# Automate informational messages using Azure IoT Architecture

In this blog, we are going to learn about automating messages and emails using Azure IoT architecture. This is an introductory blog which will help us to know more about Azure IoT and its effective usage in our daily life.

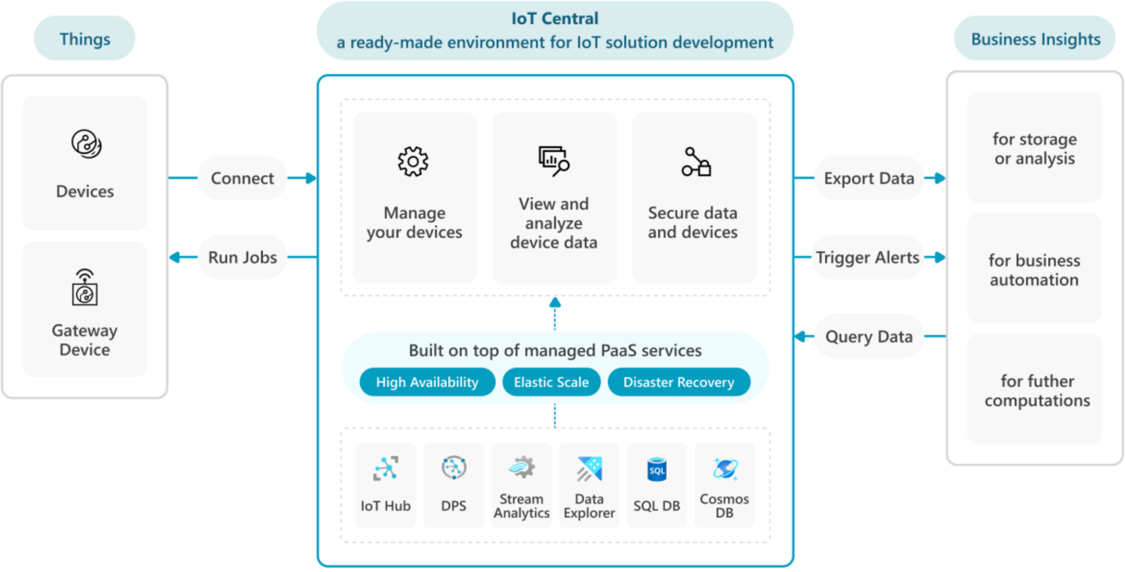
## Problem Statement

In our everyday life, messages and emails are an important factor. Starting from a professional email which has to be sent to close an important deal or texting someone about your whereabouts. We need internet to survive in this era of Emails and texts.

## Microsoft to the rescue!

**"Necessity is the mother of invention."** The primary drive of force for new invention is a need. As technology is evolving, AI is taking shape. Machines started to learn on its own. Automation is the new trend!

Microsoft Azure brings you IoT solutions by assembling Azure PaaS (Platform-as-a-service). Devices connected to cloud can access and explore the data to form customised insights about their environment. Azure IoT supports wide range of devices, supports smart server gateways, and lots more to explore. Devices can directly connect to Azure to send and receive messages on IoT solution.



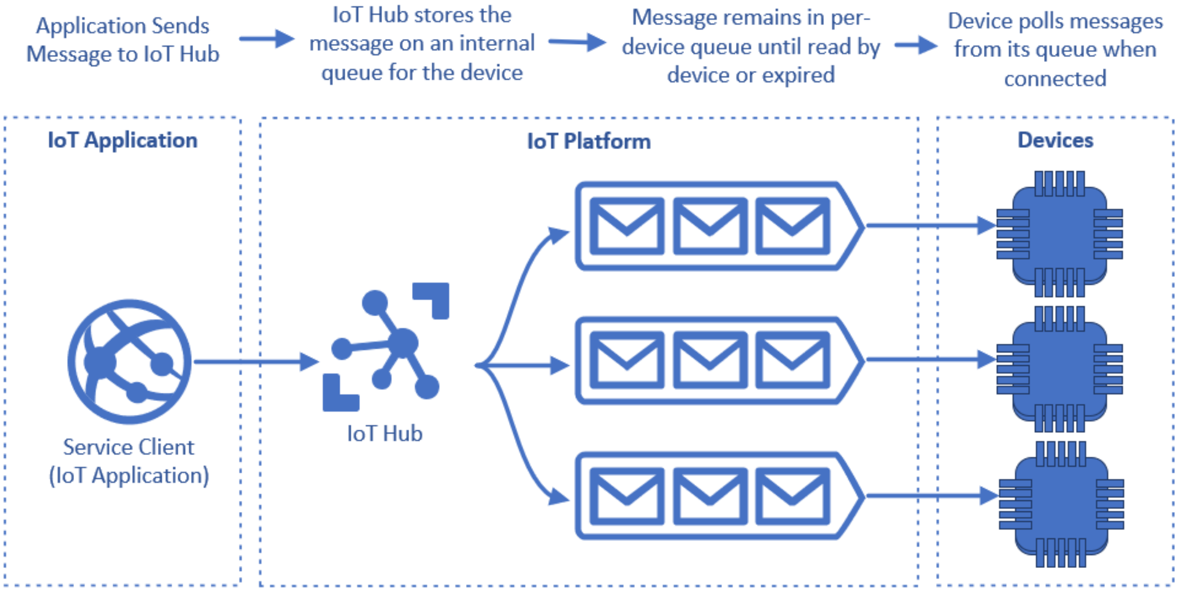
## IoT hub Message

Microsoft Azure allows bi-directional communication with devices. The IoT hub is used to communicate with your device by sending messages from your device to your IoT solution backend and vice-versa. Some of the features like cloud-to-device messaging, device-twins, and device management are only available in the standard tier of IoT hub, so keeping up to the top tier is a must.

There are two ways of creating and reading messages via IoT hub:

- Device-to-cloud messaging
- Cloud-to-device messaging

To use the above two methods for seamless interoperability across protocols IoT hub defines a common set of messaging protocols that are available in all device facing protocols.


Application Sends Message to IoT Hub
IoT Hub stores the message on an internal queue for the device
Message remains in per-device queue until read by device or expired
Device polls messages from its queue when connected
IoT Application
IoT Platform
Devices
Service Client (IoT Application)
IoT Hub

## Creating and Reading IoT Hub Messages

A device-to-cloud (D2C) has certain characteristics such as

- property names and values can only contain ASCII alphanumeric characters, plus {'!', '#', '$', '%, '&', ''', '*', '+', '-', '.', '^', '_', '`', '|', '~'}
- To use the message body in an IoT Hub routing query we must provide a valid JSON object for the message and set the content type property of the message to “application/json;charset=utf-8”

**The message body should look like this:**

```
{
  "timestamp": "2022-02-08T20:10:46Z",
  "tag_name": "spindle_speed",
  "tag_value": 100
}
```

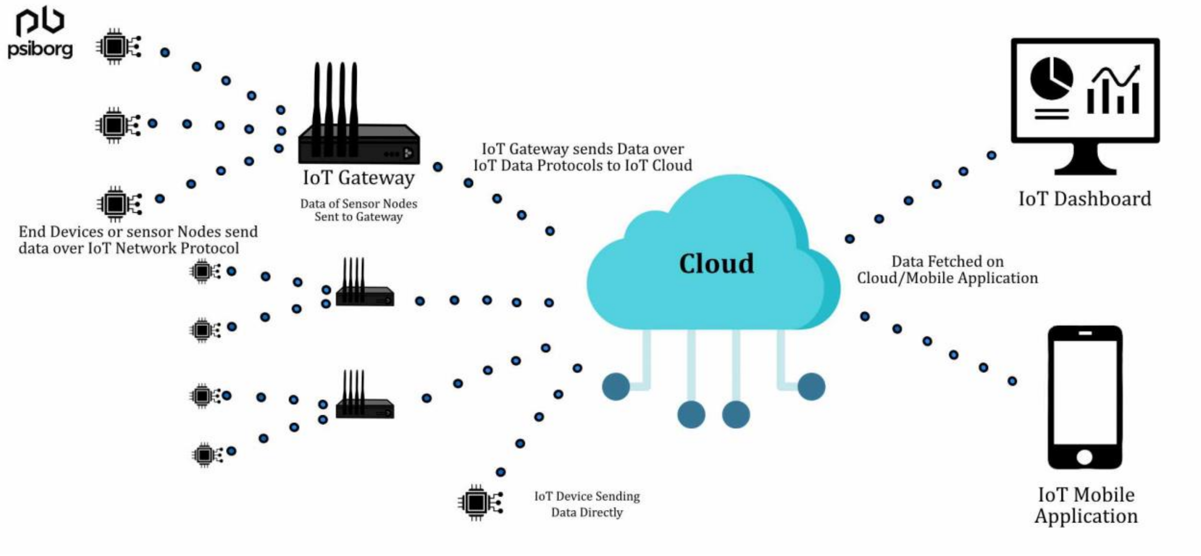
***(Source : https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-construct)***

A common use of application properties is to send a timestamp from the device using the **“iothub-creation-time-utc”** property to record when the message was sent by the device. The format of this timestamp must be UTC with no time zone information. For example, **2021-04-21T11:30:16Z** is valid, **2021-04-21T11:30:16-07:00** is invalid:

**The final message body:**

```
{
  "applicationId":"5782ed70-b703-4f13-bda3-1f5f0f5c678e",
  "messageSource":"telemetry",
  "deviceId":"sample-device-01",
  "schema":"default@v1",
  "templateId":"urn:modelDefinition:mkuyqxzgea:e14m1ukpn",
  "enqueuedTime":"2021-01-29T16:45:39.143Z",
  "telemetry":{
    "temperature":8.341033560421833
  },
  "messageProperties":{
    "iothub-creation-time-utc":"2021-01-29T16:45:39.021Z"
  },
  "enrichments":{}
}
```

***(Source : https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-construct)***


psiborg
IoT Gateway
Data of Sensor Nodes Sent to Gateway
End Devices or sensor Nodes send data over IoT Network Protocol
IoT Gateway sends Data over IoT Data Protocols to IoT Cloud
Cloud
IoT Device Sending Data Directly
IoT Dashboard
Data Fetched on Cloud/Mobile Application
IoT Mobile Application

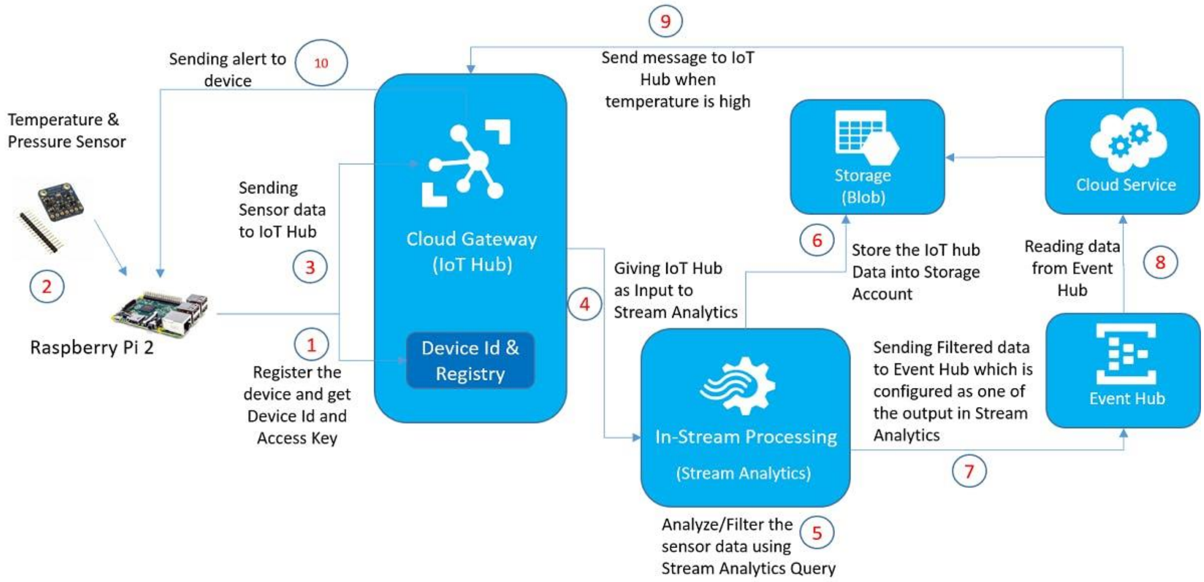
## Sending IoT hub messages

Send one-way notifications to a device app from the solution back end, send cloud-to-device messages from the IoT hub to the device. Please note, this feature is only in standard tier of IoT hub as mentioned earlier.

Send cloud-to-device messages through a service-facing endpoint, **“/messages/devicebound”**. A device then receives the messages through a device-specific endpoint, **“/devices/{deviceId}/messages/devicebound”.**

To target each cloud-to-device message at a single device, the IoT hub sets the to property

## “/devices/{deviceId}/messages/devicebound”


9
Send message to IoT Hub when temperature is high
Sending alert to device
10
Temperature & Pressure Sensor
Sending Sensor data to IoT Hub
3
2
Raspberry Pi 2
1
Register the device and get Device Id and Access Key
Cloud Gateway (IoT Hub)
Device Id & Registry
4
Giving IoT Hub as Input to Stream Analytics
Storage (Blob)
6
Store the IoT hub Data into Storage Account
Cloud Service
Reading data from Event Hub
8
Event Hub
Sending Filtered data to Event Hub which is configured as one of the output in Stream Analytics
7
In-Stream Processing
(Stream Analytics)
Analyze/Filter the sensor data using Stream Analytics Query
5

## Let’s receive messages via Iot hub using python

1. Open CMD and install Azure IoT Hub Device SDK for Python:

***“pip install azure-iot-device”***

2. create a file named SimulatedDevice.py add the following import statement:

***“import time***

***from azure.iot.device import IoTHubDeviceClient***

***RECEIVED_MESSAGES = 0”***

3. Define the following function to print received message:

***“def message_handler(message):***

***global RECEIVED_MESSAGES***

```
    RECEIVED_MESSAGES += 1
    print("")
    print("Message received:")

    # print data from both system and application (custom) properties
    for property in vars(message).items():
        print ("    {}".format(property))

    print("Total calls received: {}".format(RECEIVED_MESSAGES))”
```

4. Write this code to initialise the client and wait to receive C2D message:

```
“def main():
    print ("Starting the Python IoT Hub C2D Messaging device sample...")

    # Instantiate the client
    client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)

    print ("Waiting for C2D messages, press Ctrl-C to exit")
```

```
try:
    # Attach the handler to the client
    client.on_message_received = message_handler

    while True:
        time.sleep(1000)
except KeyboardInterrupt:
    print("IoT Hub C2D Messaging device sample stopped")
finally:
    # Graceful exit
    print("Shutting down IoT Hub Client")
    client.shutdown()”
```

5. Add the following main function and save and close “SimulatedDevice.py”:

```
“if __name__ == '__main__':
    main()”
```

## Let’s send messages via Iot hub using python

1. Open CMD and create a file named **SendCloudToDeviceMessage.py** and add the following import statement at the start:

```
“import random
import sys
```

```
from azure.iot.hub import IoTHubRegistryManager


MESSAGE_COUNT = 2

AVG_WIND_SPEED = 10.0

MSG_TXT = "{\"service client sent a message\": %.2f}"”
```

2. Add the following code to send messages:

```
“def iothub_messaging_sample_run():

    try:

        # Create IoTHubRegistryManager

        registry_manager =
IoTHubRegistryManager(CONNECTION_STRING)


        for i in range(0, MESSAGE_COUNT):

            print ( 'Sending message: {0}'.format(i) )

            data = MSG_TXT % (AVG_WIND_SPEED +
(random.random() * 4 + 2))


            props={}

            # optional: assign system properties

            props.update(messageId = "message_%d" % i)

            props.update(correlationId = "correlation_%d" % i)

            props.update(contentType = "application/json")
            # optional: assign application properties
            prop_text = "PropMsg_%d" % i
            props.update(testProperty = prop_text)

            registry_manager.send_c2d_message(DEVICE_ID,
data, properties=props)

        try:
            # Try Python 2.xx first
            raw_input("Press Enter to continue...\n")
        except:
            pass
            # Use Python 3.xx in the case of exception
            input("Press Enter to continue...\n")

    except Exception as ex:
        print ( "Unexpected error {0}" % ex )
        return
    except KeyboardInterrupt:
        print ( "IoT Hub C2D Messaging service sample
stopped" )”
```

**3.** Add the following **“main.py”** file:

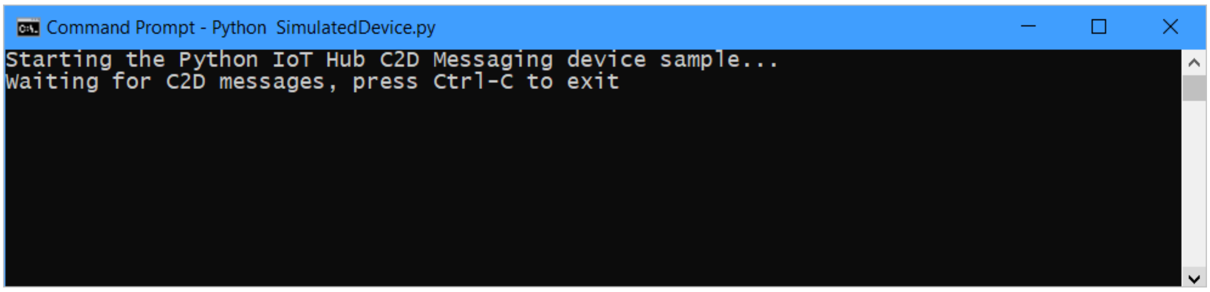
***“if __name__ == '__main__':***

***print ( "Starting the Python IoT Hub C2D Messaging service sample..." )***

***iothub_messaging_sample_run()”***

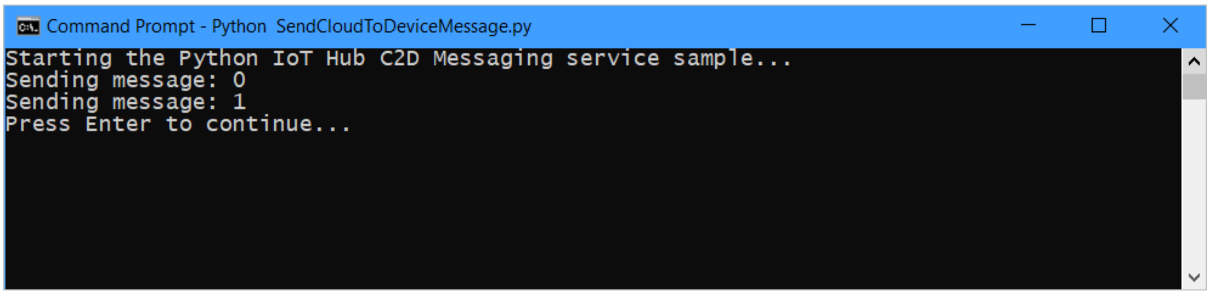
**Now we are ready to run the application!!**

- In CMD write : “**python SimulatedDevice.py**”



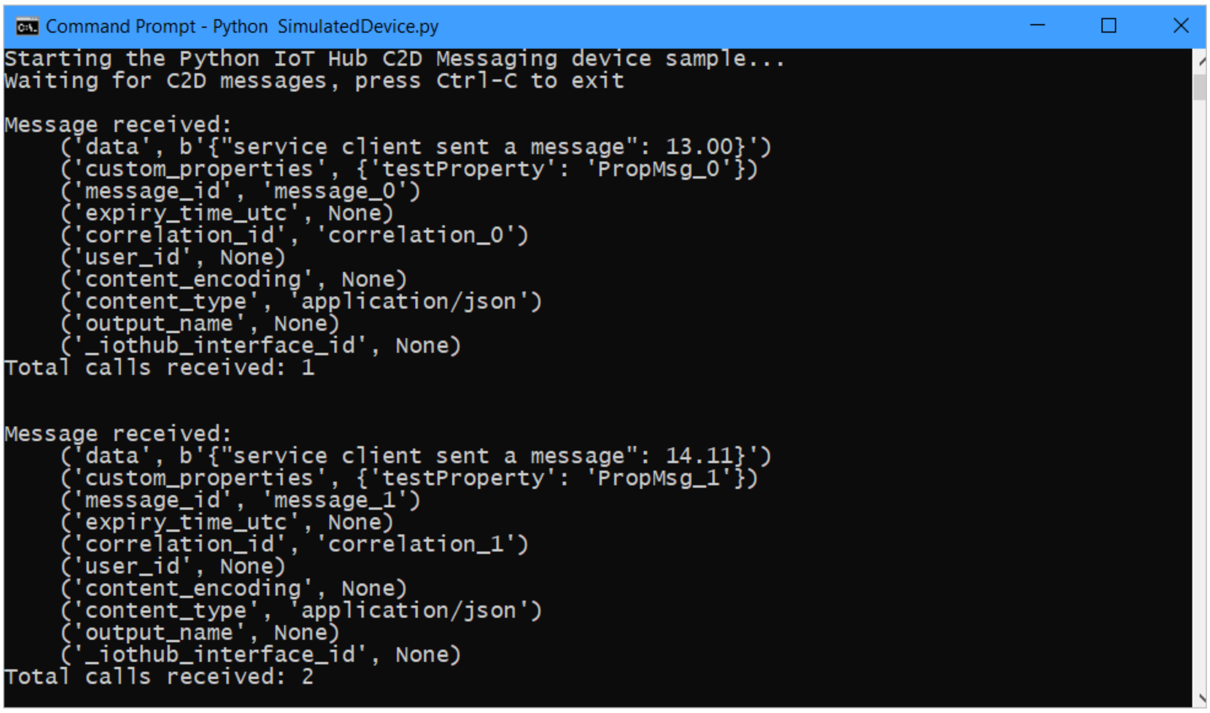
```
Starting the Python IoT Hub C2D Messaging device sample...
Waiting for C2D messages, press Ctrl-C to exit
```

- Open a new CMD and run this command : “**python SendCloudToDeviceMessage.py**”



```
Starting the Python IoT Hub C2D Messaging service sample...
Sending message: 0
Sending message: 1
Press Enter to continue...
```

- The final output should be:



```
Starting the Python IoT Hub C2D Messaging device sample...
Waiting for C2D messages, press Ctrl-C to exit

Message received:
    ('data', b'{"service client sent a message": 13.00}')
    ('custom_properties', {'testProperty': 'PropMsg_0'})
    ('message_id', 'message_0')
    ('expiry_time_utc', None)
    ('correlation_id', 'correlation_0')
    ('user_id', None)
    ('content_encoding', None)
    ('content_type', 'application/json')
    ('output_name', None)
    ('_iothub_interface_id', None)
Total calls received: 1


Message received:
    ('data', b'{"service client sent a message": 14.11}')
    ('custom_properties', {'testProperty': 'PropMsg_1'})
    ('message_id', 'message_1')
    ('expiry_time_utc', None)
    ('correlation_id', 'correlation_1')
    ('user_id', None)
    ('content_encoding', None)
    ('content_type', 'application/json')
    ('output_name', None)
    ('_iothub_interface_id', None)
Total calls received: 2
```

Hence we successfully learned how to create, read, and send a C2D and D2C messaging using Microsoft Azure IoT Hub.

## Challenges in implementing the solution

Microsoft Azure has newly launched this IoT hub and learning to use this has been an uneven journey. Since it is connected with cloud and can be written in any programming language it becomes a relief to many new users. If someone has a good knowledge of cloud and IoT they are good to go with the newest IoT hub era.

## Business benefit

Automation is the new trend and is one of the most important factor in business. This IoT hub will make life a lot easier than thought and things will escalate faster than expected. Since cloud has more security, data security is ensured.

## References

- https://learn.microsoft.com/en-us/azure/iot-hub/iot-hub-python-python-c2d
- https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/iot

Catch up with me on LinkedIn

https://www.linkedin.com/in/ishita-biswas-521b191ba/

This blog is written by

~Ishita Biswas