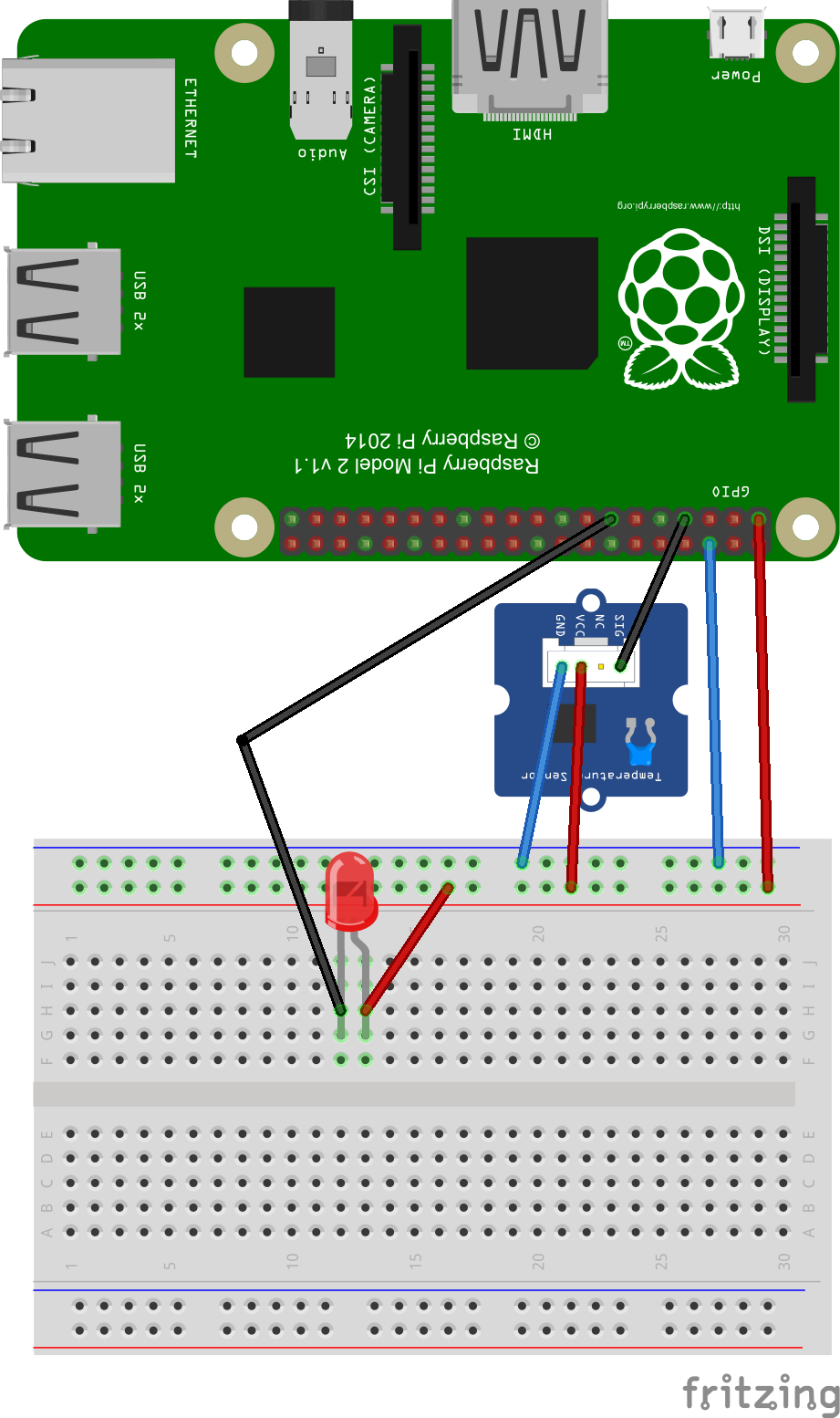
Microsoft IoT Hackathon DX

Scenario 3 – Industrial

# Overview

In this scenario we are going to simulate an Industrial use of IoT using a temperature and humidity sensor. You will connect a temperature and humidity sensor to your Raspberry Pi, using Windows 10 IoT Core and a Universal Windows Platform app to send these readings to an IoT Hub. The readings will automatically increase to simulate a fault. The readings are processed using Stream Analytics and Azure ML to detect if a fault has occurred and if it has then this fault is saved to an event hub. The event hub data is processed by an Azure Web Job which sends a cloud to device message back to your Raspberry Pi to provide a visual alert to the user. Finally, there is a companion website which shows your sensor readings in real-time using ASP.NET MVC and SignalR. The website also allows you to send commands back to your Raspberry Pi.

# Circuit Diagram

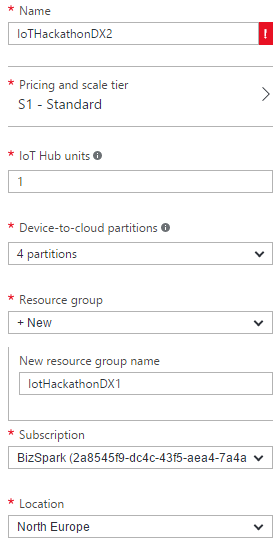
Wire up your Raspberry Pi by following the circuit diagram. 

Ensure that your Raspberry Pi now has the Windows 10 IoT Core SD card inserted, connect the Raspberry Pi to your computer with a network cable power the Pi on. If you haven’t already **download the Windows 10 IoT Core Dashboard** from <http://ms-iot.github.io/content/en-US/Downloads.htm> and install it.

# Provision the IoT Hub

For this scenario we are going to provision two IoT Hubs. The first hub will be used to send the player profile data to the Stream Analytics task. The second hub will be used to send the player answer data to the Stream Analytics task.

Follow steps 1 through 8 and then repeat. Configure your two hubs with the same name but suffix the second IoT Hub with “-profiles”.

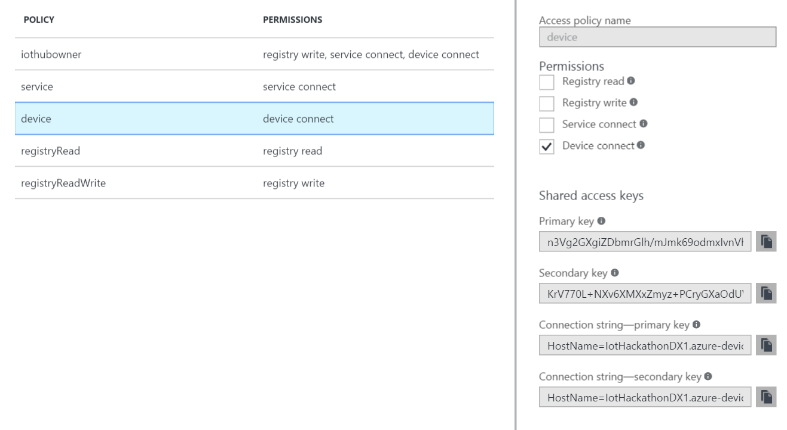
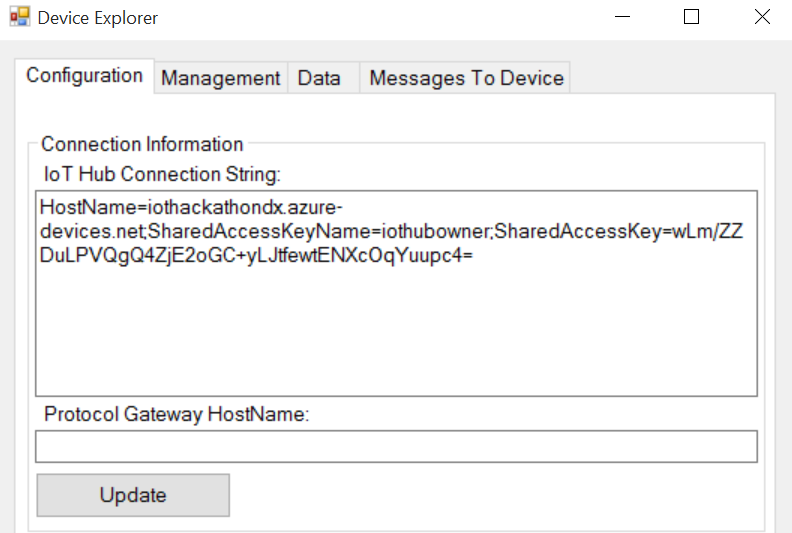
1. On the Preview Portal (remember you need to be in the Preview Portal), click **“+ NEW”** (top left) and select **Internet of Things.**
2. Select **IoT Hub** from the featured apps shown.
3. Enter a name for your hub. In this example I am using “IoTHackathonDXScenario3”.
4. The IoT Hub is charged based on the number of messages per day you want to send to the hub. There is a **“free”** tier in this service that gives you 8,000 messages/day at no charge – select this option unless you are already using the **“free”** tier, in which case select “**S1 – Standard**”.

**Each Azure subscription may only have one free tier IoT Hub provisioned**

1. Leave the IoT Hub units as 1 and the number of partitions as 4.
2. Create a new resource group by clicking **“or create new”** and enter the name you want to call this resource group. We will re-use this resource group for provisioning additional services later on in this lab.
3. For the location select a location local to you where possible.
4. Click the **create button**. You will now be taken back to your Azure dashboard where you will see a new dashboard tile appear showing the status of the provision. IoT Hubs can take a few minutes to provision.
5. Once the IoT Hub has finished being provisioned you need to configure a consumer group which will be used later. From the dashboard select the IoT Hub you just created.
6. Click **“Messaging”** on the settings panel of the IoT Hub.
7. In the **“Consumer groups”** section type **“controlapp”** followed by the enter key.
8. Click the **“save”** button.

# Using Device Explorer

You can’t configure your IoT Hub devices from the Preview Portal however as part of the Azure IoT SDK there is an application called DeviceExplorer which you can use to manage devices, view messages being sent to your hub and send cloud to device messages. For this part of the lab you will need to have the Azure IoT SDK. For this section please use the first IoT Hub you created for the solution.

1. **Download** or clone the GitHub repo for the **Azure IoT SDK** (<https://github.com/Azure/azure-iot-sdks>). If downloading extract, the zip files to your machine.
2. Navigate to \tools\DeviceExplorer\ in your local repo and **open the DeviceExplorer solution** file in Visual Studio.
3. Run the project.
4. To use Device Explorer, the first thing you need to do is enter the connection string for your IoT Hub. In the Preview Portal click your IoT Hub tile that was created for your earlier.
5. Two panes should now appear for your IoT Hub. The right-hand side pane should be titled **“settings”**
6. Select the **“Shared access policies”** menu option and select the **“iothubowner”** policy in the shared access policies tile.
7. From the **“iothubowner”** tile copy the **“Connection string – primary key”.**
8. Back in Device Explorer paste into the **“IoT Hub connection string”** textbox in Device Explorer.
9. Click the **“Update”** button and a dialog confirming the update of the settings should appear.
10. Select the **management tab** and click the **“Create”** button.
11. Enter a name for your device in the **“device id”** textbox. Copy the primary key value as you will need this in a moment. In this example I’ve called my device **“industrialdevice”** and click **“create”**. You should now see the device you created shown in the data grid. You can verify it was created by clicking the **“refresh”** button.

# Configuring the Sensor App

In this section you are going to configure the sensor app that will run on your Raspberry Pi.

1. Download the solution from **xyz** and open in Visual Studio 2015.
2. Open the **“MainPage.xaml.cs”** file and find the line “private const string IOTHUBCONNECTIONSTRING = …”
3. Replace the **“<IoTHubName>”** text with the name of the IoT Hub you just created for this solution.
4. Replace the **“<DeviceName>”** text with the name of the device you just created in device explorer.
5. Replace the **“<SharedAccessKey>”** text with primary key you copied in device explorer.
6. Save the changes to the solution.

# Deploying the Sensor App

1. Next you need to deploy the solution to your Raspberry Pi. In Visual Studio from the tool bar set the solution platform to **“ARM”** and the target as **“Remote Machine”**.



1. Open the project properties by double clicking the **“Properties”** node in solution explorer and navigate to the **“debug”** settings.
2. In the **“Remote machine:”** textbox you need to enter the IP Address/ Name of your Raspberry Pi. To do this click the “Find” button, wait a few moments for Visual Studio to search for your devices and your Rasbperry Pi should show in the “Auto Detected” section. Click your Raspberry Pi and click the **“Select”** button. If your device doesn’t show your device should be called **“minwinpc”** so enter this value.
3. Press the **“green play”** button next “Remote Machine” or press **F5** to build and deploy the solution to your Raspberry Pi.
4. Once the solution has been deployed it will run and start sending data to the IoT Hub. If you have a monitor you can connect this to your Pi and see a basic interface showing you the sensor readings.

# Provisioning Storage

1. On the Preview Portal, click “+ NEW” (top left) and select **Data + Storage**.
2. Select **Storage Account** from the featured apps shown and click the **“Create”** button.
3. Enter a name for your storage account. In this example I am using “iothackathondxScenario3”. The name must in lower case letters.
4. Select the type of storage by clicking on **“Type”**, **locally redundant** is fine for this solution so select this option followed by the **“Select”** button.
5. Chose the resource group you created for your IoT Hubs earlier by clicking the “Select existing” link, then click “Not configured”. A new pane will show with your existing resource groups. Select the correct resource group.
6. For the location select a location local to you where possible.
7. Finally click the create button. You will now be taken back to your Azure dashboard where you will see a new dashboard tile appear showing the status of the provision. Storage accounts can take a few minutes to provision.

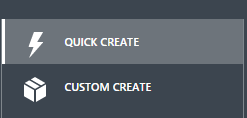
# Configuring the Control Web App

In this section you are going to configure the Control Web App and deploy this as an Azure Website.

1. Download the solution from **xyz** and open in Visual Studio 2015.
2. Open the **“web.config”** file.
3. In the Azure Portal select the IoT Hub you created for this scenario from the Dashboard.
4. From the settings panel select **“Messaging”**.
5. From the Event hub-compatible endpoint copy the text between “sb://” and “.servicebus.net…”. This is your service bus namespace.
6. Under AppSettings find the **“ServiceBusNamespace”** key. Paste the copied text from the previous step into the value section. Your setting should look something like <add key="ServiceBusNamespace" value="iothub-ns-iothackath-18733-5722b4ac43" />.
7. Copy the Event hub-compatible name from the portal.
8. Back in the web.config file find the **“EventHubName”** setting and paste the copied event hub name into the value section.
9. In the settings panel of the IoT Hub on the Azure portal click the “Shared access policies” link. Select “Service” from the list of access policies and copy the primary key.
10. Back in the web.config file find the **“SharedAccessKey”** setting and paste the copied access key into the value section.
11. From the **“service”** shared access policy, copy the **“Connection string – primary key”** value and paste this into the **“IoTHubConnectionString”** setting in the web.config file.
12. Back on the Azure portal dashboard click the storage account you created for this scenario.
13. Select Access keys in the settings panel.
14. Copy the **“Storage Account Name”** and paste it into the **“StorageAccountName”** setting value in the web.config file.
15. Copy the **“Access keys – KEY1”** value from the portal and this time paste it into the value section of the “StorageAccountKey” setting.
16. Close and save the web.config file.
17. Right click on the **“ControlWebApp”** project node in the Solution Explorer and choose publish from the popup menu.
18. Click on the **“Microsoft Azure Web Apps”** option under **“Select a publish target”.**
19. Click the **“New”** button.
20. Enter a name for the Web App in the **“Web App Name”** textbox, making a note of what you called it.
21. Select an existing App service plan or select **“Create new App Service plan”** from the drop down. Enter a name for the App service plan in the textbox.
22. Select the resource group you have used for the other services in this scenario.
23. Select a suitable region for the Web App.
24. Click the **“Ok”** button to provision the service.
25. Visual Studio will now begin to provision the Web App service. This will take a few moments.
26. Once the provisioning has been completed right click on the **“ControlWebApp”** project node and select **“Publish”** from the popup menu again.
27. Click the **“Publish”** button on the Publish Web dialog. Your Web app will now be published. Once the solution has been published a browser window should load with the new website showing.

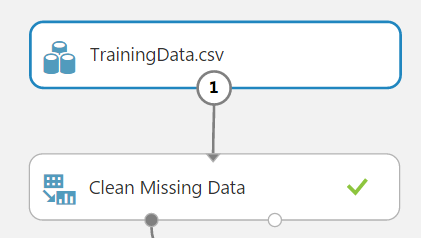
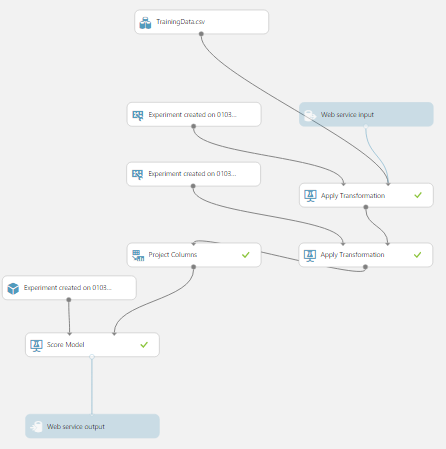
# Provisioning an Event Hub

You are now going to provision an Event Hub. This will be used by Azure Stream Analytics to output data when the Azure ML detects a fault.

1. On the Preview Portal, click “+ NEW” (top left) and select **“Internet of Things”**.
2. Select **Event Hub** from the featured apps shown and click the **“Create”** button.
3. You should be redirected to the old Azure Management Portal.
4. Click the **“Custom Create”** link.  
   
5. In the **“Event Hub Name”** textbox enter a name for your event hub.
6. Select a **“Region”** local to you where possible.
7. In the “Namespace” dropdown make sure **“Create a new namespace”** is selected.
8. Enter a name into the **“Namespace”** textbox.
9. Click the **“next”** icon.
10. In the **“Partition Count”** textbox enter **“2”**.
11. In the **“Message Retention”** textbox enter **“1”**.
12. Click the **“tick”** icon to provision the Event Hub.

# Configuring Azure ML

In this section you are going to setup an Azure ML experiment. The experiment will help predict if the equipment is going to fail based on the temperature and humidity readings.

1. Go to <http://manage.windowsazure.com> and login to your account.
2. Click the bottom **“+”** icon and select **“Data services”** from the menu.
3. Select **“Machine learning”** and then the **“Quick create”** link.
4. In the workspace box enter a name for the workspace. I’m using **“IoTHackathonScenario3”**.
5. Select a location and subscription.
6. Under storage account select **“Create a new storage account”** and enter a name for the storage account.
7. Click the **“Create an ML workspace”** link. A new ML workspace will be created, however this may take a few minutes.
8. When your ML workspace has finished being provisioned select the name of your workspace from the list of ML workspaces under the Machine Learning tab and click the bottom **“Open in studio”** icon.
9. Once in the ML Studio click the bottom **“+”** icon and select **“Dataset”** from the menu. Next select the **“From local file”** option.
10. Click the “choose file” button and select the **“Scenario3TrainingData.csv”** file.
11. In the “Enter a name for the new dataset” textbox enter a name for the dataset. I’ve called mine **“TrainingData”**.
12. From the **“Select a dataset type…”** drop down select the **“Generic CSV File with a header (.csv)”** option.
13. Finally click the **“tick”** icon. Once the file has uploaded click the **“+”** icon again.
14. This time from the menu select **“Experiment”** and choose the **“Blank experiment”** template. This will create a new experiment and load the ML experiment editor.
15. Your experiment will have a default name of something like “Experiment created….”. Click this text and give your experiment a name.
16. From the left hand menu expand the “Saved datasets” node and then expand the “My Datasets” node. The dataset you just created should appear.
17. Drag and drop the dataset onto the canvas. You will notice the background template disappears and your dataset is now shown on the canvas.
18. In the **“search experiment items”** textbox type **“Clean”**. You will see the list of items changes and a few convert options are shown. Drag and drop the “Clean Missing Data” option onto the canvas below the dataset.
19. On the canvas click back on the dataset and a number 1 should appear at the bottom of the item (output). Click and drag from this to the green dot that appeared on the Clean Missing Data item (input). The two should now show a line between them.  
    
20. Next again using the **“search experiment items”** feature type **“normalize”** and drag and drop the **“Normalize Data”** item onto the canvas again below the Clean Missing Data item.
21. Click and drag from the Clean Missing Data item to the Normalize Data item.
22. Next add a **“Project Columns”** item to the canvas. This time connect the left output of the Normalize Data to the Project Columns input.
23. Click the Project Columns item on the canvas and on the right hand side of the editor you should see properties. Click the **“Launch column selector”** button.
24. Make sure the “No columns” option is selected and in the text box type **“Temperature”** + Enter, **“Humidty”** + Enter, **“Fail”** + Enter. Finally click the **“tick”** icon.
25. Add a “**Split Data**” item below this again connecting the output of the Project Columns item to the input of the Split Data.
26. Click the Split Data item on the canvas and in the properties panel change the **“Fraction of rows in the first output dataset”** to “**.75**”.
27. Change the **“Stratified split”** to **“True”** and click the **“Launch column selector”**.
28. In the textbox enter **“Fail”** + Enter.
29. On the same level as the Split Data item add a **“Two-Class Neural Network item”** and a **“Two-Class Logistic Regression”** item. Don’t connect these for the moment.
30. Next below the Split Data item add two **“Train Model”** items **–** one to the left and one to the right of Split Data.Connect the left output of Split Data to the right input of the left Train Model item, then drag and drop a second connection from the left output of the split model item to the right input of the second Train Model item.
31. With the first Train Model item selected click the **“Launch column selector”** button and enter **“Fail”** + Enter into the textbox. Repeat this for the second Train Model item.
32. Connect the Two-Class Neural Network item to the left hand input of the first Train Model item. Connect the Two-Class Logistic Regression output to the second Train Model item left hand input.
33. Add two **“Score Model”** items to the canvas. One below each of the Train Model items. Connect one to the left hand Train Model output and connect the other to the right hand Train Model item.
34. From the right hand output of the Split Data item connect to the right hand input of the Score Model items.
35. Finally add an **“Evaluate Model”** item to the canvas below the two Score Model items and connect one of the Score Model items outputs to the left hand Evaluate Model input and the other Score Model item output to the right hand Evaluate Model input.
36. Click the **“run”** icon at the bottom of the window to test the experiment runs ok. Once the experiment has run successfully all the items on the canvas should have a green tick.
37. Once the experiment has run successfully we need to convert the experiment into a API which we can consume in other apps.
38. Click the train clustering model item on the canvas and click the bottom **“Set up web service”** button and select **“Predictive web service (recommended) option”**. Azure will now create a web service that can call the ML experiment. The experiment should be similar to the below image.  
    
39. Save the experiment and click the **“Run”** icon.
40. Once the experiment has run you should now see a **“Deploy web service”**. Once completed your will be redirected to API dashboard for your experiment.

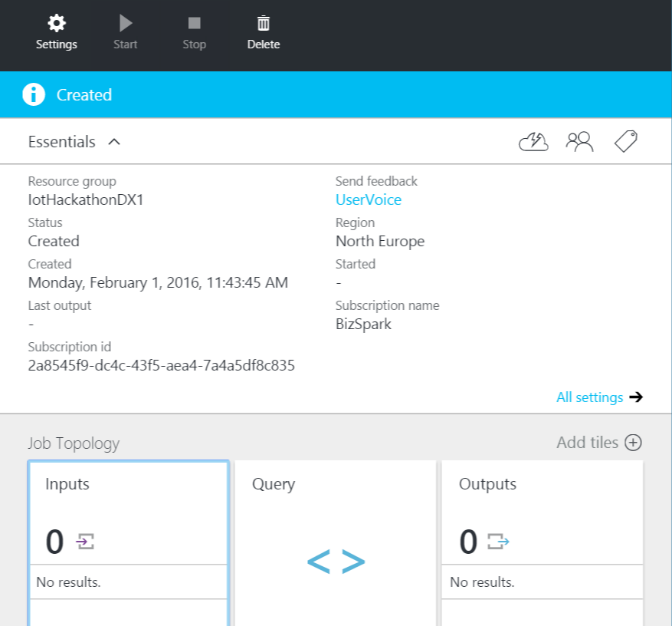
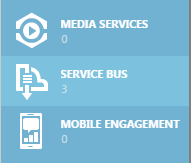
# Provisioning Stream Analytics

You are now going to provision and configure the Azure Stream Analytics service. This will be used to process your data from your IoT Hub, using an Azure ML function and output any detected faults to the Event Hub.

1. On the Preview Portal, click **“+ NEW”** (top left) and select **INTERNET OF THINGS.**
2. Select **Stream Analytics Job** from the list of featured apps.
3. Enter a suitable name for the job.
4. Make sure a subscription is selected.
5. Select a resource group for the Job. I would recommend the same resource group you created for the IoT Hub earlier.
6. Select a suitable region for the Job.
7. Click the **“Create”** button. The Stream Analytics Job will now be provisioned. As with the IoT Hub you can see the status of the provisioning process on your dashboard.

## Configuring the Stream Analytics Jobs

Once the job has provisioned the Preview Portal should automatically open up the job. For this Scenario there are two inputs and two outputs. I would recommend opening a second tab in your browser with the Azure portal in both tabs.

1. Under **“Job Topology”** click the “Inputs” box. You will then see a new tile open.
2. On the “Inputs” tile click the top left **“Add”** link.
3. In the **“Input Alias”** textbox enter“iothub”.
4. Select **“Data stream”** as the Source Type.
5. Select **“IoT Hub”** as the Source.
6. Next you need the information for the IoT Hub you created for this scenario. In the second browser tab from your **dashboard** select the IoT Hub you created earlier.
   1. Make a note of the IoT Hub name.
   2. From the settings tile select **“Shared access policies”.**
   3. Select the device policy.
   4. Copy the **“primary key”** for the policy.
7. Back in the previous browser tab enter the name of your IoT Hub in the IoT Hub textbox.
8. In the **“shared access policy name”** textbox enter **“service”**.
9. In the **“shared access policy key”** textbox paste the primary key you copied from the IoT Hub a few moments ago.
10. Leave the consumer group blank.
11. Make sure the Event serialization format is set to JSON and the Encoding is set to UFT-8.
12. Click the **“Create”** button. You will now see under the list on inputs the new input you just created.
13. Back under **“Job Topology”** click the “Outputs” box. You will then see a new tile open.
14. On the “Outputs” tile click the top left **“Add”** link.
15. In the **output alias** textbox enter **“eventhub”**.
16. Select “Event hub” from the Sink dropdown.
17. In your second browser window navigate to <http://manage.windowsazure.com/> and navigate to “Service Bus”.  
    
18. Select the service bus with the event hub namespace that you created earlier. The service bus namespace is now shown at the top of the screen. Copy this.
19. In your first browser window paste the service bus namespace into the **“Service bus namespace”** text box.
20. In your second browser window select the **“Event Hubs”** link and select the event hub. The event hub name is now shown at the top of the screen. Copy this.
21. In your first browser window paste the event hub name into the **“Event hub name”** textbox.
22. Back in the second browser window click the **“Connection Information”** icon at the bottom of the screen.
23. Click the **“Click here to configure”** link.
24. Under the “shared access policies” section enter a new policy name of **“streamanalytics”**.
25. From the permissions drop down select the **“Send”** option.
26. Now click the “Save” icon at the bottom of the screen.
27. Once saved a primary and second key will be shown. Copy the **“primary key”**.
28. Back in the first browser window in the **“Event hub policy name”** textbox enter **“streamanalytics”**.
29. In the **“Partition key column”** textbox enter **“stream”**.
30. Make sure the Event serialization format is set to JSON, encoding is set to UTF-8 and the Format is set to Line separated.
31. Click the **“create”** button.
32. Next we need to create a Stream Analytics function which will use the Azure ML web service we created earlier. At the moment configuring functions is only available in the old management portal. Navigate to <https://manage.windowsazure.com> and select Stream Analytics from the side menu. Click the name of the Stream Analytics service we just created.
33. Click the **“Functions”** link at the top of the screen.
34. Click the **“Add Function”** button at the bottom of the screen.
35. Enter **“predictfailure”** as the alias.
36. From the **“workspace”** drop down select the workspace you created earlier.
37. From the **“webservice”** drop down select the name of the ML webservice you created.
38. Finally click the **“tick”** icon to complete the setup.
39. You are now ready to add the Query. Click the **“Query”** link at the top of the screen and in the query editor enter the following query: -  
    WITH subquery AS

(SELECT temperature, humidity, predictfailure(temperature, humidity, 0) as result from iothub)

Select temperature, humidity, result.[Scored Labels]

Into eventhub

From subquery

WHERE CAST(result.[Scored Labels] as float) >= 1

1. Click the **“Save”** button at the bottom of the screen.
2. Once saved click the **“Start”** button at the bottom of the screen to start the job.

# Configure the web job

Your final task is to configure the web job. The web job reads events created by Stream Analytics on the Event Hub and sends a cloud to device message back to your Raspberry Pi to alert the user to the simulated fault.

1. Open the Scenario3-WebJob solution in Visual Studio.
2. In solution explorer right click on the top solution node and select **“Restore Nuget Packages”**. This will restore the packages the project uses for Azure.
3. In solution explorer double click the **“Properties”** node to open the project properties.
4. Set the **“IoTDeviceId”** property to the name you gave your device in the IoT Hub earlier – this should be **“industrialdevice”**.
5. Back in the Azure Portal from the dashboard select the storage account you created for this scenario. From the settings tile select **“Access Keys”**.
6. Copy the storage account name and paste into the **“StorageAccountName”** setting in the application settings.
7. Copy the **“Key1”** value from the portal and paste this into the **“StorageAccountKey”** setting in the application settings.
8. Next go back to the portals dashboard and select the IoT Hub you created for this scenario and select **“Shared access policies”**.
9. Select the **“service”** policy and copy the **“Connection string – primary key”** setting.
10. Paste this into the **“IoTHubConnectionString”** setting in your application settings.
11. Now go to <https://manage.windowsazure.com> and navigate to **“service bus”**, select the service bus namespace you created for this scenario and select the **“event hubs”** link.
12. Click the **“Configure”** link for the event hub.
13. In the **“shared access policies”** section enter **“webjob”** into the **“New Policy Name”** textbox.
14. From the drop down next to it select **“Listen”**.
15. Click the **“save”** button at the bottom of the screen.
16. Once saved click the **“Dashboard”** link.
17. Click the **“Connection Information”** button at the bottom of the screen and copy the **“webjob connection string”**.
18. Paste this value into the **“EventHubConnectionString”** setting in your application settings.
19. Save and close the properties window.
20. Right click on the project node in the Solution Explorer again and this time chose **“Publish as Azure WebJob…”**.
21. From the Publish window make sure the **WebJob run mode** is set to **“Run Continuously”**.
22. Click “Ok”. You might now see a dialog saying installing WebJob Nuget Package.
23. Under “Select a publish target” select **“Microsoft Azure Web Apps”**.
24. Next click the **“New”** button.
25. Enter a name for the **“Web App Name”**.
26. Under “App Service plan” select **“Create new App Service plan”** and enter a name for the plan.
27. Select the resource group you used for the rest of the Azure services you have provisioned.
28. Select the same region as the rest of the Azure services you created earlier.
29. Select no database from the **“Database server”** drop down.
30. Click the **“Create”** button.
31. The Web App will now be provisioned and may take a few moments.
32. Once the Web App has been provisioned click the **“Publish”** button. This will publish your Azure WebJob to the website and start running the WebJob automatically.