Connect Microchip Technology’s “SAM-IoT WG” Development Board to Azure IoT Services

Introduction

This document describes how to connect the Microchip Technology “SAM-IoT” Development Board (Part No. EV75S95A) to Azure IoT services (e.g. IoT Explorer & IoT Central) using Azure IoT Plug-and-Play Device models.

IoT Plug and Play certified device simplifies the process of building devices without custom device code. Using Solution builders can be integrated quickly using the certified IoT Plug and Play enabled device based on Azure IoT Central as well as third-party solutions.

This “Getting Started Guide” provides step by step instructions on getting the device provisioned to Azure IoT Hub using Device Provisioning Service (DPS) and then using both IoT Explorer and IoT Central to interact with the device's Plug and Play capabilities.

Diagram

Description automatically generated

### **Features of the “SAM-IoT” Development Board**

Microchip’s “SAM-IoT” Development Board combines a powerful 32-bit ATSAMD21G18A MCU, an ATECC608A CryptoAuthentication™ secure element IC, and the fully-certified ATWINC1510 Wi-Fi® network controller module - which provides the most simple and effective way to connect your embedded application to the Cloud. The board also includes an on-board debugger and requires no external hardware to program and debug the MCU.

The fully programmable flash-based ATSAMD21G18A MCU enables you to quickly connect and send data to the Azure Cloud using the on-board temperature and light sensors. Two mechanical buttons and 4 LEDs are all used to simulate telemetry events and to demonstrate read-only and writable device properties. Once you are ready to build your own custom design, you can easily generate code using the free libraries from Microchip’s MPLAB Harmony Software Framework.

Device Programming Procedure URL: <https://github.com/Azure-Samples/Microchip-SAM-IoT-Wx>

Shape

Description automatically generated with low confidence

Diagram

Description automatically generated

# Development Tools Installation

1. Install the Microchip MPLAB X Tool Chain
   * [MPLAB X IDE](https://www.microchip.com/mplab/mplab-x-ide)
   * [MPLAB XC32 Compiler](https://www.microchip.com/en-us/development-tools-tools-and-software/mplab-xc-compilers#tabs)
   * [MPLAB Harmony Software Framework](https://microchipdeveloper.com/harmony3:mhc-overview)
2. Install [Git](https://git-scm.com)
   * Download/install the latest version of [Git for Windows](https://git-scm.com/download/win)
   * Verify that you have access to Git Bash (click Start > type Git Bash)
3. Follow the procedure provided by the [SAM-IoT WG Development Board (EV75S95A) Provisioning Tools Package for Windows](https://github.com/randywu763/sam-iot-provision)

Integration with Azure IoT Explorer

### **Enroll Device into DPS (Device Provisioning Service)**

1. Set up certificates for the verification process:
   * Go to the certificates store sub-folder located at \[your\_path]\sam-iot-provision-main\SAM\_IoT\_Certs\_Generator\ChainOfTrust
   * Copy all \*.crt files and rename each to \*.pem
2. In the [Microsoft Azure Portal](https://portal.azure.com/#home), upload the root CA cert root-ca.pem in DPS and do proof-of-possession for X.509 CA certificates with your Device Provisioning Service

* Register the public part of an X.509 certificate and get a verification code
* Follow the Register the public part of an X.509 certificate and get a verification code section in [How to Verify Certificates](https://docs.microsoft.com/azure/iot-dps/how-to-verify-certificates)
* The verification code is generated by encrypting the public key portion of your X.509 certification. It will be used to validate the uploaded certificate ownership. So make sure to copy the generated verification code to clipboard for use in an upcoming step.
* Digitally sign the verification code to create a verification certificate
* Now that you've registered your root CA with Azure IoT Hub, you'll need to prove that you actually own it by:

1. Generating the Certificate Signing Request (CSR) using the verification code.
2. Generating a verification certificate using CSR above

### **Demonstrate Proof of Possession**

Upload your verification certificate to DPS for [Proof of Possession](https://tools.ietf.org/html/rfc5280#section-3.1) with the following steps:

1. Open a Git Bash window (Start menu > type Git Bash)
2. Change the directory to your certificates store folder (which was generated by the [SAM-IoT WG Provisioning Tools Package for Windows](https://github.com/randywu763/sam-iot-provision)):

* cd drive\[your path]\sam-iot-provision-main\SAM\_IoT\_Certs\_Generator\ChainOfTrust  
    
  Example: cd /C/Users/john5/Azure/sam-iot-provision-main/SAM\_IoT\_Certs\_Generator/ChainOfTrust

1. Generate a certification signing request (CSR) by entering the below command. A CSR is generated by encrypting these two key inputs:
   * The root-ca.key generated in previous step by Microchip IoT Provisioning tool. This will be used in the openssl command line below.
   * The Verification Code generated from DPS in previous step. You will be asked to provide this during the process of creating the CSR.

* **Note**: Once you enter to command below, you will then be asked to enter information that will be used to will be incorporated into your certificate request. Enter verification code (generated from Azure portal) when prompt for CommonName. For the rest, you can enter anything you want.
* openssl req -new -key root-ca.key -out azure\_root\_ca\_verification.csr
* As a result, you should see the file azure\_root\_ca\_verification.csr appear in the ChainOfTrust sub-folder:

1. Generate a verification certificate by executing the following command:

* openssl x509 -req -in azure\_root\_ca\_verification.csr -CA root-ca.crt -CAkey root-ca.key -CAcreateserial -out azure\_signer\_verification.cer -days 365 -sha256
* As a result, you should see the file azure\_signer\_verification.cer appear in the ChainOfTrust sub-folder

### **Upload the Signed Verification Certificate to DPS**

Follow the Upload the signed verification certificate section in [How to Verify Certificates](https://docs.microsoft.com/azure/iot-dps/how-to-verify-certificates). As a result, the status of your uploaded certification should be “Verified” as shown below (make sure to refresh the page to see the updated change in status)

### **Add a New Enrollment Group using the Signer Certificate**

1. In the Azure portal, navigate to your DPS > Manage enrollments > Select Enrollment Groups tab
2. Add enrollment group > Enter Group name > Choose Certificate as Attestation Type > Choose False for IoT Edge Device > Choose Intermediate Certificate as Certificate Type > Upload \[path]\.microchip-iot\signer-ca.pem to Primary Certificate > select Evenly weighted distribution for how you want to assign devices to hub > select your IoT Hub that this new enrollment group can assign to > leave the rest as their existing defaults > hit Save

* Once this has been done, your enrollment group name should show up in the Enrollment Groups tab

### **Program the Plug and Play Demo**

1. Clone/download the MPLAB X demo project by issuing the following commands in a Command Prompt or PowerShell window:
   * git clone https://github.com/Azure-Samples/Microchip-SAM-IoT-Wx.git  
     cd Microchip-SAM-IoT-Wx  
     git submodule update --init
2. Launch the MPLAB X IDE and navigate to the main toolbar's File > Open Project operation to load the demo project folder (\*.X) located at:
   * [your\_path]\Microchip-SAM-IoT-Wx\firmware\AzureIotPnpDps.X
3. Set the AzureIotPnpDps project as the main (currently focused/active) project by right-clicking on it and selecting Set as Main Project
4. Modify AzureIotPnPDps/Source Files/app.c with your wireless router’s SSID and password (keeping the surrounding quotation marks for each); for example
   * #define APP\_CFG\_MAIN\_WLAN\_SSID "MY\_WIFI\_AP\_SSID"  
     #define APP\_CFG\_MAIN\_WLAN\_PSK "MY\_WIFI\_AP\_PSWD"
   * [!TIP]  
     You may also configure the WiFi SSID and password stored in the board using the wifi command via the Command Line Interface (CLI); e.g.

* wifi <MY\_WIFI\_AP\_SSID>,<MY\_WIFI\_AP\_PSWD>,2

1. Verify the project properties are set correctly before building the project:
   * With the board connected to the PC, verify that the CURIOSITY device shows up as a disk drive in a File Explorer window
   * Right-click the project AzureIotPnPDps > select Properties > Verify that all Configuration settings are at least the minimum versions as shown in the below screenshot (and that your SAM-IoT board is selected as the Connected Hardware Tool). If any changes were made in the project properties window, make sure to hit the Apply button before hitting OK.
2. Build the project and set up a Command Line Interface (CLI) to the board:
   * Open a serial terminal (e.g. PuTTY, TeraTerm, etc.) and connect to the COM port corresponding to your board at 9600 baud (e.g. open PuTTY Configuration window > choose session > choose Serial > Enter the right COMx port). You can find the COM info by opening your PC’s Device Manager > expand Ports(COM & LPT) > take note of Curiosity Virtual COM Port.
   * Right-click the AzureIotPnPDps project and select Make and Program Device. This operation will automatically clean and build the project before attempting to program the target device. After the BUILD SUCCESSFUL message appears in the Output window, the application HEX file will be programmed onto the SAM-IoT board. Once programming has finished, the board will automatically reset and start running its application code.
3. Launch a terminal emulator window and connect to the COM port corresponding to the SAM-IoT board at 9600 baud (**disable** local echo for the terminal settings for best results). Hit [RETURN] to bring up the Command Line Interface prompt (which is simply the > character). Type help and then hit [RETURN] to get the list of available commands for the Command Line Interface (CLI). The Command Line Interface allows you to send simple ASCII-string commands to set or get the user-configurable operating parameters of the application while it is running. The CLI prompt is simply the > character
4. Look up the ID Scope corresponding to your DPS in the Microsoft Azure Portal. This value is displayed in a web browser when clicking on Overview on the DPS resource page (the DPS should have been created earlier using a web page interface on the Azure Portal). The ID Scope is programmed/saved into the SAM-IoT board in the next step using a CLI command (allowing you to change the ID Scope for the board without having to reprogram the MCU's application firmware)
5. In the terminal emulator window, confirm that local echo is **disabled** in the terminal settings. Hit [RETURN] to bring up the Command Line Interface prompt (which is simply the > character). At the CLI prompt, type in the command idscope <MY\_ID\_SCOPE> to set the ID Scope (which gets saved in the ATECC608A secure element on the SAM-IoT board) and then hit [RETURN]. To confirm it was set correctly, the ID Scope can be read out from the board by issuing the idscope command (i.e. without specifying an ID Scope value as the parameter on the command line)
6. At the CLI prompt, type in the command reset and hit [ENTER] to restart the application using the updated ID Scope to establish a connection to your DPS.
7. Wait for the SAM-IoT board to connect to your DPS and stabilize (it could take a few minutes); eventually the Blue and Green LEDs should both stay constantly on (which signifies a successful & stable connection to DPS). If the Red LED comes on and stays lit, then something was incorrectly programmed (e.g. application firmware, Wi-Fi credentials, ID Scope). If the Blue LED is not constantly on, then there is an issue with connecting to your wireless access point.
8. To enable the “full” debug messaging output to the terminal emulator window, execute the command debug 4 on the Command Line Interface (CLI). To completely disable the debug messages at any time, execute the command debug 0 (debug levels range from 0 to 4). The CLI is always active, even while debug messages are being continuously displayed on the terminal window.

### **Verify the SAM-IoT Device Registration to Azure IoT Hub**

A successful SAM-IoT to Azure DPS connection can be verified two ways:

1. Correct device ID shows up in the DPS enrollment

In the [Azure Portal](https://portal.azure.com/#home), go to your DPS > click Manage enrollments > under Enrollment Group, click your group name > click Registration Records > device should show up with the IoT Hub info that it got assigned to

1. Correct device ID shows up in the IoT Hub

In the [Azure Portal](https://portal.azure.com/#home), go to your IoT Hub > click IoT Devices > click Refresh > device should show up with the Status “Enabled” and Authentication Type of “SelfSigned”

### **SAM-IoT Board Interaction with Azure IoT Explorer**

Once the SAM-IoT connection to Azure IoT Hub has been verified, the device can be monitored & controlled using Microsoft's Azure IoT Explorer. The Azure IoT Explorer is a graphical tool for interacting with and testing your IoT device on Azure. Refer to [Install Azure IoT Explorer](https://docs.microsoft.com/azure/iot-pnp/howto-install-iot-explorer#install-azure-iot-explorer) for additional details.

1. Connect Azure IoT Explorer to IoT Hub by providing your IoT Hub’s connection string. From the Azure Portal: click on your IoT Hub > Shared access polices > iothubowner > connection string-primary key > Copy to clipboard
2. Launch Azure IoT Explorer: Click on Add connection > paste the Connection string > Save
3. In the Azure IoT Explorer window, click on the Home link near the top of the window
4. On the left-hand side of the Azure IoT Explorer window, click on IoT Plug and Play Settings
5. Please make sure Public repository is in the list. If Public repository is not listed, add it using the Add icon
6. Click on Save
7. On the left-hand side of the IoT Explorer window, click on IoT hubs
8. Verify that the name of your IoT hub is displayed, then click on View devices in this hub
9. Verify that your device ID is displayed (and status is **Enabled**), then click on it
10. On the left-hand side of the IoT Explorer window, click on IoT Plug and Play components
11. Click on Default component near the bottom of the IoT Explorer window
12. Click on Properties (read-only) near the top of the IoT Explorer window
13. Confirm that the Value each LED property matches the physical state observed on the SAM-IoT board (1 = On, 2 = Off, 3 = Blinking)
14. Click on Properties (writable) near the top of the IoT Explorer window
15. Click on the input field labeled led\_y (the property corresponding to the Yellow LED) and select Blink
16. Click on Update desired value
17. Observe the notification that the request to write the property was accepted by your device, and that the Yellow LED on the SAM-IoT board is currently blinking/toggling/flashing
18. Click on Telemetry near the top of the IoT Explorer window and then click on Start
19. Observe the telemetry data (for the temperature and light sensors) is updating every few seconds
20. Increase the ambient light shining on top of the board and observe that the value of the light sensor increases noticeably within a few seconds
21. On the SAM-IoT WG Development Board, press and release user buttons SW0 and/or SW1. The Red LED may toggle at least once on each button press event that is detected (not due to any error condition)
22. Observe the button event message (telemetry) that is generated each time a user button has been pressed/released
23. Click on Commands near the top of the IoT Explorer window
24. Click on the input field for delay and type PT10S, then click Send command. Confirm that the command was successfully invoked via a notification message, and that the SAM-IoT Development Board resets itself in approximately 10 seconds (the LEDs on the board should all go off and then re-initialize themselves)

Connect to Azure IoT Central

[Azure IoT Central](https://docs.microsoft.com/en-us/azure/iot-central/core/overview-iot-central) is an IoT application platform that reduces the burden and cost of developing, managing, and maintaining enterprise-grade IoT solutions. Choosing to build with IoT Central gives you the opportunity to focus time, money, and energy on transforming your business with IoT data, rather than just maintaining and updating a complex and continually evolving IoT infrastructure.

The web UI lets you quickly connect devices, monitor device conditions, create rules, and manage millions of devices and their data throughout their life cycle. Furthermore, it enables you to act on device insights by extending IoT intelligence into line-of-business applications.

## Program the Plug and Play Demo

1. Clone/download the MPLAB X demo project by issuing the following commands in a Command Prompt or PowerShell window

* git clone https://github.com/Azure-Samples/Microchip-SAM-IoT-Wx.git  
  cd Microchip-SAM-IoT-Wx  
  git submodule update --init

1. Launch the MPLAB X IDE and navigate to the main toolbar's File > Open Project operation to load the demo project folder (\*.X) located at [your\_path]\Microchip-SAM-IoT-Wx\firmware\AzureIotPnpDps.X
2. Set the AzureIotPnpDps project as the main (currently focused/active) project by right-clicking on it and selecting Set as Main Project
3. Modify AzureIotPnPDps/Source Files/app.c with your wireless router’s SSID and password (keeping the surrounding quotation marks for each); for example

* #define APP\_CFG\_MAIN\_WLAN\_SSID "MY\_WIFI\_AP\_SSID"  
  #define APP\_CFG\_MAIN\_WLAN\_PSK "MY\_WIFI\_AP\_PSWD"
* [!TIP]  
  You may also configure the WiFi SSID and password during run-time (i.e. dynamically when the board is set up to run the application) using the wifi command via the Command Line Interface (CLI); e.g.
* wifi <MY\_WIFI\_AP\_SSID>,<MY\_WIFI\_AP\_PSWD>,2

1. Verify the project properties are set correctly before building the project
   * Connect the board to PC, then make sure CURIOSITY device shows up as a disk drive in a File Explorer window
   * Right-click the project AzureIotPnPDps > select Properties > Verify that all Configuration settings are at least the minimum versions as shown in the below screenshot (and that your SAM-IoT board is selected as the Connected Hardware Tool). If any changes were made in the project properties window, make sure to hit the Apply button before hitting OK
2. Build the project and set up a Command Line Interface (CLI) to the board
   * Open a serial terminal (e.g. PuTTY, TeraTerm, etc.) and connect to the COM port corresponding to your board at 9600 baud (e.g. open PuTTY Configuration window > choose session > choose Serial> Enter the right COMx port). You can find the COM info by opening your PC’s Device Manager > expand Ports(COM & LPT) > take note of Curiosity Virtual COM Port
   * Right-click the AzureIotPnPDps project and select Make and Program Device. This operation will automatically clean and build the project before attempting to program the target device. After the BUILD SUCCESSFUL message appears in the Output window, the application HEX file will be programmed onto the SAM-IoT board. Once programming has finished, the board will automatically reset and start running its application code.
3. Before typing anything in the terminal emulator window, **disable** the local echo feature in the terminal settings for best results. In the terminal window, hit [RETURN] to bring up the Command Line Interface prompt (which is simply the > character). Type help and then hit [RETURN] to get the list of available commands for the Command Line Interface (CLI). The CLI allows you to send simple ASCII-string commands to set or get the user-configurable operating parameters

## Create an IoT Central Application for your Device

IoT Central allows you to create an application dashboard to monitor the telemetry and take appropriate actions based on customized rules.

1. Create a custom IoT Central application by starting with an existing [Microchip IoT Development Board Template](https://apps.azureiotcentral.com/build/new/bc6e29a0-963d-45e1-9037-b0e780e3943f) (if there is a problem with loading the template, refer to the [Create an application](https://docs.microsoft.com/en-us/azure/iot-central/core/quick-deploy-iot-central) section to create your IoT Central application from scratch)
2. Review and select the settings for your IoT Central application (if needed, refer to [Create an application](https://docs.microsoft.com/en-us/azure/iot-central/core/quick-deploy-iot-central) for additional guidance on selecting the settings for your application). It is recommended to choose a unique Application name which will result in a unique URL for accessing your application. Azure IoT Builder will populate a suggested unique Application name which can/should be leveraged, resulting in a unique URL. When finished configuring all the required settings, click the Create button to save your application
3. Create an X.509 enrollment group for your IoT Central application. Open your IoT Central application and navigate to Administration in the left pane and select Device connection
4. Select + Create enrollment group, and create a new enrollment group using any name (Group type = IoT devices, attestation type = Certificates (X.509)). Hit Save when finished
5. Now that the new enrollment group has been created, select + Manage Primary.
6. Select the file/folder icon associated with the Primary field and upload the root certificate file root-ca.crt (located in the ChainOfTrust sub-folder that was created by the SAM-IoT Provisioning Tools Package for Windows). The message "(!) Needs verification" should appear. The Subject and Thumbprint fields will automatically populate themselves
7. Click Generate verification code (this code will be copied to the clipboard which will be needed in a future step)
8. Open a Git Bash window (Start menu > type Git Bash)
9. Using the Git Bash command line, navigate to your certificates folder (the ChainOfTrust sub-folder which was generated by the [SAM-IoT Provisioning Tools Package for Windows](https://github.com/randywu763/sam-iot-provision))

* cd <path>\sam-iot-provision-main\SAM\_IoT\_Certs\_Generator\ChainOfTrust

1. Execute the below command in the Git Bash window (copy and paste for best results)

* **Note**: Once you enter the below command, you will then be asked to enter information for various fields that will be incorporated into your certificate request. Enter the verification code (which was just generated previously) when prompted for the Common Name. It's recommended to just copy the Verification code to the clipboard and paste it when it's time to enter the Common Name. For the rest of the fields, you can enter anything you want (or just hit [RETURN] to keep them blank which is fine for basic demonstration purposes). If you accidentally hit [RETURN] when asked for the Common Name, you will need to run the command again...
* openssl req -new -key root-ca.key -out azure\_root\_ca\_verification.csr

1. Generate the verification certificate by executing the following command exactly as shown (suggest copy and paste for best results)

* openssl x509 -req -in azure\_root\_ca\_verification.csr -CA root-ca.crt -CAkey root-ca.key -CAcreateserial -out azure\_signer\_verification.cer -days 365 -sha256

1. Click Verify and select the azure\_signer\_verification.cer file to upload. Confirm that the Primary certificate has been verified and that a Thumbprint has been generated for your certificate. Click on Close to exit the current window, then click on Save at the top of the web application window. The X.509 enrollment group has been successfully created and should be ready to go!
2. If not already active, launch a terminal emulator window and connect to the COM port corresponding to the SAM-IoT board at 9600 baud (**disable** local echo for the terminal settings for best results). Hit [RETURN] to bring up the Command Line Interface prompt (which is simply the > character). Type help and then hit [RETURN] to get the list of available commands for the Command Line Interface (CLI). The Command Line Interface allows you to send simple ASCII-string commands to set or get the user-configurable operating parameters of the application while it is running
3. Look up the ID Scope for the DPS created/used by your IoT Central application (navigate to your application's web page and using the left-hand navigation pane, select Administration > Device connection). The ID Scope will be programmed/saved into the [ATECC608A](https://www.microchip.com/wwwproducts/en/atecc608a) secure element on the board in the next step
4. In the terminal emulator window, hit [RETURN] to bring up the Command Line Interface prompt (which is simply the > character>). At the CLI prompt, type in the idscope <your\_ID\_scope> command to set it (which gets saved in the [ATECC608A](https://www.microchip.com/wwwproducts/en/atecc608a) secure element on the board) and then hit [RETURN]. The ID Scope can be read out from the board by issuing the idscope command without specifying any parameter on the command line
5. In the terminal emulator window, hit [RETURN] to bring up the Command Line Interface (CLI) prompt. Type in the command reset and hit [RETURN]
6. Wait for the SAM-IoT board to connect to your IoT Central application’s DPS; the Blue and Green LEDs will be flashing and/or staying on at different times/rates (which could take up to a few minutes). Eventually the Blue and Green LEDs should both remain constantly ON.

* NOTE: If the Red LED comes on, then something was incorrectly programmed (e.g. ID scope was entered incorrectly)

1. Go back to your web browser to access the Azure IoT Central application. Use the left-hand side pane and select Devices > All Devices. Confirm that your device is listed – the device name & ID is the Common Name of the device certificate (which should be sn + {17-digit device ID})
2. If desired, change the Device name by clicking on Manage device > Rename
3. Click on the Command tab; type PT5S in the Reboot delay field and then click on Run to send the command the device to reboot in 5 seconds
4. Within 5 seconds of sending the Reboot command, the SAM-IoT development board should reset itself. Once the Blue and Green LED's stay constantly ON, press the SW0 and SW1 buttons (the Red LED may toggle with each button press)
5. Click on the Raw data tab and confirm that the button press telemetry messages were received
6. Click on the Refresh icon to display all messages received since the previous page refresh operation. Confirm that periodic telemetry messages are being continuously received approximately every 10 seconds (the default interval value for the telemetryInterval property)
7. Increase the ambient light source shining on top of the board. Wait approximately 30 seconds. Click on the Refresh icon to confirm that the light sensor value has increased

## Connecting your Device to the Dashboard

1. Navigate to the left-hand vertical toolbar and click on the Dashboards icon
2. Towards the top of the web page, click on the Edit icon
3. For **all** of the existing tiles named Light or Temperature, click on the upper right-hand corner of the tile to select Configure
4. Select Device Group > SAM-IoT WM - All devices and then check the box for your specific device name for Devices
5. Under the Telemetry category, click on + Capability and select the parameter pertaining to the title of the tile (e.g. Brightness from light sensor for each of the Light tiles or Temperature for each of the Temperature tiles)
6. Click on Update and repeat the process for the remainder of the existing tiles
7. Click on the Save icon
8. Confirm that the dashboard is being continuously updated with the expected telemetry data received from the device. For example, adjust the ambient light source directed at the board and observe that the light sensor values are changing accordingly

## Expanding the Dashboard

To create additional tiles for your IoT Central dashboard, refer to [Configure the IoT Central application dashboard](https://docs.microsoft.com/en-us/azure/iot-central/core/howto-add-tiles-to-your-dashboard). The following screen captures show additional possibilities of dashboard components that can highlight the telemetry data and properties facilitated by the Plug and Play interface. Note that multiple devices can be selected for each tile to allow groups of devices to be visualized within a single tile.

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Additional Links

Please refer to the below links for additional information for Plug and Play and IoT Central

* [Manage cloud device messaging with Azure-IoT-Explorer](https://github.com/Azure/azure-iot-explorer/releases)
* [Import the Plug and Play model](https://docs.microsoft.com/en-us/azure/iot-pnp/concepts-model-repository)
* [Configure to connect to IoT Hub](https://docs.microsoft.com/en-us/azure/iot-pnp/quickstart-connect-device-c)
* [How to use IoT Explorer to interact with the device](https://docs.microsoft.com/en-us/azure/iot-pnp/howto-use-iot-explorer#install-azure-iot-explorer)
* [Create an Azure IoT Central application](https://docs.microsoft.com/en-us/azure/iot-central/core/quick-deploy-iot-central)
* [Manage devices in your Azure IoT Central application](https://docs.microsoft.com/en-us/azure/iot-central/core/howto-manage-devices)
* [How to connect devices with X.509 certificates for IoT Central](https://docs.microsoft.com/en-us/azure/iot-central/core/how-to-connect-devices-x509)
* [Configure the IoT Central application dashboard](https://docs.microsoft.com/en-us/azure/iot-central/core/howto-add-tiles-to-your-dashboard)
* [Customize the IoT Central UI](https://docs.microsoft.com/en-us/azure/iot-central/core/howto-customize-ui)

A close up of a logo

Description automatically generated

Microchip logo

A picture containing drawing

Description automatically generated