Renesas RA Family

RA Azure IoT Cloud Connectivity Solution

Introduction

The objective of this Application Project is to demonstrate the Renesas RA Azure IoT Cloud Connectivity solution by providing a fully working solution that sends sensor data to the Microsoft® Azure IoT Central cloud over Wi-Fi/Ethernet connection and using HAL drivers and middleware provided with Renesas Flexible Software Package (FSP).

You will start by setting up the project to read the necessary sensor data from the On-chip temperature sensor and then view that data within the debug context of e^2 studio IDE. Next, you will setup your Azure IoT Central software-as-a-service (SaaS) application to send and receive data to and from the evaluation kit. Lastly you will configure the necessary security credentials on your evaluation kit to securely connect to the Azure cloud.

This application project contains integrated Azure Embedded C SDK files which are designed to allow embedded (IoT) MCU devices like RA6M3 to communicate with Azure services such as Azure Device Provisioning Service (DPS) and Azure IoT Hub to demonstrate Azure cloud connectivity.

Hardware/Software Prerequisites:

Hardware:

- Renesas EK-RA6M3 kit with USB micro B cables (2) (renesas.com/ra/ek-ra6m3)
- Silex SX-ULPGN Wi-Fi PMOD module (or) Ethernet cable.(renesas.com/wi-fi-pmod)
- PC running Windows® 10 with at least 2 USB ports

Software:

- Renesas Flexible Software Package platform installation, which includes:
  - e^2 studio 2.10.0 or later
  - FSP v 2.0.0 or greater
  - GCC Arm Embedded 9.2.1 (2019-q4-major)
- SEGGER J-Link USB driver
- Tera Term v4.0 or newer
- Web browser (Google Chrome, Mozilla Firefox or Microsoft Edge)

Prerequisites and Intended Audience

This application note assumes that the user has some experience with the Renesas e^2 studio IDE and Flexible Software Package (FSP). Before running this application, follow the procedure in the FSP User Manual to build and run the Blinky project. Doing so enables you to become familiar with the e^2 studio and the FSP and ensures proper functioning of debug connection to your board. In addition, this application note assumes the user have some knowledge of MQTT/TLS protocols.

The intended audience are users who want to develop Azure Cloud Connectivity applications using FSP MQTT/TLS modules on Renesas RA6M3 MCU series.

Contents

1. RA MQTT/TLS Application Overview

This application project demonstrates the Renesas RA IoT Cloud Connectivity solution using the FSP MQTT/TLS modules and uses Microsoft® Azure as the cloud provider. Ethernet or Wi-Fi can be used as primary communication interface between the MQTT device and the Azure IoT Services.

The EK-RA6M3 kit acts as an MQTT node, connects to the Azure IoT service using MQTT/TLS protocol over Ethernet or Wi-Fi network interface. The application periodically reads the on-chip temperature sensor values and publishes this information to the Azure IoT Core. It also subscribes to a User LED state MQTT topic. You can turn the User LEDs ON/OFF by publishing the LED state remotely. This application reads the updated LED state and turns the Users LEDs ON/OFF.

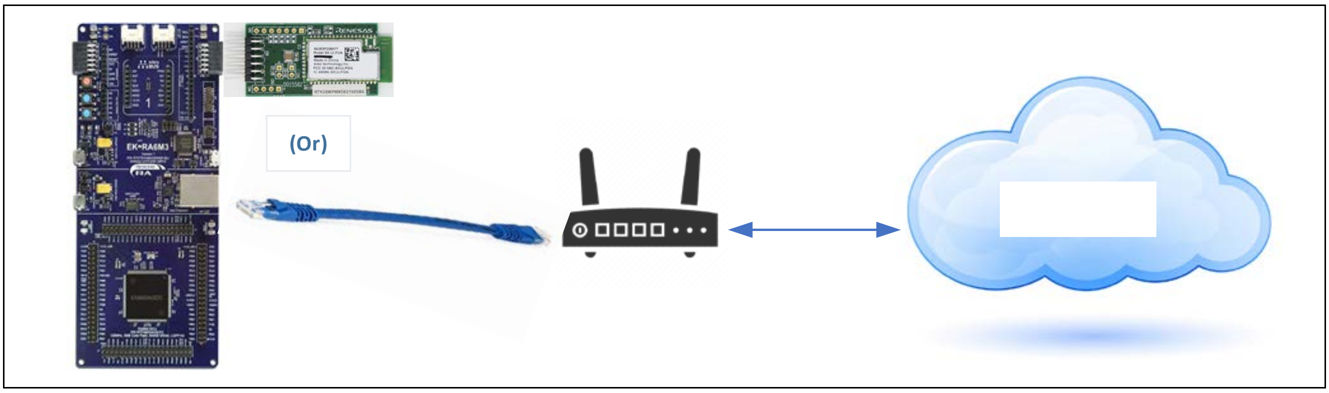


Figure 1. RA MQTT/TLS Application Overview

This application project is a single user thread application and the main thread handles the following major functions:

1. Initialize network interface (Ethernet in case of Azure_EK_RA6M3_Eth project or Wi-Fi in case of Azure_EK_RA6M3_WiFi project).
2. Initialize sensors.
3. Initialize USB PCDC interface to print debug message over serial console.
4. Connect to Azure Cloud services.
5. Read sensor data periodically and publish the data on MQTT topics.
6. Toggle the user LED state based on the incoming MQTT message.

2. Powering Up the EK-RA6M3 Evaluation Kit

The following hardware components are needed for this application. Refer to EK-RA6M3 user's manual (renesas.com/ra/ek-ra6m3) for the default jumper settings.

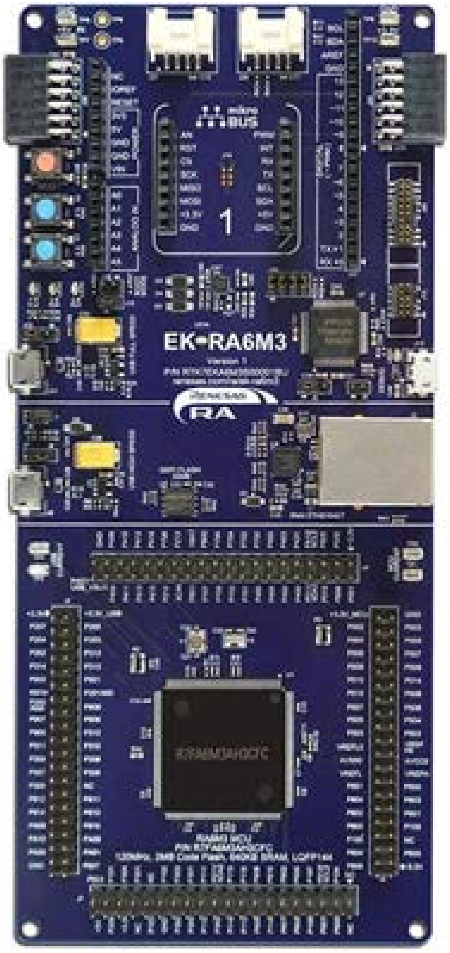


Figure 2. EK-RA6M3 Board

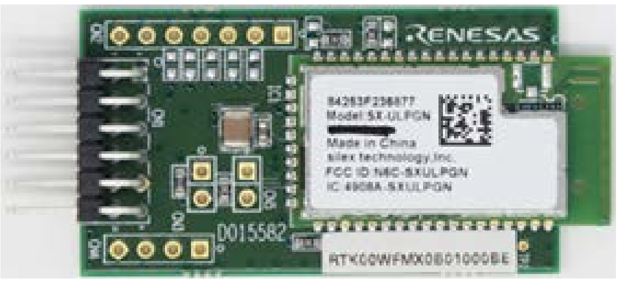


Figure 3. Silex SX-ULPGN Wi-Fi PMOD Module

Note: One micro USB cable is provided with the EK-RA6M3 kit. Users will need to supply one additional cable.

Figure 4. Micro USB Cable

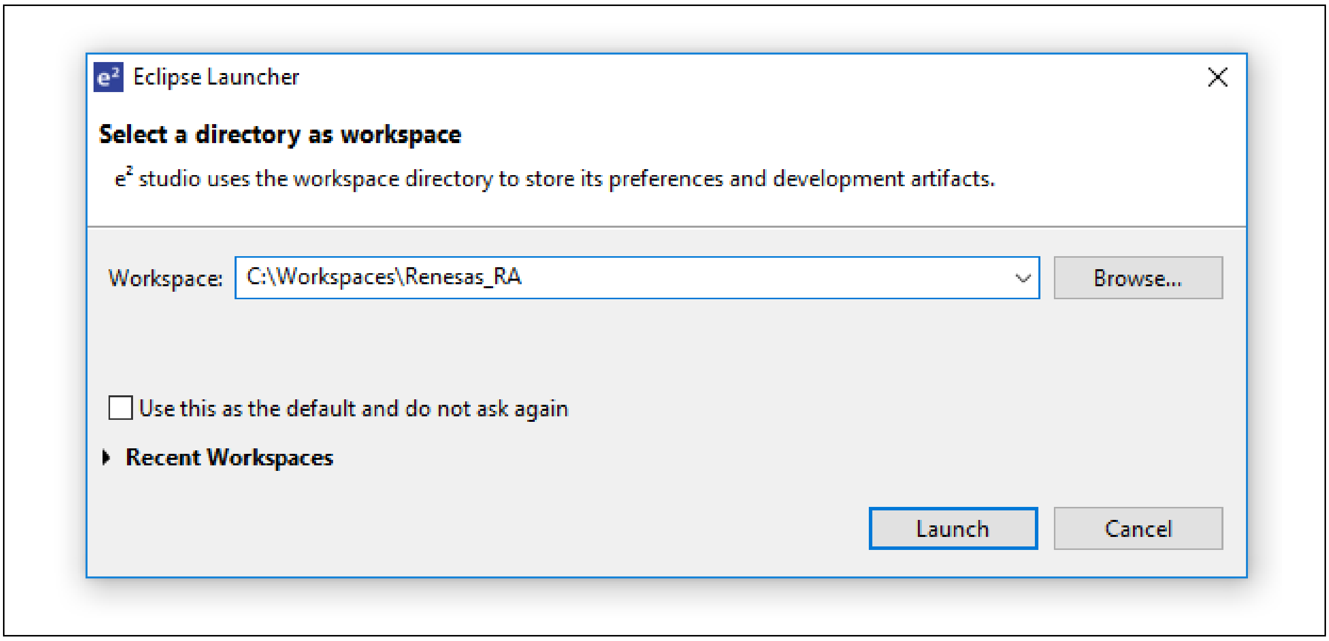
1. In case of using Wi-Fi network interface, attach Silex SX-ULPGN Wi-Fi Pmod module to PMOD2 (J25, upper right). The Silex module and components should be facing up.
2. In case of using Ethernet network interface, attach an Ethernet cable to the EK_RA6M3 Ethernet LAN Port.
3. Connect the EK-RA6M3 kit to the PC using a USB micro-B cable connected to DEBUG (J10) on the board (right side of board above Ethernet jack).
4. Connect the second USB micro-B cable to the PC and to the USB high speed (J6) port on the opposite edge of the board. Make sure Jumper J7 has pin 2-3 connected.
5. Once completed, the LEDs should be illuminated as described below.

LED1, LED2, LED3 (below user buttons)	Flashing red, green and blue
LED4 (middle of board)	Illuminating white
LED5 (right side, near USB debug port)	Illuminating yellow

3. Importing the FSP Application Project

1. Launch e^2 studio. e^2 studio can be launched from the Windows start menu or directly from the FSP folder. If you have multiple versions of e^2 studio installed, please make sure to launch the version of e^2 studio that was called out in the prerequisites section on page 1 of this application project.

2. In the Eclipse Launcher window, specify the destination for the new workspace. It is recommended to keep the path simple and avoid using spaces.



Eclipse Launcher

Select a directory as workspace

e^2 studio uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Workspaces\Renesas_RA

□ Use this as the default and do not ask again

Figure 5. Eclipse Launcher Window

3. Click **Launch** to start e^2 studio in the specified path. When prompted, press **Apply** to dismiss pop-up window asking for permission to log and report usage (it will remain disabled).

4. The welcome screen may show inside the new workspace. It can be dismissed by clicking on the **Workbench** button in the top-right corner.

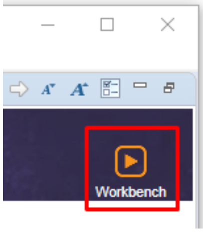


Figure 6. Workbench Button

5. Go to **File** > **Import** option.

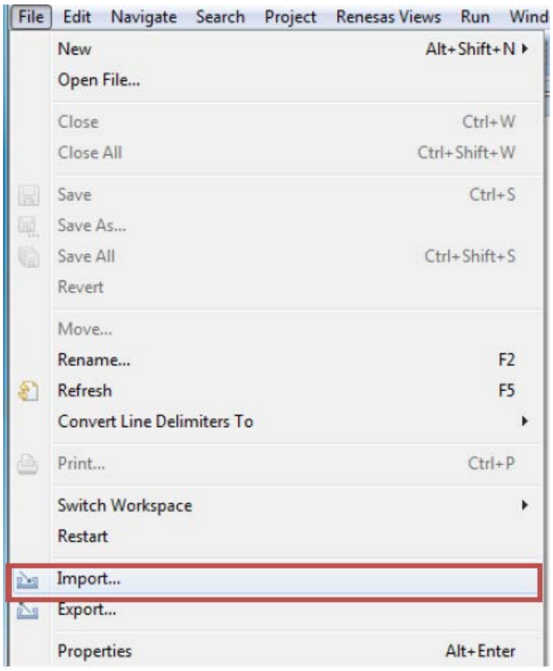
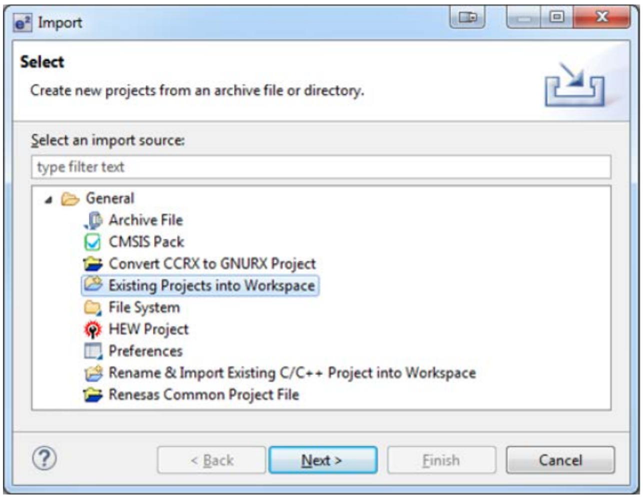


Figure 7. Import Option

6. In the import dialog box, select the **General** option, and then select **Existing Projects into Workspace** to import the project into the current workspace and click **Next**.


Import

Select

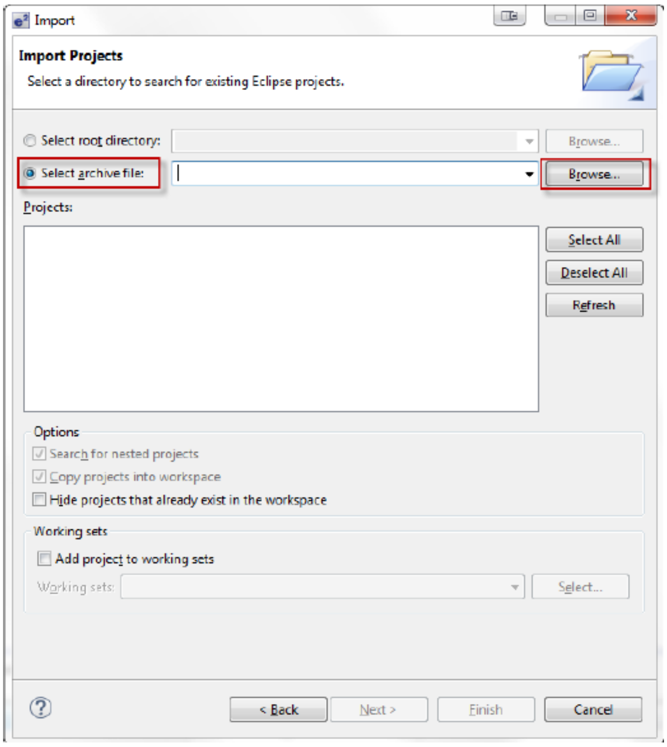
Create new projects from an archive file or directory.

Select an import source:

type filter text
General
Archive File
CMSIS Pack
Convert CCRX to GNURX Project
Existing Projects into Workspace
File System
HEW Project
Preferences
Rename & Import Existing C/C++ Project into Workspace
Renesas Common Project File
< Back
Next >
Finish
Cancel

Figure 8. Importing Existing Projects into Workspace

7. Click **Select archive file** and Click **Browse**.



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory:

● Select archive file:

Projects:

Options

☑ Search for nested projects

☑ Copy projects into workspace

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

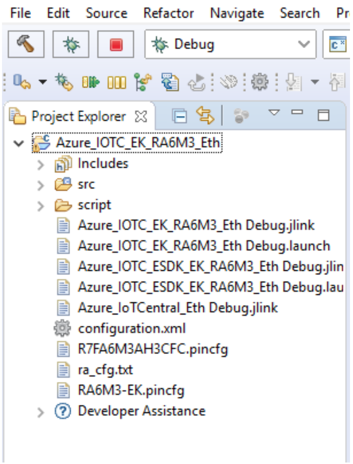
Working sets:

Figure 9. Selecting Archive File

8. Browse to the folder where the zip file for the project you want to import is located. Select the file for import. In our case, it is `Azure_EK_RA6M3_WiFi.zip` or `Azure_EK_RA6M3_Eth.zip` file depending up on the network interface.

9. Click **Finish** to import the project.
   Now that the project has been successfully imported, you can start configuring the project for the hardware.

10. Open the RA Configuration, if not already open, by double-clicking the configuration.xml file in the project explorer window.



Azure_IOTC_EK_RA6M3_Eth
- Includes
- src
- script
- Azure_IOTC_EK_RA6M3_Eth Debug.jlink
- Azure_IOTC_EK_RA6M3_Eth Debug.launch
- Azure_IOTC_ESDK_EK_RA6M3_Eth Debug.jlin
- Azure_IOTC_ESDK_EK_RA6M3_Eth Debug.lau
- Azure_IoTCentral_Eth Debug.jlink
- configuration.xml
- R7FA6M3AH3CFC.pincfg
- ra_cfg.txt
- RA6M3-EK.pincfg
- Developer Assistance

Figure 10. Project Explorer Window

Note: At this point, the ra_cfg and ra folders have not been created. These two folders contain files generated by e^2 studio and the FSP. The next step generates these files.

11. Click the **Generate Project Content** button in the top-right corner of the Configurator window. When prompted to *Proceed with save and generate*, tick the box next to **Always save and generate without asking** and click **Proceed**.

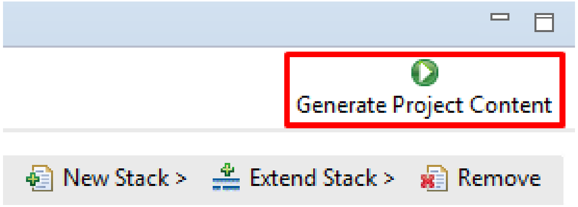


Figure 11. Generate Project Content

12. The RA Configurator will extract all the necessary drivers and generate the code based on the configurations provided in the **Properties** tab.

4. Setting up Azure IoT Central Application

In this section the cloud-side of the application will be created using Microsoft's Azure IoT Central services. A device template will be used to structure the devices that will connect to the cloud application. EK-RA6M3 kit will be created as an instance of this template and will be used for communication with IoT Central in the upcoming sections.

1. Open a web browser and go to http://azureiotcentral.com. Use the link in the top-right corner to **Sign in**. If you have not signed up yet, you will have a chance to do so (sign up is free).

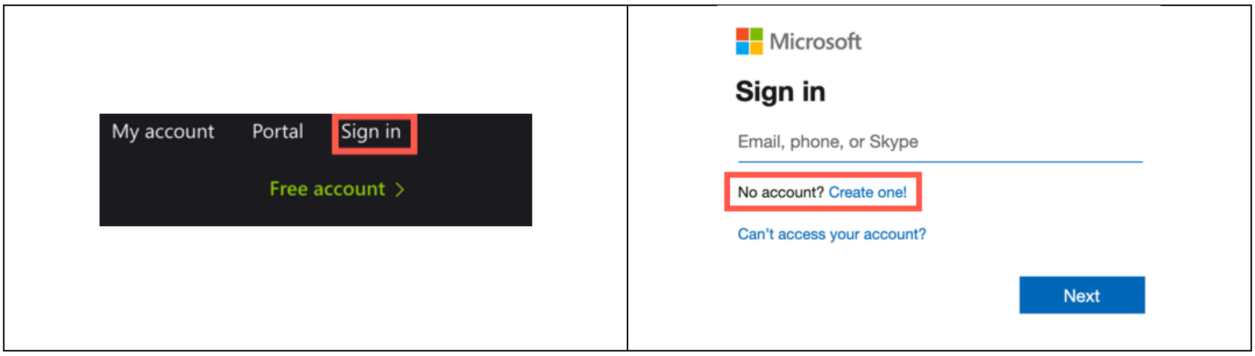
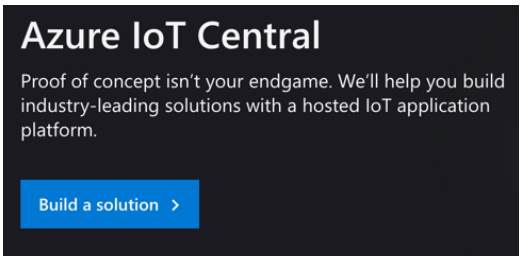


Figure 12. Signing In

2. Once logged in select **Build a solution**.


Azure IoT Central

Proof of concept isn't your endgame. We'll help you build industry-leading solutions with a hosted IoT application platform.

Figure 13. Selecting 'Build a solution'

3. On the upper left, click on the hamburger icon to **Expand Side Navigation** and then click **My apps**. Click ***+ New application*** in **My Apps**.

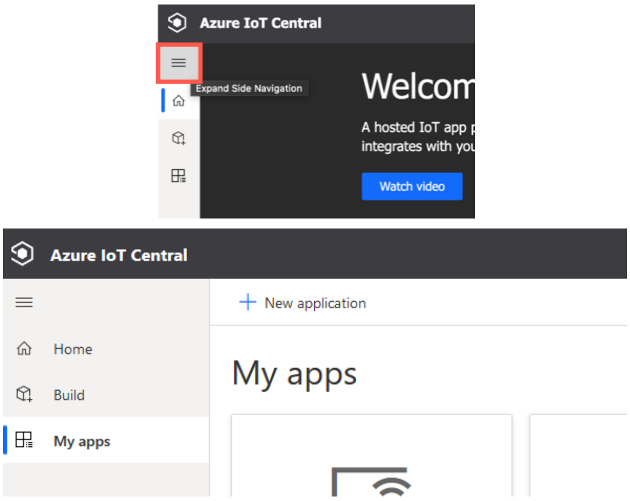
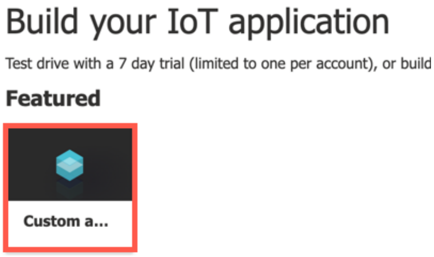


Figure 14. Clicking New Application

4. Create a custom application with the parameters listed below. Once completed, press **Create**.


Build your IoT application

Test drive with a 7 day trial (limited to one per account), or build

Featured

Figure 15. Creating a Custom Application

Pricing plan:	Free
Select an application template:	Custom application
Application name:	(use the generated default or specify a unique name)
Contact information:	(specify required details)

5. The application dashboard will be shown. Press the **Device Templates** icon on the left navigation menu and click **+ New** to create a device template.

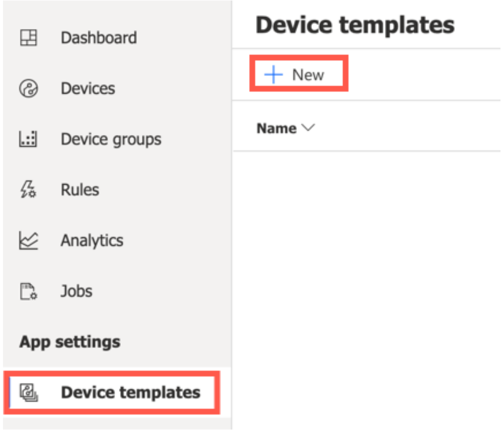
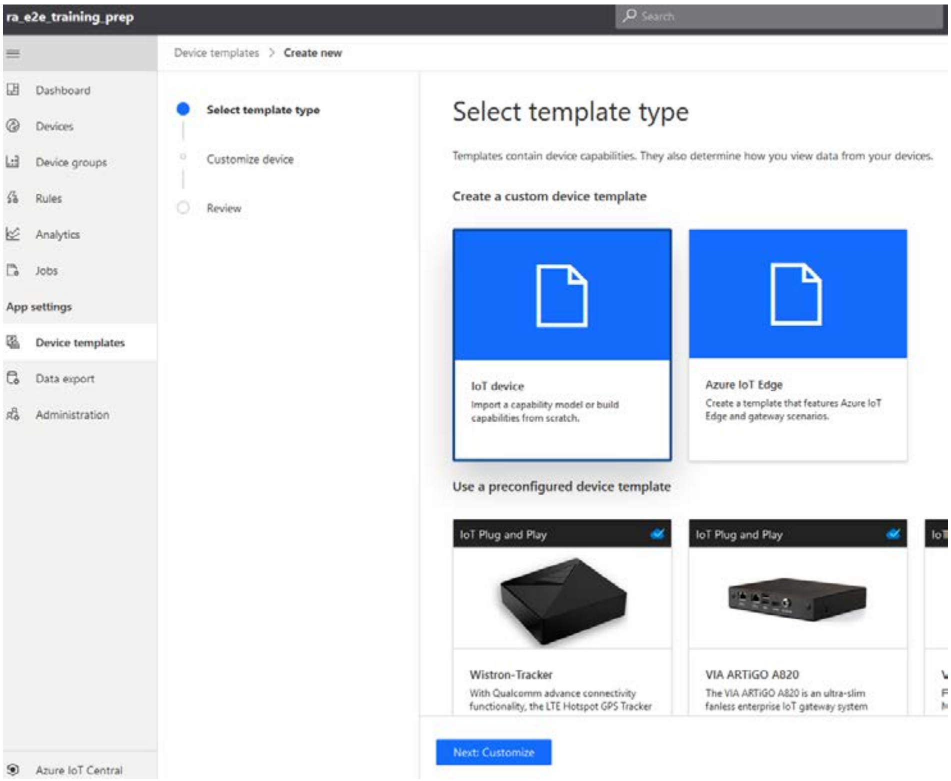


Figure 16. Creating a New Device Template

6. Select **IoT device** to create a custom device template for an **IoT device** and click **Next: Customize** button at the bottom of the page.



Templates contain device capabilities. They also determine how you view data from your devices.

Import a capability model or build capabilities from scratch.

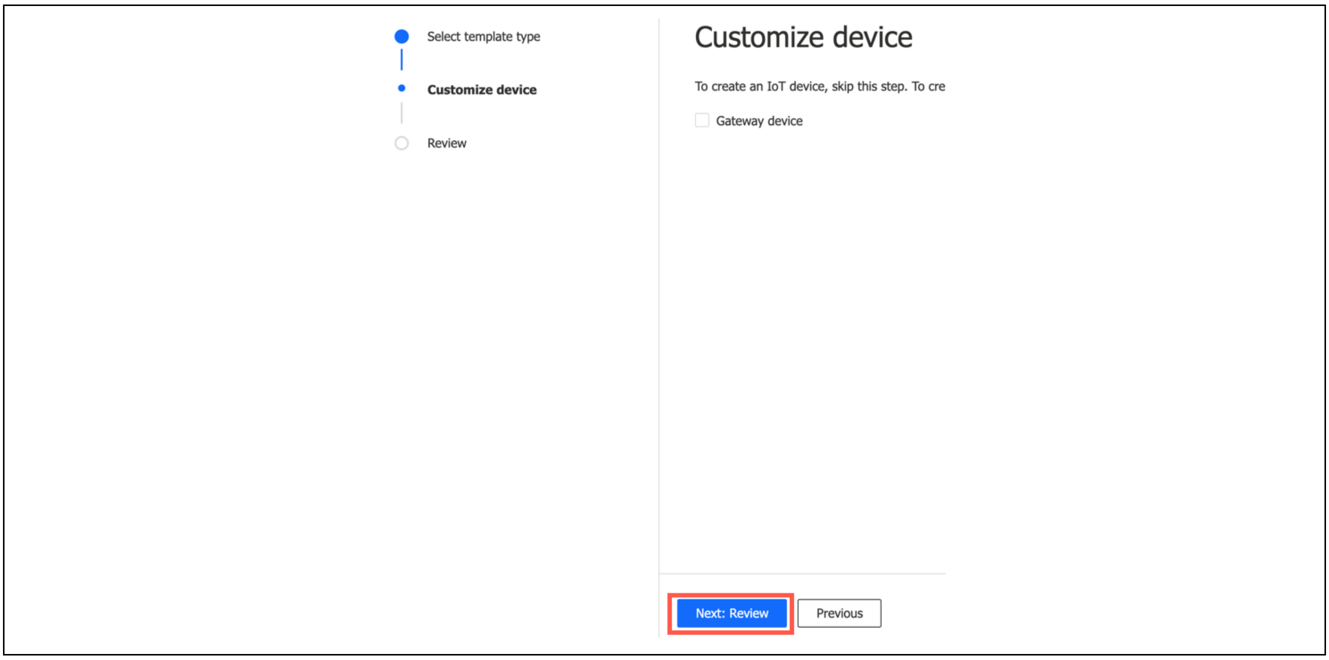
Create a template that features Azure IoT Edge and gateway scenarios.

With Qualcomm advance connectivity functionality, the LTE Hotspot GPS Tracker

The VIA ARTiGO A820 is an ultra-slim fanless enterprise IoT gateway system

Figure 17. Customizing the Device Template

7. On the next screen, we are not configuring a gateway device, so skip this step and click the **Next: Review** button at the bottom of the page.



Customize device

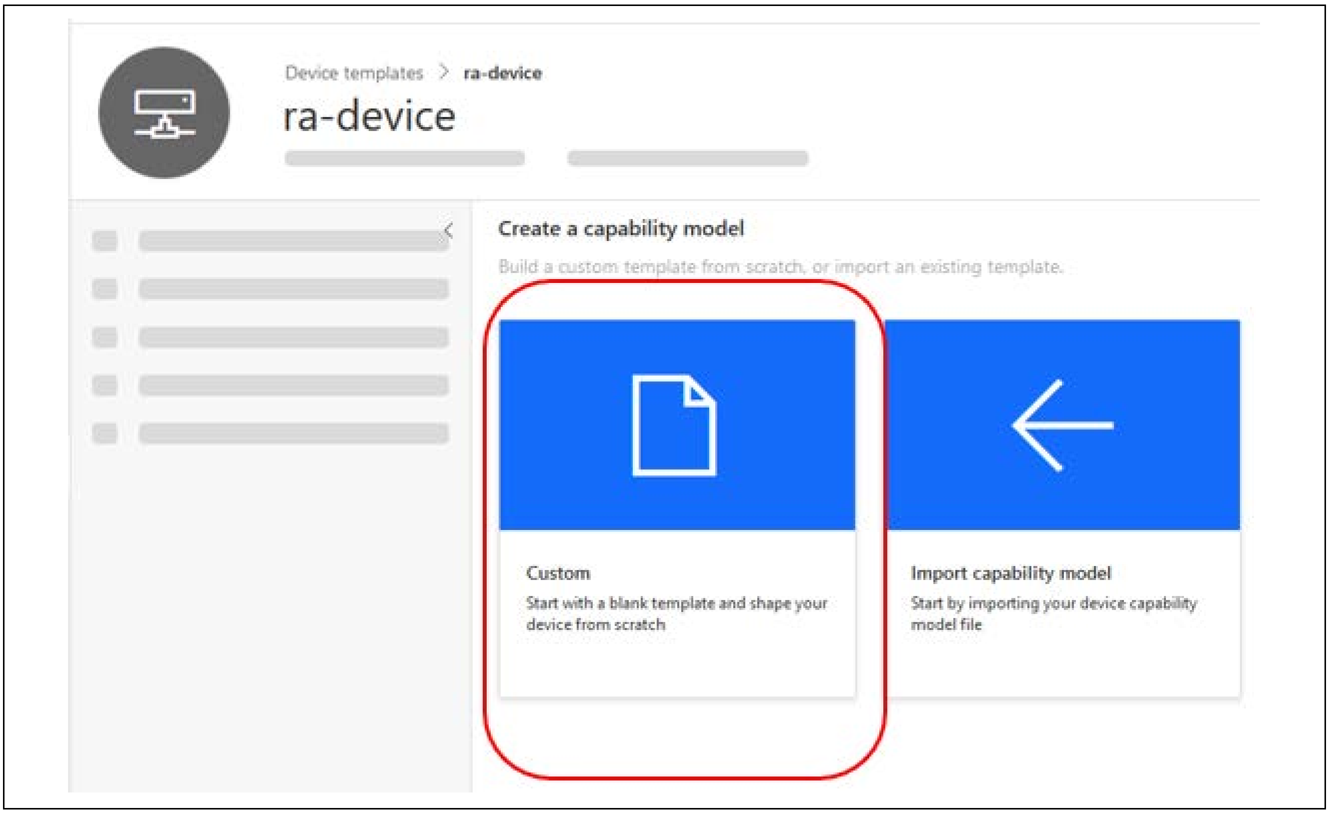
To create an IoT device, skip this step. To cre

□ Gateway device

Figure 18. Clicking Next: Review Button

8. On the next page (Review), Click on **Create** button. Enter a name for your device template and press **Enter**.

9. Next, create a capability model for our IoT device. Click on **Custom** button.



Create a capability model

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

Custom

Start with a blank template and shape your device from scratch

Import capability model

Start by importing your device capability model file

Figure 19. Creating a Custom Capability Model

10. Now you should create an interface for our capability model. Click **+ Add interface** button.

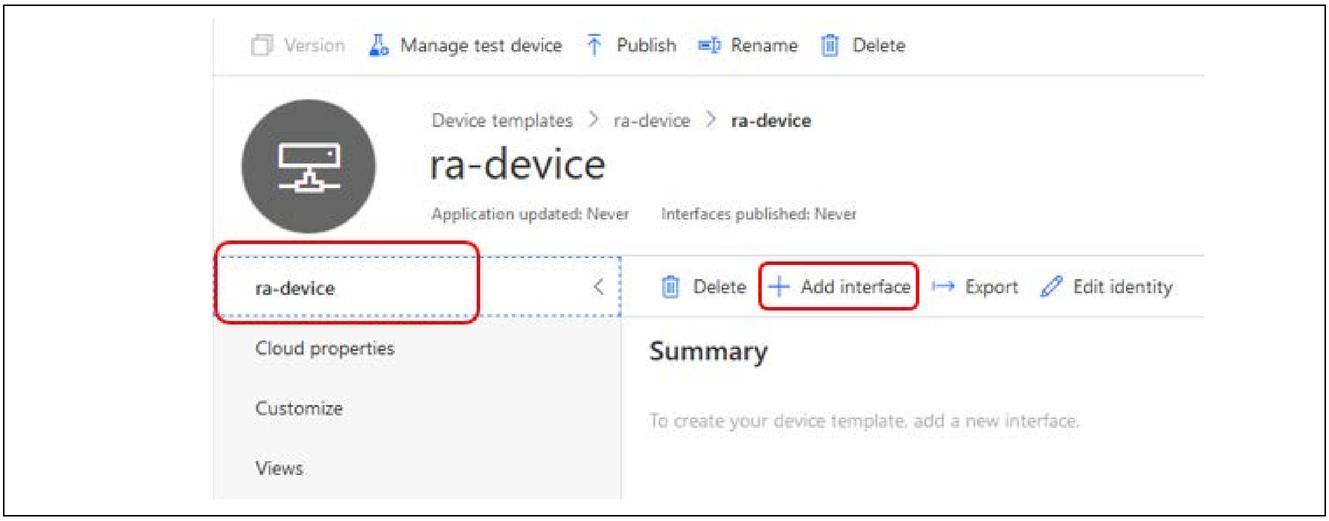

Version
Manage test device
Publish
Rename
Delete
Device templates > ra-device > ra-device
ra-device
Application updated: Never
Interfaces published: Never
ra-device
Delete
Add interface
Export
Edit identity
Cloud properties
Summary
Customize
To create your device template, add a new interface.
Views

Figure 20. Creating an Interface for Capability Model

11. Choose Custom interface as shown below.

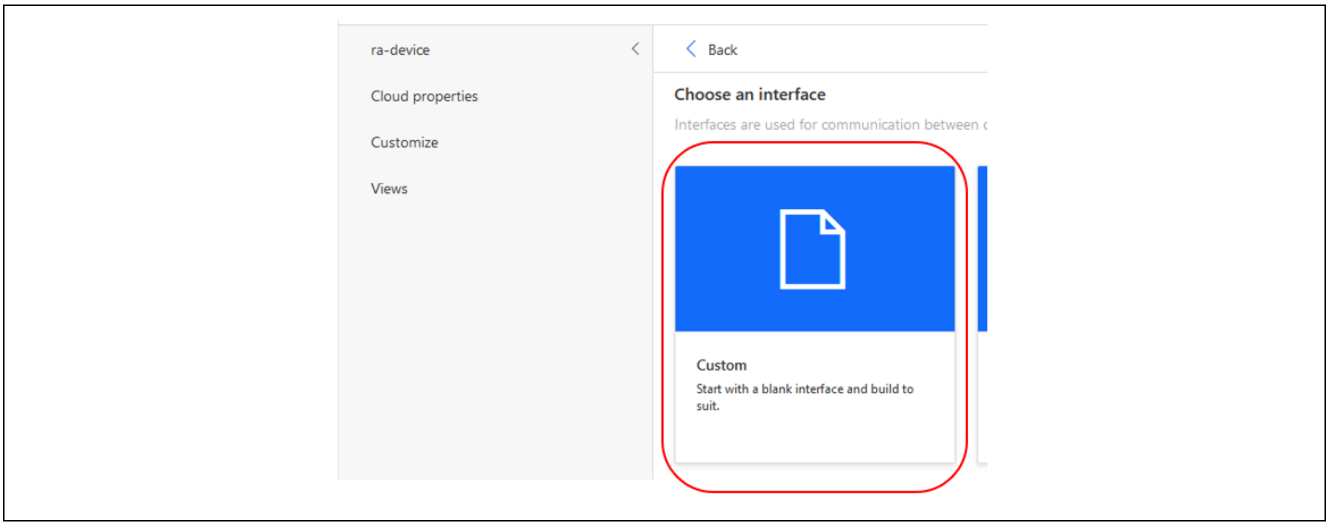

ra-device
Back
Cloud properties
Choose an interface
Interfaces are used for communication between
Customize
Views
Custom
Start with a blank interface and build to suit.

Figure 21. Choosing a Custom Interface

12. Click the **+ Add capability** button to create capabilities.

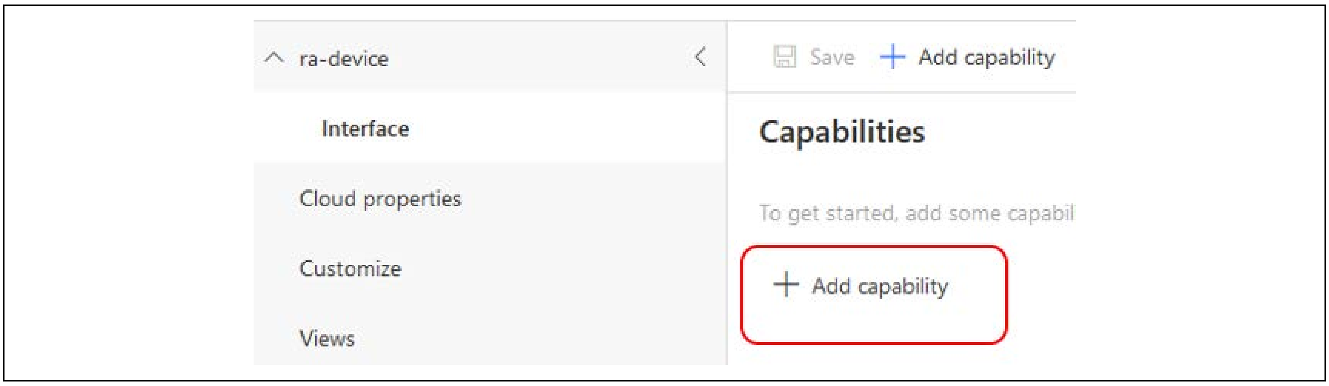


Figure 22. Clicking Capability Button

13. Start creating the capabilities for the temperature sensor data as shown below. Start with the temperature capability. Click **+ Add capability** button to add new sensor data capabilities.

14. For this application project, use the following properties:

Temperature Sensor:

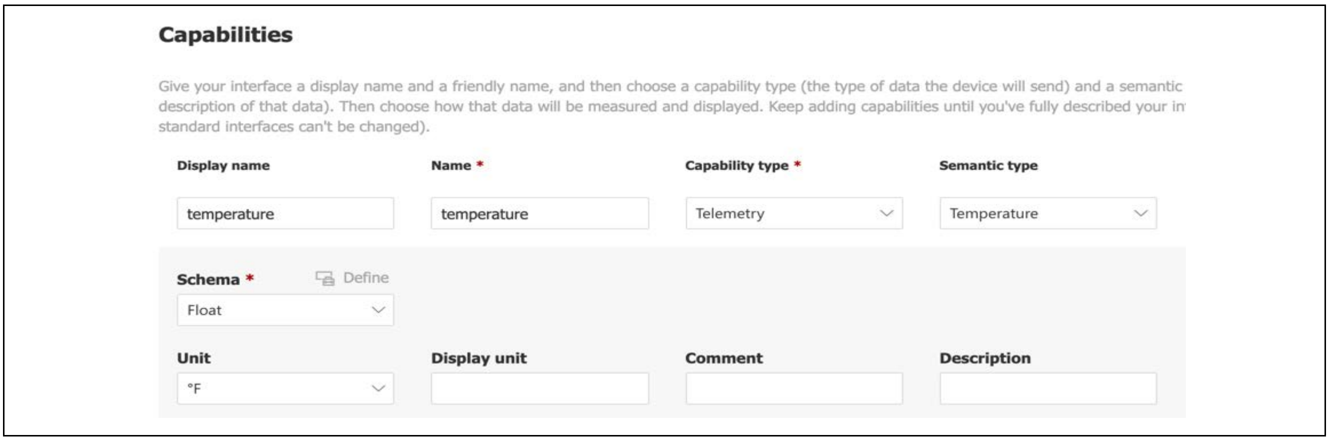
Display Name:	“temperature”
Capability type:	“Telemetry”
Semantic type:	“Temperature”
Schema:	“Float”
Unit:	“°F”

Manufacturer String:

Display Name:	“manufacturer”
Capability type:	“Property”

Toggle LED:

Display Name:	“led_toggle”
Capability type:	“Command”
Command:	“synchronous”



Capabilities

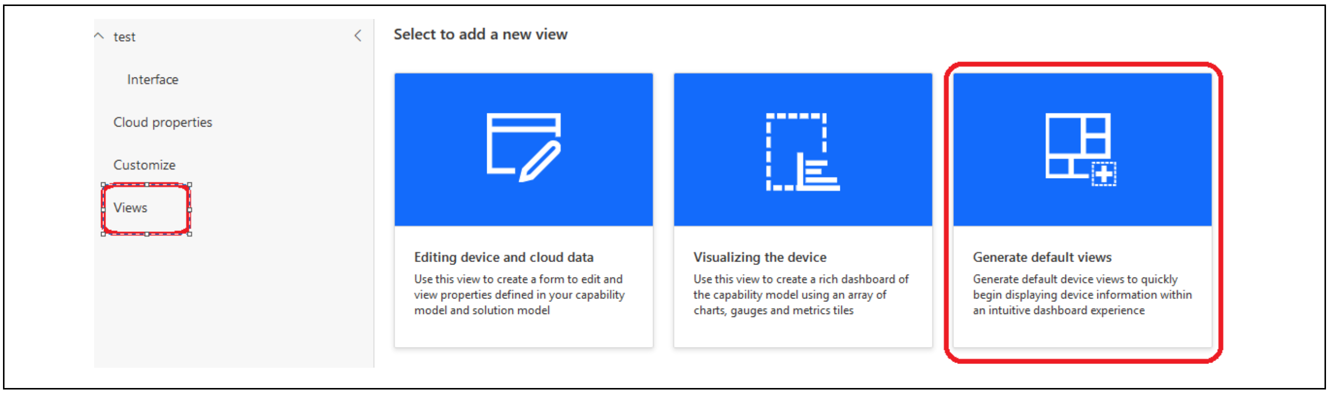
Give your interface a display name and a friendly name, and then choose a capability type (the type of data the device will send) and a semantic description of that data). Then choose how that data will be measured and displayed. Keep adding capabilities until you've fully described your in standard interfaces can't be changed).

Display name: temperature
Name *: temperature
Capability type *: Telemetry
Semantic type: Temperature
Schema *: Float
Unit: °F
Display unit:
Comment:
Description:

Figure 23. Selecting Capabilities for the Application Project

15. Click the **Save button** to save the capabilities.

16. Click **Views** tab to **Generate default views** for the device to quickly begin displaying device information.



Select to add a new view

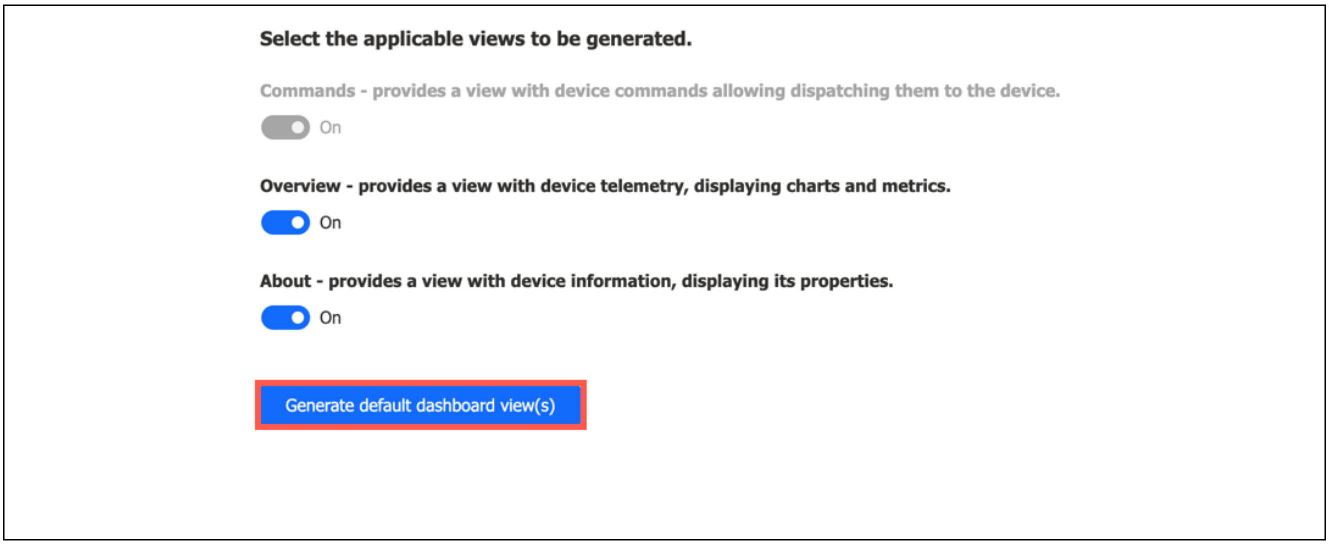
Editing device and cloud data
Use this view to create a form to edit and view properties defined in your capability model and solution model

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

Figure 24. Generating Default Views for the Device

17. In the next window, click **Generate default dashboard view(s)** button as shown below.



Select the applicable views to be generated.

Commands - provides a view with device commands allowing dispatching them to the device.

On

Overview - provides a view with device telemetry, displaying charts and metrics.

On

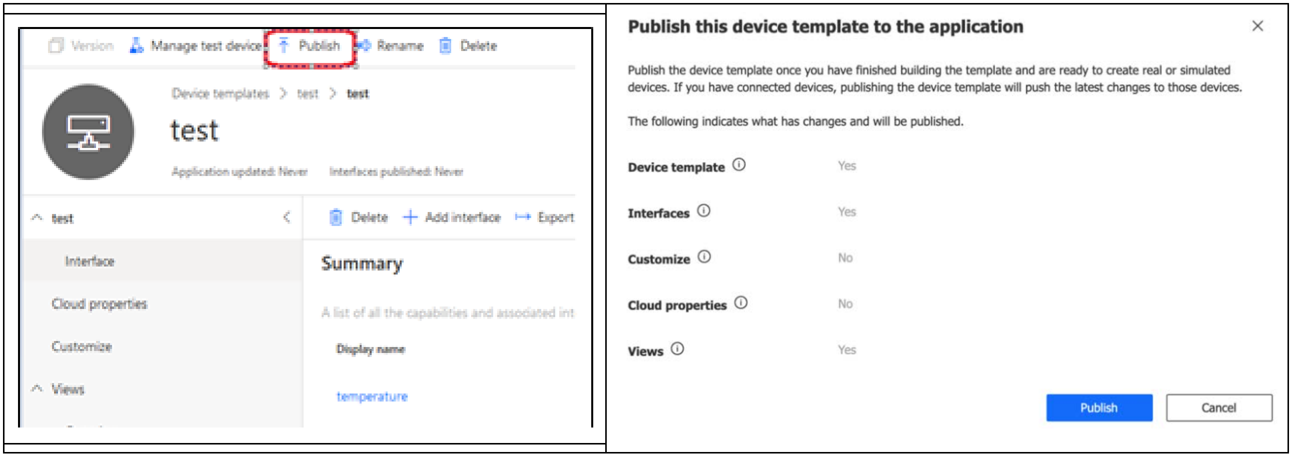
About - provides a view with device information, displaying its properties.

On

Figure 25. Clicking 'Generate default dashboard view(s)' Button

18. Publish the newly created device template. To publish, click the **Publish** button as highlighted below and then click **Publish** again in the pop-up window.

    Note: Once the template is published it cannot be edited.



Publish this device template to the application

Publish the device template once you have finished building the template and are ready to create real or simulated devices. If you have connected devices, publishing the device template will push the latest changes to those devices.

The following indicates what has changes and will be published.

Device template	Yes
Interfaces	Yes
Customize	No
Cloud properties	No
Views	Yes

Figure 26. Publishing the Newly Created Device Template

At this stage, the device templates which includes the capability model and corresponding interfaces are created and the corresponding view options are configured. Now we need to create Devices. To create devices, click **Devices** tab, followed by **+ New** option as highlighted below.

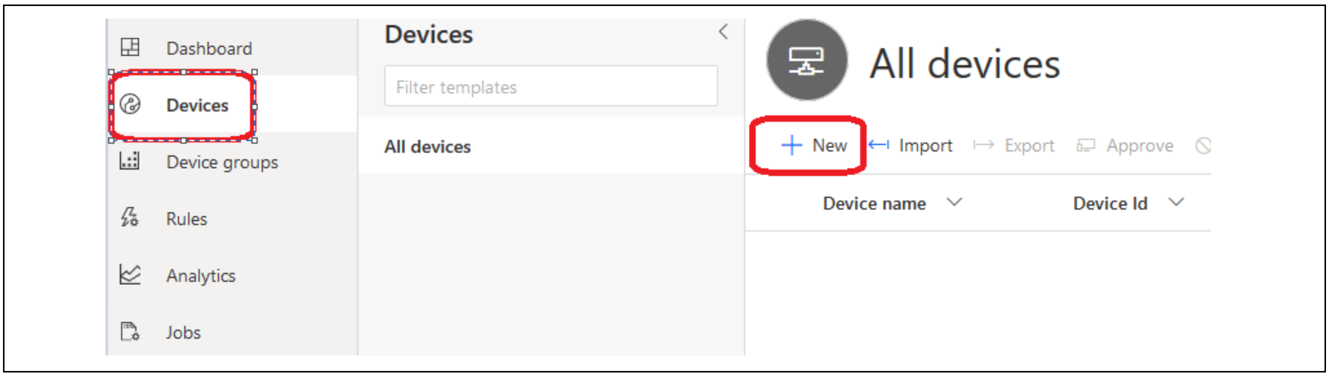
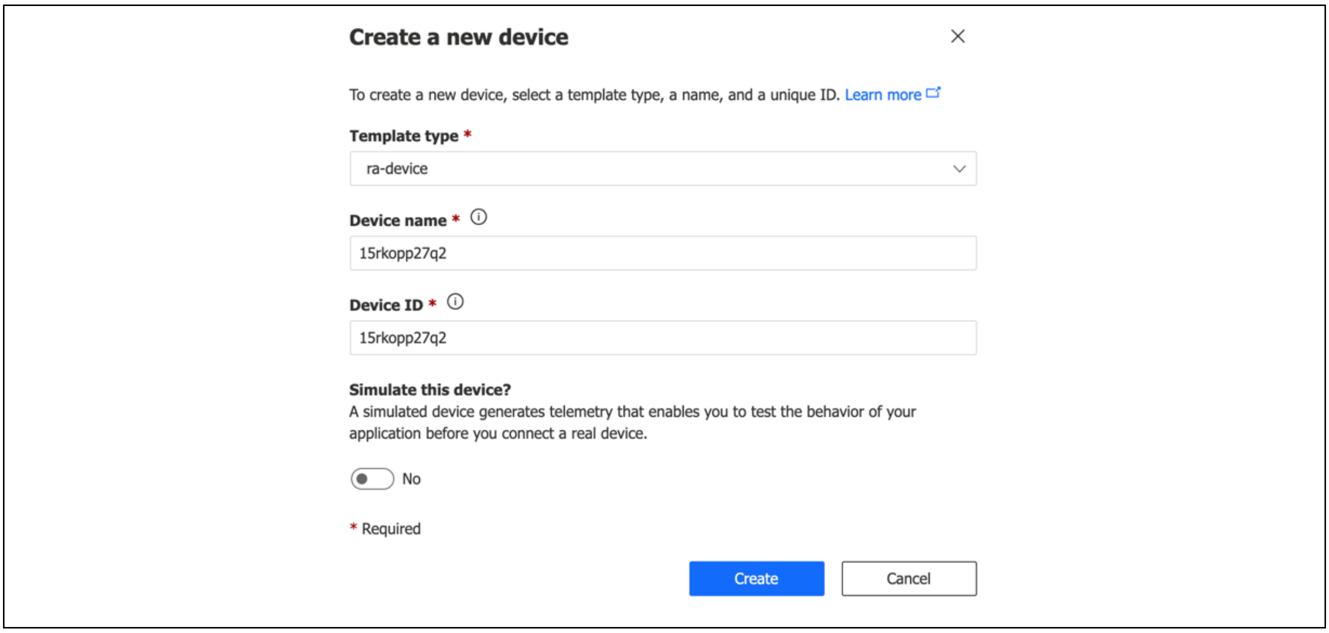


Figure 27. Creating Devices Option

19. A pop-up window appears, choose the **Template type** for the device template we just created. Leave the default value for the **Device name** and click the **Create** button.



Create a new device

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

Template type *: ra-device

Device name *: 15rkopp27q2

Device ID *: 15rkopp27q2

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

Figure 28. Choosing Template Type and Creating New Device

20. Click on the newly created device and click **Connect** button as highlighted below.

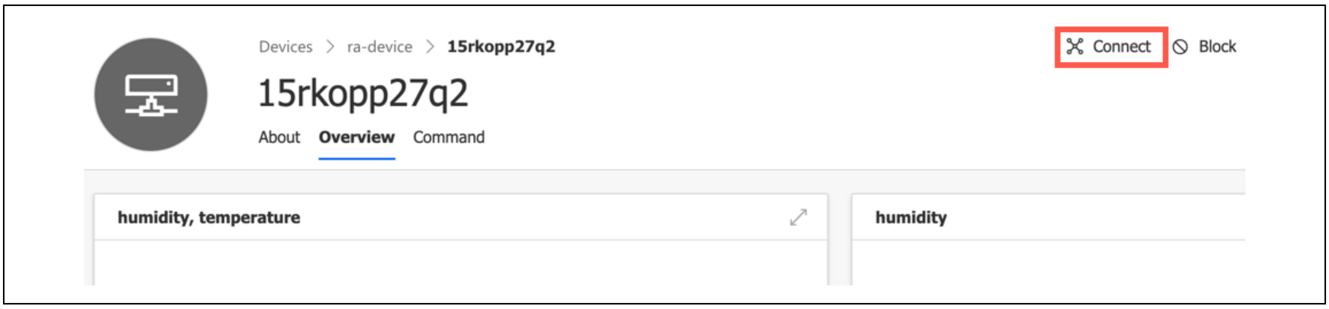
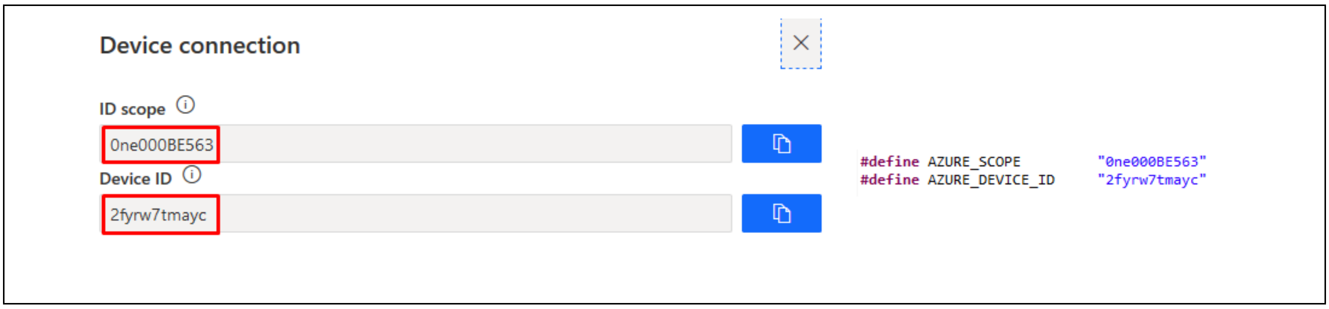


Figure 29. Connecting the Newly Created Device

A new pop-up window appears with Device Connection information. This information needs to match with the corresponding values in our e^2 studio project. Open *user_cfg.h* file in e^2 studio and copy the **Scope ID** to **AZURE_SCOPE** macro, **Device ID** to **AZURE_DEVICE_ID** macro. On the Azure side, it's recommended to use the blue copy buttons to reduce the chance of errors in matching these values from Azure to the application project.



```
#define AZURE_SCOPE         "0ne000BE563"
#define AZURE_DEVICE_ID     "2fyrw7tmayc"
```

Figure 30. Device Connection Information

5. Setting up X.509 Certificate based Authentication

To enable X.509 Certificate based authentication mechanism, follow the instructions mentioned in the following sub-sections to generate device credentials that will be used in this project.

5.1 Setting up Environment

1. Download PowerShell script for creating test CA and leaf certificates. Go to the script file on **CA tools** page in **Azure IoT SDK repository**: https://github.com/Azure/azure-iot-sdk-c/blob/master/tools/CACertificates/ca-certs.ps1. Right-click *Raw* and select **Save link as**… (in Google chrome) or equivalent (other browsers).

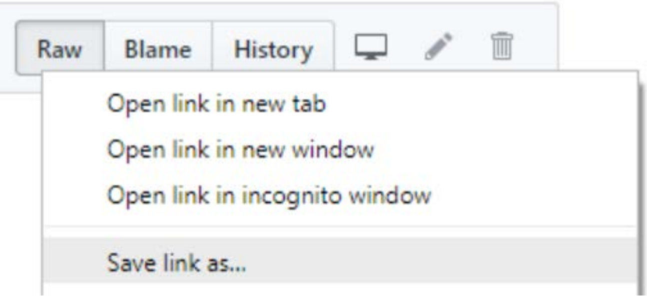


Figure 31. Save Link as Option

2. Save the file into designated work folder. This folder will later be used to output certificates for Azure IoT Central and the RA device.

3. If your machine already has *OpenSSL* installed, you can go directly to step 5.1.5. *OpenSSL* is typically installed with other commonly used utilities such as *Git for Windows*.

4. Download **Win32** or **Win64 OpenSSL** from https://slproweb.com/products/Win32OpenSSL.html and run the installer. Make note of the installation directory specified in the wizard.

5. Run Windows PowerShell as an administrator. Open Start Menu, type *powershell* and right-click *Windows* **PowerShell** to select **Run as Administrator**.

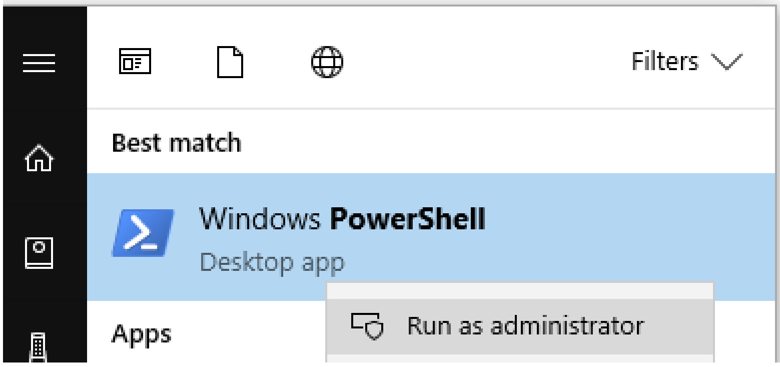
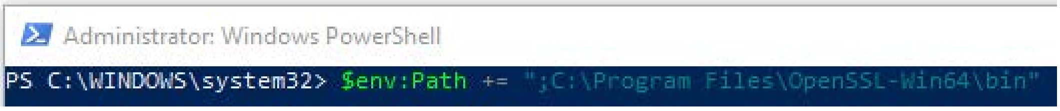


Figure 32. Running Windows PowerShell as an Administrator

6. In the PowerShell window, type `openssl` and hit **Enter**. If red error message is shown, OpenSSL path is not present in the system PATH variable. If no error is shown and OpenSSL shell starts executing, write q, press Enter and skip to step 8.

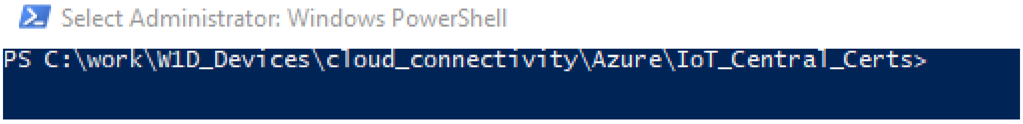
7. Add OpenSSL directory to temporary PATH variable in current PowerShell session. Execute command shown below, where text inside quotation marks after the semicolon is replaced with path to bin directory inside OpenSSL installation directory (specified in step 4):



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> $env:Path += ";C:\Program Files\OpenSSL-Win64\bin"
```

Figure 33. ..Path to Bin Directory Inside OpenSSL

8. Change active *PowerShell* directory to work folder used in step 2. Use *cd* command followed by directory path and press *Enter*. Include quotation marks at the start and end of the path if the path includes any whitespaces.

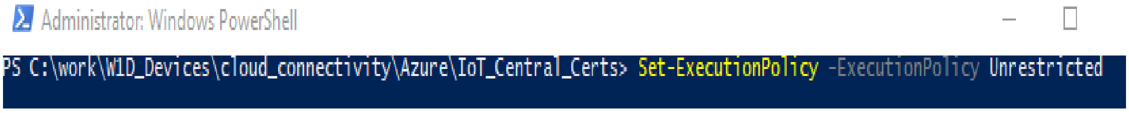


```
Select Administrator: Windows PowerShell
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs>
```

Figure 34. Changing Active PowerShell Directory to Work Folder

9. Get current execution policy by executing *Get-ExecutionPolicy* command and make note of the output.

10. If output from step 9 is other than Unrestricted, execute command below to set ExecutionPolicy. Original setting can be restored after CA and leaf certificates are generated (in section 4.3). Using Unrestricted execution policy will let PowerShell execute script used in this application project.

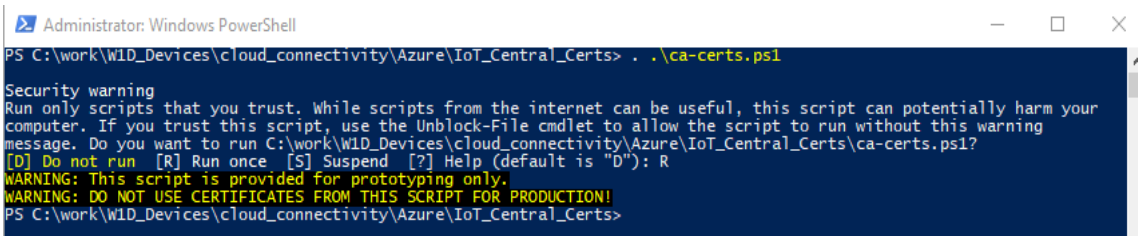


```
Administrator: Windows PowerShell
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs> Set-ExecutionPolicy -ExecutionPolicy Unrestricted
```

Figure 35. Get-ExecutionPolicy Command

11. Confirm changes to execution policy by typing y and pressing **Enter**.

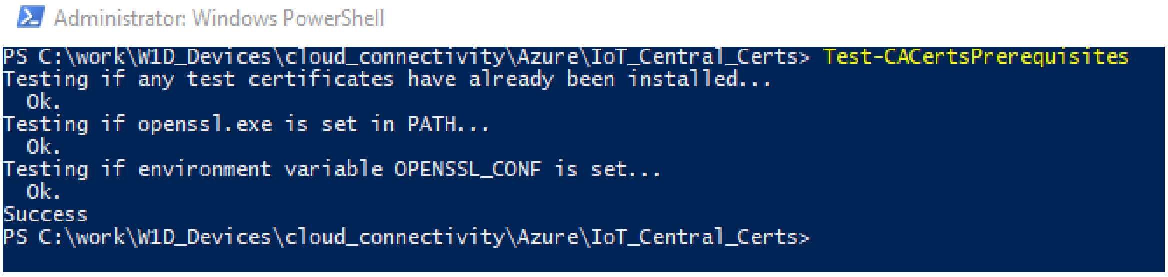
12. Execute the script with `. .\ca-certs.ps1` (notice the first two dot characters before backward-slash). This will introduce several new functions into the current PowerShell session. When prompted to confirm script execution, press **r** and hit **Enter**.



```
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs> . .\ca-certs.ps1

Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning
message. Do you want to run C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs\ca-certs.ps1?
[D] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"): R
WARNING: This script is provided for prototyping only.
WARNING: DO NOT USE CERTIFICATES FROM THIS SCRIPT FOR PRODUCTION!
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs>
```

Figure 36. Executing Script with '`. .\ca-certs.ps1`'

13. Run `Test-CACertsPrerequisites` command. First two tests should return ok. Final test will return an error associated with missing OPENSSL_CONF definition. Since OpenSSL will use built-in defaults under such circumstances, this error message can be ignored.

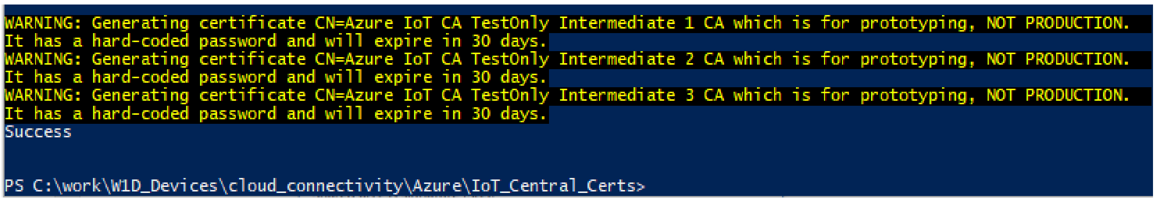


```
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs> Test-CACertsPrerequisites
Testing if any test certificates have already been installed...
  Ok.
Testing if openssl.exe is set in PATH...
  Ok.
Testing if environment variable OPENSSL_CONF is set...
  Ok.
Success
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs>
```

Figure 37. Running 'Test-CACertsPrerequisites' Command

5.2 Creating and Verifying a Certificate Chain

1. Create root certificate, type `New-CACertsCertChain rsa` into *PowerShell* and confirm with **Enter**. Verify that the function outputs `Success`, as shown below. The output certificates will be added to the *Windows Certificate store* automatically and can be removed at the end of this application project.



```
WARNING: Generating certificate CN=Azure IoT CA TestOnly Intermediate 1 CA which is for prototyping, NOT PRODUCTION.
It has a hard-coded password and will expire in 30 days.
WARNING: Generating certificate CN=Azure IoT CA TestOnly Intermediate 2 CA which is for prototyping, NOT PRODUCTION.
It has a hard-coded password and will expire in 30 days.
WARNING: Generating certificate CN=Azure IoT CA TestOnly Intermediate 3 CA which is for prototyping, NOT PRODUCTION.
It has a hard-coded password and will expire in 30 days.
Success


PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs>
```

Figure 38. Creating the Root Certificate

2. Go to the **IoT Central application** in the web browser and select **Administration** from the menu on the left side. Once in the **Administration** view, select **Device Connection**.

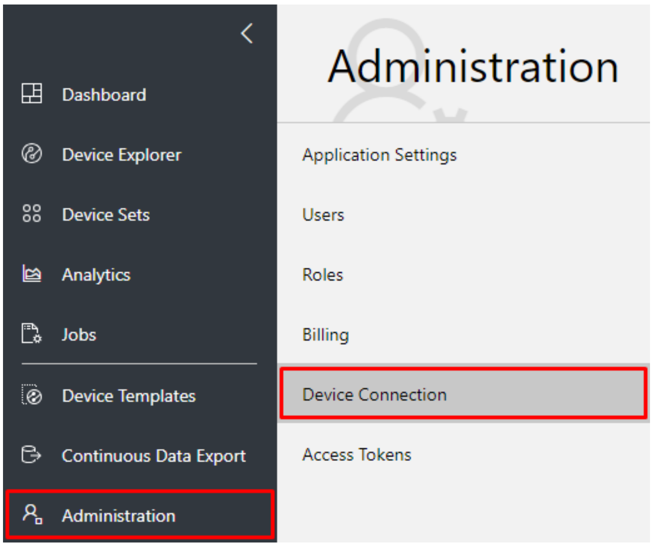
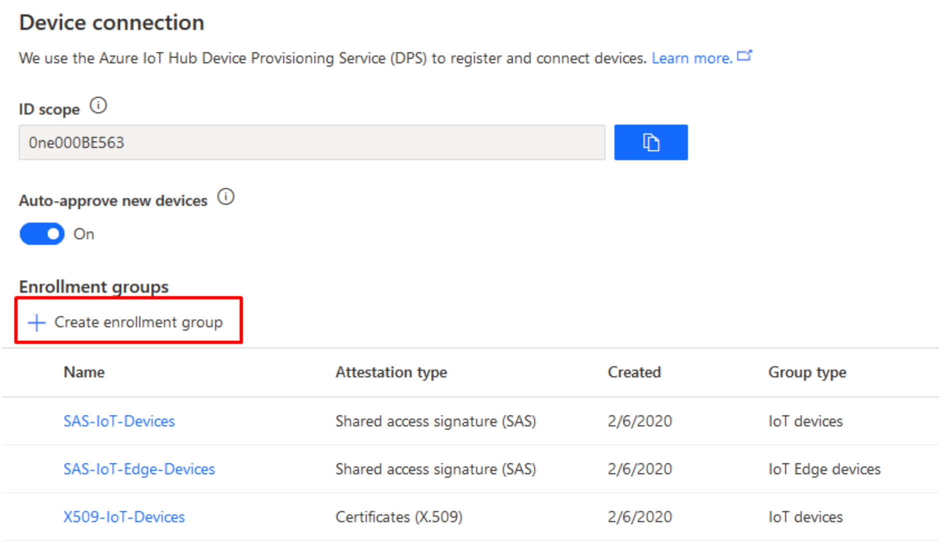


Figure 39. Selecting the Device Connection

3. Click ‘**+ Create enrollment group**’ under Device Connection.



Device connection

We use the Azure IoT Hub Device Provisioning Service (DPS) to register and connect devices. Learn more.

ID scope ⓘ

0ne000BE563

Auto-approve new devices ⓘ

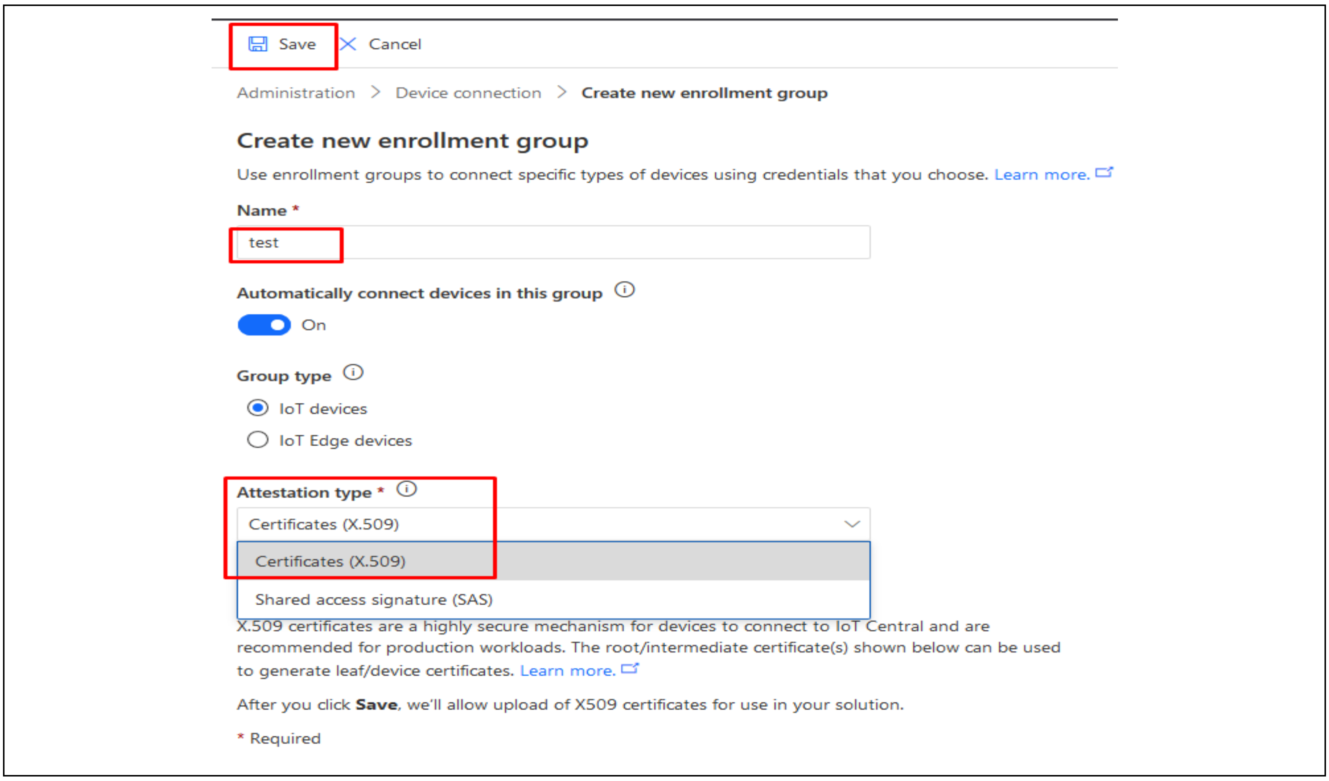
On

Enrollment groups

Name	Attestation type	Created	Group type
SAS-IoT-Devices	Shared access signature (SAS)	2/6/2020	IoT devices
SAS-IoT-Edge-Devices	Shared access signature (SAS)	2/6/2020	IoT Edge devices
X509-IoT-Devices	Certificates (X.509)	2/6/2020	IoT devices

Figure 40. Creating the Enrollment Group

4. Enter the **Name** for the new enrollment group, **Group type** as **IoT devices**, choose the Attestation type as **Certificate (X.509)** option and **Save** button.



Save
Cancel
Administration > Device connection > Create new enrollment group

Create new enrollment group

Use enrollment groups to connect specific types of devices using credentials that you choose. Learn more.

Name *

test

Automatically connect devices in this group

On

Group type

● IoT devices

○ IoT Edge devices

Attestation type *

Certificates (X.509)

Certificates (X.509)
Shared access signature (SAS)

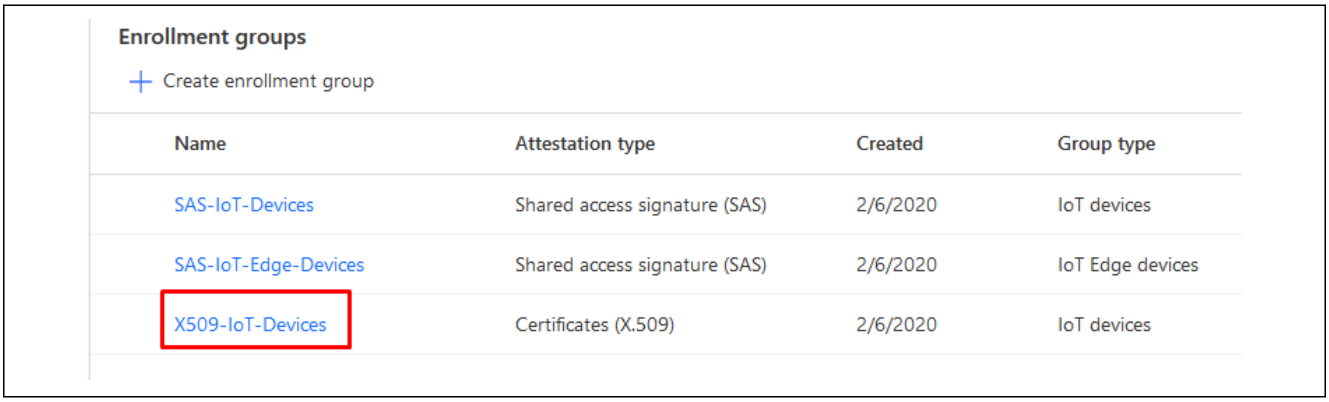
X.509 certificates are a highly secure mechanism for devices to connect to IoT Central and are recommended for production workloads. The root/intermediate certificate(s) shown below can be used to generate leaf/device certificates. Learn more.

After you click **Save**, we'll allow upload of X509 certificates for use in your solution.

* Required

Figure 41. Selecting the Options for the New Enrollment Group

5. Open the **enrollment group** just created as shown below.

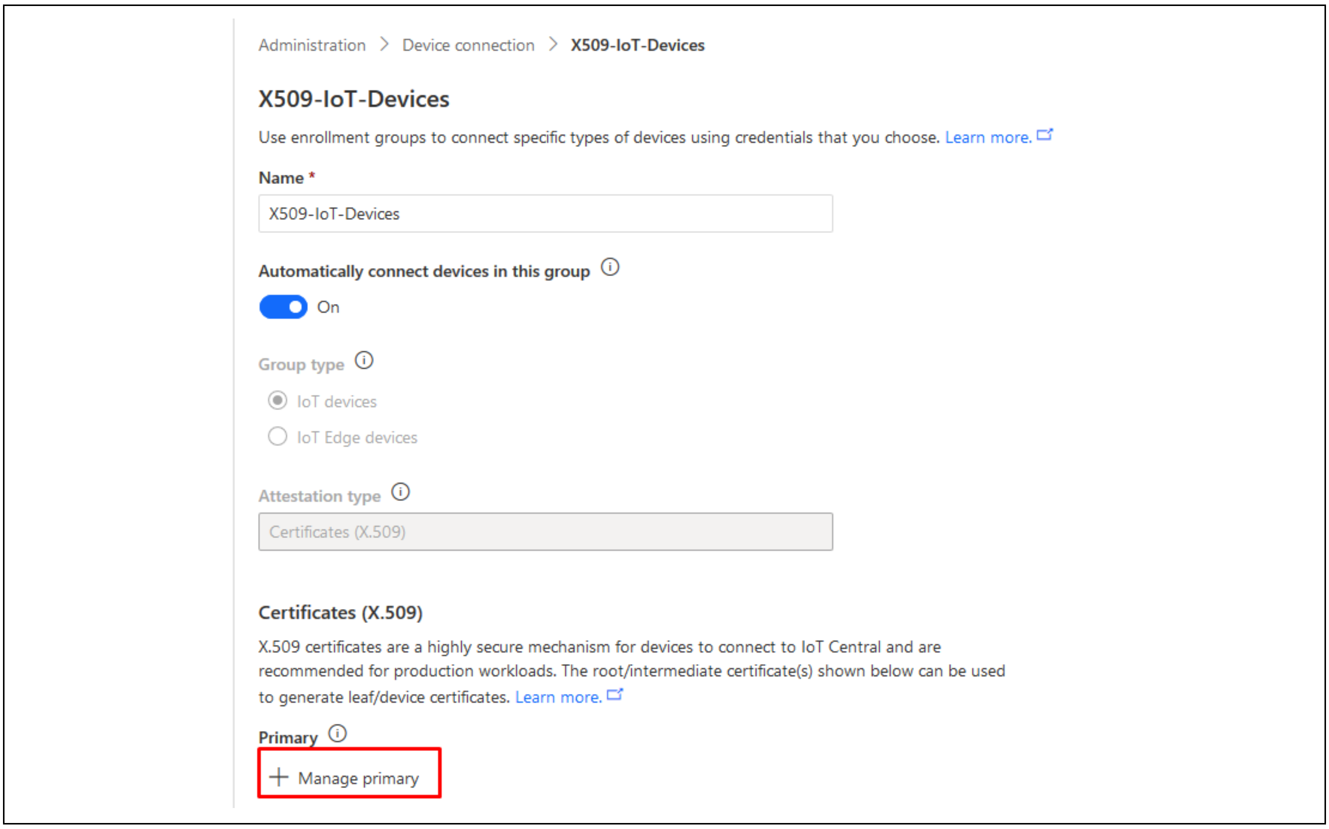


Enrollment groups

Name	Attestation type	Created	Group type
SAS-IoT-Devices	Shared access signature (SAS)	2/6/2020	IoT devices
SAS-IoT-Edge-Devices	Shared access signature (SAS)	2/6/2020	IoT Edge devices
X509-IoT-Devices	Certificates (X.509)	2/6/2020	IoT devices

Figure 42. Opening the Enrollment Group Created

6. Click **+ Manage primary** option as shown below.



Administration > Device connection > X509-IoT-Devices

X509-IoT-Devices

Use enrollment groups to connect specific types of devices using credentials that you choose. Learn more.

Name * X509-IoT-Devices

Automatically connect devices in this group: On

Group type: ● IoT devices ○ IoT Edge devices

Attestation type: Certificates (X.509)

Certificates (X.509)

X.509 certificates are a highly secure mechanism for devices to connect to IoT Central and are recommended for production workloads. The root/intermediate certificate(s) shown below can be used to generate leaf/device certificates. Learn more.

Primary

Figure 43. Clicking the '+Manage primary' Option

7. Click on **Upload** button for **Primary** certificate.

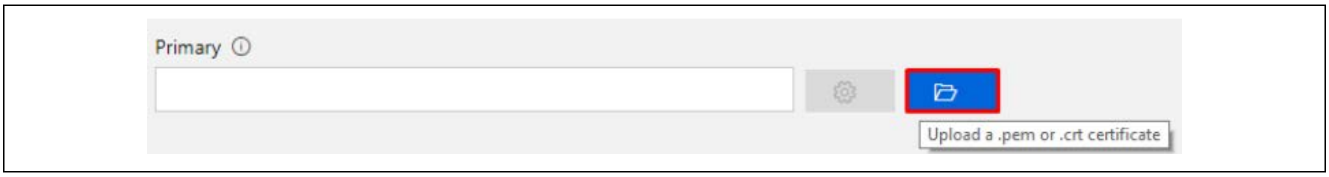
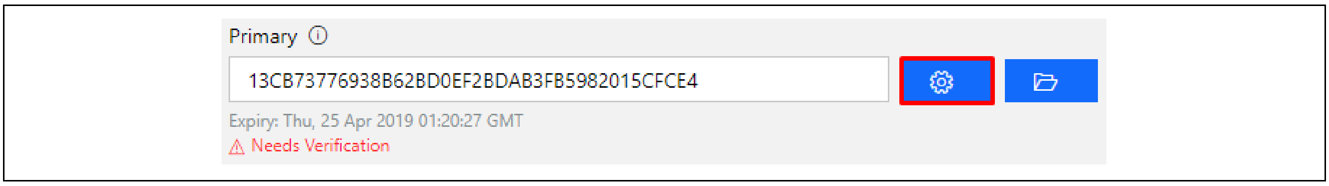


Figure 44. Clicking 'Upload' Button for the Primary Certificate

8. File upload prompt will now open. Navigate to the work folder used in step **1** and select *RootCA.pem* file.

9. Azure will parse the root certificate and will report that it needs to be verified. Click on the **Gear icon** to bring up **Primary Certificate** pop-up.



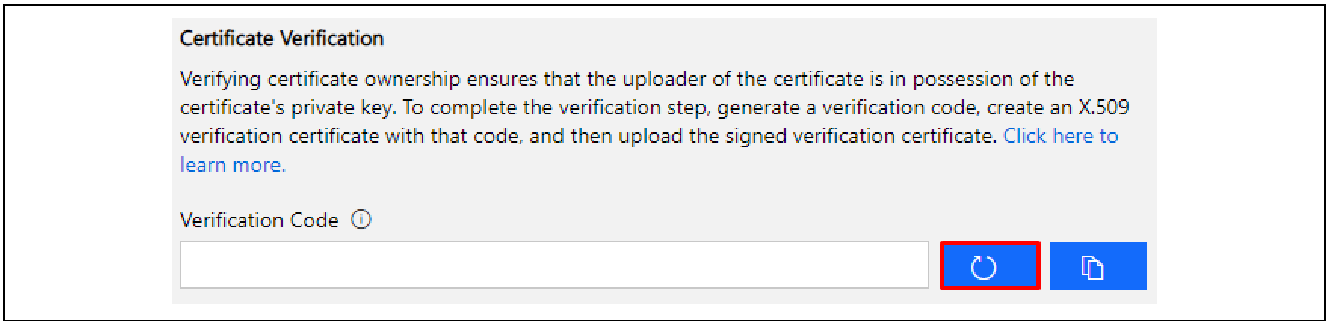
Primary: 13CB73776938B62BD0EF2BDAB3FB5982015CFCE4

Expiry: Thu, 25 Apr 2019 01:20:27 GMT

⚠ Needs Verification

Figure 45. Primary Certificate Pop-up

10. In the **Certificate Verification** section of the window, press *Refresh* button to generate ***Verification Code*** against the root certificate.



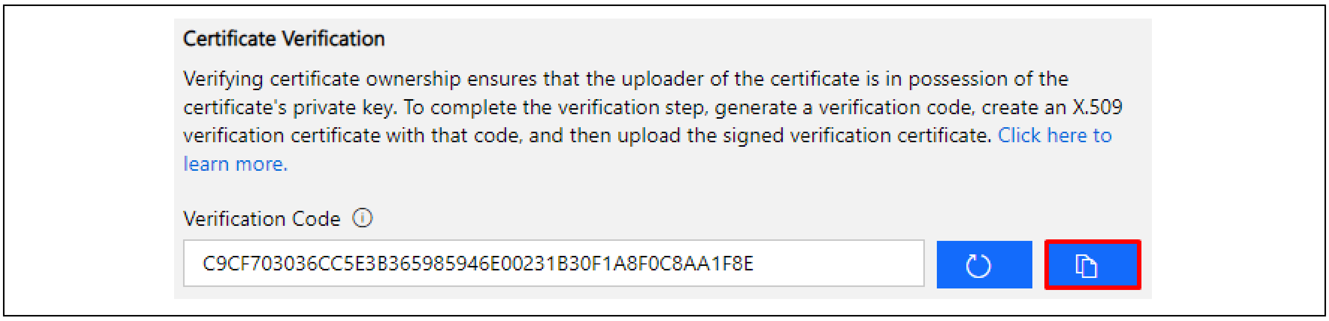
Certificate Verification

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 ⓘ

Figure 46. Generating Verification Code

11. Copy the generated **Verification Code** into the clipboard. Use **Copy button** to avoid mistakes.



Certificate Verification

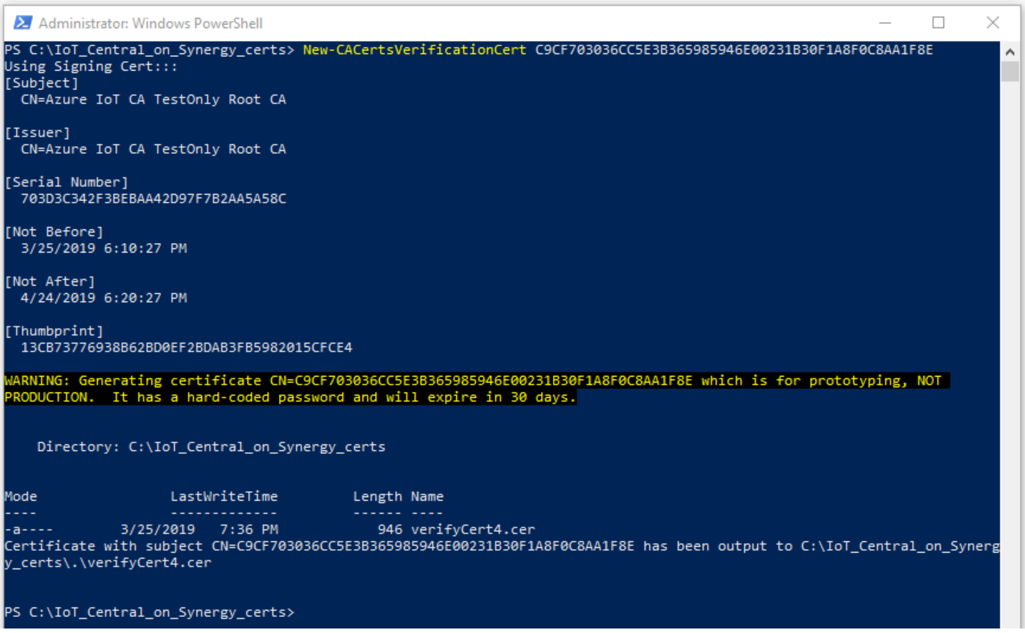
Verifying certificate ownership ensures that the uploader of the certificate is in possession of the certificate's private key. To complete the verification step, generate a verification code, create an X.509 verification certificate with that code, and then upload the signed verification certificate.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

Verification Code ⓘ

C9CF703036CC5E3B365985946E00231B30F1A8F0C8AA1F8E

Figure 47. Certificate Verification Window

12. Keep **Primary Certificate** pop-up open in the browser and go back to the **PowerShell window**. Type `New-CACertsVerificationCert` followed by space and the verification code copied to the clipboard in previous step. Right-clicking inside *PowerShell* window will paste content from the clipboard automatically. Confirm command execution by hitting *Enter*.

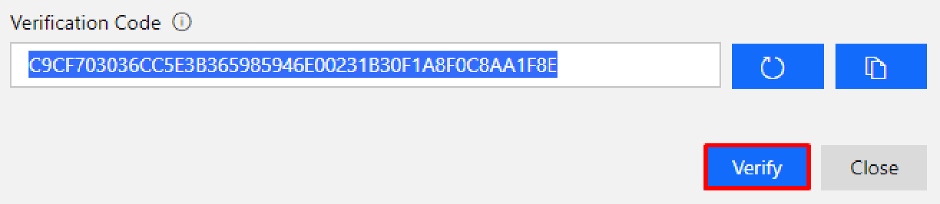


Administrator: Windows PowerShell

```
PS C:\IoT_Central_on_Synergy_certs> New-CACertsVerificationCert C9CF703036CC5E3B365985946E00231B30F1A8F0C8AA1F8E
Using Signing Cert:::
[Subject]
  CN=Azure IoT CA TestOnly Root CA

[Issuer]
  CN=Azure IoT CA TestOnly Root CA

[Serial Number]
  703D3C342F3BEBAA42D97F7B2AA5A58C

[Not Before]
  3/25/2019 6:10:27 PM

[Not After]
  4/24/2019 6:20:27 PM

[Thumbprint]
  13CB73776938B62BD0EF2BDAB3FB5982015CFCE4

WARNING: Generating certificate CN=C9CF703036CC5E3B365985946E00231B30F1A8F0C8AA1F8E which is for prototyping, NOT
PRODUCTION.  It has a hard-coded password and will expire in 30 days.


    Directory: C:\IoT_Central_on_Synergy_certs


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        3/25/2019   7:36 PM            946 verifyCert4.cer
Certificate with subject CN=C9CF703036CC5E3B365985946E00231B30F1A8F0C8AA1F8E has been output to C:\IoT_Central_on_Synerg
y_certs\.\verifyCert4.cer


PS C:\IoT_Central_on_Synergy_certs>
```

Figure 48. Verifying Certification

13. Go back to the web browser where **Primary Certificate** pop-up should still be left open. Press **Verify** button at the bottom of the window.

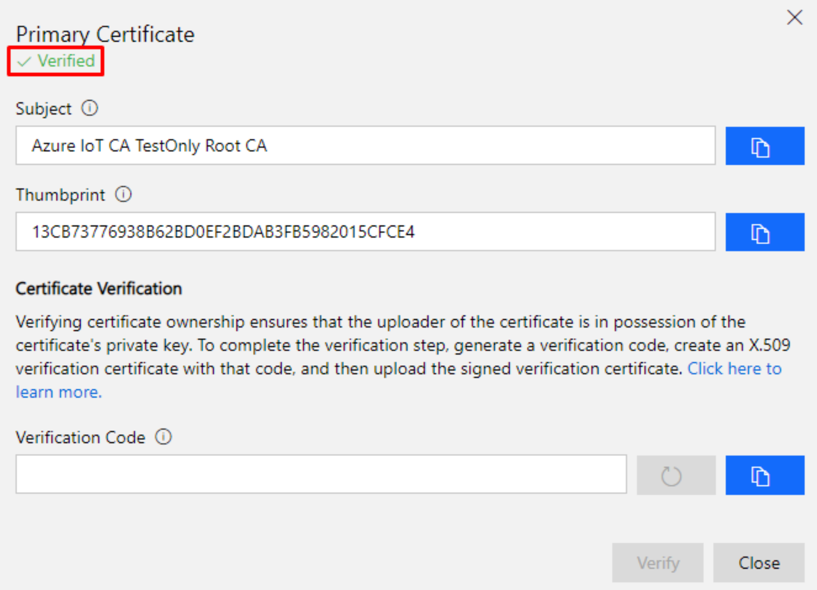


Verification Code ⓘ

C9CF703036CC5E3B365985946E00231B30F1A8F0C8AA1F8E

Figure 49. Pressing Verify Button

14. Another file upload prompt will open. Select `verifyCert4.cer` file and confirm choice. Azure will parse the uploaded file and will produce notification that the primary certificate is now verified. Click **Close** to go back to **IoT Central application**.



Primary Certificate

✓ Verified

Subject ⓘ

Azure IoT CA TestOnly Root CA

Thumbprint ⓘ

13CB73776938B62BD0EF2BDAB3FB5982015CFCE4

Certificate Verification

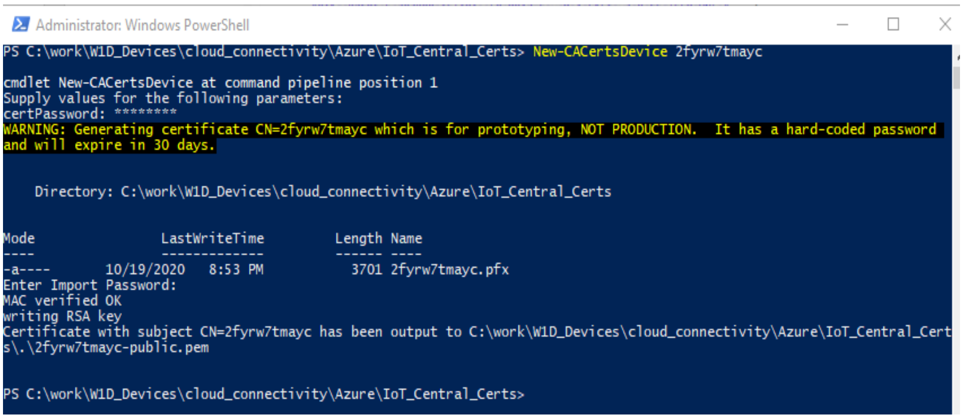
Verifying certificate ownership ensures that the uploader of the certificate is in possession of the certificate's private key. To complete the verification step, generate a verification code, create an X.509 verification certificate with that code, and then upload the signed verification certificate.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

Verification Code ⓘ

Figure 50. Primary Certificate Verification

5.3 Creating a Device Certificate

1. To create leaf certificate, type `New-CACertsDevice` followed by space and the device name into the **PowerShell console**. Use name (Device ID) provided on **IoT Central** in earlier sections (as specified in section 4, step 20). Function will prompt for user password twice – any text can be provided here however it must be the same on both occasions. Script will output certificate files to the work folder.



```
PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs> New-CACertsDevice 2fyrw7tmayc

cmdlet New-CACertsDevice at command pipeline position 1
Supply values for the following parameters:
certPassword: ********
WARNING: Generating certificate CN=2fyrw7tmayc which is for prototyping, NOT PRODUCTION.  It has a hard-coded password
and will expire in 30 days.


    Directory: C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       10/19/2020   8:53 PM           3701 2fyrw7tmayc.pfx
Enter Import Password:
MAC verified OK
writing RSA key
Certificate with subject CN=2fyrw7tmayc has been output to C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Cert
s\.\2fyrw7tmayc-public.pem


PS C:\work\W1D_Devices\cloud_connectivity\Azure\IoT_Central_Certs>
```

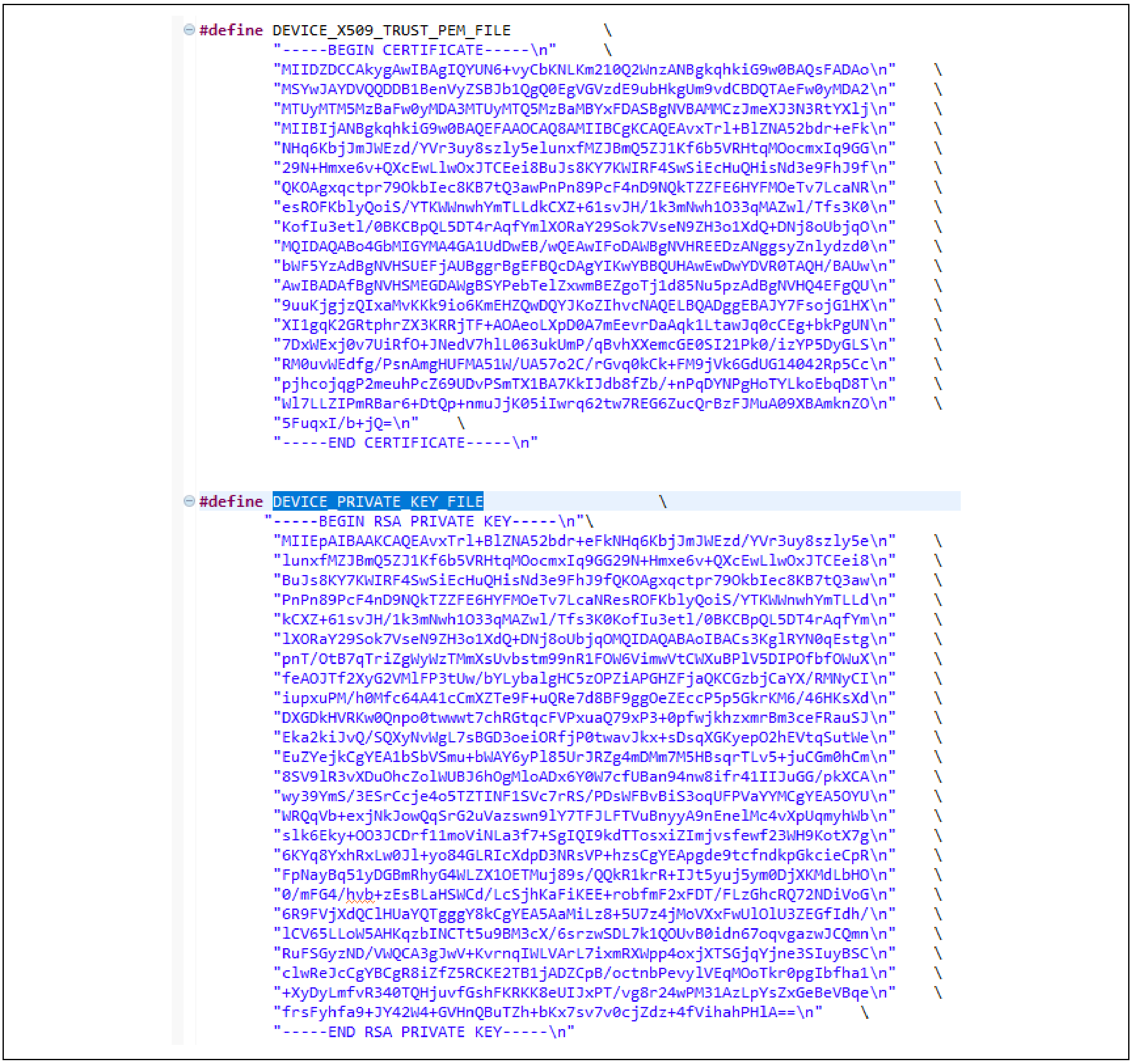
Figure 51. Creating Leaf Certificate

2. Open the device certificate **<device_id>-public.pem** file using Notepad++ or similar utility. Copy the contents and convert them into strings. Open **azure.h** file in src directory, copy the converted string to `DEVICE_X509_TRUST_PEM_FILE` macro. Refer to example shown in the following image.

   Note: By default, the application project contains a sample device certificate. Refer to the existing format and replace it with newly generated device certificate.

3. Open the device private key <device_id>-private.pem file using Notepad++ or similar utility. Copy the contents and convert them into strings. Open azure.h file in src directory, copy the converted string to `DEVICE_PRIVATE_KEY_FILE` macro. Refer to example shown in the following image.

   Note: By default, the application project contains a sample device private key. Refer to the existing format and replace it with newly generated device private key.

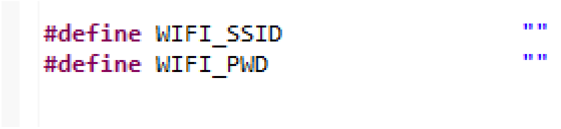


```
#define DEVICE_X509_TRUST_PEM_FILE          \
        "-----BEGIN CERTIFICATE-----\n"     \
        "MIIDZDCCAkygAwIBAgIQYUN6+vyCbKNLKm210Q2WnzANBgkqhkiG9w0BAQsFADAo\n"    \
        "MSYwJAYDVQQDDB1BenVyZSBJb1QgQ0EgVGVzdE9ubHkgUm9vdCBDQTAeFw0yMDA2\n"    \
        "MTUyMTM5MzBaFw0yMDA3MTUyMTQ5MzBaMBYxFDASBgNVBAMMCzJmeXJ3N3RtYXlj\n"    \
        "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvxTrl+BlZNA52bdr+eFk\n"    \
        "NHq6KbjJmJWEzd/YVr3uy8szly5elunxfMZJBmQ5ZJ1Kf6b5VRHtqMOocmxIq9GG\n"    \
        "29N+Hmxe6v+QXcEwLlwOxJTCEei8BuJs8KY7KWIRF4SwSiEcHuQHisNd3e9FhJ9f\n"    \
        "QKOAgxqctpr79OkbIec8KB7tQ3awPnPn89PcF4nD9NQkTZZFE6HYFMOeTv7LcaNR\n"    \
        "esROFKblyQoiS/YTKWWnwhYmTLLdkCXZ+61svJH/1k3mNwh1O33qMAZwl/Tfs3K0\n"    \
        "KofIu3etl/0BKCBpQL5DT4rAqfYmlXORaY29Sok7VseN9ZH3o1XdQ+DNj8oUbjqO\n"    \
        "MQIDAQABo4GbMIGYMA4GA1UdDwEB/wQEAwIFoDAWBgNVHREEDzANggsyZnlydzd0\n"    \
        "bWF5YzAdBgNVHSUEFjAUBggrBgEFBQcDAgYIKwYBBQUHAwEwDwYDVR0TAQH/BAUw\n"    \
        "AwIBADAfBgNVHSMEGDAWgBSYPebTelZxwmBEZgoTj1d85Nu5pzAdBgNVHQ4EFgQU\n"    \
        "9uuKjgjzQIxaMvKKk9io6KmEHZQwDQYJKoZIhvcNAQELBQADggEBAJY7FsojG1HX\n"    \
        "XI1gqK2GRtphrZX3KRRjTF+AOAeoLXpD0A7mEevrDaAqk1LtawJq0cCEg+bkPgUN\n"    \
        "7DxWExj0v7UiRfO+JNedV7hlL063ukUmP/qBvhXXemcGE0SI21Pk0/izYP5DyGLS\n"    \
        "RM0uvWEdfg/PsnAmgHUFMA51W/UA57o2C/rGvq0kCk+FM9jVk6GdUG14042Rp5Cc\n"    \
        "pjhcojqgP2meuhPcZ69UDvPSmTX1BA7KkIJdb8fZb/+nPqDYNPgHoTYLkoEbqD8T\n"    \
        "Wl7LLZIPmRBar6+DtQp+nmuJjK05iIwrq62tw7REG6ZucQrBzFJMuA09XBAmknZO\n"    \
        "5FuqxI/b+jQ=\n"     \
        "-----END CERTIFICATE-----\n"


#define DEVICE_PRIVATE_KEY_FILE                     \
       "-----BEGIN RSA PRIVATE KEY-----\n"\
        "MIIEpAIBAAKCAQEAvxTrl+BlZNA52bdr+eFkNHq6KbjJmJWEzd/YVr3uy8szly5e\n"    \
        "lunxfMZJBmQ5ZJ1Kf6b5VRHtqMOocmxIq9GG29N+Hmxe6v+QXcEwLlwOxJTCEei8\n"    \
        "BuJs8KY7KWIRF4SwSiEcHuQHisNd3e9FhJ9fQKOAgxqctpr79OkbIec8KB7tQ3aw\n"    \
        "PnPn89PcF4nD9NQkTZZFE6HYFMOeTv7LcaNResROFKblyQoiS/YTKWWnwhYmTLLd\n"    \
        "kCXZ+61svJH/1k3mNwh1O33qMAZwl/Tfs3K0KofIu3etl/0BKCBpQL5DT4rAqfYm\n"    \
        "lXORaY29Sok7VseN9ZH3o1XdQ+DNj8oUbjqOMQIDAQABAoIBACs3KglRYN0qEstg\n"    \
        "pnT/OtB7qTriZgWyWzTMmXsUvbstm99nR1FOW6VimwVtCWXuBPlV5DIPOfbfOWuX\n"    \
        "feAOJTf2XyG2VMlFP3tUw/bYLybalgHC5zOPZiAPGHZFjaQKCGzbjCaYX/RMNyCI\n"    \
        "iupxuPM/h0Mfc64A41cCmXZTe9F+uQRe7d8BF9ggOeZEccP5p5GkrKM6/46HKsXd\n"    \
        "DXGDkHVRKw0Qnpo0twwwt7chRGtqcFVPxuaQ79xP3+0pfwjkhzxmrBm3ceFRauSJ\n"    \
        "Eka2kiJvQ/SQXyNvWgL7sBGD3oeiORfjP0twavJkx+sDsqXGKyepO2hEVtqSutWe\n"    \
        "EuZYejkCgYEA1bSbVSmu+bWAY6yPl85UrJRZg4mDMm7M5HBsqrTLv5+juCGm0hCm\n"    \
        "8SV9lR3vXDuOhcZolWUBJ6hOgMloADx6Y0W7cfUBan94nw8ifr41IIJuGG/pkXCA\n"    \
        "wy39YmS/3ESrCcje4o5TZTINF1SVc7rRS/PDsWFBvBiS3oqUFPVaYYMCgYEA5OYU\n"    \
        "WRQqVb+exjNkJowQqSrG2uVazswn9lY7TFJLFTVuBnyyA9nEnelMc4vXpUqmyhWb\n"    \
        "slk6Eky+OO3JCDrf11moViNLa3f7+SgIQI9kdTTosxiZImjvsfewf23WH9KotX7g\n"    \
        "6KYq8YxhRxLw0Jl+yo84GLRIcXdpD3NRsVP+hzsCgYEApgde9tcfndkpGkcieCpR\n"    \
        "FpNayBq51yDGBmRhyG4WLZX1OETMuj89s/QQkR1krR+IJt5yuj5ym0DjXKMdLbHO\n"    \
        "0/mFG4/hvb+zEsBLaHSWCd/LcSjhKaFiKEE+robfmF2xFDT/FLzGhcRQ72NDiVoG\n"    \
        "6R9FVjXdQClHUaYQTgggY8kCgYEA5AaMiLz8+5U7z4jMoVXxFwUlOlU3ZEGfIdh/\n"    \
        "lCV65LLoW5AHKqzbINCTt5u9BM3cX/6srzwSDL7k1QOUvB0idn67oqvgazwJCQmn\n"    \
        "RuFSGyzND/VWQCA3gJwV+KvrnqIWLVArL7ixmRXWpp4oxjXTSGjqYjne3SIuyBSC\n"    \
        "clwReJcCgYBCgR8iZfZ5RCKE2TB1jADZCpB/octnbPevylVEqMOoTkr0pgIbfha1\n"    \
        "+XyDyLmfvR340TQHjuvfGshFKRKK8eUIJxPT/vg8r24wPM31AzLpYsZxGeBeVBqe\n"    \
        "frsFyhfa9+JY42W4+GVHnQBuTZh+bKx7sv7v0cjZdz+4fVihahPHlA==\n"    \
        "-----END RSA PRIVATE KEY-----\n"
```

Figure 52. Device Private Key Generation Example

6. Building the Application

1. In case of running **Azure_EK_RA6M3_WiFi** project, open *user_cfg.h* insert your Wi-Fi network name (SSID) and password in the designated defines between the quotation marks.



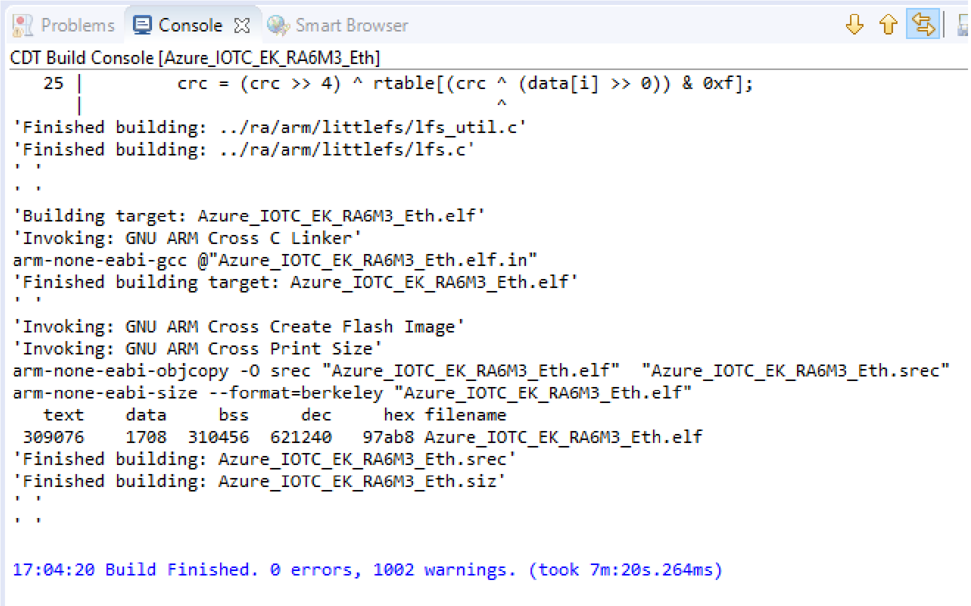
```
#define WIFI_SSID                  ""
#define WIFI_PWD                   ""
```

Figure 53. Entering Wi-Fi Network Name and Password

2. The project is now ready to compile. Press the “hammer” icon to start building the project.

Figure 54. Starting to Build the Project

3. The toolchain will report compilation and build status to the console pane in the lower-right corner of e^2 studio. When the build has completed, it should confirm that there are zero errors and four warnings. All warnings come from the third-party code.



```
CDT Build Console [Azure_IOTC_EK_RA6M3_Eth]
   25 |         crc = (crc >> 4) ^ rtable[(crc ^ (data[i] >> 0)) & 0xf];
      |                                         ^
'Finished building: ../ra/arm/littlefs/lfs_util.c'
'Finished building: ../ra/arm/littlefs/lfs.c'
' '
' '
'Building target: Azure_IOTC_EK_RA6M3_Eth.elf'
'Invoking: GNU ARM Cross C Linker'
arm-none-eabi-gcc @"Azure_IOTC_EK_RA6M3_Eth.elf.in"
'Finished building target: Azure_IOTC_EK_RA6M3_Eth.elf'
' '
'Invoking: GNU ARM Cross Create Flash Image'
'Invoking: GNU ARM Cross Print Size'
arm-none-eabi-objcopy -O srec "Azure_IOTC_EK_RA6M3_Eth.elf"  "Azure_IOTC_EK_RA6M3_Eth.srec"
arm-none-eabi-size --format=berkeley "Azure_IOTC_EK_RA6M3_Eth.elf"
   text    data     bss     dec     hex filename
 309076    1708  310456  621240   97ab8 Azure_IOTC_EK_RA6M3_Eth.elf
'Finished building: Azure_IOTC_EK_RA6M3_Eth.srec'
'Finished building: Azure_IOTC_EK_RA6M3_Eth.siz'
' '
' '


17:04:20 Build Finished. 0 errors, 1002 warnings. (took 7m:20s.264ms)
```

Figure 55. Compilation and Build Status Report

7. Connecting to Azure IoT Cloud Server

This section covers establishing connectivity with the Azure Device Provisioning Service (DPS) server. Establishing a connection with DPS will be necessary to detect additional connection details such as the assigned hub – these will be needed in subsequent sections when using MQTT.

Using the connection string from DPS, the RA device will establish connection to the corresponding Azure IoT Hub. Upon, connection, the RA device will be able to periodically upload the sensor telemetry data to the Microsoft Azure hub. Microsoft's server-side backend will manage the information so that it will later be displayed inside an IoT Central application.

1. The application is now ready to be programmed and run on the EK-RA6M3 board. Press the "bug" icon to begin the debug session.

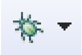
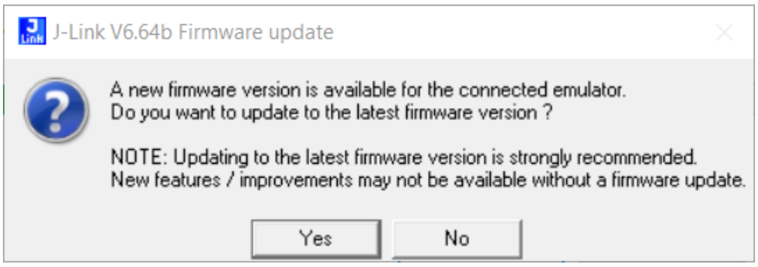


Figure 56. Starting the Debug Session

2. You may be prompted to update the J-Link debugger firmware. You can click *Yes* to update. It should not take long to complete.



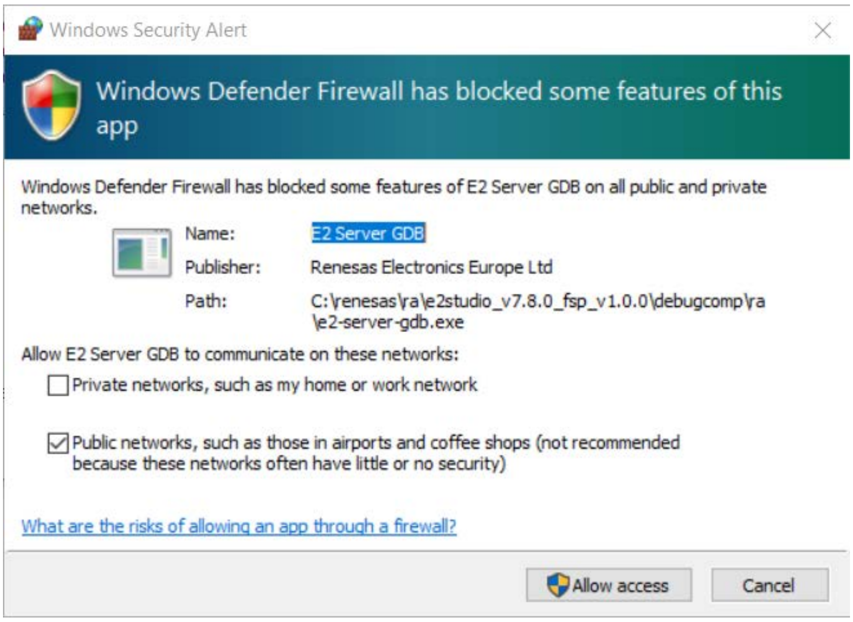
J-Link V6.64b Firmware update

A new firmware version is available for the connected emulator.
Do you want to update to the latest firmware version ?

NOTE: Updating to the latest firmware version is strongly recommended.
New features / improvements may not be available without a firmware update.

Figure 57. J-Link Debugger Firmware Update Prompt

3. Windows could also prompt you to allow the GBD server through your firewall. Click the checkbox to allow it through private networks, then **Allow access**.



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of E2 Server GDB on all public and private networks.

Name: E2 Server GDB
Publisher: Renesas Electronics Europe Ltd
Path: C:\renesas\ra\e2studio_v7.8.0_fsp_v1.0.0\debugcomp\ra\e2-server-gdb.exe

Allow E2 Server GDB to communicate on these networks:

□ Private networks, such as my home or work network

☑ Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

Figure 58. Allowing Access to the GBD Server

4. e^2 studio will perform flash programming routines and prompt to switch to **Debug** perspective. Select the check box by **Remember my decision** and click **Yes**. LED5 near the debug USB port will blink while programming.

5. The debug session is now started, and the application is paused at its entry function (`Reset_Handler`). At this point, you can set up additional debug features such as variable and expressions views before the program is executed. Select the **Expressions** tab on the top right pane of the e^2 studio window:

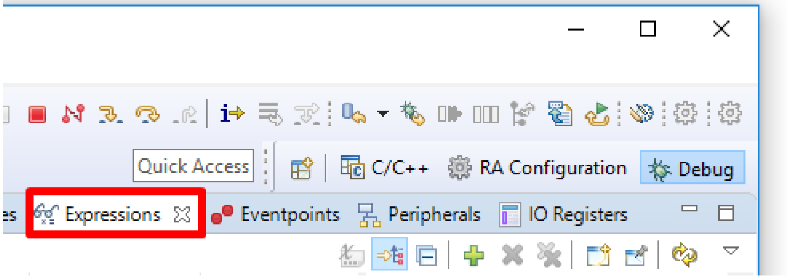
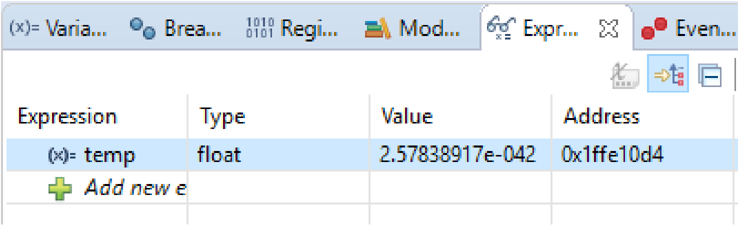


Figure 59. Selecting Expressions Tab on the Top Right Pane

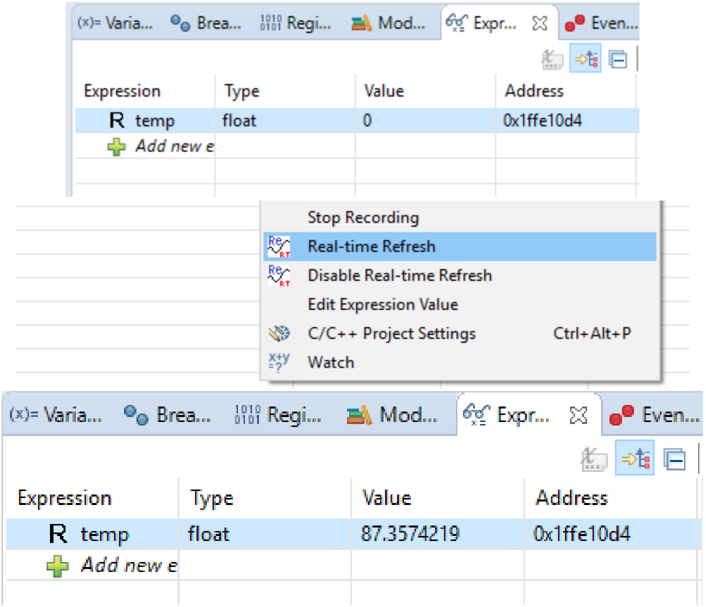
6. Click **Add new expression** and type “temp” for the symbol. Confirm the symbol name by pressing *Enter* on the keyboard. This variable holds temperature readings from the On-chip temperature sensor.



Expression	Type	Value	Address
(x)= temp	float	2.57838917e-042	0x1ffe10d4
Add new e			

Figure 60. Adding New Expression of Type ‘temp’

7. Right-click the **temp** table entry and select **Real-time Refresh**. R symbol to the left of *temp* shows that real-time refresh has been enabled.



Expression	Type	Value	Address
R temp	float	0	0x1ffe10d4
Add new e			

Expression	Type	Value	Address
R temp	float	87.3574219	0x1ffe10d4
Add new e			

Figure 61. Selecting Real-time Refresh

8. Click the **Resume** button or press **F8** on the keyboard to start the application.

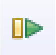
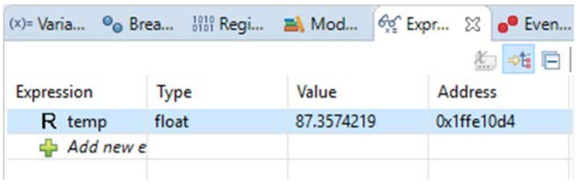


Figure 62. Starting the Application

9. The Program will stop again, this time at the start of the `main` function. Low-level initialization routines are now completed, and the temp and hum values reported on the Expressions pane should be 0. Press Resume or F8 again to resume the application and begin executing user code.

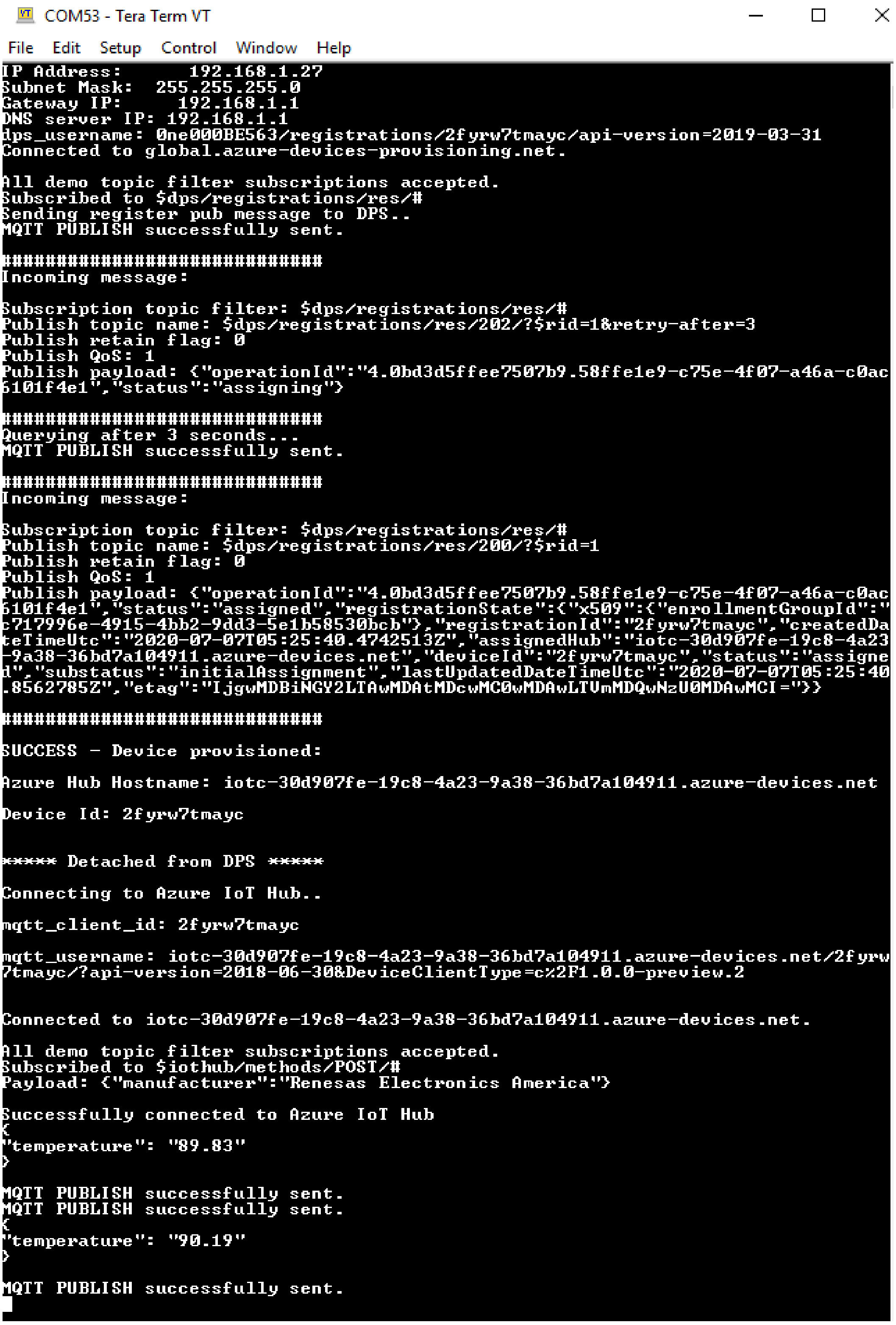
10. As the program is executing the **Expressions** view will report changing value for the *temp* and *hum* variables.



Expression	Type	Value	Address
R temp	float	87.3574219	0x1ffe10d4

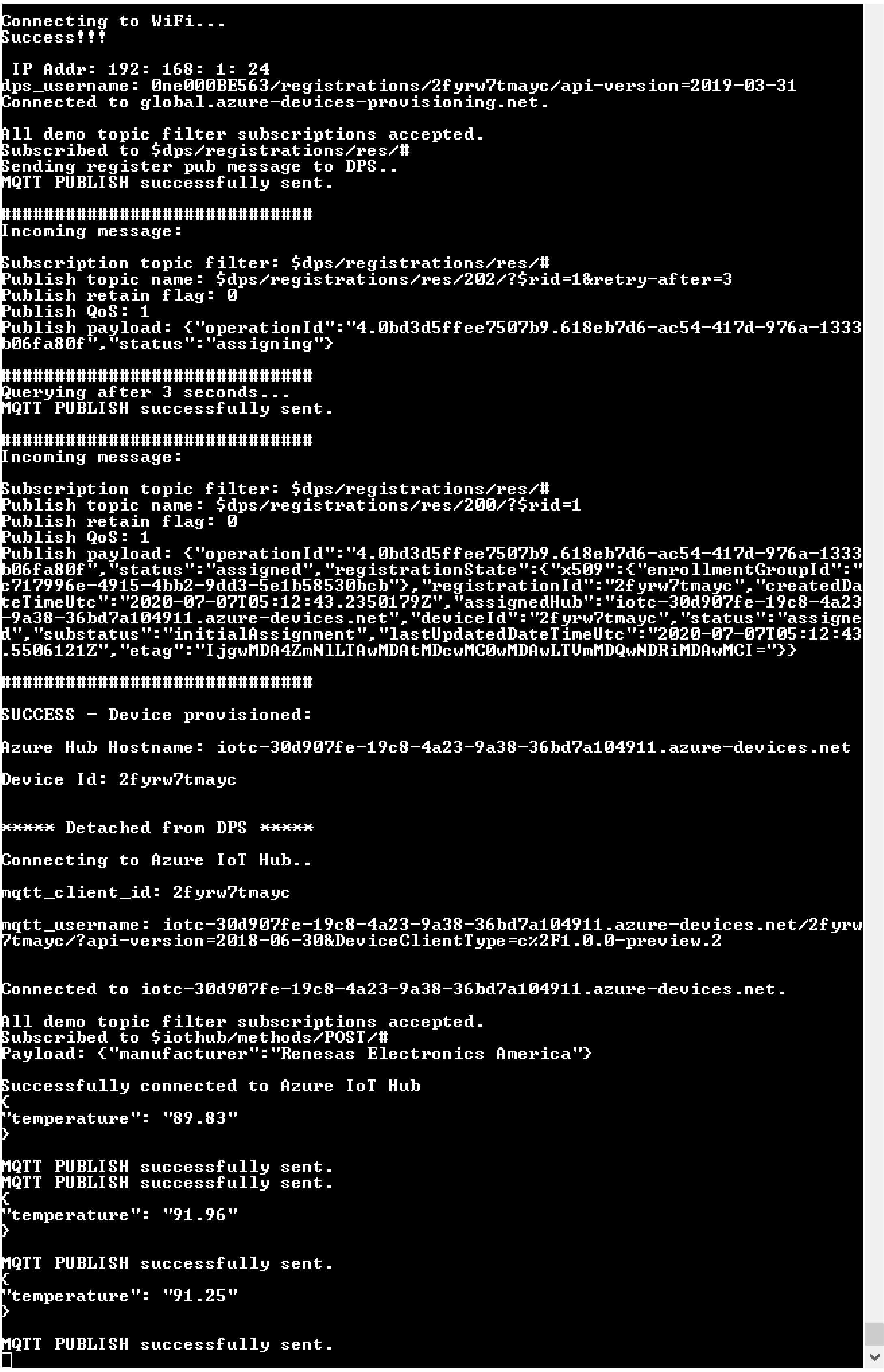
Figure 63. Expressions View

11. Open the serial console such as Tera Term to check the serial debug messages from the EK-RA6M3 kit. The RA device establishes connection with the Azure DPS to obtain the connection string. The RA device establishes connection with the Azure IoT Hub using the connection string obtained from Azure DPS and periodically publishes sensor data.



```
IP Address:       192.168.1.27
Subnet Mask:   255.255.255.0
Gateway IP:      192.168.1.1
DNS server IP: 192.168.1.1
dps_username: 0ne000BE563/registrations/2fyrw7tmayc/api-version=2019-03-31
Connected to global.azure-devices-provisioning.net.

All demo topic filter subscriptions accepted.
Subscribed to $dps/registrations/res/#
Sending register pub message to DPS..
MQTT PUBLISH successfully sent.

#############################
Incoming message:

Subscription topic filter: $dps/registrations/res/#
Publish topic name: $dps/registrations/res/202/?$rid=1&retry-after=3
Publish retain flag: 0
Publish QoS: 1
Publish payload: {"operationId":"4.0bd3d5ffee7507b9.58ffe1e9-c75e-4f07-a46a-c0ac
6101f4e1","status":"assigning"}

#############################
Querying after 3 seconds...
MQTT PUBLISH successfully sent.

#############################
Incoming message:

Subscription topic filter: $dps/registrations/res/#
Publish topic name: $dps/registrations/res/200/?$rid=1
Publish retain flag: 0
Publish QoS: 1
Publish payload: {"operationId":"4.0bd3d5ffee7507b9.58ffe1e9-c75e-4f07-a46a-c0ac
6101f4e1","status":"assigned","registrationState":{"x509":{"enrollmentGroupId":"
c717996e-4915-4bb2-9dd3-5e1b58530bcb"},"registrationId":"2fyrw7tmayc","createdDa
teTimeUtc":"2020-07-07T05:25:40.4742513Z","assignedHub":"iotc-30d907fe-19c8-4a23
-9a38-36bd7a104911.azure-devices.net","deviceId":"2fyrw7tmayc","status":"assigne
d","substatus":"initialAssignment","lastUpdatedDateTimeUtc":"2020-07-07T05:25:40
.8562785Z","etag":"IjgwMDBiNGY2LTAwMDAtMDcwMC0wMDAwLTVmMDQwNzU0MDAwMCI="}}

#############################

SUCCESS - Device provisioned:

Azure Hub Hostname: iotc-30d907fe-19c8-4a23-9a38-36bd7a104911.azure-devices.net

Device Id: 2fyrw7tmayc


***** Detached from DPS *****

Connecting to Azure IoT Hub..

mqtt_client_id: 2fyrw7tmayc

mqtt_username: iotc-30d907fe-19c8-4a23-9a38-36bd7a104911.azure-devices.net/2fyrw
7tmayc/?api-version=2018-06-30&DeviceClientType=c%2F1.0.0-preview.2


Connected to iotc-30d907fe-19c8-4a23-9a38-36bd7a104911.azure-devices.net.

All demo topic filter subscriptions accepted.
Subscribed to $iothub/methods/POST/#
Payload: {"manufacturer":"Renesas Electronics America"}

Successfully connected to Azure IoT Hub
{
"temperature": "89.83"
}

MQTT PUBLISH successfully sent.
MQTT PUBLISH successfully sent.
{
"temperature": "90.19"
}

MQTT PUBLISH successfully sent.
```

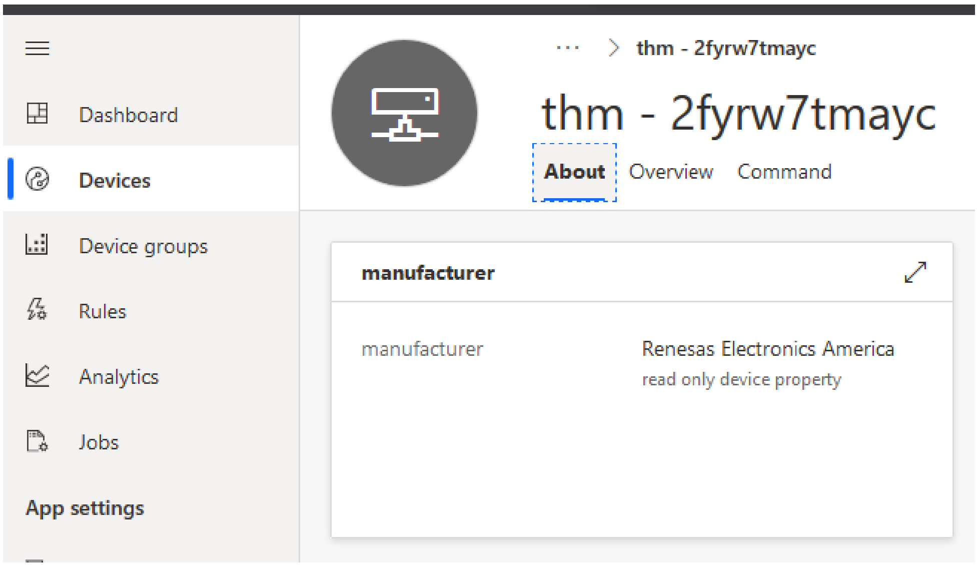
Figure 64. Serial Log Message for X.509 Based Authentication using Ethernet Interface



```
Connecting to WiFi...
Success!!!

 IP Addr: 192: 168: 1: 24
dps_username: 0ne000BE563/registrations/2fyrw7tmayc/api-version=2019-03-31
Connected to global.azure-devices-provisioning.net.

All demo topic filter subscriptions accepted.
Subscribed to $dps/registrations/res/#
Sending register pub message to DPS..
MQTT PUBLISH successfully sent.

#############################
Incoming message:

Subscription topic filter: $dps/registrations/res/#
Publish topic name: $dps/registrations/res/202/?$rid=1&retry-after=3
Publish retain flag: 0
Publish QoS: 1
Publish payload: {"operationId":"4.0bd3d5ffee7507b9.618eb7d6-ac54-417d-976a-1333
b06fa80f","status":"assigning"}

#############################
Querying after 3 seconds...
MQTT PUBLISH successfully sent.

#############################
Incoming message:

Subscription topic filter: $dps/registrations/res/#
Publish topic name: $dps/registrations/res/200/?$rid=1
Publish retain flag: 0
Publish QoS: 1
Publish payload: {"operationId":"4.0bd3d5ffee7507b9.618eb7d6-ac54-417d-976a-1333
b06fa80f","status":"assigned","registrationState":{"x509":{"enrollmentGroupId":"
c717996e-4915-4bb2-9dd3-5e1b58530bcb"},"registrationId":"2fyrw7tmayc","createdDa
teTimeUtc":"2020-07-07T05:12:43.2350179Z","assignedHub":"iotc-30d907fe-19c8-4a23
-9a38-36bd7a104911.azure-devices.net","deviceId":"2fyrw7tmayc","status":"assigne
d","substatus":"initialAssignment","lastUpdatedDateTimeUtc":"2020-07-07T05:12:43
.5506121Z","etag":"IjgwMDA4ZmNlLTAwMDAtMDcwMC0wMDAwLTVmMDQwNDRiMDAwMCI="}}

#############################

SUCCESS - Device provisioned:

Azure Hub Hostname: iotc-30d907fe-19c8-4a23-9a38-36bd7a104911.azure-devices.net

Device Id: 2fyrw7tmayc


***** Detached from DPS *****

Connecting to Azure IoT Hub..

mqtt_client_id: 2fyrw7tmayc

mqtt_username: iotc-30d907fe-19c8-4a23-9a38-36bd7a104911.azure-devices.net/2fyrw
7tmayc/?api-version=2018-06-30&DeviceClientType=c%2F1.0.0-preview.2


Connected to iotc-30d907fe-19c8-4a23-9a38-36bd7a104911.azure-devices.net.

All demo topic filter subscriptions accepted.
Subscribed to $iothub/methods/POST/#
Payload: {"manufacturer":"Renesas Electronics America"}

Successfully connected to Azure IoT Hub
{
"temperature": "89.83"
}

MQTT PUBLISH successfully sent.
MQTT PUBLISH successfully sent.
{
"temperature": "91.96"
}

MQTT PUBLISH successfully sent.
{
"temperature": "91.25"
}

MQTT PUBLISH successfully sent.
```

Figure 65. Serial Log Message for X.509 Based Authentication using Wi-Fi Interface

12. The RA device also publishes the read-only device property (*manufacturer*) string "*Renesas Electronics America*" only once after connecting to Azure IoT Central application. Go back to the web browser with the Azure IoT Central application open. Select the **Devices** tab and click on the device name. If the About window is not displayed, then select About tab to see this property update.



manufacturer

manufacturer: Renesas Electronics America
read only device property

Figure 66. Read only Device Property in About Tab

13. Go back to the web browser with the Azure IoT Central application open. Select the ***Devices*** tab and click on the device name. If the **Overview** window is not displayed, then select **Overview** to see the temperature sensor readings received from the device.

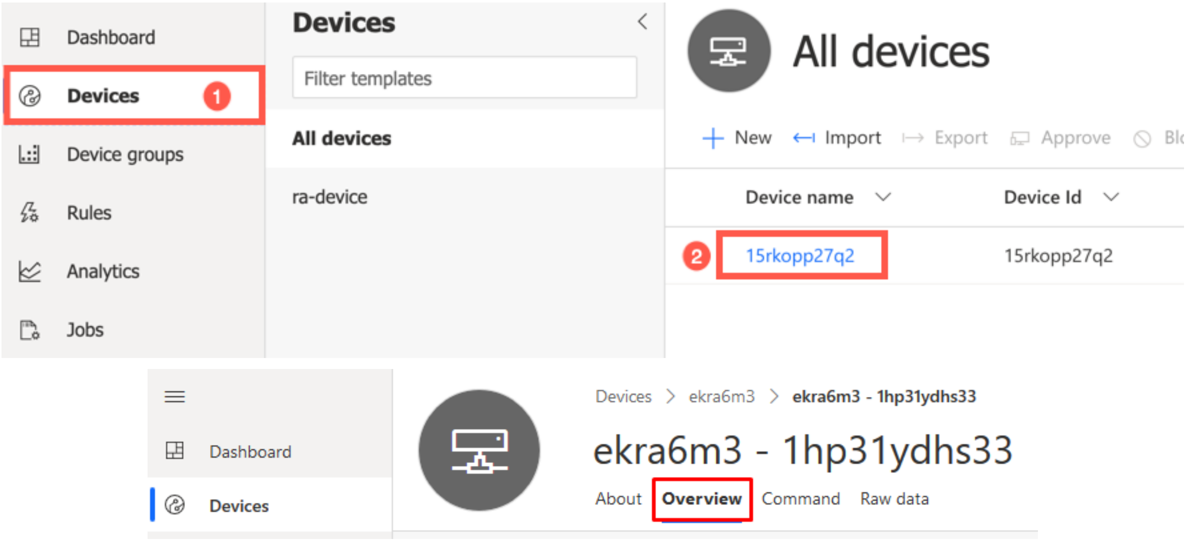
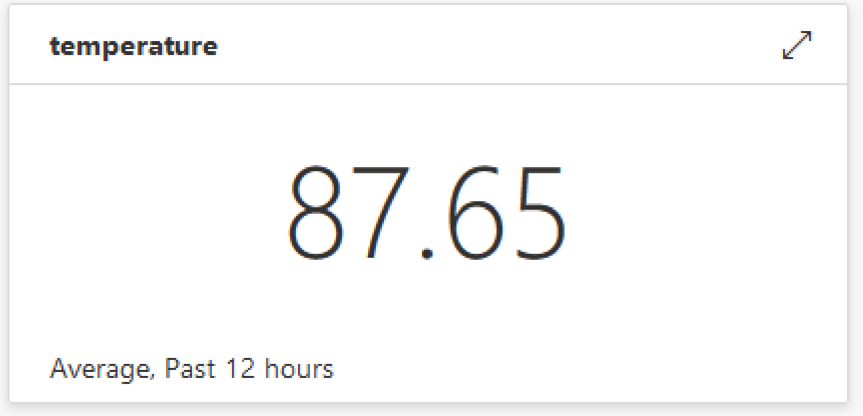


Figure 67. Viewing Temperature Sensor Readings in Overview Window

14. Note that the RA device is only sending the value every few seconds to conserve network bandwidth and the IoT Central application is performing averaging over fixed periods of time, so it may take some time (15-20 seconds) before the first value is shown on the graph or for subsequent values to be posted to IoT Central.



temperature

87.65

Average, Past 12 hours

Figure 68. Temperature Sensor Readings

15. Click the *About* tab to check the device property reported from EK-RA6M3 kit.

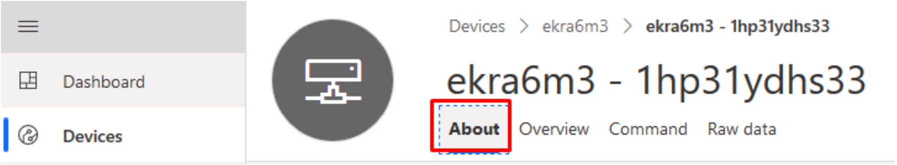


Figure 69. Checking Device Property in About Tab

16. Click the **Terminate** button or press **Ctrl + F2** on the keyboard to stop the application and terminate the debug session.

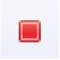


Figure 70. Stopping the Application and Terminating the Debug Session

8. Receiving Messages from the IoT Central Application

The previous sections showed how we can send MQTT data from our device up to the cloud. We can also receive information and commands from the cloud and act on them. We already configured a led_toggle command in IoT Central previously and we will now hook that up to toggle LEDs on the dev kit.

1. Build the project and start the debug session once the project is compiled and error-free.
2. Click the **Resume** button twice and switch to the *Tera Term* window. Go to **Edit** and select **Clear Buffer**.
3. The RA device establishes connection with the Azure IoT Hub using the connection string obtained from Azure DPS and periodically publish sensor data.

4. In the Azure IoT Central application, click on the **Devices** tab and select the device we created for this application project. In the device window, go to **command** tab and click the **Run** button as highlighted below.

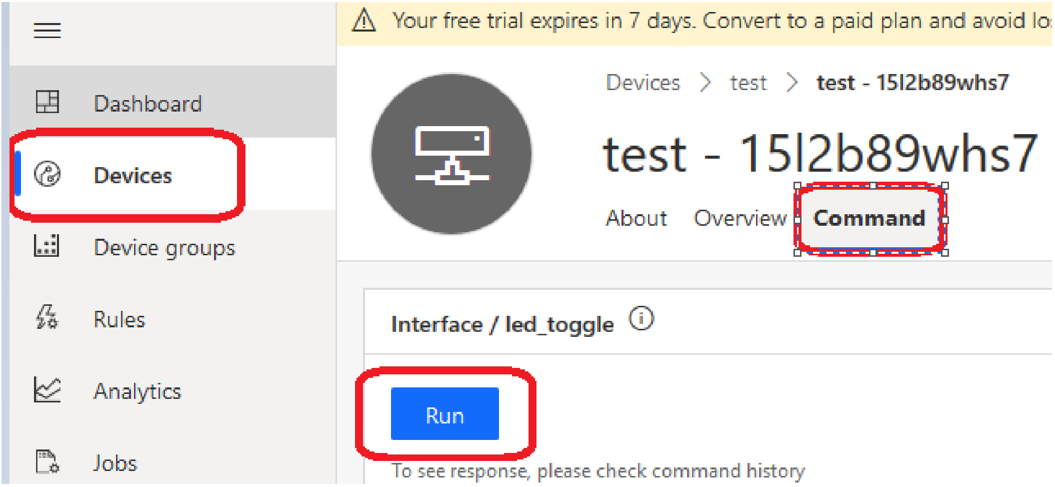
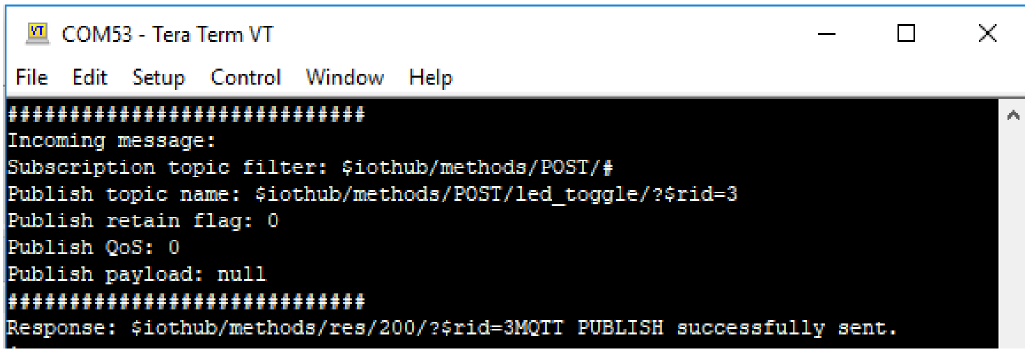

Your free trial expires in 7 days. Convert to a paid plan and avoid lo
Dashboard
Devices
Device groups
Rules
Analytics
Jobs
Devices > test > test - 15l2b89whs7
test - 15l2b89whs7
About
Overview
Command
Interface / led_toggle
Run
To see response, please check command history

Figure 71. Clicking Run Button on the Command Tab

5. *IoT Central* will report that the message was sent, and the *Tera Term* console will display its contents. The three LEDs under the user buttons on the EK-RA6M3 kit will toggle on and off for each press of the **Run** button.



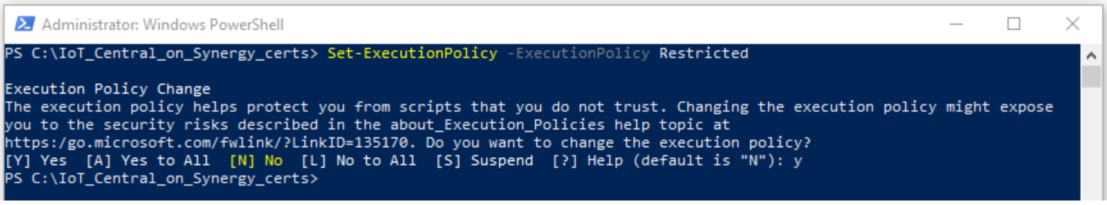
```
############################
Incoming message:
Subscription topic filter: $iothub/methods/POST/#
Publish topic name: $iothub/methods/POST/led_toggle/?$rid=3
Publish retain flag: 0
Publish QoS: 0
Publish payload: null
############################
Response: $iothub/methods/res/200/?$rid=3MQTT PUBLISH successfully sent.
```

Figure 72. Tera Term Console Display

Congratulations! You've reached the end of this application project. Your RA application is now complete. You can use it as a starting point for your own Azure cloud applications by utilizing other components of the Flexible Software Package and Azure IoT Central.

9. Cleaning up PowerShell and Windows Certificate Store

1. Go back to the *PowerShell window* and type `Set-ExecutionPolicy -ExecutionPolicy` followed by space and the execution policy name retrieved in section 5, step 9. Usually this setting is `Restricted`. Press *Enter* to execute the command. When prompted, write `y` and press *Enter* again.



```
PS C:\IoT_Central_on_Synergy_certs> Set-ExecutionPolicy -ExecutionPolicy Restricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y
PS C:\IoT_Central_on_Synergy_certs>
```

Figure 73. Setting the Execution Policy

2. The **PowerShell window** can now be closed.

3. Open start menu and type *manage computer certificates* and select the *best match* suggestion pointing to *Control Panel* widget.

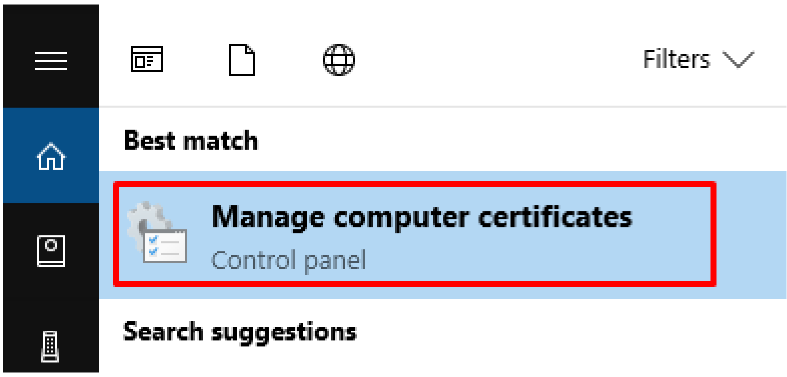
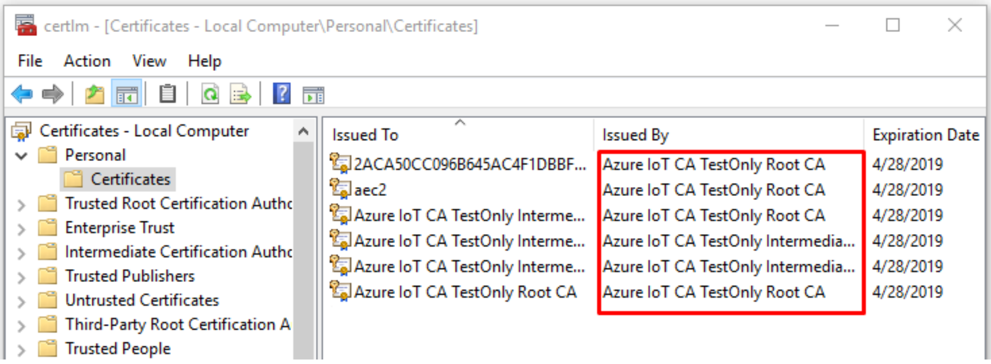


Figure 74. Managing Computer Certificates

4. **Certificate Manager (certlm)** will now open. From the menu on the left side, expand Personal and select **Certificates**.

5. Select and delete (that is, by pressing **Delete** on the keyboard) every certificate issued by entity starting with *Azure IoT CA TestOnly*.


certlm - [Certificates - Local Computer\Personal\Certificates]
File
Action
View
Help
Certificates - Local Computer
Personal
Certificates
Trusted Root Certification Autho
Enterprise Trust
Intermediate Certification Autho
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification A
Trusted People

Issued To	Issued By	Expiration Date
2ACA50CC096B645AC4F1DBBF...	Azure IoT CA TestOnly Root CA	4/28/2019
aec2	Azure IoT CA TestOnly Root CA	4/28/2019
Azure IoT CA TestOnly Interme...	Azure IoT CA TestOnly Root CA	4/28/2019
Azure IoT CA TestOnly Interme...	Azure IoT CA TestOnly Intermedia...	4/28/2019
Azure IoT CA TestOnly Interme...	Azure IoT CA TestOnly Intermedia...	4/28/2019
Azure IoT CA TestOnly Root CA	Azure IoT CA TestOnly Root CA	4/28/2019

Figure 75. Deleting Certificates Issued by Entities Starting with 'Azure IoT Test CA TestOnly'

6. Expand **Trusted Root Certification Authorities** on the left side and select **Certificates**. Repeat the process from step 5 – this category should only have one certificate issued by Azure IoT CA TestOnly Root CA.

7. Close Certificate Manager.

10. Next Steps and References

- Refer to the following GitHub repository for various FSP modules example projects and application projects (https://github.com/renesas/ra-fsp-examples/)
- Refer to Establishing and Protecting Device Identity using SCE7 and Security MPU (R11AN0449) on renesas.com
- Refer to Securing Data at Rest utilizing the RA Security MPU (R11AN0416) on renesas.com
- Refer to Azure GitHub link for more details on Azure SDK for Embedded C (https://github.com/Azure/azure-sdk-for-c)

Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct.23.20	—	First release document

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.