Tutorial: Machine Learning and IoT Edge – send device data

1 Send device data

1.1 Role

This step is typically performed by a cloud or device developer.

1.2 Introduction

In this section, we will once again use development machine as a simulated device, but instead of sending data directly to the IoT Hub the device will send data to the edge device configured as a transparent gateway.

We will monitor the operation of the edge device while the simulated device is sending data. Once the device is finished running, we will look at the data in our storage account to validate everything worked as expected.

1.3 Review device harness

We will be reusing the DeviceHarness project to simulate the downstream (or leaf) device. Connecting to the transparent gateway requires 2 additional things:

1. Register the certificate to make the downstream device (in this case our development machine) trust the certificate authority being used by the IoT Edge runtime.
2. Add the edge gateway fully qualified domain name (FQDN) to the device connection string

Look at the code to see how these two items are implemented.

1. From your development machine open Visual Studio Code
2. Use **File -> Open Folder…** to open C:\source\IoTEdgeAndMlSample\DeviceHarness
3. Look at the InstallCertificate() method in Program.cs
4. Note that if the code finds the certificate path, it calls the CertificateManager.InstallCACert method to install the certificate on the machine.
5. Now look at the GetIotHubDevice method on the TurbofanDevice class.
6. When the user specifies the FQDN of the gateway using the "-g" option, that value is passed to this method as gatewayFqdn, which gets appended to the device connection string.

```
connectionString =
$"{connectionString};GatewayHostName={gatewayFqdn.ToLower()}";
```

1.4 Build and run leaf device

1. With the DeviceHarness project still open in Visual Studio Code
2. Build the project (Ctrl+Shift+B) or **Terminal->Run Build Task…** and select "build" from the dialog
3. Find the fully qualified domain name (FQDN) for your edge device by navigating to your edge device virtual machine in the portal and copying the "DNS name" from the overview.
4. Open the Visual Studio Code terminal (Ctrl+Shift+`) and run the command below

```
dotnet run -- --gateway-host-name " <edge_device_fqdn>" --certificate
C:\edgeMlCertificates\azure-iot-test-only.root.ca.cert.pem --max-devices
1
```

5. The application will attempt to install the certificate onto your development machine, when it does you will need to accept the security warning
6. When prompted for the IoT Hub connection string click the "…" on the "AZURE IOT HUB DEVICES" panel and choose "Copy IoT Hub Connection String"
7. You will see output like:

```
Found existing device: Client_001
Using device connection string: HostName=<your hub>.azure-
devices.net;DeviceId=Client_001;SharedAccessKey=xxxxxxx;GatewayHostName=
iotedge-xxxxxx.<region>.cloudapp.azure.com
Device: 1 Message count: 50
Device: 1 Message count: 100
Device: 1 Message count: 150
Device: 1 Message count: 200
Device: 1 Message count: 250
```

Note the addition of the "GatewayHostName" to the device connection string, which causes the device to communicate through the IoT Hub through the IoT Edge transparent gateway.

1.5 Check output

1.5.1 Edge machine output

The output from the avroFileWriter module can be readily observed by looking at the edge device.

1. SSH into your edge machine
2. Look for files written to disk by running

```
ls -lR /data/avrofiles | grep [0-9].avro
```

3. Output will look like

```
-rw-r--r-- 1 iotuser iotusr 59239 Feb 15 22:52 50.avro
```

Note: you may have more than a single file depending on timing of the run.

4. Pay attention to the time stamps, the avroFileWriter module will upload the files once the last modification time is more than ten minutes in the past (see MODIFIED_FILE_TIMEOUT in uploader.py in the avroFileWriter module)
5. Once the ten minutes has elapsed, the module should upload the files and if successful delete the files from disk.

1.5.2 Azure storage

We can observe the results of our leaf device sending data by looking at the storage accounts where we expect data to be routed.

1. On the development machine open Visual Studio Code

2. In the “AZURE STORAGE” panel in the explore window navigate the tree to find your storage account.
3. Expand the [Blob Containers] node
4. From Reconfigure IoT Hub we expect that the “ruldata” container should contain messages with RUL. Expand the “ruldata” node
5. You will see on or more blob files named like: <IoT Hub Name>/<partition>/<year>/<month>/<day>/<hour>/<minute>
6. Right click on one of the files and choose “Download Blob” to save the file to your VM
7. Download one or more of the files to your development machine
8. Next expand the “uploadturbofanfiles”. In Configure file upload, we set this location as the target for files uploaded by the avroFileWriter module.
9. Right click on the files and choose “Download Blob” to save it to your VM

1.5.3 Read Avro file contents

We included a very simple command line utility for reading an Avro file and returning a JSON string of the messages in the file. In this section, we will install and run it.

1. Open a terminal in Visual Studio Code (Ctrl+Shift+`)
2. Run the command to install hubavroreader:

```
pip install  c:\source\IoTEdgeAndMlSample\HubAvroReader
```

3. Use hubavroreader to read an Avro file downloaded from “ruldata”

```
hubavroreader <avro file with path> | more
```

4. Note that body of the message looks as we expected with device ID and predicted RUL.

```
{
    "Body": {
        "ConnectionDeviceId": "Client_001",
        "CorrelationId": "3d0bc256-b996-455c-8930-99d89d351987",
        "CycleTime": 1.0,
        "PredictedRul": 170.1723693909444
    },
    "EnqueuedTimeUtc": "<time>",
    "Properties": {
        "ConnectionDeviceId": "Client_001",
        "CorrelationId": "3d0bc256-b996-455c-8930-99d89d351987",
        "CreationTimeUtc": "01/01/0001 00:00:00",
        "EnqueuedTimeUtc": "01/01/0001 00:00:00"
    },
    "SystemProperties": {
        "connectionAuthMethod":
"{\"scope\":\"module\",\"type\":\"sas\",\"issuer\":\"iothub\",\"acceptin
gIpFilterRule\":null}",
        "connectionDeviceGenerationId": "636857841798304970",
        "connectionDeviceId": "aaTurbofanEdgeDevice",
        "connectionModuleId": "turbofanRouter",
        "contentEncoding": "utf-8",
        "contentType": "application/json",
        "correlationId": "3d0bc256-b996-455c-8930-99d89d351987",
        "enqueuedTime": "<time>",
        "iotHubName": "mledgeiotwalkthroughhub"
    }
}
```

5. Run the same command passing an Avro file downloaded from “uploadturbofanfiles”
6. As expected, these messages contain all the sensor data and operational settings from the original message. This is the data that we would use to continue to improve the RUL model on our edge device.

```
{
    "Body": {
        "CycleTime": 1.0,
        "OperationalSetting1": -0.0005000000237487257,
        "OperationalSetting2": 0.00039999998989515007,
        "OperationalSetting3": 100.0,
        "PredictedRul": 170.17236328125,
        "Sensor1": 518.6699829101562,
        "Sensor10": 1.2999999523162842,
        "Sensor11": 47.29999923706055,
        "Sensor12": 522.3099975585938,
        "Sensor13": 2388.010009765625,
        "Sensor14": 8145.31982421875,
        "Sensor15": 8.424599647521973,
        "Sensor16": 0.029999999329447746,
        "Sensor17": 391.0,
        "Sensor18": 2388.0,
        "Sensor19": 100.0,
        "Sensor2": 642.3599853515625,
        "Sensor20": 39.11000061035156,
        "Sensor21": 23.353700637817383,
        "Sensor3": 1583.22998046875,
        "Sensor4": 1396.8399658203125,
        "Sensor5": 14.619999885559082,
        "Sensor6": 21.610000610351562,
        "Sensor7": 553.969970703125,
        "Sensor8": 2387.9599609375,
        "Sensor9": 9062.169921875
    },
    "ConnectionDeviceId": "Client_001",
    "CorrelationId": "70df0c98-0958-4c8f-a422-77c2a599594f",
    "CreationTimeUtc": "0001-01-01T00:00:00+00:00",
    "EnqueuedTimeUtc": “<time>”
}
```

1.6 Summary

In this section we used our development machine to simulate a leaf device sending sensor and operational data to our edge device. We validated that the modules on the device routed, classified, persisted and uploaded the data first by examining the real time operation of the edge device and then by looking at the files uploaded to the storage account.

More information can be found at the following pages:

1. Connect a downstream device to an Azure IoT Edge gateway
2. Store data at the edge with Azure Blob Storage on IoT Edge (preview)

2 Clean up resources

If you plan to explore the resources used by this walk-through, wait until you are done to clean up the resources created in this walk-through. If you do not plan to continue, use the following steps to delete them.

1. Delete the resource group(s) created to hold the Dev VM, Edge VM, IoT Hub, storage account, ML workspace service (and created resources: container registry, application insights, key vault, storage account).
2. Delete the ML project at http://notebooks.azure.com
3. If you cloned the repo locally
    1. Close any PowerShell windows, VS Code windows referring to the local repo
    2. Delete the repo directory
    3. If you created certificates locally, delete the folder c:\edgeCertificates