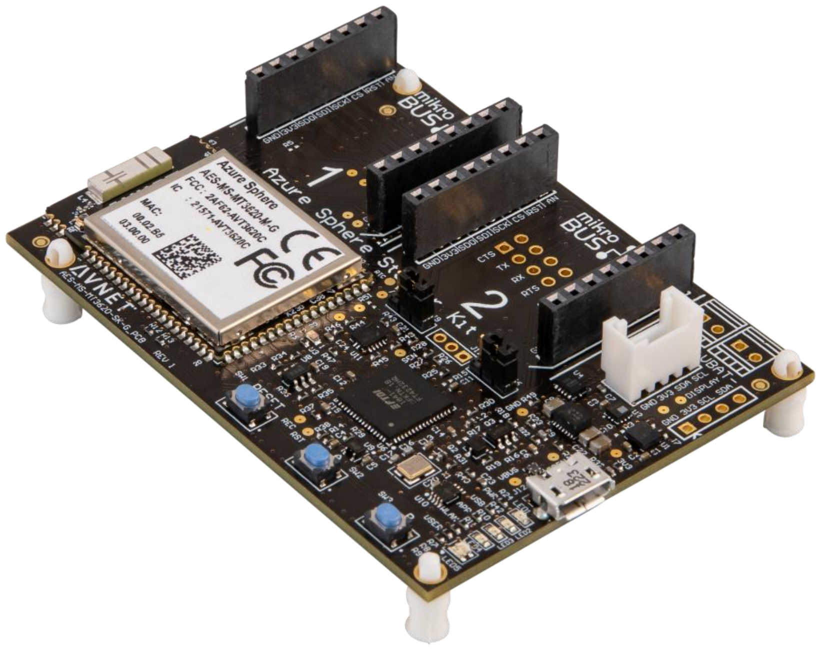
Avnet Technical Training Course

Azure Sphere: Accelerating your Cloud Dashboard Implementation

Lab 5



Azure Sphere SDK:	19.05
Training Version:	v1
Date:	1 July 2019

Introduction

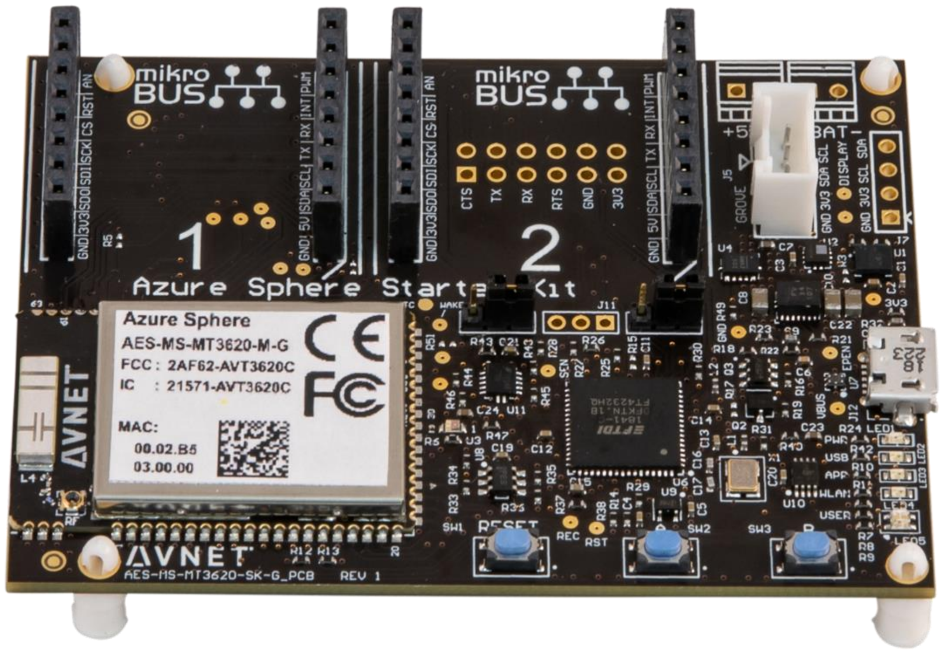
This Lab will walk the student through connecting the example application to an IoT Central application. We'll use a template that I've prepared to make creating the IoT Central application fast and easy. I'll provide a coding assignment to help reinforce some of the Azure Sphere coding that we've covered in the course. We will then modify the IoT Central application display the new item we create with the coding assignment.

Avnet Azure Sphere Starter Kit Overview

The Avnet Azure Sphere Starter Kit from Avnet Electronics Marketing provides engineers with a complete system for prototyping and evaluating systems based on the MT3620 Azure Sphere device.

The Avnet Azure Sphere MT3620 Starter Kit supports rapid prototyping of highly secure, end-to-end IoT implementations using Microsoft's Azure Sphere. The small form-factor carrier board includes a production-ready MT3620 Sphere module with Wi-Fi connectivity, along with multiple expansion interfaces for easy integration of off-the-shelf sensors, displays, motors, relays, and more.

The Starter Kit includes Avnet's MT3620 Module. Having the module on the Starter Kit means that you can do all your development work for your IoT project on the Starter Kit and then easily migrate your Azure Sphere Application to your custom hardware design using Avnet's MT3620 Module.



Avnet Azure Sphere Starter Kit

Lab 5: Objectives

The Lab-5 objectives are to teach the student how to stand up an IoT Central application and send telemetry data from the Azure Sphere device to the cloud so we can visualize the telemetry data.

- Learn how to create an IoT Central application from a template
- Learn how to provision our device to the IoT Central application
- Configure the example application for the IoT Central configuration
- Complete a code assignment
- Modify the IoT Central Application to display the new item created in the coding assignment

Lab-5 builds on the previous labs and should not be started until Labs 0-3 have been completed.

Requirements

Hardware

- A PC running Windows 10 Anniversary Update or later (Version 1607 or greater)
- An unused USB port on the PC
- An Avnet Azure Sphere Starter Kit
- A micro USB cable to connect the Starter Kit to your PC

Software

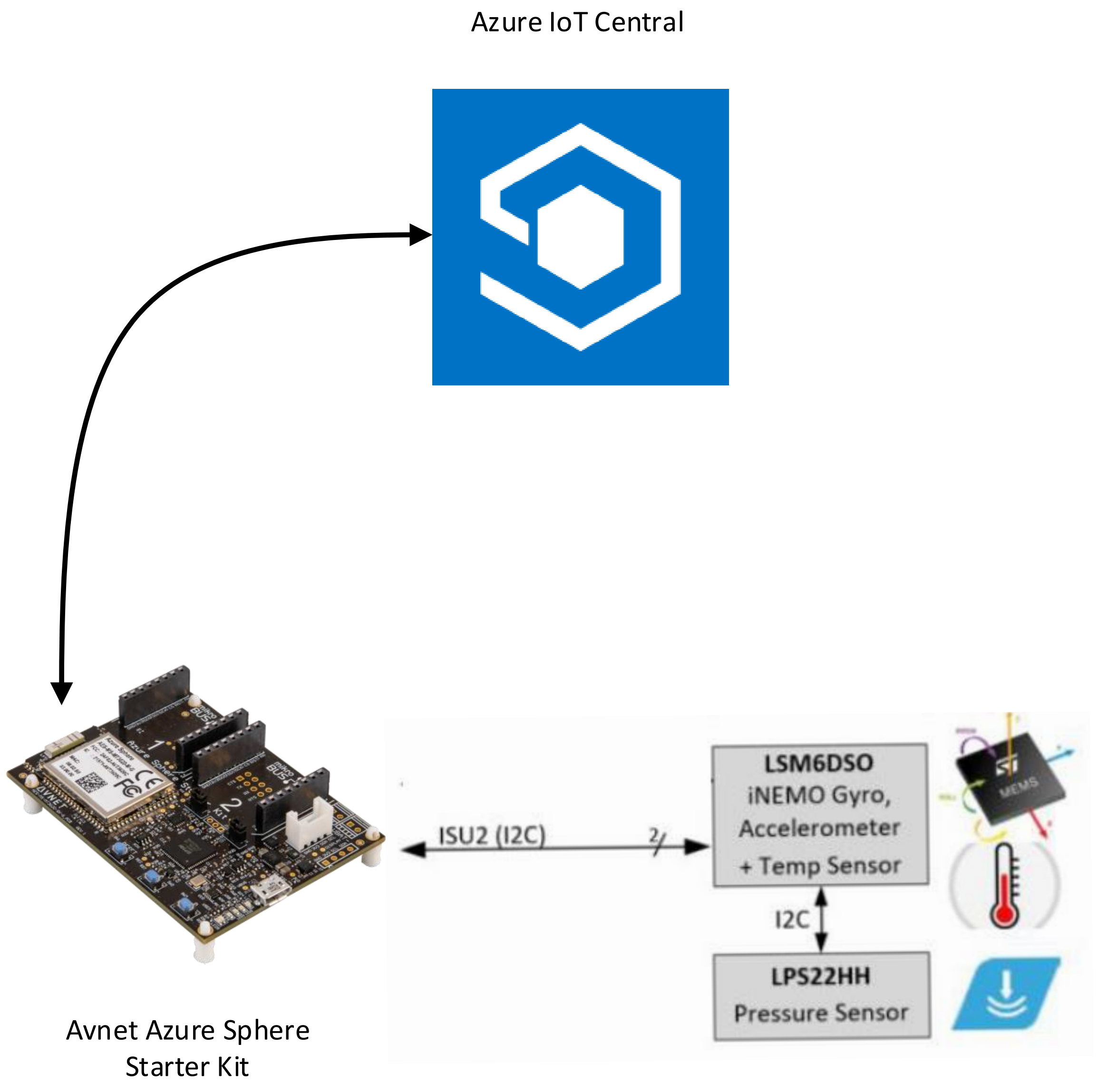
- Visual Studio 2019 Enterprise, Professional, or Community version 16.04 or later; or Visual Studio 2017 version 15.9 or later **installed**
- Azure Sphere SDK 19.05 or the current SDK release **installed**

Other

- Your Azure Sphere device must be connected to a Wi-Fi access point or hotspot with access to the internet.
- Labs 4 and 5 both require an Azure Account. You can sign up for a free Azure account with a $200 credit here.

The Big Picture

The diagram below shows the system we'll be building out in this lab. Starting at the bottom right of the graphic, we'll use the on-board I2C sensors to read accelerometer data. That data will be sent as telemetry to an IoT Central application.



Create an IoT Central Application

Azure IoT Central

IoT Central is a Software-as-a-Service (SaaS) offering from Microsoft. Microsoft has made it easy to connect IoT devices and display telemetry data from connected devices. Additionally, there are some cloud to device (C2D) controls available. IoT Central Applications are created on-line without any coding necessary; all IoT Central configuration is driven from the web interface.

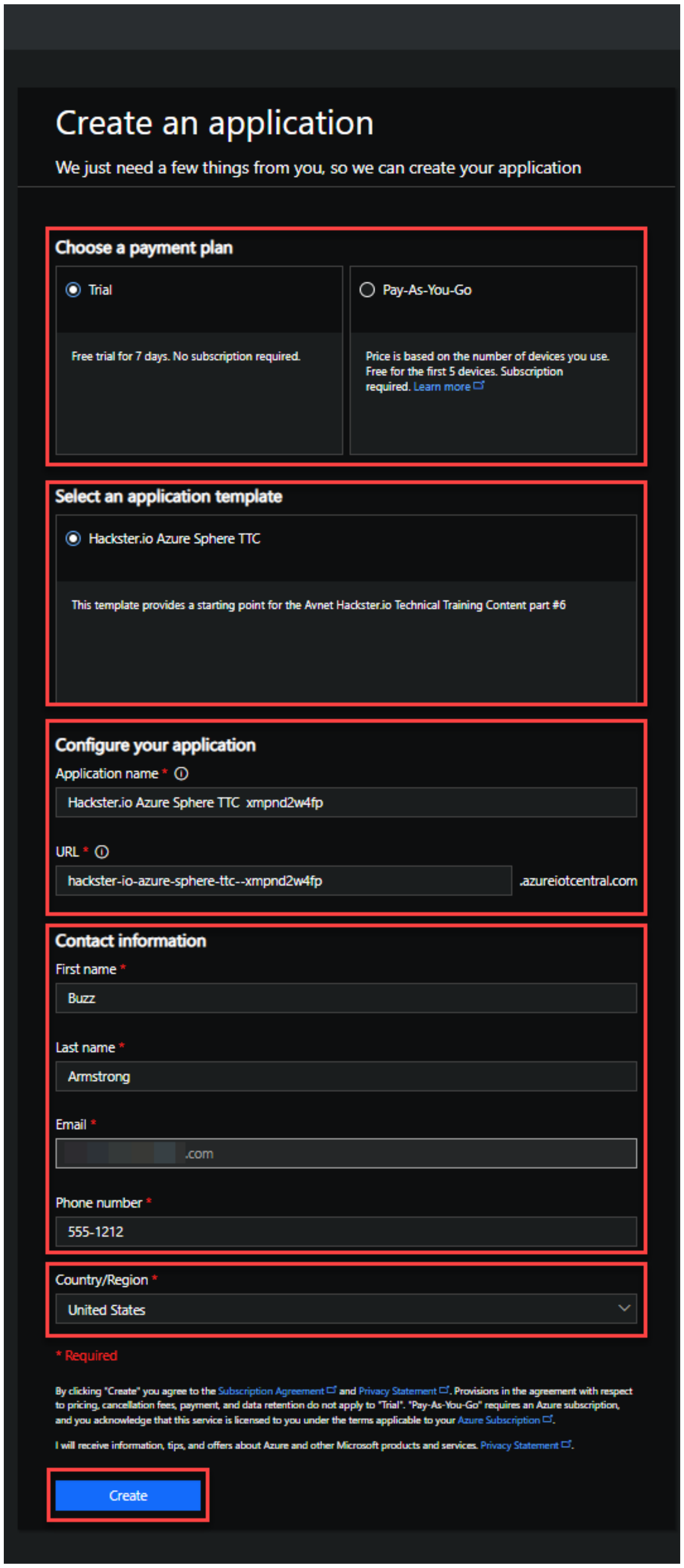
Create IoT Central Application

Recently Microsoft released a new IoT Central feature called “Application Template Export” for IoT central. This is a really nice feature that allows you to basically clone an IoT Central application that someone else created. We’ll use this feature for the lab, but I also wanted to provide a link that includes all the details for creating an IoT Central application from scratch. The process is very simple, and is fully documented here. Additionally, there is a blog on Element14.com that includes all the details for creating the IoT Central Application that I used to create the template that we’ll be using today: here.

Open the Template

The template that we’ll use is here. When you click on the link, a browser window will open like the one shown on the next page. If you get an error with the link above, try to copy and paste the URL below directly into your browser window.

https://apps.azureiotcentral.com/create?appTemplate=08eb9226-8bce-4204-852a-eb2f6942bd5f



Create an application

We just need a few things from you, so we can create your application

Choose a payment plan

[✓ Trial] Pay-As-You-Go

Trial: Free trial for 7 days. No subscription required.

Pay-As-You-Go: Price is based on the number of devices you use. Free for the first 5 devices. Subscription required. Learn more

Select an application template

[✓ Hackster.io Azure Sphere TTC]

This template provides a starting point for the Avnet Hackster.io Technical Training Content part #6

Configure your application

Application name *: Hackster.io Azure Sphere TTC xmpnd2w4fp

URL *: hackster-io-azure-sphere-ttc--xmpnd2w4fp .azureiotcentral.com

Contact information

First name *: Buzz

Last name *: Armstrong

Email *: [redacted].com

Phone number *: 555-1212

Country/Region *: United States

* Required

By clicking "Create" you agree to the Subscription Agreement and Privacy Statement. Provisions in the agreement with respect to pricing, cancellation fees, payment, and data retention do not apply to "Trial". "Pay-As-You-Go" requires an Azure subscription, and you acknowledge that this service is licensed to you under the terms applicable to your Azure Subscription.

I will receive information, tips, and offers about Azure and other Microsoft products and services. Privacy Statement.

Choose a payment plan: You can choose either the 7-Day free trial, or the Pay-As-You-Go option. If you choose the 7-Day trial, after 7-days you'll be prompted to delete your application, or convert it to the "Pay-As-You-Go" option.

If you choose the "Pay-As-You-Go" option the application is free for up to 5 devices. You will need an Azure account for this option.

Select an application template: Leave this item selected.

Configure your application

Application Name: Feel free to modify this as you see fit.

URL: The URL needs to be unique across all of Azure since it's a URL! You can leave this as the default, or change it. There will be real-time feedback showing if the name you choose is acceptable or not.

Contact Information

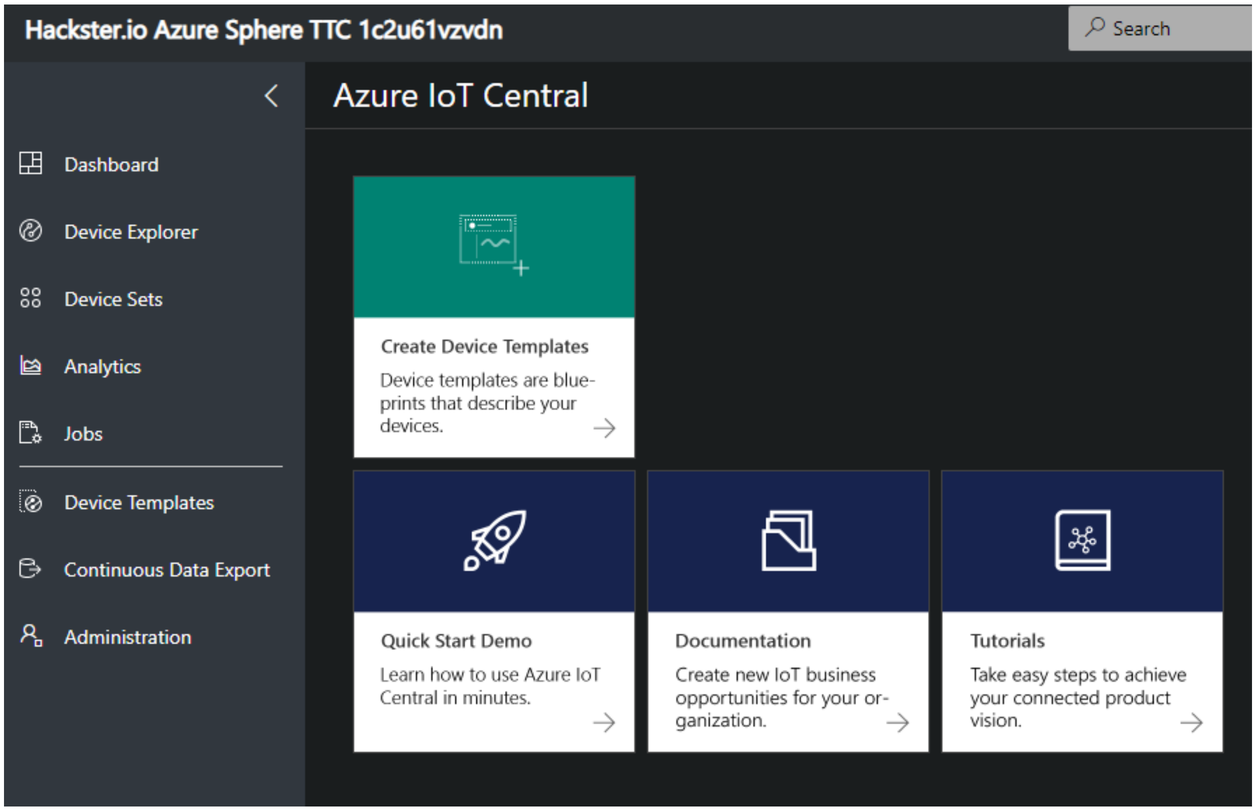
Supply your contact information

Country/Region:

Select your Country/Region from the pull down menu.

Click **Create**

Your IoT Central application will be provisioned and will open in your browser.



Create Device Templates

Device templates are blueprints that describe your devices.

Quick Start Demo

Learn how to use Azure IoT Central in minutes.

Documentation

Create new IoT business opportunities for your organization.

Tutorials

Take easy steps to achieve your connected product vision.

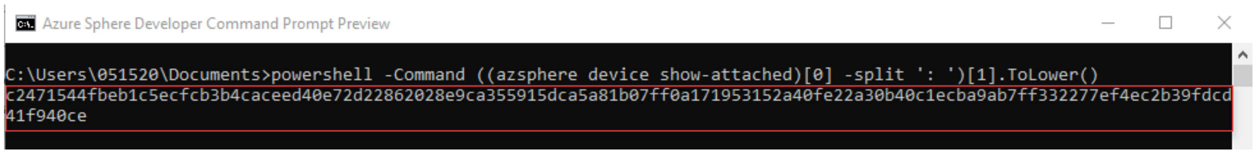
Provision our device to the IoT Central application

It's easy to add a real device to IoT Central. The steps below show how to manually add our device.

- On the left side menu, select Device Explorer
- On the Explorer page, click "+" to add a new device, select Real from the drop down menu
- The Create New Device form will open

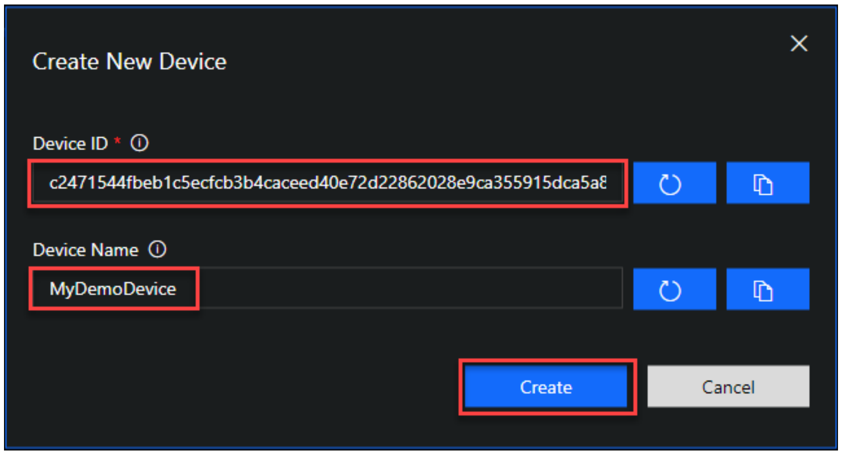
We need to get our device ID for this form, and it needs to be all lower case

- Back in our command prompt application paste in the following command
  - `powershell -Command ((azsphere device show-attached)[0] -split ': ')[1].ToLower()`



```
C:\Users\051520\Documents>powershell -Command ((azsphere device show-attached)[0] -split ': ')[1].ToLower()
c2471544fbeb1c5ecfcb3b4caceed40e72d22862028e9ca355915dca5a81b07ff0a171953152a40fe22a30b40c1ecba9ab7ff332277ef4ec2b39fdcd
41f940ce
```

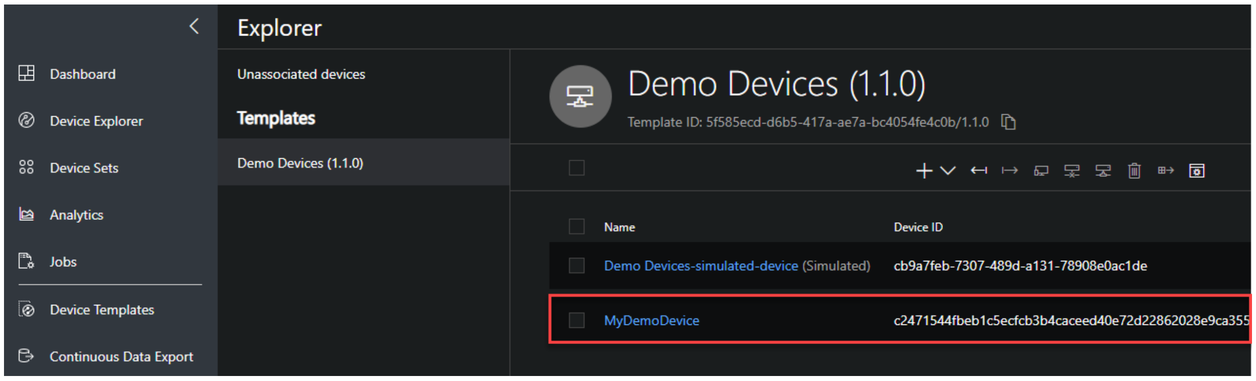
- Copy the output device id, and past it into the Create New Device form in the Device ID entry box
- Change the Device Name if you wish
- Click on the “Create” link



Device ID *: c2471544fbeb1c5ecfcb3b4caceed40e72d22862028e9ca355915dca5a8

Device Name: MyDemoDevice

Now click on Device Explorer → Demo Devices (1.1.0) and you will see a simulated device (that gets created automatically for any new template) and your new real device.



Demo Devices (1.1.0)

Template ID: 5f585ecd-d6b5-417a-ae7a-bc4054fe4c0b/1.1.0

Name	Device ID
Demo Devices-simulated-device (Simulated)	cb9a7feb-7307-489d-a131-78908e0ac1de
MyDemoDevice	c2471544fbeb1c5ecfcb3b4caceed40e72d22862028e9ca355

Configure DPS for our Application

The next thing we need to do is to configure the IoT Central application so that devices from our Azure Sphere Tenant are allowed to connect to our application. IoT Central also uses a Device Provisioning Service (DPS) to provision devices. Remember in Lab-4 we had to prove to the DPS that we owned the Azure Sphere Tenant, we need to do the same thing for the IoT Central application.

Download the tenant authentication CA certificate

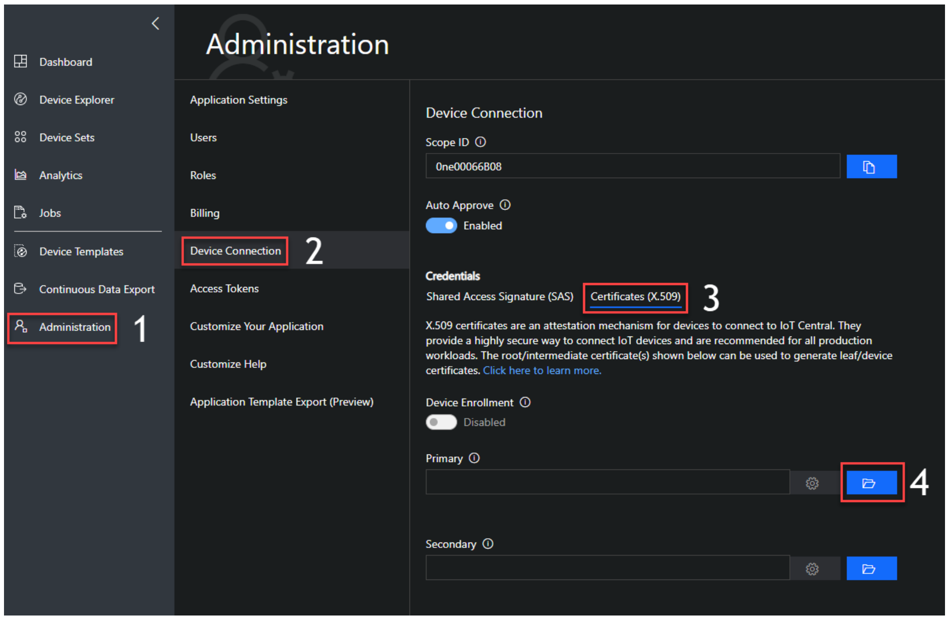
If you completed Lab-4, then you already have this certificate. If not, follow the instructions below.

- Go back to your “Azure Sphere Developer Command Prompt Preview” application
  - Start → Azure Sphere → Azure Sphere Developer Command Prompt Preview
- Make sure you’re logged into your tenant:
  - `azsphere login`
- Copy and paste in the command:
  - `azsphere tenant download-CA-certificate --output CAcertificate.cer`
  - Note the output file must have the .cer extension

Upload the tenant CA certificate to Azure IoT Central and generate a verification code

In your new IoT Central application

- Select the “Administration” link from the left menu
- Select the “Device Connection” option
- Select the Certificates (X.509) link under the credentials section
- Click on the Primary folder icon and browse to the tenant CA certificate you downloaded from your tenant. Make sure the view filter is set to All files (*) if you don’t see your certificate.



Scope ID: 0ne00066B08

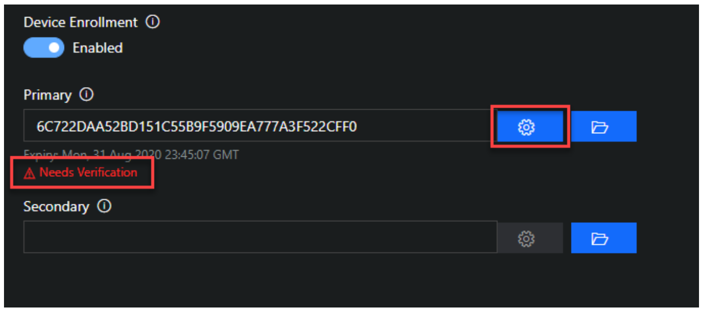
Auto Approve: Enabled

X.509 certificates are an attestation mechanism for devices to connect to IoT Central. They provide a highly secure way to connect IoT devices and are recommended for all production workloads. The root/intermediate certificate(s) shown below can be used to generate leaf/device certificates. Click here to learn more.

Device Enrollment: Disabled

- Your certificate is uploaded and a notification is displayed stating that the certificate "Needs Verification"

- Click on the gear icon to verify the certificate


Device Enrollment
Enabled
Primary
6C722DAA52BD151C55B9F5909EA777A3F522CFF0
Expires Mon, 31 Aug 2020 23:45:07 GMT
Needs Verification
Secondary

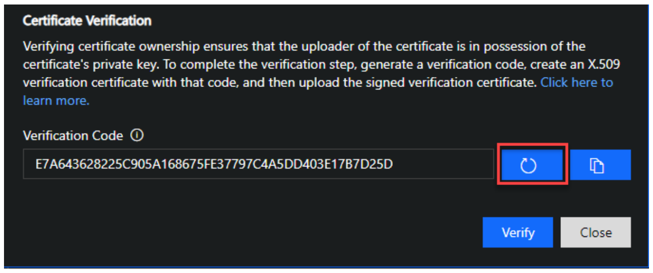
Return to the Azure Sphere Developer Command Prompt application

- Copy and paste the following command into the application, don't execute the command yet, we need to get the validation code first

- azsphere tenant download-validation-certificate --output ValidationCertification.cer --verificationcode <code>

Back in your IoT Central Application

Click on the refresh link (the circle with an arrow), a Verification code is generated and displayed in the "Verification Code" box.



Certificate Verification

Verifying certificate ownership ensures that the uploader of the certificate is in possession of the certificate's private key. To complete the verification step, generate a verification code, create an X.509 verification certificate with that code, and then upload the signed verification certificate. Click here to learn more.

Verification Code: E7A643628225C905A168675FE37797C4A5DD403E17B7D25D

- Copy the verification code, there's a copy link to the right of the box

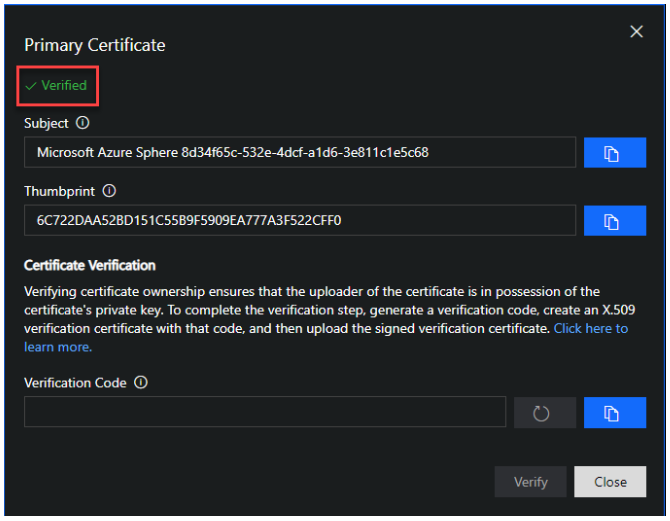
- Back in the command window, replace the <code> text with your verification code and execute the command

The Azure Sphere Security Service signs the validation certificate with the verification code to prove that you own the CA and downloads a Verification certificate.

- Back in your IoT Central application, click on the Verify button at the bottom of the form

- Upload the new Verification Certificate by browsing to the file you just downloaded from the tenant

The Primary Certificate window will update and show the status as Verified


Primary Certificate

✓ Verified

Subject

Microsoft Azure Sphere 8d34f65c-532e-4dcf-a1d6-3e811c1e5c68

Thumbprint

6C722DAA52BD151C55B9F5909EA777A3F522CFF0

Certificate Verification

Verifying certificate ownership ensures that the uploader of the certificate is in possession of the certificate's private key. To complete the verification step, generate a verification code, create an X.509 verification certificate with that code, and then upload the signed verification certificate. Click here to learn more.

Verification Code

At this point the application is ready to provision new devices that connect to the application with the correct identifiers (DPS Scope ID and Tenant GUID).

Configure the example application

Now that we have the IoT Central side ready, let's go back to our application code and configure the application to connect to IoT Central.

Remember in Lab-4 we used the "Connected Service" utility in Visual Studio to update our app_manifest.json file with the details needed to connect to the IoT Hub and DPS that we created. Microsoft has not added IoT Central configurations to the "Connected Service" utility yet, so we need to get the following information using a utility that Microsoft provides.

Remember the data we need to configure our application is . . .

- The FQDN for the DPS
- The FQDN for the IoT Hub that exists behind the IoT Central SaaS implementation
- The DPS scope ID
- Our tenant GUID

Some of this information we have easy access to, however the FQDN for the IoT Hub is buried in the implementation and we need to use a utility that Microsoft provides to get that information.

ShowIoTCentralConfig.exe

When you downloaded the example application project from GitHub, you also downloaded a tools folder. We need to change to the tools directory and execute the utility.

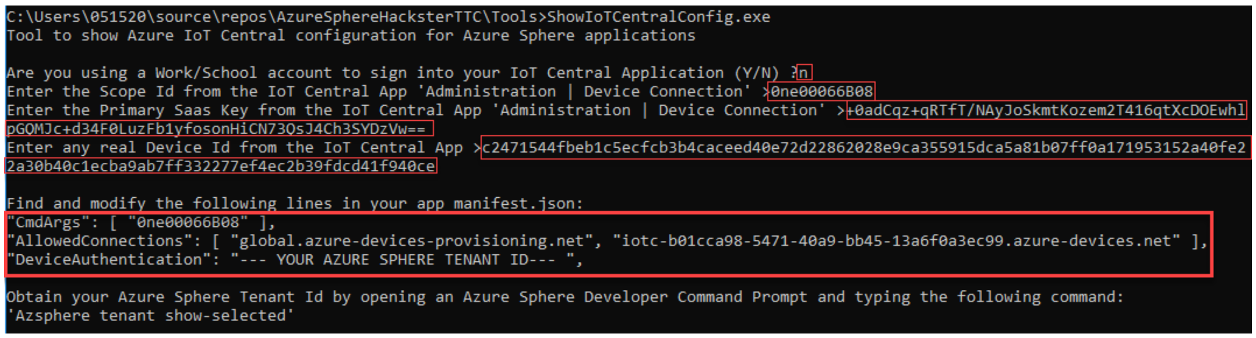
- From the "Azure Sphere Developer Command Prompt Preview" application change to the tools directory
- On my PC this directory is C:\Users\xxxx\source\repos\AzureSphereHacksterTTC\Tools
- From my command prompt I can execute the following command to change to this directory

  - Cd C:\Users\xxxx\source\repos\AzureSphereHacksterTTC\Tools

- Then execute the command
  - showIoTCentralConfig.exe
- You'll be prompted to answer a few questions, please use the details below to complete the task
  - **Work or School account** answer: **N** for No
  - **Scope ID**: This can be found in your IoT Central Application on the Administration → Device Connection page. Copy the Scope ID at the top of the form and paste it into the command prompt
  - **Primary Saas Key**: This detail is on the same page as the Scope ID, copy the Primary Key and paste it into the command prompt
  - **Device ID**: Next the utility is asking for a "real" device ID. We just read the device ID from our Azure Sphere Board. Use that ID and paste it in for your answer.

The utility will contact Azure and using the details you provided, it will provide details required for your device to connect to the IoT Central Application.

Below is a capture showing my responses, and the output from the utility.



```
C:\Users\051520\source\repos\AzureSphereHacksterTTC\Tools>ShowIoTCentralConfig.exe
Tool to show Azure IoT Central configuration for Azure Sphere applications

Are you using a Work/School account to sign into your IoT Central Application (Y/N) ?n
Enter the Scope Id from the IoT Central App 'Administration | Device Connection' >0ne00066B08
Enter the Primary Saas Key from the IoT Central App 'Administration | Device Connection' >+0adCqz+qRTfT/NAyJoSkmtKozem2T416qtXcDOEwhl
pGQMJc+d34F0LuzFb1yfosonHiCN73QsJ4Ch3SYDzVw==
Enter any real Device Id from the IoT Central App >c2471544fbeb1c5ecfcb3b4caceed40e72d22862028e9ca355915dca5a81b07ff0a171953152a40fe2
2a30b40c1ecba9ab7ff332277ef4ec2b39fdcd41f940ce

Find and modify the following lines in your app manifest.json:
"CmdArgs": [ "0ne00066B08" ],
"AllowedConnections": [ "global.azure-devices-provisioning.net", "iotc-b01cca98-5471-40a9-bb45-13a6f0a3ec99.azure-devices.net" ],
"DeviceAuthentication": "--- YOUR AZURE SPHERE TENANT ID--- ",

Obtain your Azure Sphere Tenant Id by opening an Azure Sphere Developer Command Prompt and typing the following command:
'Azsphere tenant show-selected'
```

Keep this information handy for the next step.

Modify the Azure Sphere source code

- Launch the Visual Studio application and open the “**AvnetStarterKitReferenceDesign**” project. Visual Studio keeps a list of recent projects, your project should be found in that list.

If you completed Lab-4 then you already have entries for the CmdArgs, AllowedConnections, and DeviceAuthentication in your app_manifest.json file. If you want to save these settings, make a backup copy of your app_manifest.json file.

- Open the app_manifest.json file from the Solution Explorer

- The output from the ShowIoTCentralConfig.exe utility shows us what we need to change. **Note that your entries will be different.**

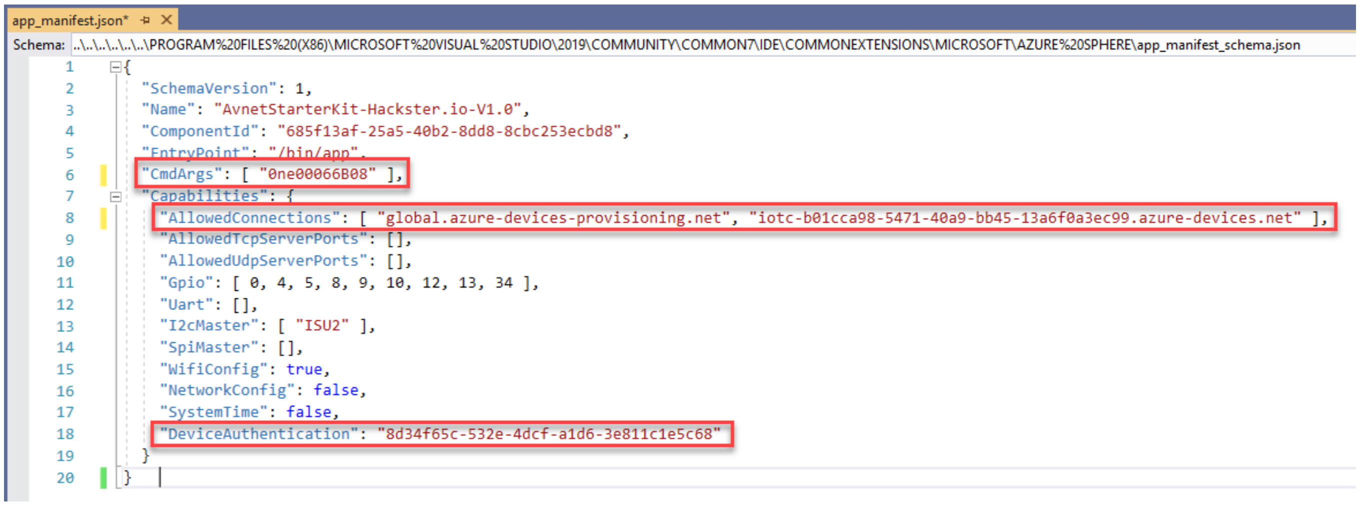
    - "CmdArgs": ["0ne00066B08"],

    - "AllowedConnections": ["global.azure-devices-provisioning.net", "iotc-b01cca98-5471-40a9-bb45-13a6f0a3ec99.azure-devices.net"],

    - "DeviceAuthentication": "--- YOUR AZURE SPHERE TENANT ID--- ",

        - If you completed Lab-4 then you already have your Azure Sphere Tenant ID, if not you can use the azsphere tenant show-selected command to retrieve the ID

- Save the file

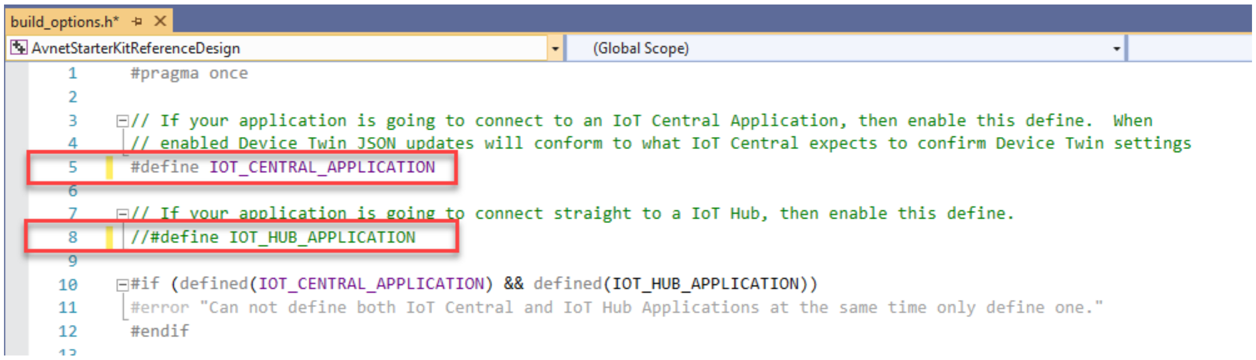


app_manifest.json*

Schema: ..\..\..\..\..\PROGRAM%20FILES%20(X86)\MICROSOFT%20VISUAL%20STUDIO\2019\COMMUNITY\COMMON7\IDE\COMMONEXTENSIONS\MICROSOFT\AZURE%20SPHERE\app_manifest_schema.json

```
{
  "SchemaVersion": 1,
  "Name": "AvnetStarterKit-Hackster.io-V1.0",
  "ComponentId": "685f13af-25a5-40b2-8dd8-8cbc253ecbd8",
  "EntryPoint": "/bin/app",
  "CmdArgs": [ "0ne00066B08" ],
  "Capabilities": {
    "AllowedConnections": [ "global.azure-devices-provisioning.net", "iotc-b01cca98-5471-40a9-bb45-13a6f0a3ec99.azure-devices.net" ],
    "AllowedTcpServerPorts": [],
    "AllowedUdpServerPorts": [],
    "Gpio": [ 0, 4, 5, 8, 9, 10, 12, 13, 34 ],
    "Uart": [],
    "I2cMaster": [ "ISU2" ],
    "SpiMaster": [],
    "WifiConfig": true,
    "NetworkConfig": false,
    "SystemTime": false,
    "DeviceAuthentication": "8d34f65c-532e-4dcf-a1d6-3e811c1e5c68"
  }
}
```

- Open the build_options.h file
- On line #5 remove the "//"s to enable the IOT_CENTRAL_APPLICATION build option
- Confirm that line #8 is commented out


build_options.h*

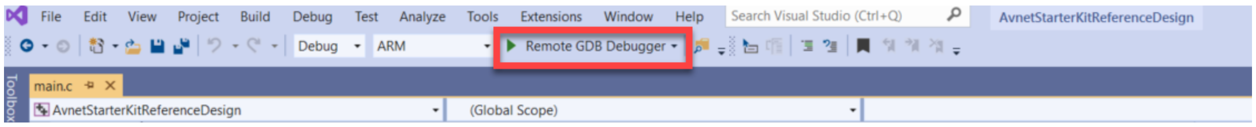
AvnetStarterKitReferenceDesign | (Global Scope)

```
1   #pragma once
2
3   // If your application is going to connect to an IoT Central Application, then enable this define.  When
4   // enabled Device Twin JSON updates will conform to what IoT Central expects to confirm Device Twin settings
5   #define IOT_CENTRAL_APPLICATION
6
7   // If your application is going to connect straight to a IoT Hub, then enable this define.
8   //#define IOT_HUB_APPLICATION
9
10  #if (defined(IOT_CENTRAL_APPLICATION) && defined(IOT_HUB_APPLICATION))
11  #error "Can not define both IoT Central and IoT Hub Applications at the same time only define one."
12  #endif
```

Build and Run the Application

Now let's build and run our application.

- Make sure your device is connected to your PC
- Ensure your device has a Wi-Fi connection
    - azsphere device wifi show-status
- Click on the "Remote GDB Debugger" button at the top of the application. Your application will build, link, side-load, and run.



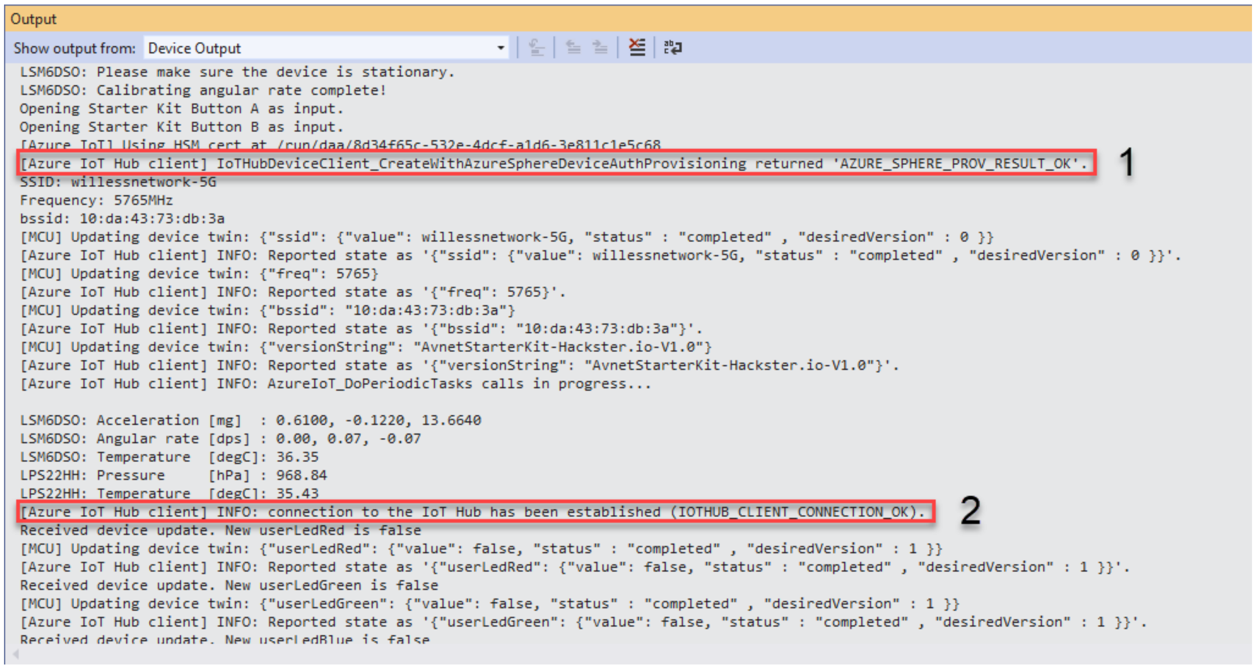
Your output should look similar to the screenshot below. There are two debug lines that I want to point out.

1: This debug shows that our device connected to our DPS and that the DPS successfully provisioned our device to our IoT Hub

2: This debug shows that our application has successfully connected to our IoT Hub

If you have errors . . .

- Confirm your device is connected to a Wi-Fi access point or hot spot that has internet connectivity



```
LSM6DSO: Please make sure the device is stationary.
LSM6DSO: Calibrating angular rate complete!
Opening Starter Kit Button A as input.
Opening Starter Kit Button B as input.
[Azure IoT] Using HSM cert at /run/daa/8d34f65c-532e-4dcf-a1d6-3e811c1e5c68
[Azure IoT Hub client] IoTHubDeviceClient_CreateWithAzureSphereDeviceAuthProvisioning returned 'AZURE_SPHERE_PROV_RESULT_OK'.
SSID: willessnetwork-5G
Frequency: 5765MHz
bssid: 10:da:43:73:db:3a
[MCU] Updating device twin: {"ssid": {"value": willessnetwork-5G, "status" : "completed" , "desiredVersion" : 0 }}
[Azure IoT Hub client] INFO: Reported state as '{"ssid": {"value": willessnetwork-5G, "status" : "completed" , "desiredVersion" : 0 }}'.
[MCU] Updating device twin: {"freq": 5765}
[Azure IoT Hub client] INFO: Reported state as '{"freq": 5765}'.
[MCU] Updating device twin: {"bssid": "10:da:43:73:db:3a"}
[Azure IoT Hub client] INFO: Reported state as '{"bssid": "10:da:43:73:db:3a"}'.
[MCU] Updating device twin: {"versionString": "AvnetStarterKit-Hackster.io-V1.0"}
[Azure IoT Hub client] INFO: Reported state as '{"versionString": "AvnetStarterKit-Hackster.io-V1.0"}'.
[Azure IoT Hub client] INFO: AzureIoT_DoPeriodicTasks calls in progress...

LSM6DSO: Acceleration [mg]  : 0.6100, -0.1220, 13.6640
LSM6DSO: Angular rate [dps] : 0.00, 0.07, -0.07
LSM6DSO: Temperature  [degC]: 36.35
LPS22HH: Pressure     [hPa] : 968.84
LPS22HH: Temperature  [degC]: 35.43
[Azure IoT Hub client] INFO: connection to the IoT Hub has been established (IOTHUB_CLIENT_CONNECTION_OK).
Received device update. New userLedRed is false
[MCU] Updating device twin: {"userLedRed": {"value": false, "status" : "completed" , "desiredVersion" : 1 }}
[Azure IoT Hub client] INFO: Reported state as '{"userLedRed": {"value": false, "status" : "completed" , "desiredVersion" : 1 }}'.
Received device update. New userLedGreen is false
[MCU] Updating device twin: {"userLedGreen": {"value": false, "status" : "completed" , "desiredVersion" : 1 }}
[Azure IoT Hub client] INFO: Reported state as '{"userLedGreen": {"value": false, "status" : "completed" , "desiredVersion" : 1 }}'.
Received device update. New userLedBlue is false
```

Once your application is running, I invite you to go to the IoT Central Application to see the telemetry and play with the settings page. I'll review the application in the lecture material.

Modify the IoT Central Application

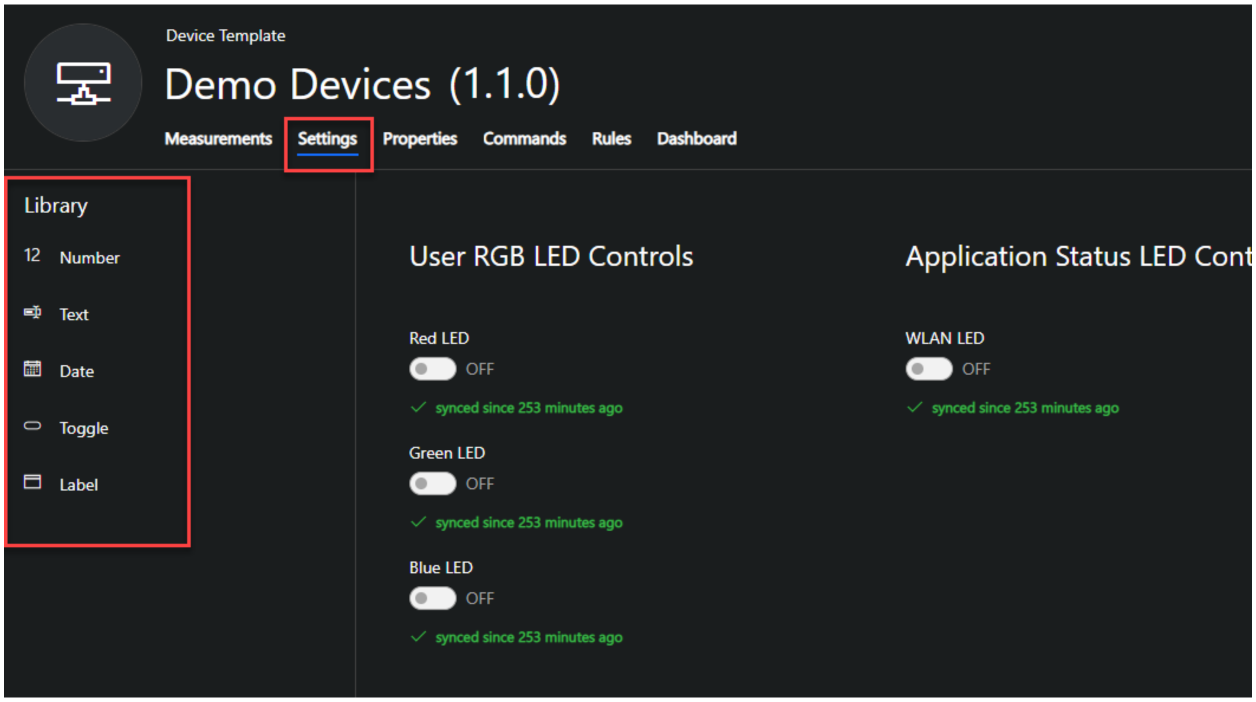
Adding telemetry or settings items to IoT Central is very easy. For all the details on how we built the IoT Central application you should read through this blog on Element14.com. The blog shows how to add additional telemetry and settings to the application.

For this section, I'm going to walk us through adding a new item to the settings tab for our application, Application LED control. The code assignment will ask the student to add the Azure Sphere code to support this change.

- Open the IoT Central Application you created in this Lab
- From the left menu, select "Device Templates", then the only template listed "Demo Devices (1.1.0).

A screen opens up that shows all the current Measurements, or telemetry data. The data displayed is from a simulated device. See the blog for details on adding telemetry data to this tab. We want to change the Settings Tab.

- Click on the "Settings" tab, you'll see a screen similar to the graphic below
- The items on the right side have already been configured
- The Library at the left side shows items that can be added. The cool thing about the settings entries is that they are all linked to the Device Twin items. So if you have a device twin item implemented in your Azure Sphere application you can control that item using this interface!



User RGB LED Controls

Red LED
OFF
synced since 253 minutes ago

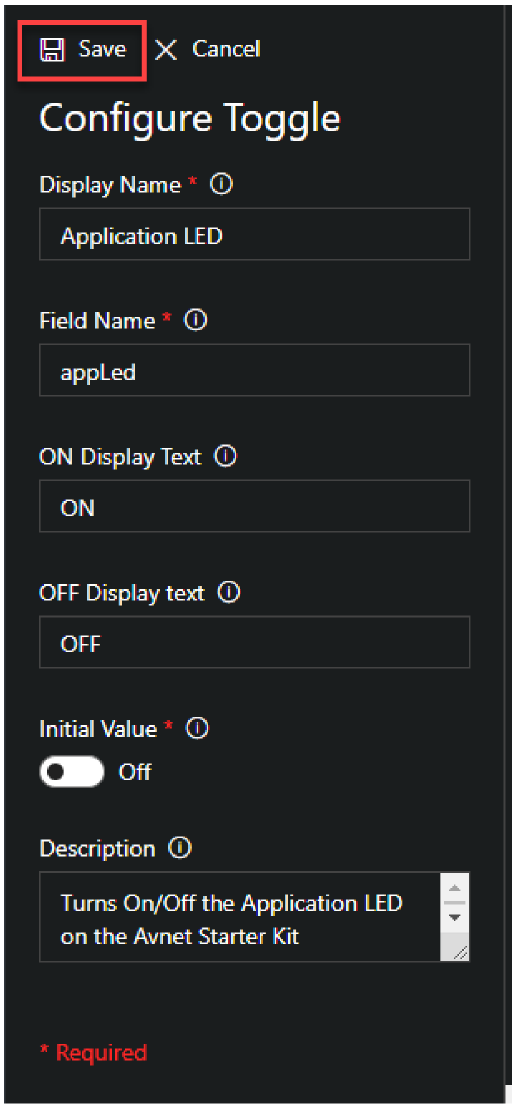
Green LED
OFF
synced since 253 minutes ago

Blue LED
OFF
synced since 253 minutes ago

Application Status LED Cont

WLAN LED
OFF
synced since 253 minutes ago

- We want to add a new toggle, so click on the Toggle text in the Library, the "Configure Toggle" form opens


Save
Cancel

Configure Toggle

Display Name *
Application LED

Field Name *
appLed

ON Display Text
ON

OFF Display text
OFF

Initial Value *
Off

Description
Turns On/Off the Application LED on the Avnet Starter Kit

* Required

Fill in the form as shown here. I'll describe each field.

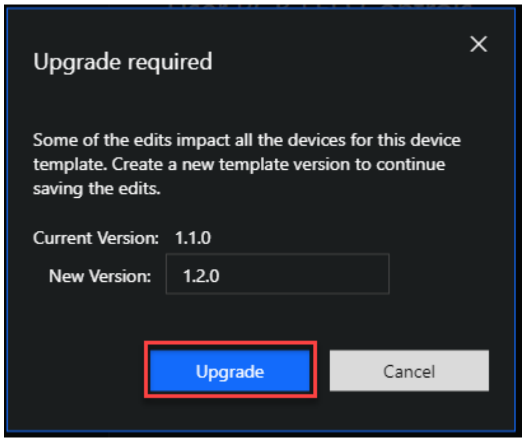
Display Name: This can be anything you like. It will be displayed over the toggle control.

Field Name: This is the Key from the {"key": value} device twin entry. This entry is case sensitive and must match the key from your device twin exactly.

On Display Text: Text to be displayed with the toggle is in the on position.

Off Display Text: Text to be displayed with the toggle is in the off position.

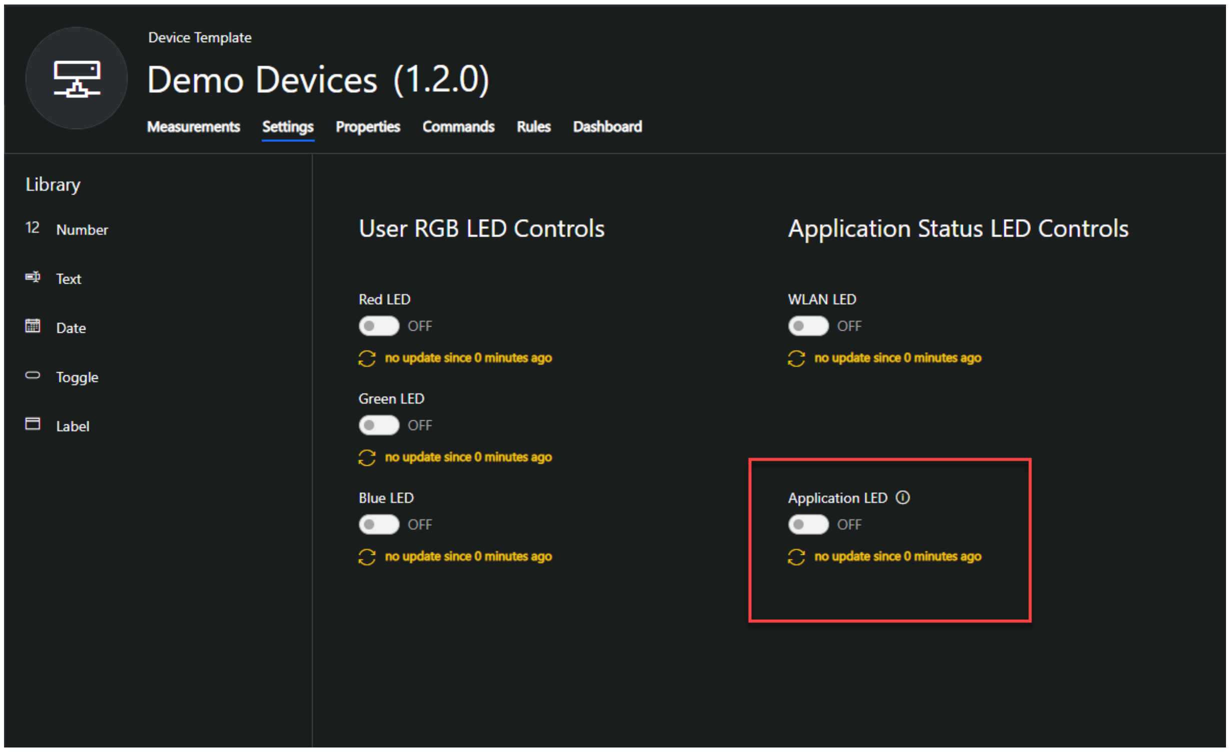
Once you have the form filled out as shown, click on the "Save" button at the top of the form. Since we added a new item to the template you'll be prompted to create a new template version, version 1.2.0. Click the "Upgrade" button to accept the change.


Upgrade required

Some of the edits impact all the devices for this device template. Create a new template version to continue saving the edits.

Current Version: 1.1.0
New Version: 1.2.0

The new control is shown on the new template



User RGB LED Controls

Red LED
OFF
no update since 0 minutes ago

Green LED
OFF
no update since 0 minutes ago

Blue LED
OFF
no update since 0 minutes ago

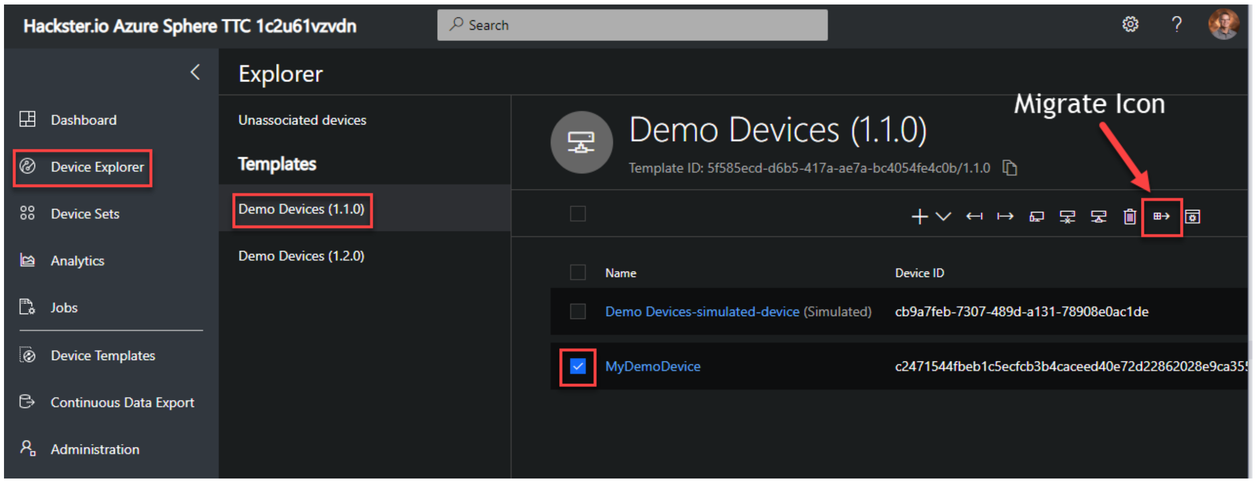
Application Status LED Controls

WLAN LED
OFF
no update since 0 minutes ago

Application LED
OFF
no update since 0 minutes ago

Next we need to move our device into the new template.

- From the left menu select "Device Explorer" → Demo Devices (1.1.0)
- Check the box next to your device
- Select the migrate icon as shown below

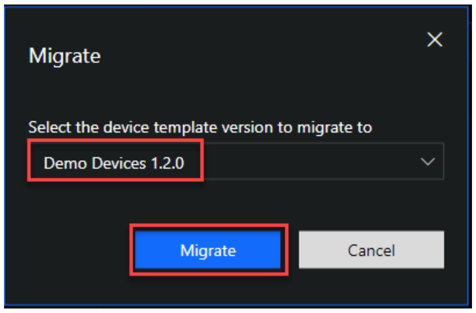


Demo Devices (1.1.0)
Template ID: 5f585ecd-d6b5-417a-ae7a-bc4054fe4c0b/1.1.0

Name	Device ID
Demo Devices-simulated-device (Simulated)	cb9a7feb-7307-489d-a131-78908e0ac1de
MyDemoDevice	c2471544fbeb1c5ecfcb3b4caceed40e72d22862028e9ca35

The “Migrate” dialog will open

- ○ Select “Demo Devices 1.2.0 from the list and click on the “Migrate” Button

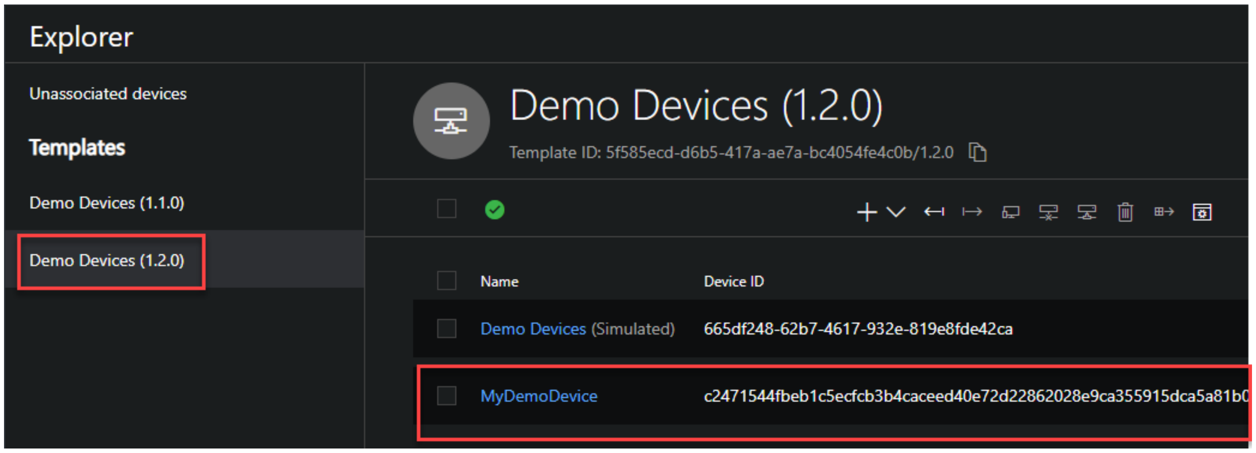


Migrate

Select the device template version to migrate to

Demo Devices 1.2.0

Your device will be removed from the “Demo Devices (1.1.0)” template and placed into the “Demo Devices (1.2.0)” template where the new setting control for the Application LED will be active.



Demo Devices (1.2.0)

Template ID: 5f585ecd-d6b5-417a-ae7a-bc4054fe4c0b/1.2.0

Name	Device ID
Demo Devices (Simulated)	665df248-62b7-4617-932e-819e8fde42ca
MyDemoDevice	c2471544fbeb1c5ecfcb3b4caceed40e72d22862028e9ca355915dca5a81b0

We have modified our IoT Central Application but the device twin item we just added is not implemented in the Azure Sphere application, let’s fix that.

Complete a code assignment

The assignment is to add a new Device Twin item to control the Application LED on the Avnet Starter Kit.

In Lecture-4 we reviewed the example application Device Twin implementation. Use that information and the details below to add a new device twin item that can be used to control the App LED on the board. Remember that the Lecture content for Device Twins was different than the Lab content on Device Twins. You can use the Lab content to understand what all has to happen when we process device twins. The lecture content discusses the actual implementation in the example application.

Key: “appLed”

GPIO: AVT_LED_APP

Active_high: false

Hint: Look at how one of the other LED device twin items is implemented.

What does success look like?

When the new device twin item is implemented and the Azure Sphere application is running, you should be able to go to your device in the IoT Central Application, toggle the “Application LED” control and see the App LED on the Avnet Starter Kit toggle!

Bonus Assignment

As a bonus assignment add the pressure telemetry (from the Lab-4 coding assignment) to the Measurements Tab on your IoT Central Application.

Wrap Up

In this Lab we learned about IoT Central and how to implement changes in the application.

- How to create an IoT Central Application from a Template
- How to add a device to IoT Central
- How to add a new Settings item to IoT Central
- Reinforced making changes in the Azure Sphere Application to implement a new Device Twin item

Revision History

Date	Version	Revision
1 July 19	01	Preliminary release