading
dumping	dumping	dumping

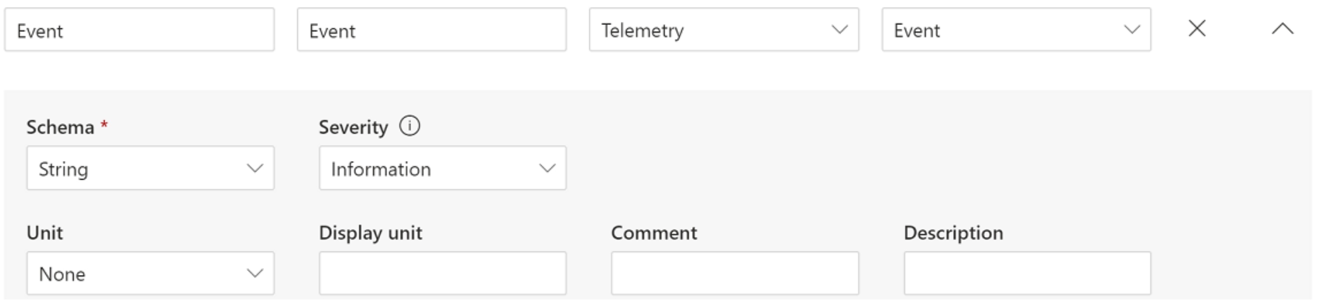
5. Add and Event Capability. One event a device might trigger is a conflicting command. An example might be when an empty truck that's returning from a customer receives a command to deliver its contents to another customer. If a conflict occurs, the device should trigger an event to warn the operator of the IoT Central app.

Another event might just acknowledge and record the customer ID that a truck is to deliver to.

To create an event, select **Add capability**. Then fill in the following information.

Entry Summary	Value				
Display Name	Event				
Name	Event				
Capability type	Telemetry				
Semantic type	Event	Schema	String	Severity	Information

Your settings should look like the image below:



Event	Event	Telemetry	Event
Schema *	String	Severity	Information
Unit	None	Display unit	
Comment		Description	

6. Add a Location capability following the below information:

Entry Summary	Value		
Display Name	Location		
Name	Location		
Capability type	Telemetry		
Semantic type	Location	Schema	Geopoint

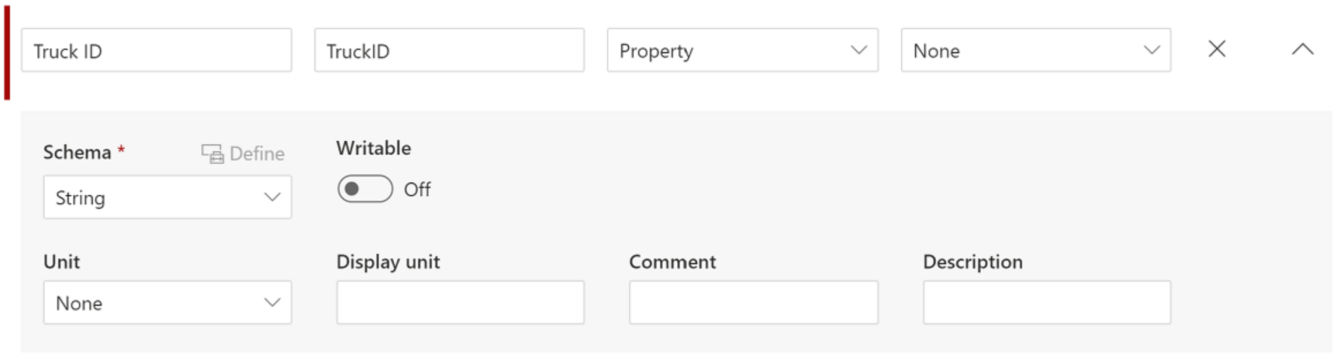
Task 3: Add Capabilities - Properties

You'll define an optimal temperature for the truck contents as a property.

1. Select Add capability. Then add the truck ID property.

Entry Summary	Value						
Display Name	Truck ID						
Name	TruckID						
Capability type	Property						
Semantic type	None	Schema	String	Writable	Off	Unit	None

You should see your property set up as this one below:

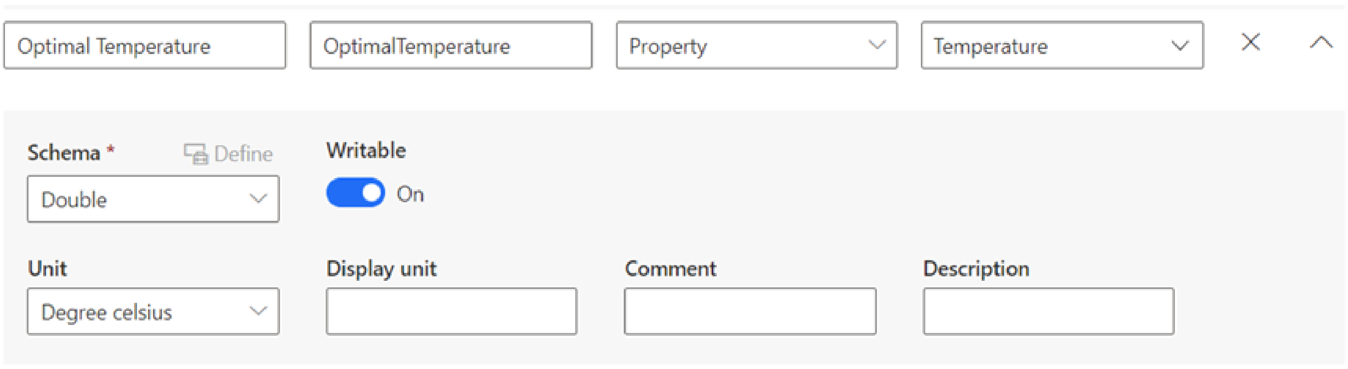


Truck ID	TruckID	Property	None
Schema *	String	Writable	Off
Unit	None	Display unit	
Comment		Description	

2. Add the optimal temperature property.

Entry Summary	Value						
Display Name	Optimal Temperature						
Name	OptimalTemperature						
Capability type	Property						
Semantic type	Temperature	Schema	Double	Writable	On	Unit	Degree celsius

Now, should look like the below image:


Optimal Temperature | OptimalTemperature | Property | Temperature

Schema * Double

Writable On

Unit Degree celsius

Display unit

Comment

Description

Task 4: Add Capabilities - Commands

For refrigerated trucks, you should add two commands:

A command to deliver the contents to a customer A command to recall the truck to base

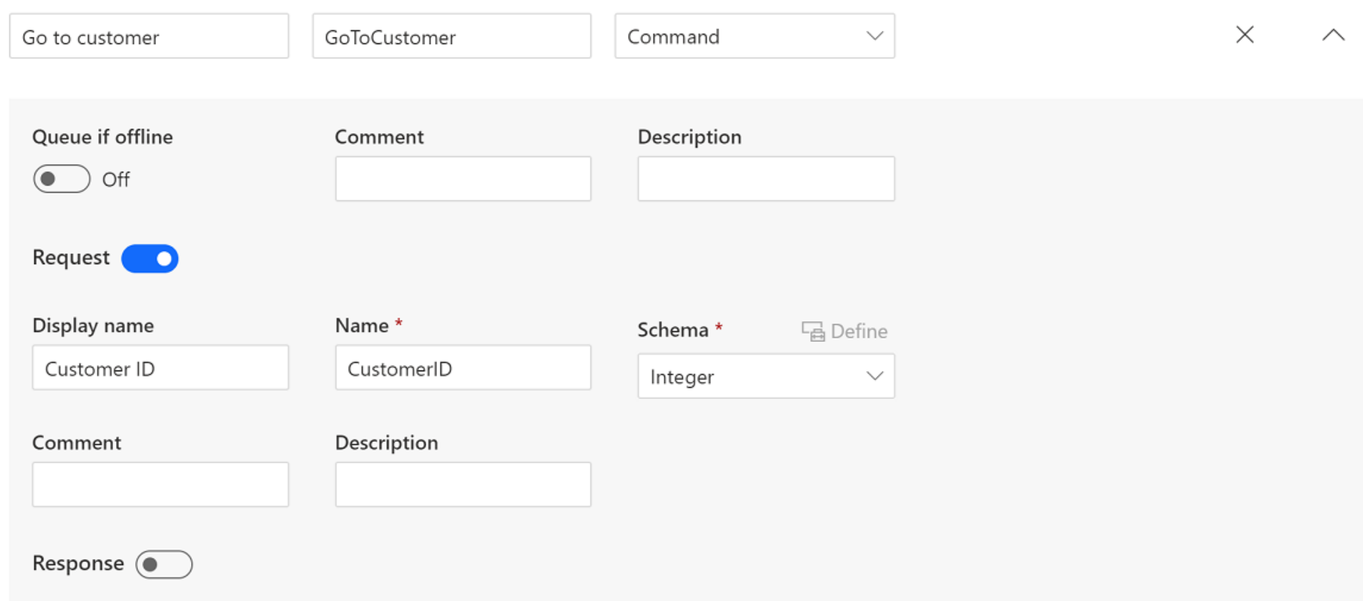
1. To add the commands, select **Add capability**. Then add the first command.

Entry Summary	Value
Display Name	Go to customer
Name	GoToCustomer
Capability type	Command

Turn on the **Request** option to enter more command details.

Entry Summary	Value
Request	On
Display name	Customer ID
Name	CustomerID
Schema	Integer

Validate your inputs with the below image:

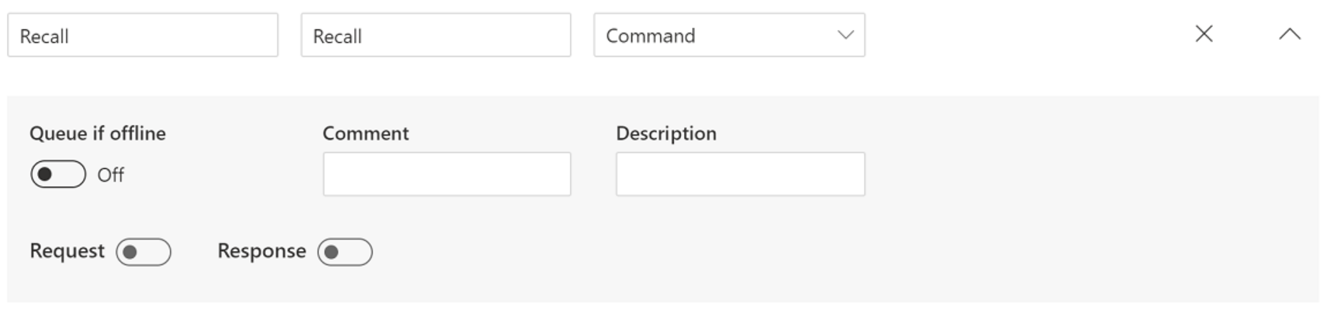


Go to customer	GoToCustomer	Command
Queue if offline: Off	Comment:	Description:
Request: On		
Display name: Customer ID	Name *: CustomerID	Schema *: Integer
Comment:	Description:	
Response: Off		

2. Create a command to recall the truck.

Entry Summary	Value
Display Name	Recall
Name	Recall
Capability type	Command

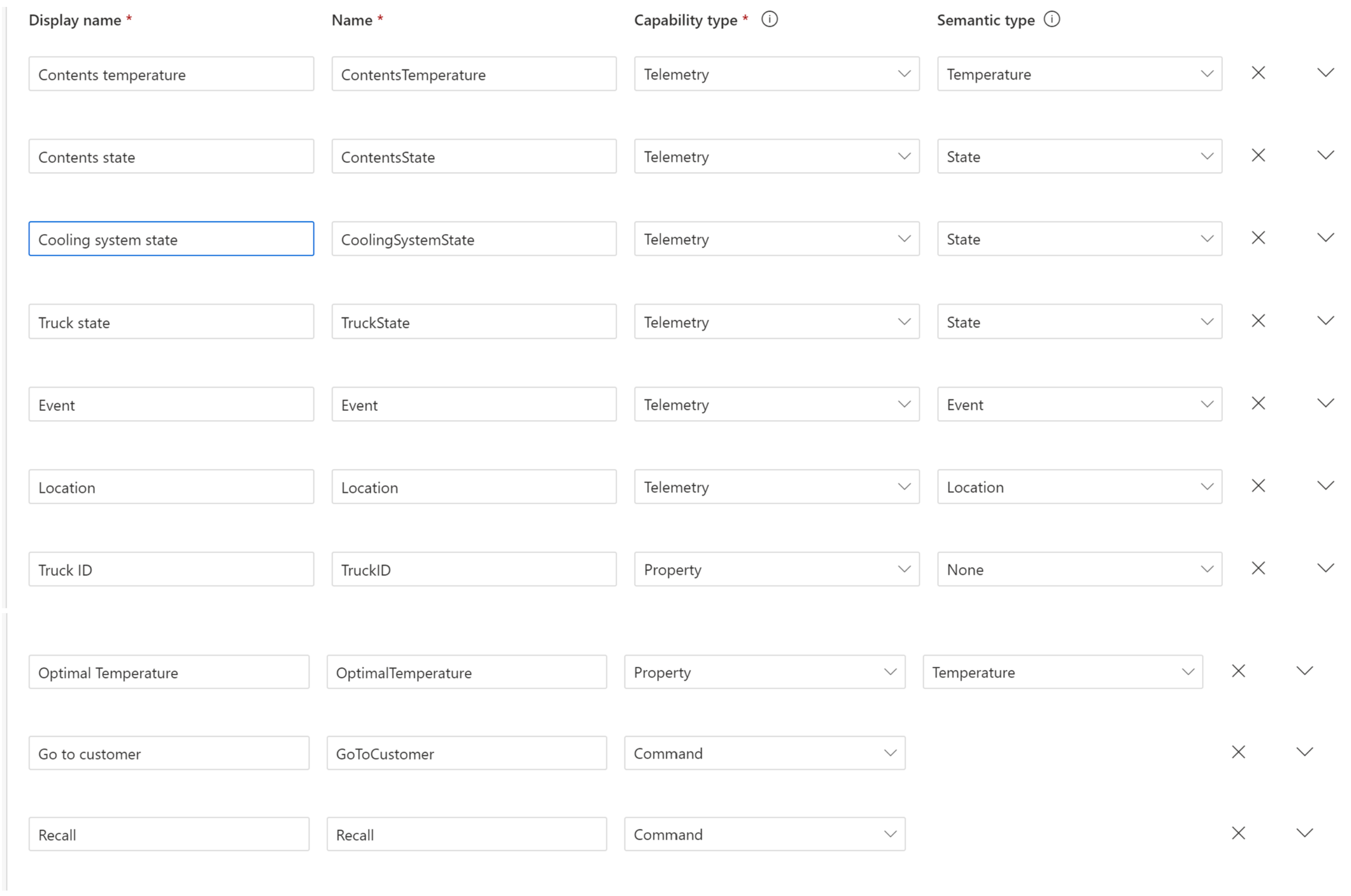
Your recall property should look like the below one:



Recall	Recall	Command
Queue if offline: Off	Comment:	Description:
Request: Off	Response: Off	

3. Select **Save**. Before you go any further, carefully double-check your interface. After an interface is published, editing options are limited. So you should get it right before publishing.

When you select the name of the device template, the menu that ends with the Views option summarizes the capabilities, 6 Telemetry based, 2 Properties and 2 Commands:



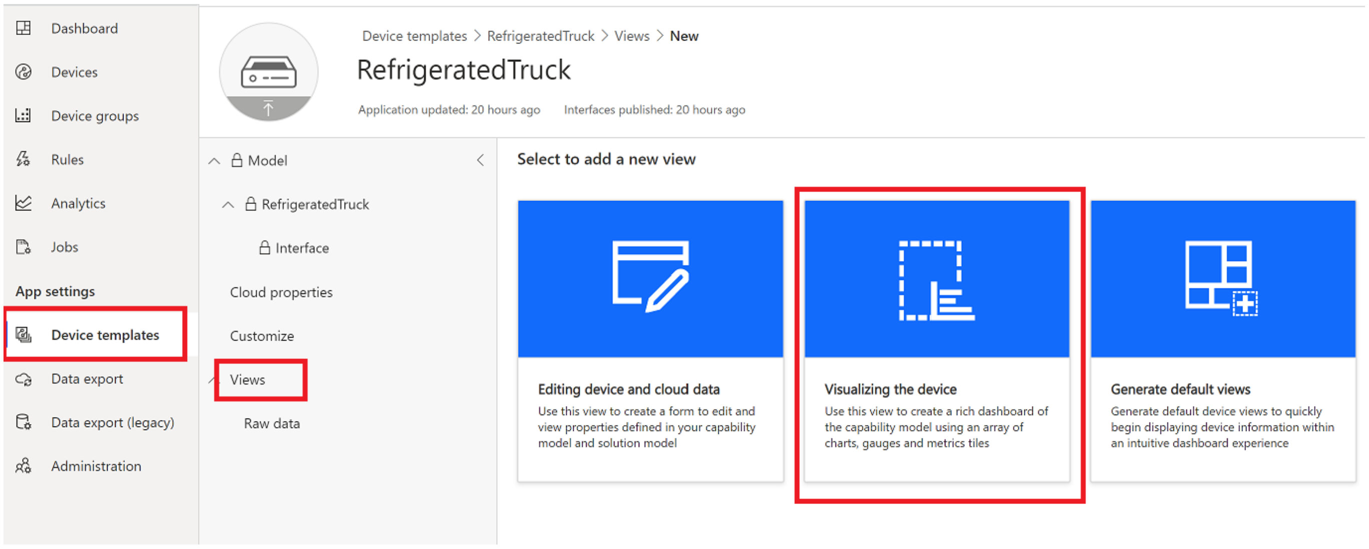
Display name *	Name *	Capability type *	Semantic type
Contents temperature	ContentsTemperature	Telemetry	Temperature
Contents state	ContentsState	Telemetry	State
Cooling system state	CoolingSystemState	Telemetry	State
Truck state	TruckState	Telemetry	State
Event	Event	Telemetry	Event
Location	Location	Telemetry	Location
Truck ID	TruckID	Property	None
Optimal Temperature	OptimalTemperature	Property	Temperature
Go to customer	GoToCustomer	Command	
Recall	Recall	Command	

4. Select **Publish**. Then in the dialog box, select **Publish** again. The annotation should change from Draft to Published.

Exercise 2: Create a Dashboard

Task 1: Visualizing the device

1. Select **Views**. Then select **Visualizing the device**. You see a list of all the Telemetry, Property, and Commands elements you created, each with a check box. You also see a list of Cloud properties and Custom tiles. Ignore these two lists for now.



Editing device and cloud data
Use this view to create a form to edit and view properties defined in your capability model and solution model

Visualizing the device
Use this view to create a rich dashboard of the capability model using an array of charts, gauges and metrics tiles

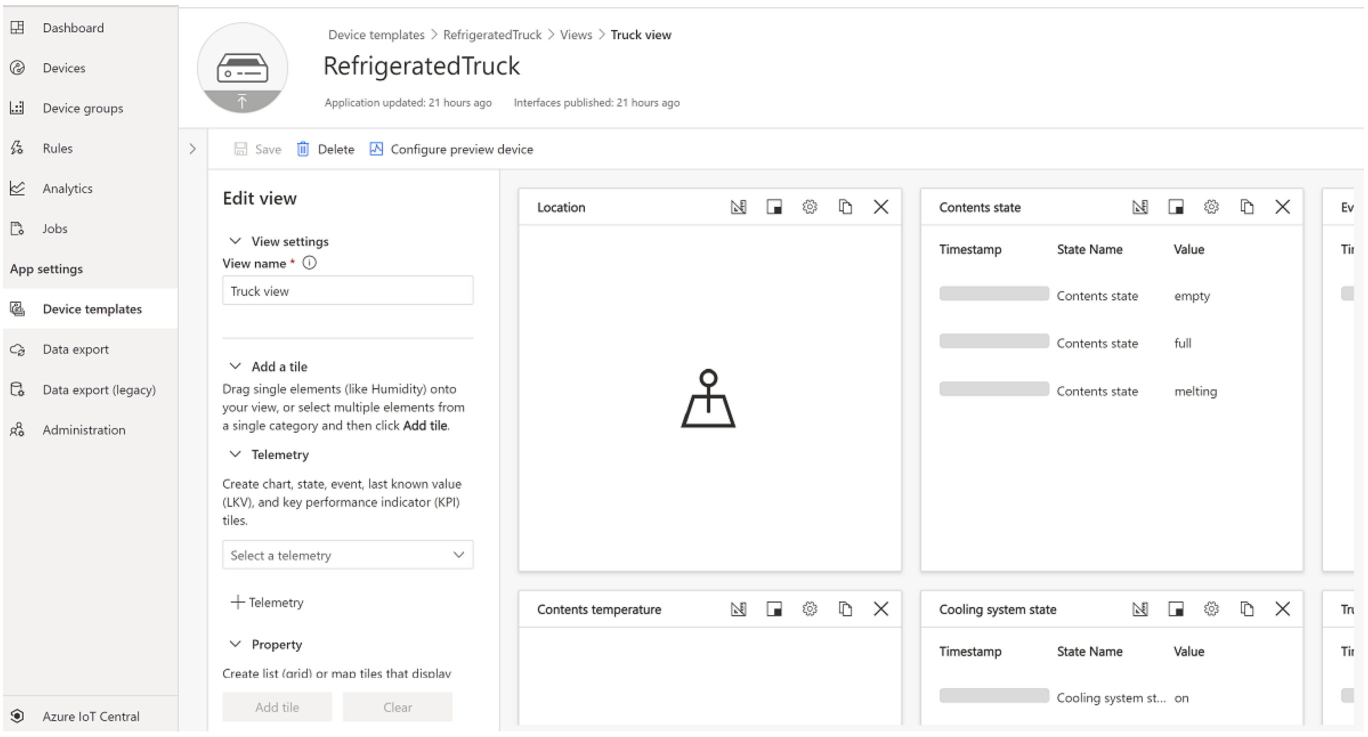
Generate default views
Generate default device views to quickly begin displaying device information within an intuitive dashboard experience

2. Under **Telemetry**, select **Location** > **Add tile**. Dashboards are made of tiles. We choose the location tile first because we want to expand it.

3. Change the View name to something more specific, for example, **Truck view**

4. Select each of the rest of the telemetry and property capabilities in turn, starting at the top. For each capability, select **Add tile**

Your new Dashboard should look like this one:



5. Select **Save** to save this view.

Task 2: Writable Properties View

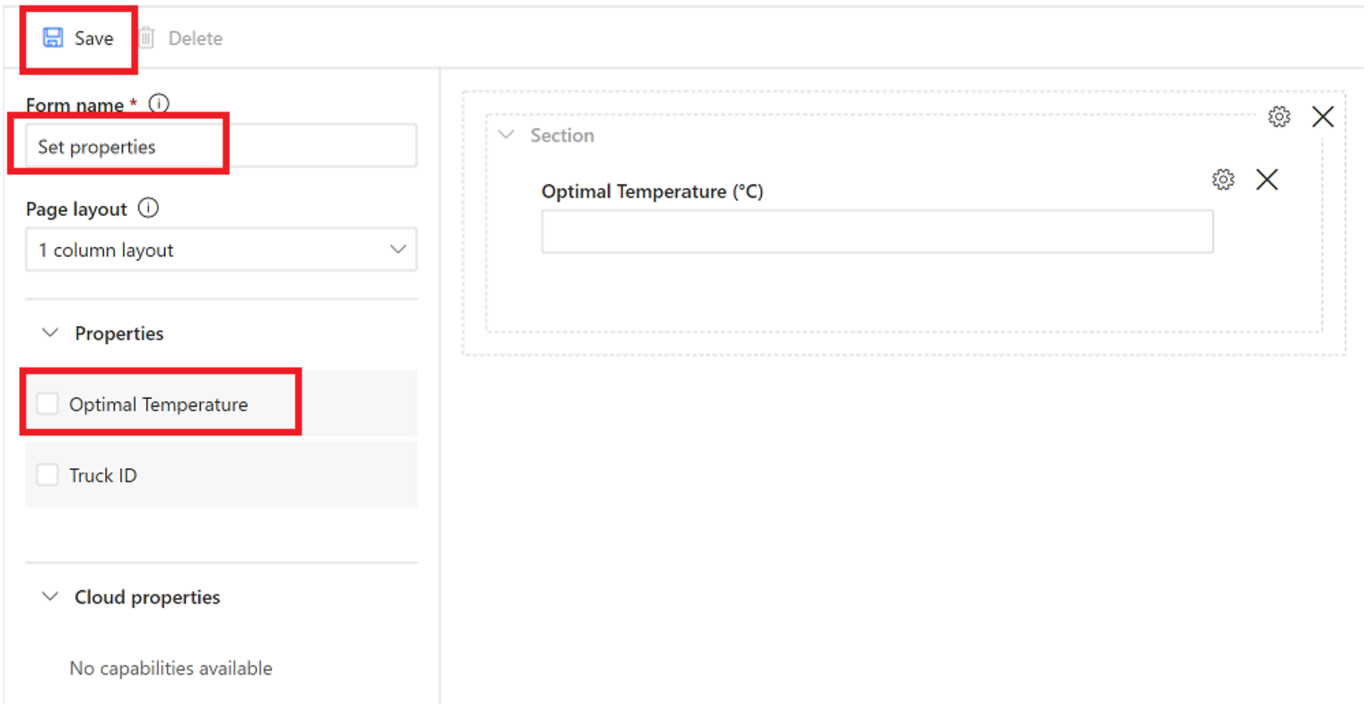
We need to create a separate view. Its sole purpose will be to set writable properties.

1. Select **Views**, and then select the **Editing device and cloud data tile**

2. Change the form name to something like **Set properties**.

3. Select the **Optimal temperature** property check box. Then select **Add section**.

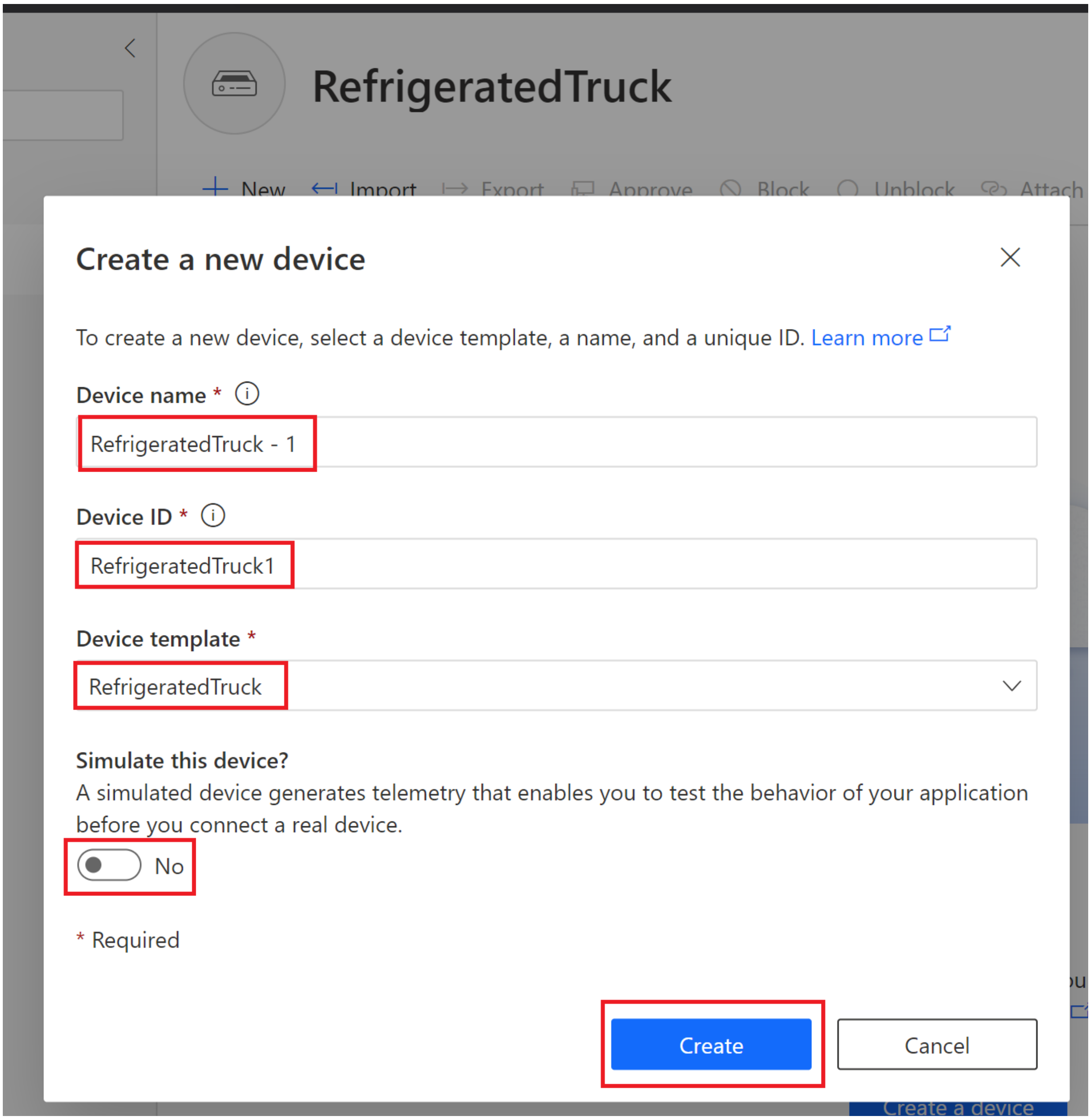
4. Verify that your view looks similar to the following image. Then select **Save**.



5. Select **Publish**. Then in the dialog box, select **Publish** again. The annotation should change from Draft to Published.

Task 3: Create a Device

1. On the menu on the left, select **Devices**.
2. To ensure the new device uses this device template, in the Devices menu, select **RefrigeratedTruck**.
3. Select **New**. In the Create a new device dialog box, verify that the device template is **RefrigeratedTruck**.
   - **Device name**: RefrigeratedTruck - 1
   - **Device ID**: RefrigeratedTruck1
   - **Simulate this device?**: setting at No



Create a new device

To create a new device, select a device template, a name, and a unique ID. Learn more

Device name *
RefrigeratedTruck - 1

Device ID *
RefrigeratedTruck1

Device template *
RefrigeratedTruck

Simulate this device?
A simulated device generates telemetry that enables you to test the behavior of your application before you connect a real device.
No

* Required

4. Then click **Create**

Notice that the Device status is **Registered**. Only after the device status is **Provisioned** will the IoT Central app accept a connection to the device. The coding unit that follows shows how to provision a device.

5. Select RefrigeratedTruck -1. You see the live dashboard. It includes lots of Waiting for data messages. On the bar that includes Truck view, select Commands. Notice that the two commands you entered are ready to run.

6. Record the connection keys. In the upper-right menu, select **Connect**. Do not select **Attach to gateway**.

In the Device connection dialog box that opens, carefully copy the **ID scope**, **Device ID**, and P**rimary key**. The ID scope identifies the app. The device ID identifies the real device. And the primary key gives you permission for the connection.

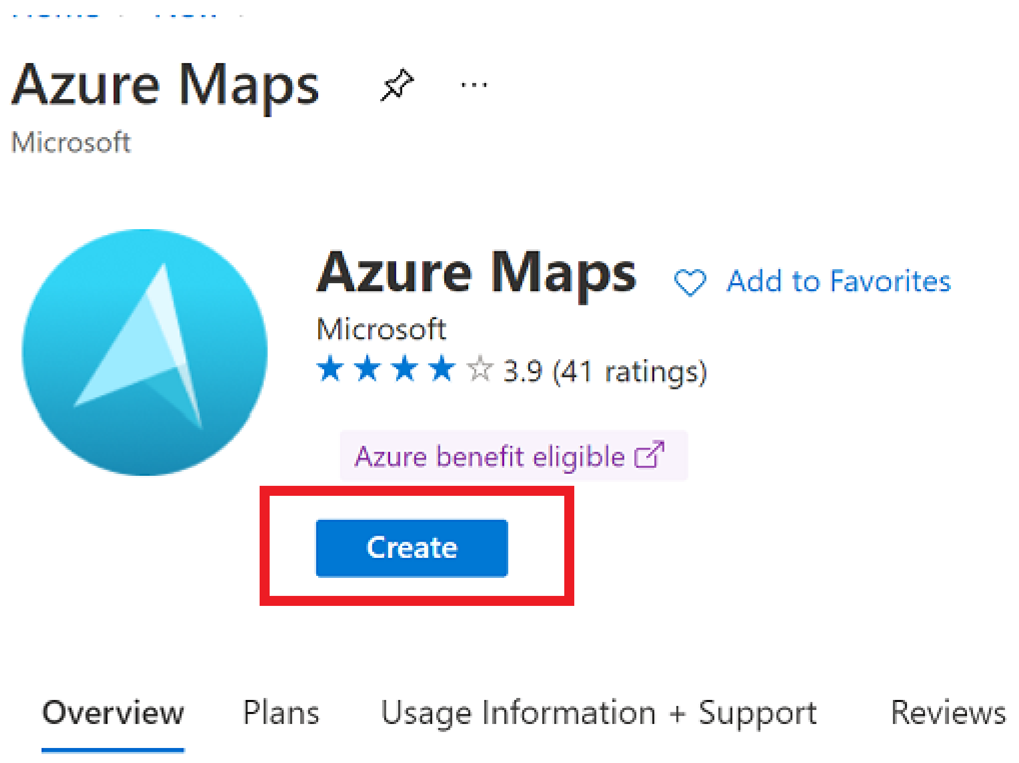
Paste this information in a text file.

Leave the Authentication type setting as **Shared access signature (SAS)**.

After you save the IDs and the key, select Close on the dialog box.

Exercise 3: Azure Maps

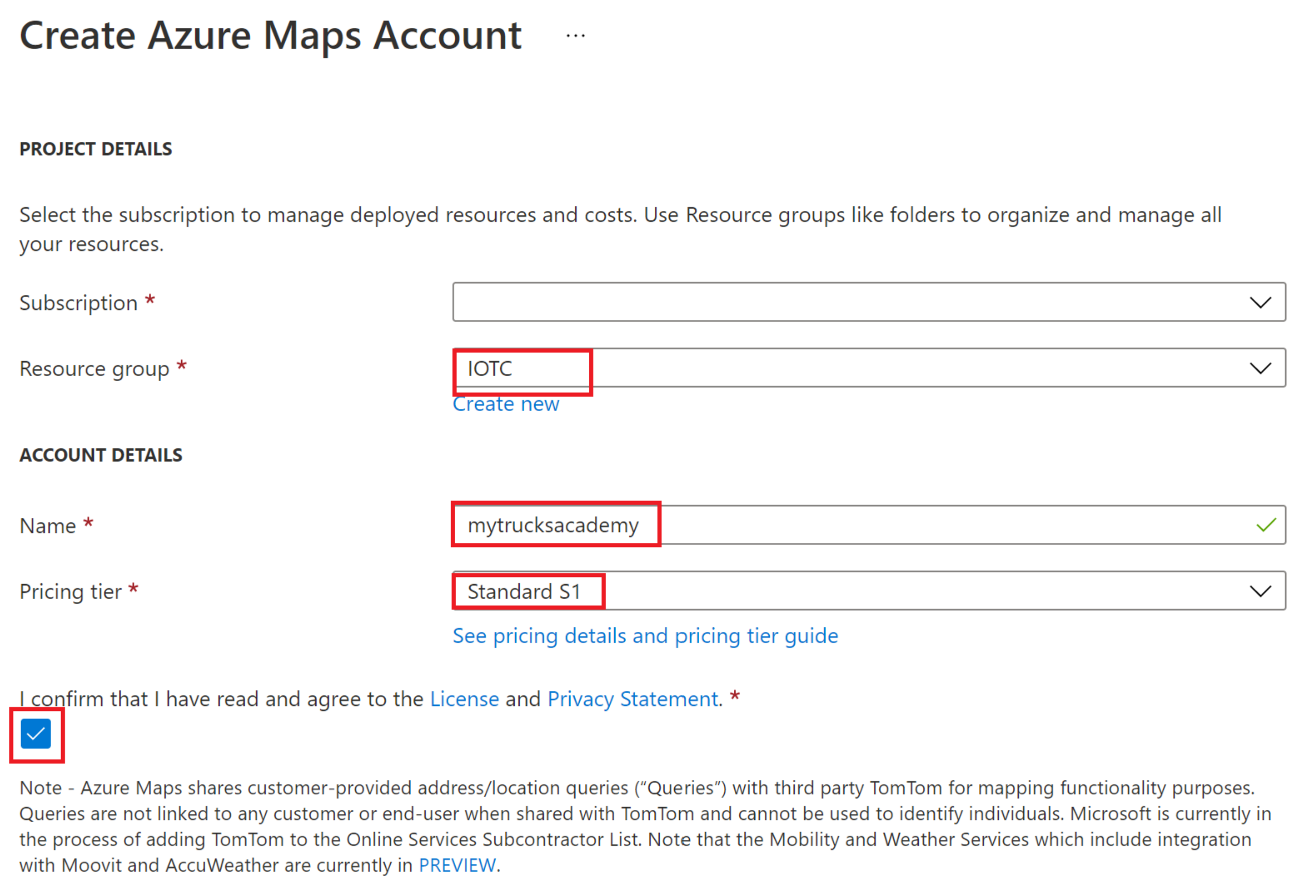
1. Go to Azure Portal: https://ms.portal.azure.com/
2. Select **Create a Resource**, in the marketplace look for **Azure Maps**, select Azure Maps and then click **create**



Complete the creation form:

- **Subscription**: Select the subscription you are using for this training.
- **Resource Group**: Select the resource group you are using for this training. -**Name**: mytrucksacademySUFFIX -**Pricing Tier**: Standard S1 -**Confirm the license and Privacy terms** make sure it is check.

Then click **Create** at the bottom of the page.



Create Azure Maps Account ...

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use Resource groups like folders to organize and manage all your resources.

Subscription *	
Resource group *	IOTC
	Create new

ACCOUNT DETAILS

Name *	mytrucksacademy
Pricing tier *	Standard S1
	See pricing details and pricing tier guide

I confirm that I have read and agree to the License and Privacy Statement. * ✓

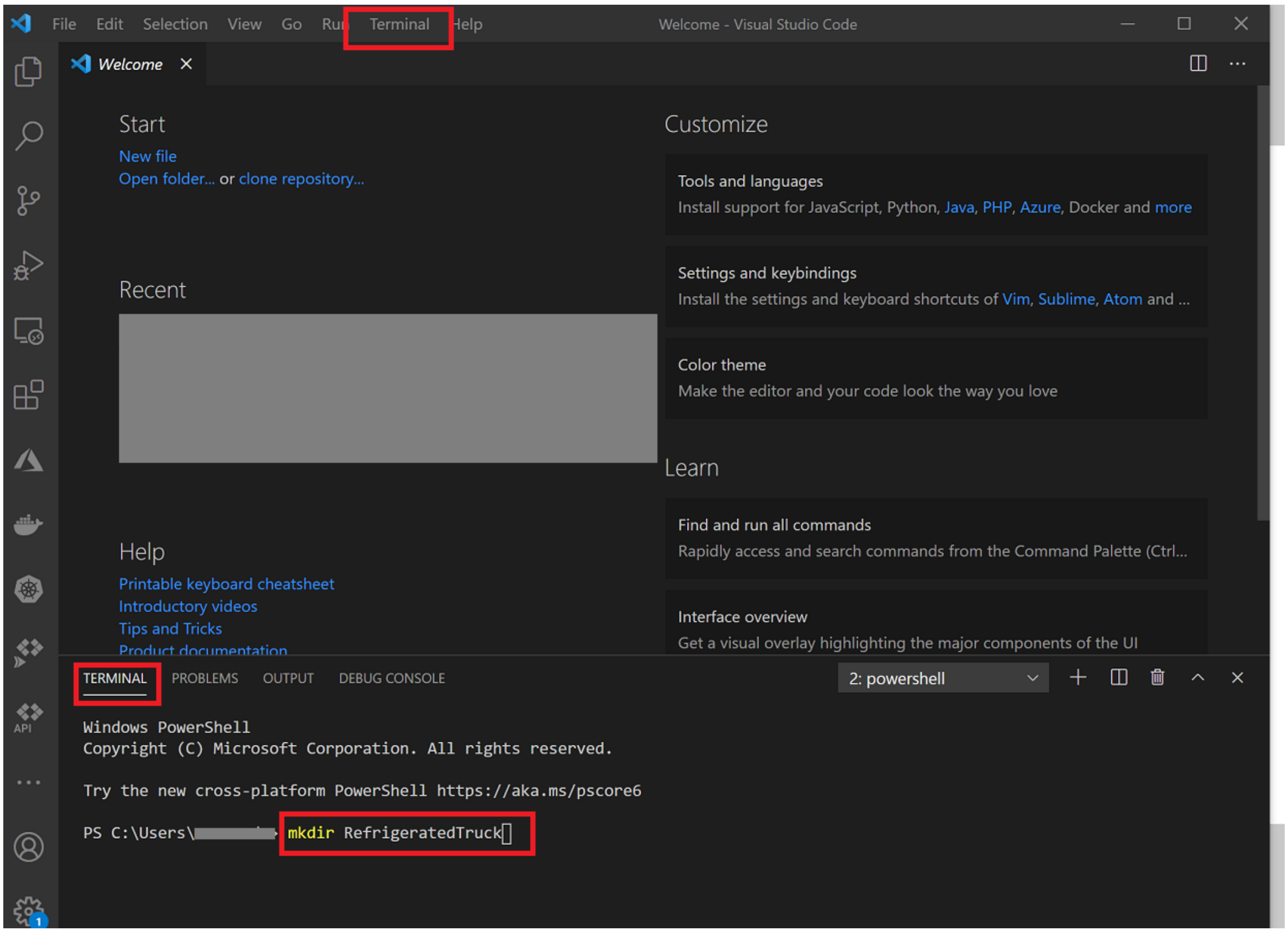
Note - Azure Maps shares customer-provided address/location queries ("Queries") with third party TomTom for mapping functionality purposes. Queries are not linked to any customer or end-user when shared with TomTom and cannot be used to identify individuals. Microsoft is currently in the process of adding TomTom to the Online Services Subcontractor List. Note that the Mobility and Weather Services which include integration with Moovit and AccuWeather are currently in PREVIEW.

Once Azure Maps resource is created, find the key by selecting **Authentication**. Copy the **primary key** and paste it into your notepad.

Exercise 4: Create the device app

Task 1: Set up your environment

1. Open Visual Studio Code locally
2. On the top bar select **Terminal** and then **New Terminal** in Visual Studio Code.
3. Create a folder called RefrigeratedTruck by entering **mkdir RefrigeratedTruck** and then enter. Go to the folder by entering **cd RefrigeratedTruck**.



Start
New file
Open folder... or clone repository...

Recent
[redacted]

Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation

Customize

Tools and languages
Install support for JavaScript, Python, Java, PHP, Azure, Docker and more

Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and ...

Color theme
Make the editor and your code look the way you love

Learn

Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl...

Interface overview
Get a visual overlay highlighting the major components of the UI

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\[redacted]> mkdir RefrigeratedTruck
```

4. Enter **dotnet new console**. This command creates a Program.cs file and a project file in your folder.

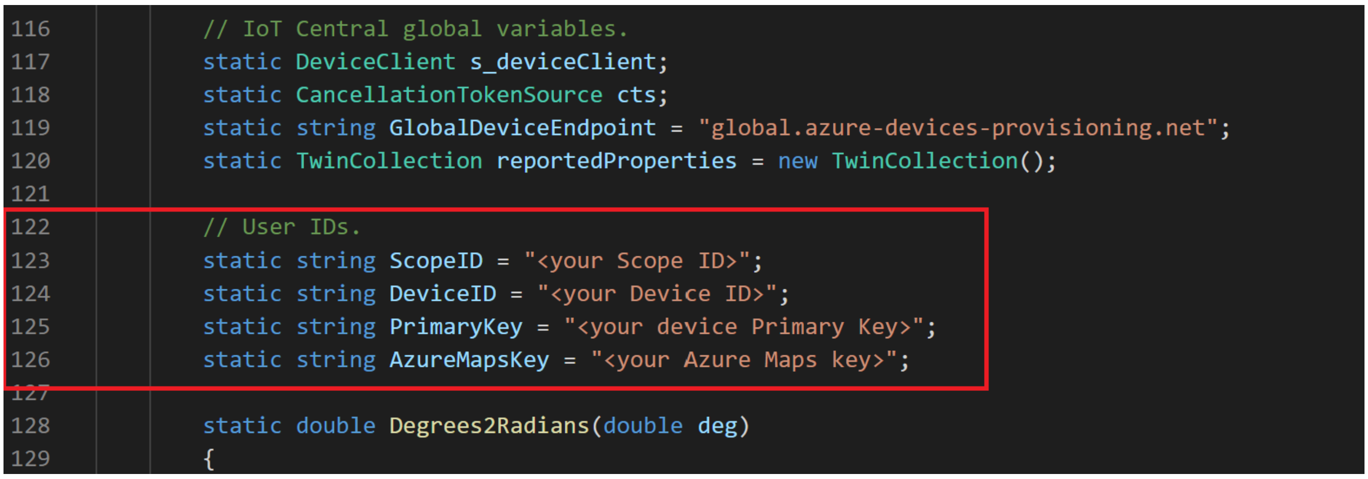
5. Enter **dotnet restore**. This command gives your app access to the required .NET packages.

Install the required libraries, copy and paste the below code in the terminal.

```
dotnet add package AzureMapsRestToolkit
dotnet add package Microsoft.Azure.Devices.Client
dotnet add package Microsoft.Azure.Devices.Provisioning.Client
dotnet add package Microsoft.Azure.Devices.Provisioning.Transport.Mqtt
dotnet add package System.Text.Json
```

6. From the File menu, open the Program.cs file just created. Then replace the whole content copying and pasting from the file **Program.cs** you will find in **code_sample** folder.

7. Once you replace the content of the files, we need to add our keys to connect with our services. Look for lines **123** to **126**. Replace accordingly based on the keys you were adding to your notepad in previous exercises.



```
        // IoT Central global variables.
        static DeviceClient s_deviceClient;
        static CancellationTokenSource cts;
        static string GlobalDeviceEndpoint = "global.azure-devices-provisioning.net";
        static TwinCollection reportedProperties = new TwinCollection();

        // User IDs.
        static string ScopeID = "<your Scope ID>";
        static string DeviceID = "<your Device ID>";
        static string PrimaryKey = "<your device Primary Key>";
        static string AzureMapsKey = "<your Azure Maps key>";

        static double Degrees2Radians(double deg)
        {
```

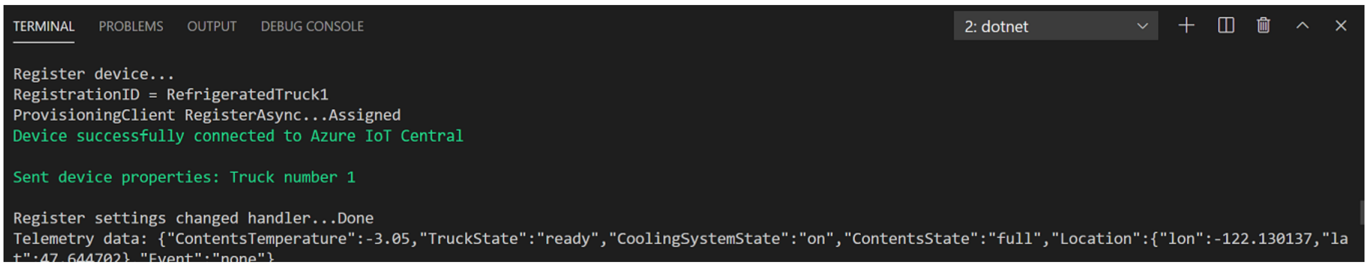
After the changes are made, save the file with **Ctrl+S**

Task 2: Launch your device

To begin testing, open the Azure IoT Central app in a browser. Then run the device app.

1. In the terminal, enter **dotnet run**.

A console screen opens with the message Starting Truck number 1.

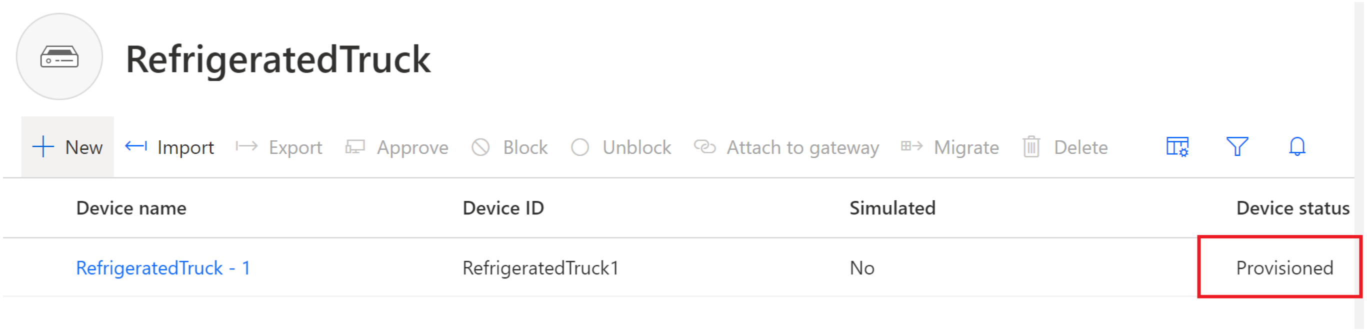


```
Register device...
RegistrationID = RefrigeratedTruck1
ProvisioningClient RegisterAsync...Assigned
Device successfully connected to Azure IoT Central

Sent device properties: Truck number 1

Register settings changed handler...Done
Telemetry data: {"ContentsTemperature":-3.05,"TruckState":"ready","CoolingSystemState":"on","ContentsState":"full","Location":{"lon":-122.130137,"la
t":47.644702},"Event":"none"}
```

Once your device in registered through VS Code, you should see in your IoT Central an status change to **Provisioned**:



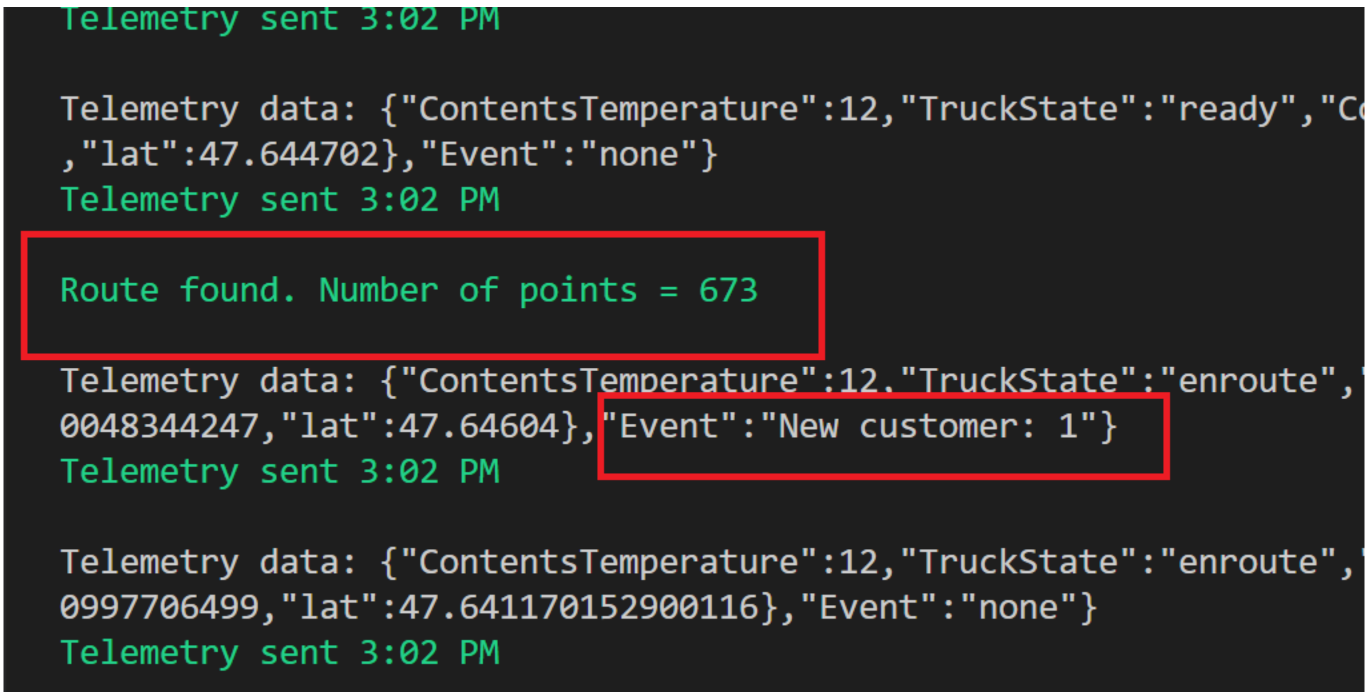
Device name	Device ID	Simulated	Device status
RefrigeratedTruck - 1	RefrigeratedTruck1	No	Provisioned

At this point in the Track View dashboard you should see data flowing thorught it, the map should show a blue dot with your truck and the chart receiving telemetry data should show some data points already.

2. Select the device's **Commands** tab. This control should be under the truck name, to the right of the Truck view control.

3. Enter a customer ID, say **1**. (Numerals 0 through 9 are valid customer IDs.) Then select **Run**.

In the console for the device app, you see both a New customer event and a Route found message



```
Telemetry sent 3:02 PM

Telemetry data: {"ContentsTemperature":12,"TruckState":"ready","C
,"lat":47.644702},"Event":"none"}
Telemetry sent 3:02 PM

Route found. Number of points = 673

Telemetry data: {"ContentsTemperature":12,"TruckState":"enroute",
0048344247,"lat":47.64604},"Event":"New customer: 1"}
Telemetry sent 3:02 PM

Telemetry data: {"ContentsTemperature":12,"TruckState":"enroute",
0997706499,"lat":47.641170152900116},"Event":"none"}
Telemetry sent 3:02 PM
```

4. On the dashboard's Location tile, check to see whether your truck is on its way. You might have to wait a short time for the apps to sync.

5. Verify the event text on the Event tile. You should see a new Customer Event.

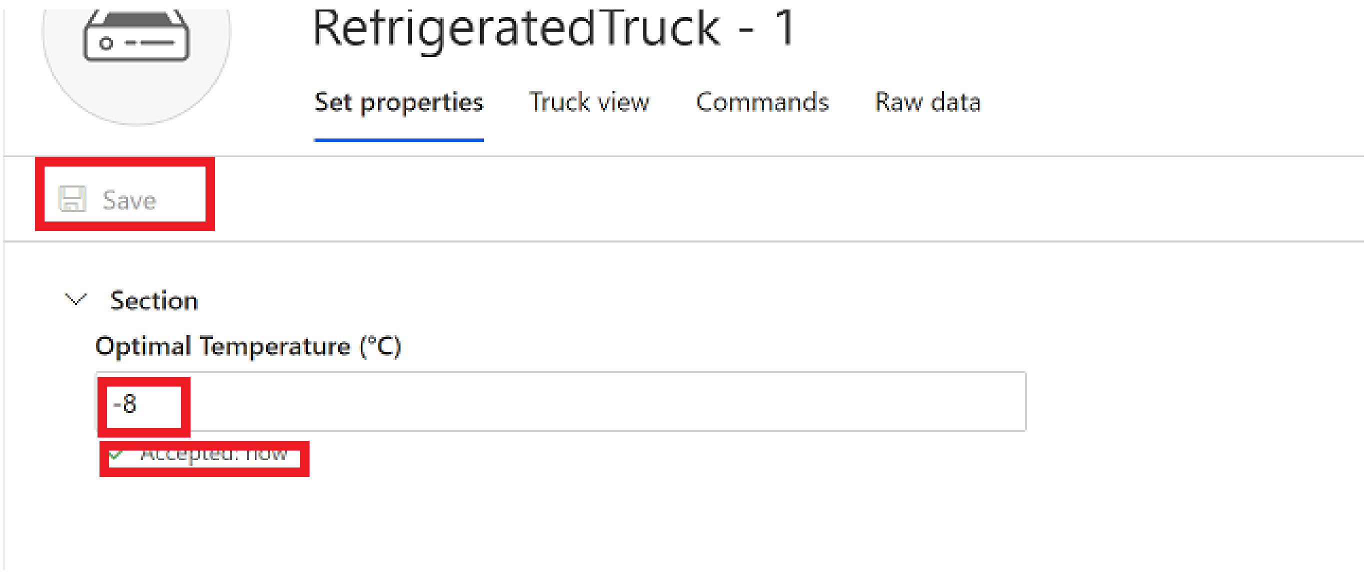
6. When the truck returns to base and is reloaded with contents, its state is ready. Try issuing another delivery command. Choose another customer ID.

7. Before the truck reaches the customer, make a recall command to check whether the truck responds.

Task 3: Set up Properties

The next test is to check the writable property, **OptimalTemperature**. To change this value, select the **Set properties** view.

Set the optimal temperature to any value, say **-8**. Select **Save** and then notice the Pending status.



RefrigeratedTruck - 1

Optimal Temperature (°C)

-8

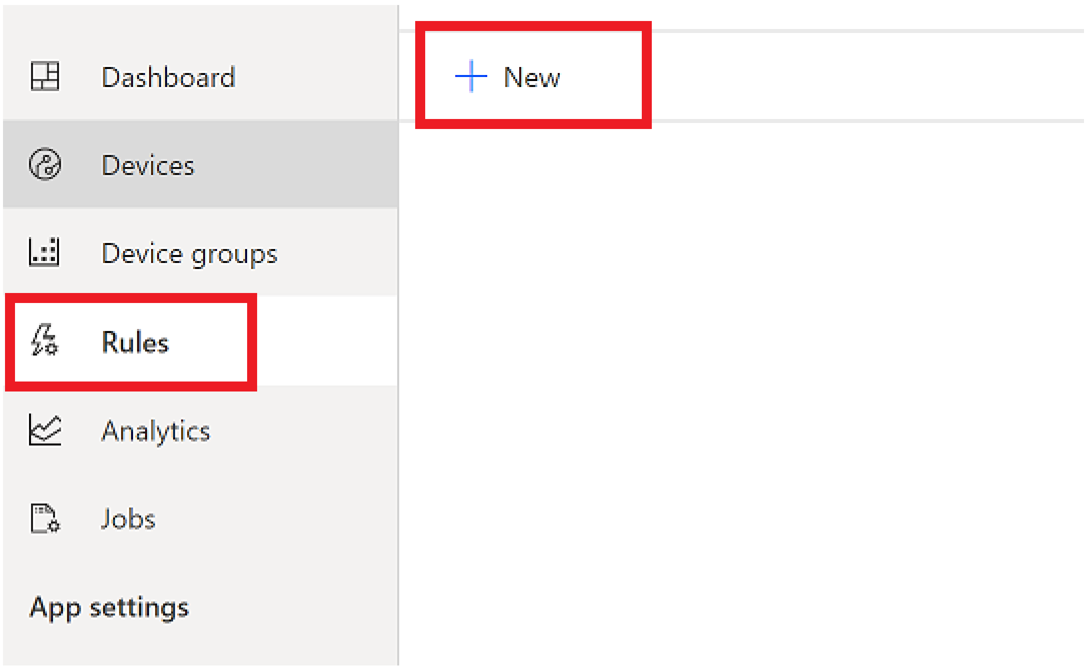
✓ Accepted: now

Now you should see the new Optimal temperature is set to -8. in the **Optimal Temperature** Tile.

Exercise 5: Create Rules

Task 1: Cooling system state

1. In the IoT portal, select **Rules** in the left-hand menu, then **+ New**. Enter a meaningful name for the rule, such as **"Cooling system failed"**. Press Enter.



2. Select **RefrigeratedTruck** for the **device template**.

3. Under **Conditions** notice that all the telemetry elements of the device template are available. Select **Cooling system state**.

For Operator, select **Equals**.

For value, type the word **"failed"**, then click on Select: "failed".

Leave Time aggregation as Off.

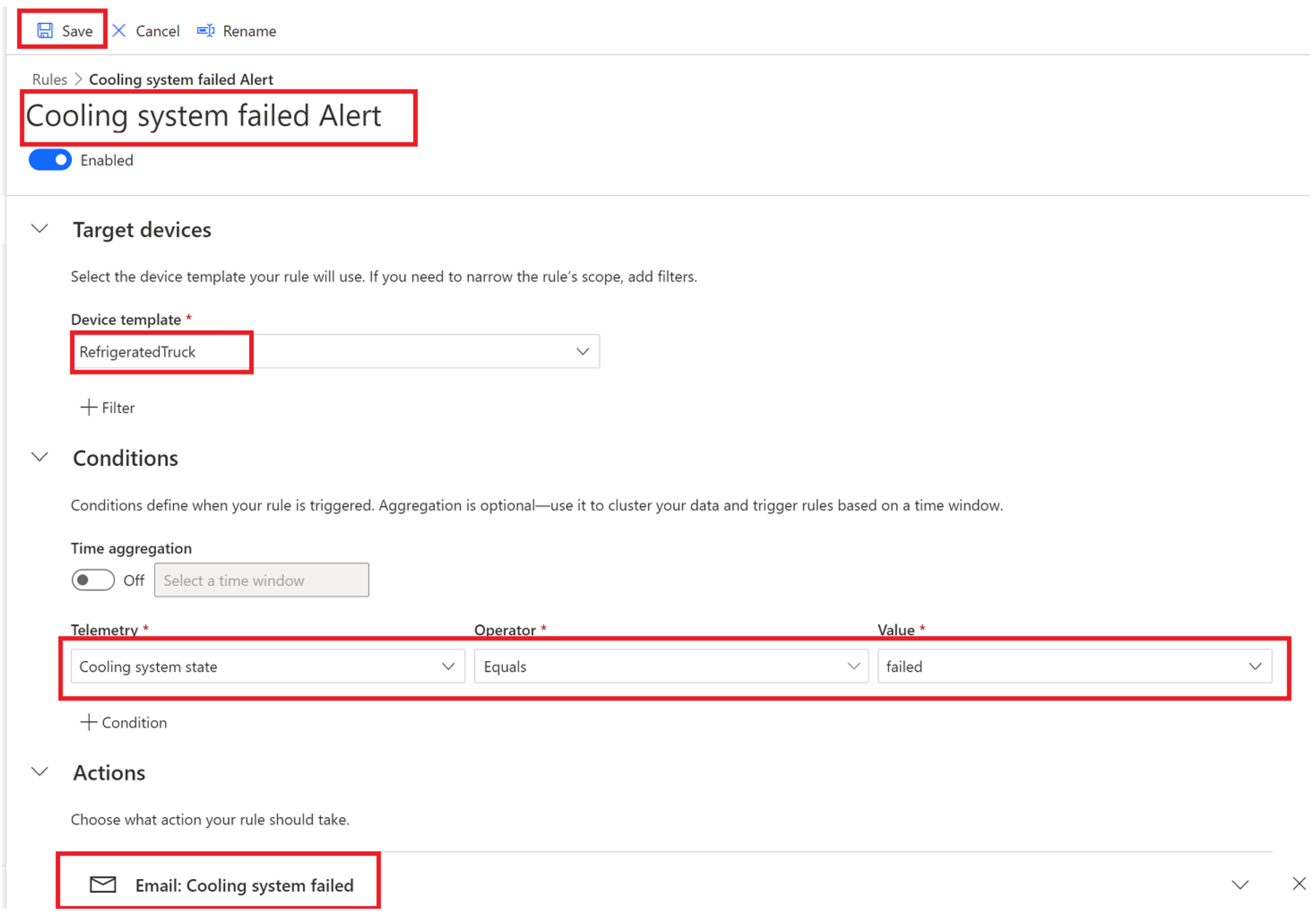
For **Actions**, click on **+ Email**.

In Display name, enter a title for the email, perhaps "Cooling system failed"!

For To, enter the email you've used for your IoT Central account. And for Note enter some descriptive text that will form the body of the email.

Note: To receive emails the account you select has to be login to IoT central at least one time, otherwise you will not receive any emails.

Your new rule should look like the below image.



Rules > Cooling system failed Alert

Cooling system failed Alert

Enabled

Target devices

Select the device template your rule will use. If you need to narrow the rule's scope, add filters.

Device template *: RefrigeratedTruck

+ Filter

Conditions

Conditions define when your rule is triggered. Aggregation is optional—use it to cluster your data and trigger rules based on a time window.

Time aggregation: Off, Select a time window

Telemetry *	Operator *	Value *
Cooling system state	Equals	failed

+ Condition

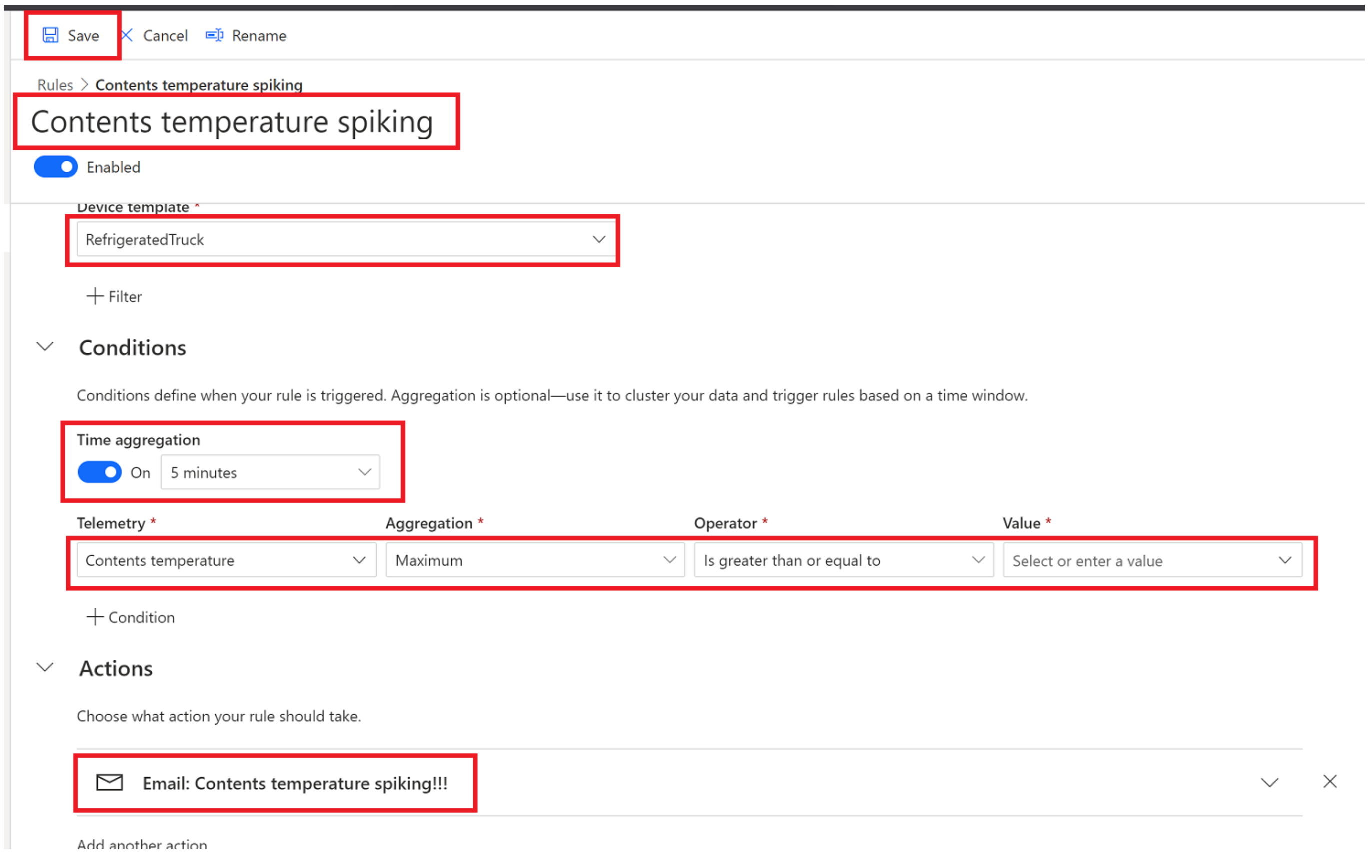
Actions

Choose what action your rule should take.

Email: Cooling system failed

Task 2: Temperature spiking

1. Create a new rule with a name such as **"Contents temperature spiking"**.
2. Turn on **Time aggregation**, and select an interval of **5 minutes**.
3. Select **Contents Temperature** for Telemetry.
4. In the range Aggregation values, select **Maximum.**
5. For Operator. select Is greater than or equal to. Then enter **"0"** for the value, and select that as the value.
6. For Actions, fire off another email. Give the email an appropriate title and note.
7. Make sure to click Save, to save off this rule.


Save Cancel Rename

Rules > Contents temperature spiking

Contents temperature spiking

Enabled

Device template *

RefrigeratedTruck

+ Filter

Conditions

Conditions define when your rule is triggered. Aggregation is optional—use it to cluster your data and trigger rules based on a time window.

Time aggregation

On 5 minutes

Telemetry *	Aggregation *	Operator *	Value *
Contents temperature	Maximum	Is greater than or equal to	Select or enter a value

+ Condition

Actions

Choose what action your rule should take.

Email: Contents temperature spiking!!!

Add another action

Task 3: Truck leaves base

1. Select **Rules** in the left-hand menu, then **+ New**. Enter a meaningful name for the rule, such as **"Truck leaving base"**. Press Enter.

Now, enter the following five conditions.

- Location / Latitude: doesn't equal => **47.644702**
- Location / Longitude: doesn't equal => **-122.130137**
- Truck state: Equals => **enroute**

Task 4: Temperature of the contents

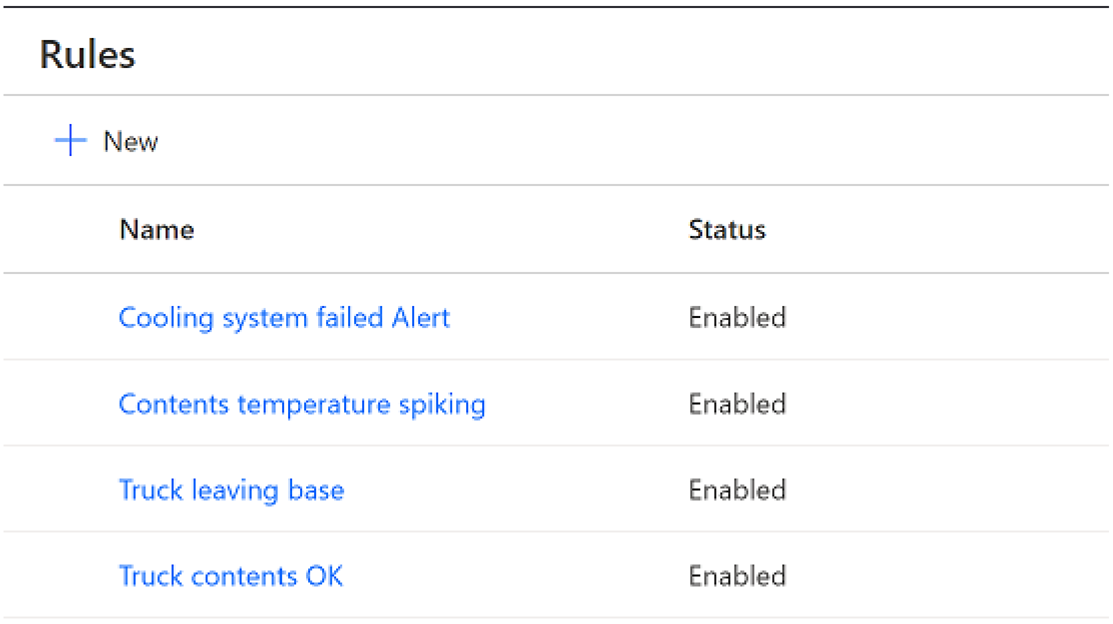
1. Enter a rule with a name such as **"Truck contents OK"**.

2. Turn on Time aggregation, with a period of **five minutes**.

3. Enter conditions that fire if the average Contents Temperature is less than **-1** degrees Celsius, and greater than **-18** degrees Celsius.

4. Again, enter an appropriate **email action**, and click **Save**.

At this point you should see all the rules listed as below:


Rules

Name	Status
Cooling system failed Alert	Enabled
Contents temperature spiking	Enabled
Truck leaving base	Enabled
Truck contents OK	Enabled

At this point it is time to test your Rules Go to your Device Dashboard, sent a Command to trigger a new Customer trip, remember use numbers from 1 to 9. In a few minutes you should start receiving emails.

Exercise 6: Clean up

Once you completed all the exercises, go to Azure Portal, look for the azure IoT Central Application and delete resource.

Resources Needed

- Azure IoT Central
- VS Code