d2c2d

Lab Workbook Five

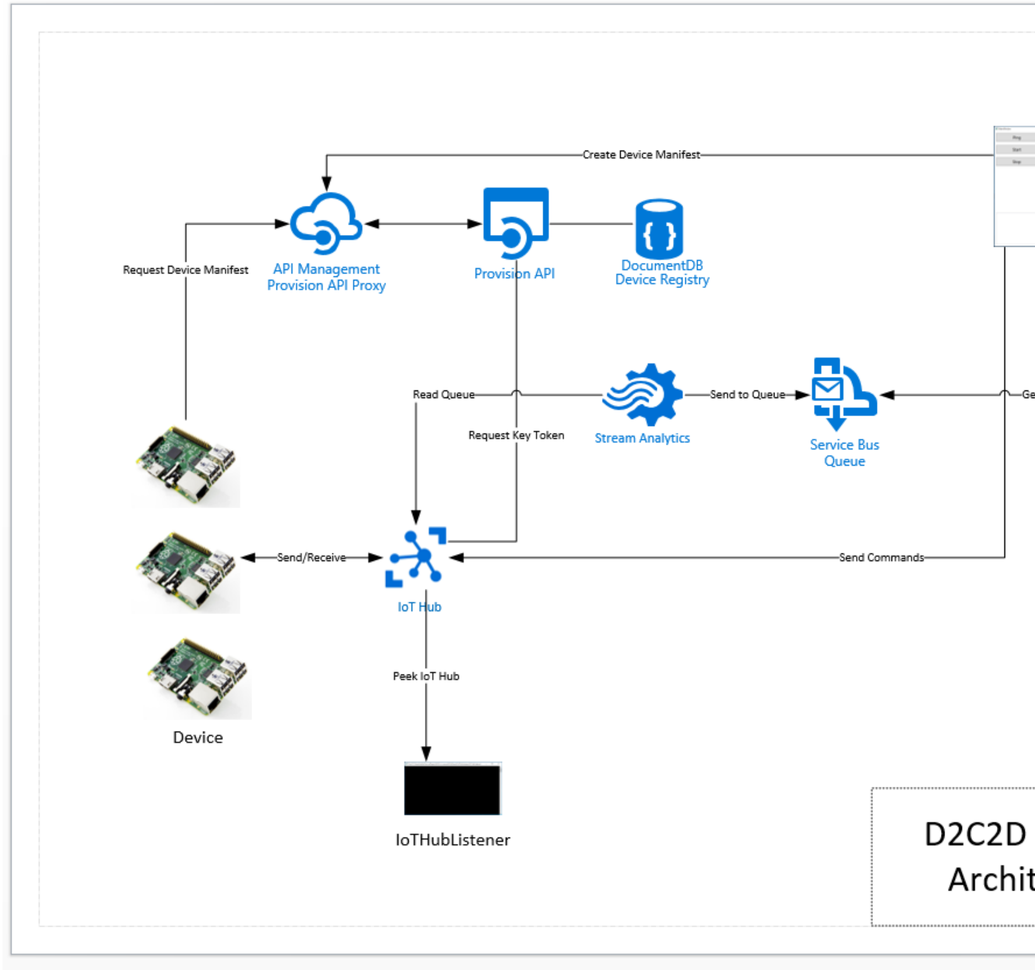
Device to Cloud to Device - a workshop for learning about Windows 10 Cor
development, Azure IoT Hub, Stream Analytics and automating Azure using

Workshop Overview

This training program provides foundational knowledge in how to architect and implem
solution using Windows 10 Core IoT hardware devices and Azure IoT Hub and Stream A
Device-to-Cloud and Cloud-to-Device communication patterns are discussed, designed
implemented using best practices.

At the conclusion of this workshop you will have provisioned, using PowerShell, an Azu
that contains IoT Hub, Stream Analytics Jobs that identify telemetry events and alarm s
Service Bus Namespace and set of message queues for backend integration.

You will also have developed a Windows 10 Core IoT application ("device") that sends t
receives incoming commands from the cloud as well as a real-time dashboard that can
directionally with the device (e.g., displaying telemetry readings and sending commanc
device).



Solution Architecture

The solution you will build and deploy consists of the following components:

- **Device** – a Windows 10 IoT Core IoT solution that dynamically connects to IoT h
heartbeat and climate telemetry as well as responds to command from a dashb
application can run on your local machine or be deployed to a Windows 10 Cor
as a Raspberry Pi.
- **Dashboard** – a Windows 10 WPF application that lists registered devices, maps
Bing Maps, and displays incoming device telemetry and alarms.
- **Provision API** – a ReST API that provides endpoints for device and device manif

- **IoT Hub –** IoT Hub provides device registration, incoming telemetry at scale, an
message services
- **DocumentDb** – DocumentDb is a NoSQL database service that is used for mana
Manifests, i.e., a Device Registry
- **Stream Analytics –** the solution leverages two Stream Analytics jobs, one that l
incoming messages and another that identifies alarm states and routes those m
second queue.

Lab Five Overview

In this lab you will deploy a new Stream Analytics Job that identifies alarm states in the
telemetry based on business rules that have been staged in Blob storage. These messag
the alarms Service Bus Queue and displayed in the Dashboard.

Lab

Step *Details*

1 **Provision a Stream Analytics Job that uses Reference Data**

In order to identify alarm states, you first need to know what the rules are for
i.e. what are the upper and lower bounds for temperature and humidity. Thes
lower bounds can be placed in a JSON file and staged in Blob storage.

The rules file which you will use is located in the automation/deploy/rules fol
devicerules.json. It contains a field for message type and the upper and lower
temperature and humidity. The rules file can be joined with any message that
Climate messages, enableing you to compare compare Temperature and Hum
the upper and lower bounds described by the rule.

```
[
  {
    "MessageType": 2,
    "TempUpperBound": 100.0,
    "TempLowerBound": 32.0,
    "HumidityUpperBound": 75.0,
    "HumidityLowerBound": 10.0
  }
]
```

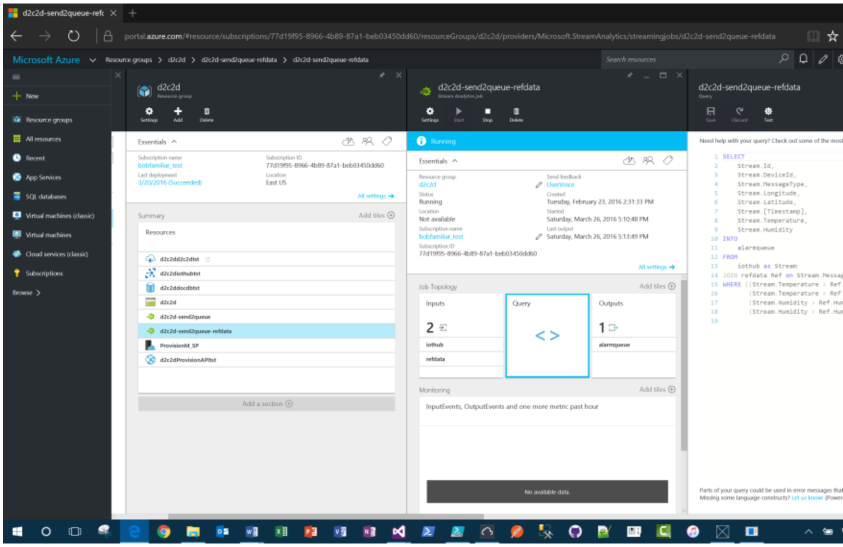
Stream Analytics has a feature where reference data can be applied as an add

```
        Stream.Humidity
    INTO
        alarmqueue
    FROM
        iothub as Stream
    JOIN refdata Ref on Stream.MessageType = Ref.MessageType
    WHERE ((Stream.Temperature > Ref.TempUpperBound) or
           (Stream.Temperature < Ref.TempLowerBound) or
           (Stream.Humidity > Ref.HumidityUpperBound) or
           (Stream.Humidity < Ref.HumidityLowerBound)
```

To provision this Stream Analytics job, run the 05-Provision-SAJob-2.ps1 script
the parameters as prompted:

➢ .\05-Provision- SAJob-2.ps1
Subscription: [the name of your subscription]
ResourceGroup: [the name of your resource group, d2c2d for exampl
Azure Location: [East US for example]
Prefix: [a unique prefix to be used in the naming of service componen
Suffix: [dev | tst | stg | prd]

Validate that the script provisions the Stream Analytics job – named 'd2c2d-al
by navigating to the Azure Portal Resource Groups screen and clicking on that
that the job has two inputs, one for IoT Hub and the other for the reference d


portal.azure.com/#resource/subscriptions/77d19f95-8966-4b89-87a1-beb03450dd60/resourceGroups/d2c2d/providers/Microsoft.StreamAnalytics/streamingjobs/d2c2d-send2queue-refdata

Resource groups > d2c2d > d2c2d-send2queue-refdata > d2c2d-send2queue-refdata

d2c2d
Resource group

Subscription name: bobfamiliar_test
Subscription ID: 77d19f95-8966-4b89-87a1-beb03450dd60
Last deployment: 3/20/2016 (Succeeded)
Location: East US

Resources
d2c2dd2c2dtst
d2c2diothubtst
d2c2ddocdbtst
d2c2d
d2c2d-send2queue
d2c2d-send2queue-refdata
ProvisionM_SP
d2c2dProvisionAPItst

d2c2d-send2queue-refdata
Stream Analytics job

Running

Resource group: d2c2d
Send feedback: UserVoice
Status: Running
Created: Tuesday, February 23, 2016 2:31:33 PM
Location: Not available
Started: Saturday, March 26, 2016 5:10:48 PM
Subscription name: bobfamiliar_test
Last output: Saturday, March 26, 2016 5:13:49 PM
Subscription ID: 77d19f95-8966-4b89-87a1-beb03450dd60

Job Topology
Inputs 2: iothub, refdata
Query
Outputs 1: alarmqueue

Monitoring
InputEvents, OutputEvents and one more metric past hour
No available data.

d2c2d-send2queue-refdata
Query

Need help with your query? Check out some of the most

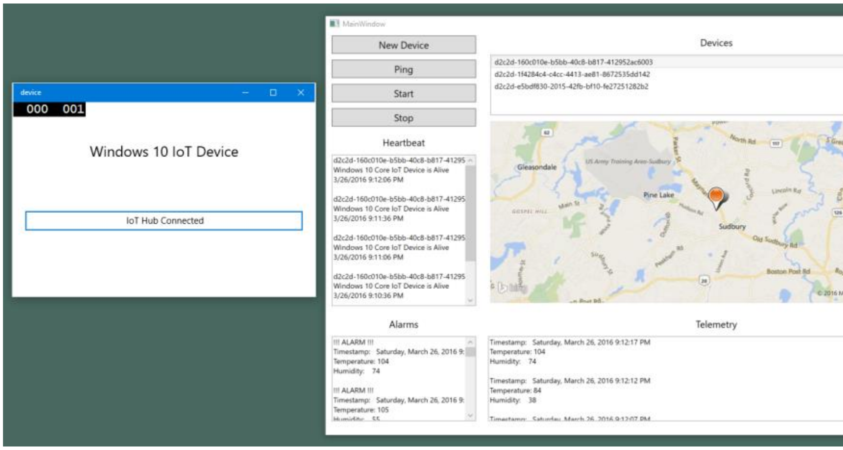
```
SELECT
    Stream.Id,
    Stream.DeviceId,
    Stream.MessageType,
    Stream.Longitude,
    Stream.Latitude,
    Stream.[Timestamp],
    Stream.Temperature,
    Stream.Humidity
INTO
    alarmqueue
FROM
    iothub as Stream
JOIN refdata Ref on Stream.Messag
WHERE ((Stream.Temperature > Ref.
       (Stream.Temperature < Ref.
       (Stream.Humidity > Ref.Hum
       (Stream.Humidity < Ref.Hum
```

Parts of your query could be used in error messages that
Missing some language constructs? Let us know! (Power

2 Update the Dashboard to display Alarm Messages

```
                {
                    var alarm = _alarmClient.Receive();
                    var messageBody = string.Empty;
                    if (alarm == null) continue;

                    try
                    {
                        messageBody = alarm.GetBody<string>();
                        var obj = JsonConvert.DeserializeObject<MessageBase>(me
                        switch (obj.MessageType)
                        {
                            case MessageTypeEnum.NotSet:
                                throw new Exception("Message Type Not Set");

                            case MessageTypeEnum.Ping:
                                break;

                            case MessageTypeEnum.Climate:

                                var climate = JsonConvert.DeserializeObject<Cli
                                     messageBody);

                                Application.Current.Dispatcher.Invoke(
                                      DispatcherPriority.Background,
                                    new ThreadStart(delegate
                                    {
                                        var currAlarm = AlarmFeed.Text;
                                        AlarmFeed.Text = string.Empty;
                                        AlarmFeed.Text += $"!!! ALARM !!!\r\n";
                                        AlarmFeed.Text += $"Timestamp:
{climate.Timestamp.ToLongDateString()} {climate.Timestamp.ToLongTimeString()}\r
                                        AlarmFeed.Text += $"Temperature:
{climate.Temperature}\r\n";
                                        AlarmFeed.Text += $"Humidity:
{climate.Humidity}\r\n\r\n";
                                        AlarmFeed.Text += $"{currAlarm}\r\n\r\r
                                    }));
                                break;

                            case MessageTypeEnum.Command:
                                // noop
                                break;

                            default:
                                throw new ArgumentOutOfRangeException();
                        }

                        alarm.Complete();
                    }
                    catch (Exception err)
                    {
                        Application.Current.Dispatcher.Invoke(
                            DispatcherPriority.Background, new ThreadStart(dele
                        {
                            var currAlarm = AlarmFeed.Text;
                            AlarmFeed.Text = string.Empty;
```

- ○ Start the Dashboard Solution
- ○ Click the Start Button



```
d2c2d-160c010e-b5bb-40c8-b817-412952ac6003
d2c2d-1f4284c4-c4cc-4413-ae81-8672535dd142
d2c2d-e5bdf830-2015-42fb-bf10-fe27251282b2
```

```
d2c2d-160c010e-b5bb-40c8-b817-41295
Windows 10 Core IoT Device is Alive
3/26/2016 9:12:06 PM

d2c2d-160c010e-b5bb-40c8-b817-41295
Windows 10 Core IoT Device is Alive
3/26/2016 9:11:36 PM

d2c2d-160c010e-b5bb-40c8-b817-41295
Windows 10 Core IoT Device is Alive
3/26/2016 9:11:06 PM

d2c2d-160c010e-b5bb-40c8-b817-41295
Windows 10 Core IoT Device is Alive
3/26/2016 9:10:36 PM
```

```
!!! ALARM !!!
Timestamp:   Saturday, March 26, 2016 9:
Temperature: 104
Humidity:    74

!!! ALARM !!!
Timestamp:   Saturday, March 26, 2016 9:
Temperature: 105
Humidity:    55
```

```
Timestamp:   Saturday, March 26, 2016 9:12:17 PM
Temperature: 104
Humidity:    74

Timestamp:   Saturday, March 26, 2016 9:12:12 PM
Temperature: 84
Humidity:    38

Timestamp:   Saturday, March 26, 2016 9:12:07 PM
```

3 Congratulations! You have competed Lab 5

Let's review:

- You deployed a new Stream Analytics job that uses a reference data fi business rules for temperature and humidity
- You updated the Dashboard with a background thread that listens for and displays them on the screen