# Prerequisites

**Create Required Azure Services**

1. Open Notepad or a similar application to record details that you will reference during the lab.
2. Sign into an Azure CLI by navigating to <https://shell.azure.com>.
3. Select **Bash** in the upper left of the window if it’s not already selected.



1. Run the following script in the Azure Shell CLI:  
    *This will take about 5 minutes to complete.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# IoT LevelUp

az extension add --name azure-iot

az extension update --name azure-iot

az provider register --namespace 'Microsoft.EventGrid'

let "randomIdentifier=$RANDOM\*$RANDOM"

location="East US"

resourceGroup="IoT-LevelUp-$randomIdentifier"

dpsName="IoT-LevelUp-DPS-$randomIdentifier"

iotHubName="IoT-LevelUp-Hub-$randomIdentifier"

az group create --name $resourceGroup --location "$location"

az iot dps create --name $dpsName --resource-group $resourceGroup

SCOPEID=$(az iot dps show --name $dpsName --query "properties.idScope")

az iot hub create --name $iotHubName --resource-group $resourceGroup --sku S1

CONNECTIONSTRING=$(az iot hub connection-string show -n $iotHubName --policy-name iothubowner --key-type primary --query "connectionString")

echo " "

echo "--- Copy the Following Lines to NotePad for Reference Throughout the Lab ---"

echo "Resource Group Name: $resourceGroup"

eval echo "IoT Hub Connection String: $CONNECTIONSTRING"

echo "RaspberryPi Connection String: <Provided during Hands-On Lab #1>"

eval echo "DPS Scope ID: $SCOPEID"

echo "DPS Primary Key: <Provided during Hands-On Lab #3>"

echo "SimulatedDevice Connection String: <Provided during Hands-On Lab #3>"

echo "HTTP POST URL: <Provided during Hands-On Lab #4>"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Copy the last 7 lines for later reference.

**Install Visual Studio Code**

1. Click [here](https://code.visualstudio.com/Download) to navigate to the Visual Studio Code download page.
2. Select the blue button of the appropriate operating system (i.e. Windows).
3. Open the downloaded setup file from your **Downloads** folder.
4. Walk through the setup wizard, selecting all the default options.

**Install the .NET SDK 6**

For Windows:

* + Click [here](https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/sdk-6.0.405-windows-x86-installer) to download the installation file.
  + Open the downloaded setup file from your **Downloads** folder.
  + Click the Install button to complete the installation.

*or...*

For MacOS or Linux:

* + Click [here](https://dotnet.microsoft.com/en-us/download/dotnet/6.0) to navigate to the Git download page.
  + Select the appropriate operating system under the most current version in the upper left corner.
  + Follow the directions to complete the installation.

**Install the C# Extension for Visual Studio Code**

1. Click [here](vscode:extension/ms-dotnettools.csharp) to download the installation file.
2. Click **Open** to launch Visual Studio Code.
3. Once Visual Studio Code opens, it should display the C# Extension offering.

*This may take 1-2 minutes to load*

1. Click the blue **Install** button (the button will display ***Installing***).
2. Once complete, you may close Visual Studio Code

**Install Git**

For Windows:

* + Click [here](https://github.com/git-for-windows/git/releases/download/v2.39.0.windows.2/Git-2.39.0.2-64-bit.exe) to download the installation file.
  + Open the downloaded setup file from your Downloads folder.
  + Walk through the setup wizard, selecting all the default options.

*or...*

For MacOS or Linux:

* + Click [here](https://git-scm.com/downloads) to navigate to the Git download page.
  + Select the appropriate operating system under **Downloads**.
  + Follow the directions to complete the installation.

**Install the Azure IoT Explorer**

1. Return to the Azure Shell
2. Once the script is complete, copy & paste the last 7 lines into Notepad
3. Then, copy the **IoT Hub Connection String**.
4. Click [here](https://github.com/Azure/azure-iot-explorer/releases/download/v0.15.4/Azure.IoT.Explorer.Preview.0.15.4.msi) to download and install.
5. Launch the application if it does not automatically start.
6. Click the **+ Add Connection** button.
7. Paste your **IoT Hub's Connection String** from Notepad and click the **Save** button.
8. Note that there are no devices listed.

**Create a Free Outlook Email Account**

1. Using a browser, open a **New InPrivate** window.
2. *Ctrl*-Click [here](https://outlook.com) to navigate to Outlook online.
3. Click the **Create Free Account** button at the bottom of the window.
4. Enter a unique email address and select the **@outlook.com** domain.
5. Click the **Next** button and create a valid password.
6. Uncheck the option for ***Tips and Offers***.
7. Record your **Email Address** and **Password** for later reference.
8. Click the **Next** button.
9. Enter your **First** and **Last Name.**
10. Click the **Next** button.
11. Select your **Country/Region** and enter your **Birthday.**
12. Click the **Next** button and solve the puzzle to confirm your human.

*That's all -You're ready to start with the first lab.*

# Hands-On Labs

#### [Simulate an IoT Device with the Online RaspberryPi Simulator](#_Hand-On_Lab_#1)

#### [Configure the Device Provisioning Service (DPS)](#_Hand-On_Lab_#2)

#### [Simulate an IoT Device with C# Code](#_Hand-On_Lab_#3)

#### [Use Event Grid to Alert for IoT Hub Events](#_Hand-On_Lab_#4)

#### [Clean-up Your Azure Resources](#_Clean-Up_Azure_Resources)

# Hand-On Lab #1

# Demonstrate the IoT Hub with an online Raspberry Pi Simulator

|  |  |
| --- | --- |
| This exercise will use an online Raspberry Pi simulator to send telemetry to an Azure IoT Hub. This is a faster and simpler approach to demonstrate our IoT platform. | [EventGridDiagram](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/RaspberryPiSimulator.png) |

### Add a New IoT Device to the IoT Hub Manually

1. Open the Azure portal by navigating to [https://portal.azure.com](https://portal.azure.com/)
2. Locate the **Resource Group** that was created in the [**Prerequisites**](https://github.com/glbwell0/IoT-LevelUp/blob/main/prerequisites.md)
3. Open the **IoT Hub** with the name that starts **IoT-LevelUp-Hub**
4. Under the **Device Management** section, select **Devices**
5. Click **+ Add Device** at the top of the window
6. For the **Device ID**, enter **RaspberryPi** and leave all other default settings
7. Click the **Save** at the bottom of the window
8. Click the **Refresh** button at the top of the window
9. Click the device titled **RaspberryPi**
10. Click the **Copy** icon after the **Primary Connection String** field
11. Paste the string in your NotePad, titled **RaspberryPi Connection String**

### Load Azure IoT Explorer to Monitor for Telemetry

1. Launch the Azure IoT Explorer you installed from the [**Prerequisites**](https://github.com/glbwell0/IoT-LevelUp/blob/main/prerequisites.md)
2. Click the **Refresh** button at the top of the window
3. Click the device titled **RaspberryPi**
4. Select the **RaspberryPi** device
5. In panel on the left, select **Telemetry**
6. Click the **Start** button at the top  
   *Note: There will not be any telemetry appearing ...yet.*

### Send Telemetry with the Raspberry Pi Simulator

1. ***Ctrl***-Click [here](https://azure-samples.github.io/raspberry-pi-web-simulator/#getstarted) to launch the **Raspberry Pi Simulator**.
2. Copy your device connection string into the code on line 15 between the single quotes.

[](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/RaspSimConnString.png)

1. Click the **Run** link located just below the code window.
2. Return to the **Azure IoT Explorer** and watch the telemetry stream.
3. Stop monitoring the telemetry by clicking the **Stop** button at the top.
4. Return to the **Raspberry Pi Simulator** and click the **Stop** link located just below the code window.

# Hand-On Lab #2

# Configure the Device Provisioning Service (DPS)

|  |  |
| --- | --- |
| This exercise will enable Azure's DPS to automatically register the device when its initially started. | [EventGridDiagram](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/DPS.png) |

### Configure the Device's Registration

1. Return to the **Resource Group** that was created in the [**Prerequisites**](https://github.com/glbwell0/IoT-LevelUp/blob/main/prerequisites.md).
2. Open the **Device Provisioning Service (DPS)** with the name that starts **IoT-LevelUp-DPS**.
3. Select the **Linked IoT Hubs** in the left panel and note that there are no associated IoT Hubs.
4. Click the **+Add** at the top of the window.
5. In the **IoT Hub** dropdown list, select the hub with the name that starts **IoT-LevelUp-Hub**.

*Note –It may require up to 20 mins for newly created IoT Hubs to appear.*

1. Click the **Save** button at the bottom of the panel.
2. Select the **Manage Enrollments** in the left panel.
3. Click the **+Add Individual Enrollment** at the top of the window.
4. For **Mechanism**, select **Symmetric Key.**
5. In the **Registration ID** textbox, enter **SimulatedDeviceRegistration.**
6. In the **IoT Hub Device ID** textbox, enter **SimulatedDevice.**
7. In the **Initial Device Twin State**, replace with the following:

{

"tags": {},

"properties": {

"desired": {"IntervalFrequency": 5}

}

}

1. Click the **Save** button at the top of the window

### Get the Primary Key for the Device Enrollment

1. Select the **Individual Enrollments** tab towards the top of the panel.
2. Select the **SimulatedDeviceRegistration** registration.
3. Copy the **Primary Key** by clicking the blue **Copy** icon to the right of the entry and save for later reference, titled **DPS Primary Key**.

# Hand-On Lab #3

# Simulate an IoT Device with C# Code

|  |  |
| --- | --- |
| This exercise will demonstrate a sample of code to manage a device and send device-to-cloud telemetry. | [EventGridDiagram](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/DeviceCode.png) |

### Download & Prepare Code for an IoT Device

1. Open a command prompt by searching for **cmd** from the Windows Search or Start menu.
2. Enter the following the download the sample code & launch Visual Studio Code:

cd\

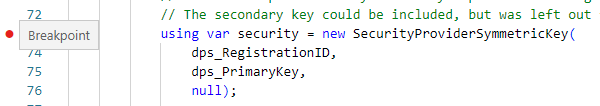
git clone <https://github.com/glbwell0/IoT-LevelUp-Code>

cd\IoT-LevelUp-Code

dotnet restore .

code .

1. Select the **Program.cs** file in the Explorer panel on the left.
2. Enter the **Scope ID** in line 27 between the double quotes.
3. Enter the **DPS Primary Key** in line 29 between the double quotes.
4. Press *Ctrl*-**S** to save your updates.
5. Set a **Breakpoint** on line **73** by clicking just to the left of the line number.

[](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/CodeBreakpoint.png)

1. Set additional breakpoints on code lines **151** and **174**.

### Run the Code

*Note -The first function of this code registers the device in the IoT Hub using the Device Provisioning Service (DPS)*

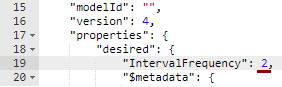
1. Press **F5** to start debugging the simulator.
2. The code will stop for observation at the start of the **ProvisionIoTDevice** function.
3. Walk the code line-by-line by pressing **F10** until line **102**, pause here.
4. Select the **DEBUG CONSOLE** at the bottom of the window *(If the Debug Console is not visible, press Ctrl-Shift-****Y****to toggle it on/off).*
5. Copy the Connection String from the Debug Console and save it in NotePad titled **SimulatedDevice Connection String**.
6. Press **F5** to continue running the application.
7. Return to the Azure portal and locate the hub with a name starting with **IoT-LevelUp-Hub**.
8. Select **Devices** in the left panel and the new device should be listed.

### Monitor the Incoming Telemetry from the Device

1. If the **Azure IoT Explorer** if it's not already running -
   * Search for **iot** in Windows Search to quickly find the application.
   * Click **Azure IoT Explorer** to launch the program.
   * Select the hub with a title starting with **IoT-LevelUp-Hub**.
2. Click the **Refresh** button at the top of the window.
3. Select **SimulatedDevice** from the device list *(if the device is not listed, please ask for help)*.
4. Select the **Telemetry** panel on the left.
5. Click the **Start** button at the top of the window.
6. Within about 10 seconds, you should see telemetry flowing every 5 seconds.

### Remotely Control Your Device with its Device Twin

1. From the Azure IoT Explorer, select the **Device Twin** panel on the left.
2. Locate the **IntervalFrequency** setting under **Properties**-**Desired**.
3. Change the setting from **5** to **2**.

[](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/DeviceTwinFrequency.png)

1. Click **Save** at the top of the window.
2. Return to **Visual Studio Code**.
3. Note the code has stopped in the **OnDesiredPropertyChanged** function.
4. Walk the code by pressing **F10** for each line until you reach the end of the function at line **167**.
5. Remove the breakpoint from line **151** by clicking the red dot to the left of the line number.
6. Press **F5** to continue running the code.
7. Return to the **Azure IoT Explorer**.
8. Click the **Refresh** button at the top of the **Device Twin** window.
9. Note that the **IntervalFrequency** under the **ReportedProperties** has updated to **2**.
10. Switch to the **Telemetry** window & press the **Start** button if it's not already running.
11. Note the telemetry is now streaming data every 2 seconds.

### Remotely Reset the Device from the Cloud

1. Select the **Direct Method** panel on the left.
2. Type **reset** in the textbox under **Method name** *(must be all lowercase)*.
3. Click the **Invoke Method** located at the top of the window.
4. Return to **Visual Studio Code** and note the debugger stopped in the **ResetMethod** function.
5. Remove the breakpoint by clicking the red dot next to the line number.
6. Press **F5** to continue running the code.
7. Return to the **Azure IoT Explorer**.
8. Note there is an error indicating that the direct method failed.

*This is because we paused the code and the device did not acknowledge the call within 30 seconds*

1. Click the **Invoke Method** located at the top of the window again *-you will receive a successful response*.
2. Return to **Visual Studio Code** and stop the devices application by clicking the **Stop** button at the top of the window.

[Graphical user interface, text, application

Description automatically generated with medium confidence](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/VisualStudio-Stop.png)

1. You may now close **Visual Studio Code**.

# Hand-On Lab #4

# Use Event Grid to Alert for IoT Hub Events

|  |  |
| --- | --- |
| This exercise will use Azure Event Grid to trigger an email to notify someone when an IoT device connects or disconnects. | [EventGridDiagram](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/EventGridDiagram.png) |

### Create a Logic App That Sends an Email

1. In the Azure portal, open the **Resource Group** with the title that starts with **IoT-LevelUp**.
2. At the top of the window, click **Create** and search for **Logic App**.
3. Click the tile titled **Logic App** by **Microsoft**.
4. Click the **Create** button.
5. Enter the following details:
   * **Resource Group:** Select the **Resource Group** with the title starting **IoT-LevelUp**.
   * **Logic App Name:** **IoTDeviceConnectionAlert**-***your initials*** -must be a unique name across Azure
   * **Region:** **East US** *-because that is where your IoT Hub is located*
   * **Plan Type:**  **Consumption**
6. Click **Review + Create**, and then click **Create**.
7. Once the resource is created, select **Go to resource**.
8. Under **Start with a common trigger** click **When a HTTP request is received**.
9. Click the **Use sample payload to generate schema** link and enter the following:

[{

"id": "f6bbf8f4-d365-520d-a878-17bf7238abd8",

"topic": "/SUBSCRIPTIONS//RESOURCEGROUPS//PROVIDERS/MICROSOFT.DEVICES/IOTHUBS/",

"subject": "devices/LogicAppTestDevice",

"eventType": "Microsoft.Devices.DeviceConnected",

"eventTime": "2018-06-02T19:17:44.4383997Z",

"data": {

"deviceConnectionStateEventInfo": {

"sequenceNumber":

"000000000000000001D4132452F67CE200000002000000000000000000000001"

},

"hubName": "",

"deviceId": "LogicAppTestDevice",

"moduleId" : "DeviceModuleID"

},

"dataVersion": "1",

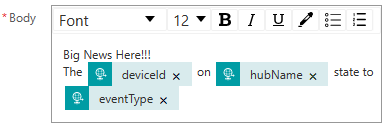
"metadataVersion": "1"

}]

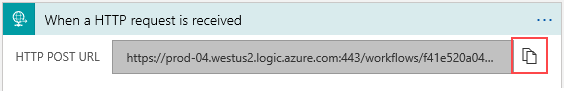
1. Select **+ New Step**. A window appears, prompting you to **Choose an operation**.
2. Search for **Outlook** and select **Outlook.com**.
3. Select the **Send an email (V2)** action. This will open a window to login to an Outlook.com account.

*(Do not use your corporate account)*

1. Select **Sign in** and sign in with the **free Outlook email account**, created during the **Prerequisites**.
2. Select **Yes** to let the app access your info.
3. Build your email template with the following:
   * **To:** Enter your email address (or an account where you want the notification sent)
   * **Subject:**
     + Type **IoT Device Connection Alert:**, then click the **Add dynamic content** link
     + Search for **eventType** and select it
   * **Body:** Write the text for your email and include some dynamic content based on event data. If you can't see the Dynamic content, select the **Add dynamic content** hyperlink under the **Body** text box. If it doesn't show you the fields you want, click **See more** in the Dynamic content screen to include the fields from the previous action.

[](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/IoTDeviceConnectionStateAlert-Body.png)

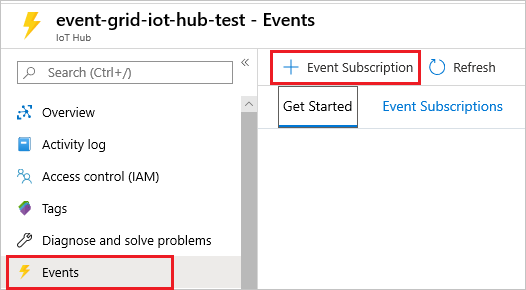
1. Click the **Save** button at the top of the page.
2. Return to the first task titled **When a HTTP request is received** by clicking the box.
3. Copy the **HTTP POST URL** by clicking the **Copy** button to the right of the field.

[](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/HTTPWebHook-URL.png)

1. Paste this with the title **HTTP Post URL** for later reference.

### Create & Configure an Event Grid Subscription for IoT Hub Events

1. In the Azure portal, navigate to your IoT Hub and select the **Events** panel

[](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/IotHub-EventGridSubscription.png)

1. Click the **+Event Subscription** button.
2. Enter the following values for the new subscription:
   * In the **EVENT SUBSCRIPTION DETAILS** section:
     + Enter **IoT-Hub-Alerts** for the **Name**.
     + Select **Event Grid Schema** for **Event Schema**.
   * In the **TOPIC DETAILS** section:
     + Confirm that the **Topic Type** is set to **IoT Hub**.
     + Confirm that your IoT Hub is set as the value for the **Source Resource** field.
     + Enter **IoTDevice-ConnectionStateChanged** for the **System Topic Name**.
   * In the **EVENT TYPES** section:
     + Open the **Filter to Event Types** dropdown list and select only **Device Connected** and **Device Disconnected**.
   * In the **ENDPOINT DETAILS** section:
     + Select **Endpoint Type** as **Web Hook**.
     + Click **Select an Endpoint**, paste the URL that you copied from your logic app, and confirm selection.

*When you're done, the pane should look like the following example:*

[Graphical user interface, text, application, email

Description automatically generated](https://github.com/glbwell0/IoT-LevelUp/blob/main/images/CreateEventSubscription.png)

1. Click the **Create** button.

### Simulate a New Device Connecting and Sending Telemetry

*The prior labs introduced the following two approaches to demonstrate connectivity between an IoT device and Azure:*

* *An online Raspberry Pi Simulator, and*
* *Using C# code on your workstation*

*We are going to present a third approach that uses a single, simple command from the Azure CLI. This approach is ideal to quickly simulate an IoT device with no preparation.*

1. Sign into an Azure CLI by navigating to [https://shell.azure.com](https://shell.azure.com/)
2. Run the following command to simulate connecting your device to IoT Hub and sending telemetry:

**az iot device simulate -d SimulatedDevice -n** ***{YourIoTHubName}***

\*\* Note... ***{YourIoTHubName}*** *should only be the hub’s name, do not include* “**.azure-devices.net**”

1. After the simulated device connects to IoT Hub, within ~2 minutes you'll receive an email notifying you of a **"DeviceConnected"** event
2. Enter ***Ctrl*-C** to cancel the simulation, you'll receive an email notifying you of a **"DeviceDisconnected"** event

***Congrats!!! You're All Done!***

# Clean-Up Azure Resources

1. Sign into an Azure CLI by navigating to [https://shell.azure.com](https://shell.azure.com/)
2. Run the following script in the Azure Shell CLI:  
    *This will take about 5 minutes to complete*

**az group delete --name** ***{YourResourceGroupName}***

***That’s All Folks!!!***