BLU

d2c2d

Lab Workbook Two

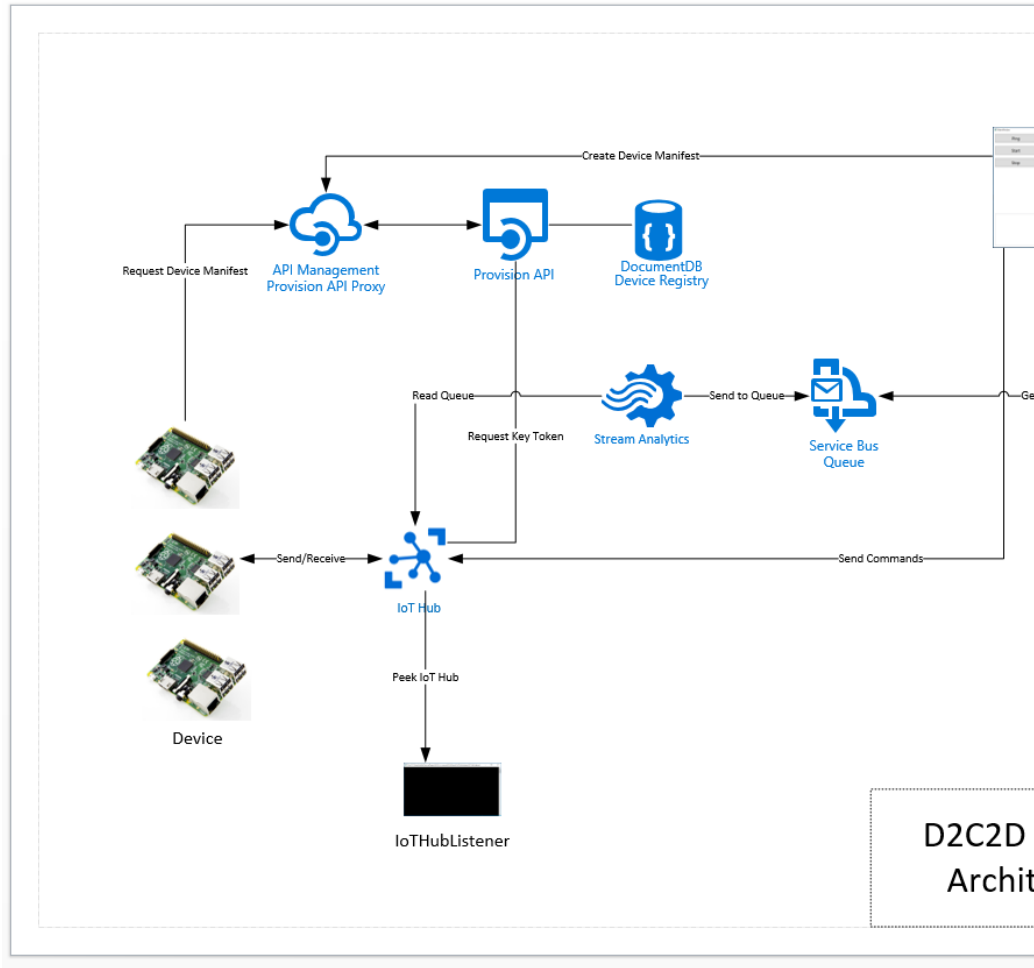
Device to Cloud to Device - a workshop for learning about Windows 10 Cor
development, Azure IoT Hub, Stream Analytics and automating Azure using

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 command
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

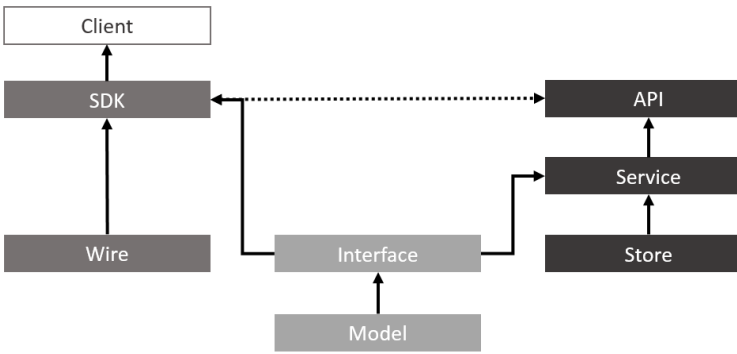
- **IoT Hub –** IoT Hub provides device registration, incoming telemetry at scale, an message services
- **DocumentDb** – DocumentDb is a NoSQL database service that is used for mana Manifests, i.e., a Device Registry
- **Stream Analytics –** the solution leverages two Stream Analytics jobs, one that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

Lab Two Overview

In this lab you will update, build, and deploy a microservice called Provision that provid registering a new device with IoT Hub registration and storing the device's manifest in will then create the Dashboard solution, configured with the appropriate connection st Dashboard to provision a new device.

Lab

Step	*Details*
1	**Complete the Provision Microservice Implementation** The Provision Microservice consists of an ASP.NET WebAPI solution and sever are built as NuGet packages. The WebAPI solution references the other packa the ReST routes deployed to Azure. The relationships among the projects are depicted here:



Microservice Project Construct

What you will note is that the Model and the Interface are reused by both the
SDK. The API solution defines the HTTP ReST routes, and the Service provides
implementation. The SDK exposes the same interface as the Service and uses
to invoke the ReST API.

In this step, you will complete the implementation of the API.

- From within Visual Studio, navigate to the microservices\provision\AP
  open the ProvisionAPI solution file.
- Open the Controllers\ProvisionMControllers.cs file and add an additio
  retrieving a device manifest by id
- Cut-and-paste the following code within the ProvisionMController cla
  constructor method:

```
[Route("provision/devicemanifests/id/{id}")]
[RequireHttps]
[HttpGet]
public DeviceManifest GetById(string id)
{
    return _provisionM.GetById(id);
}
```

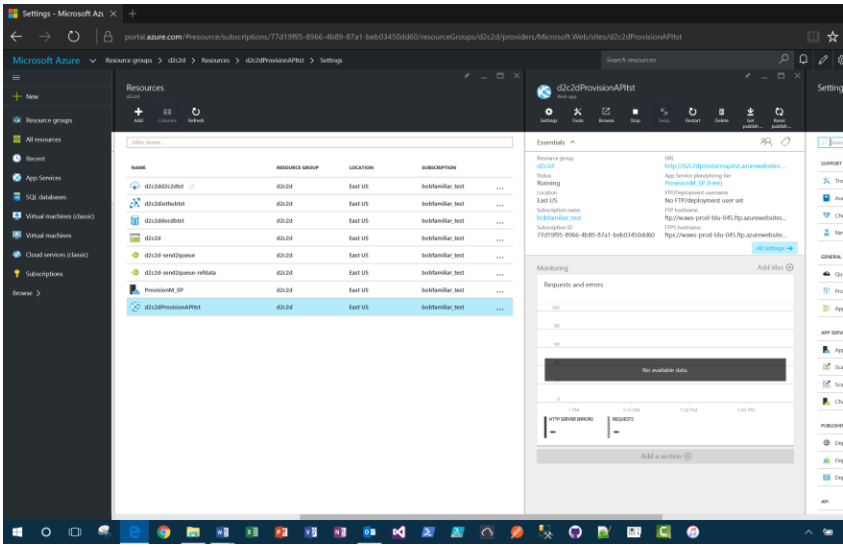
Save the solution and close Visual Studio.

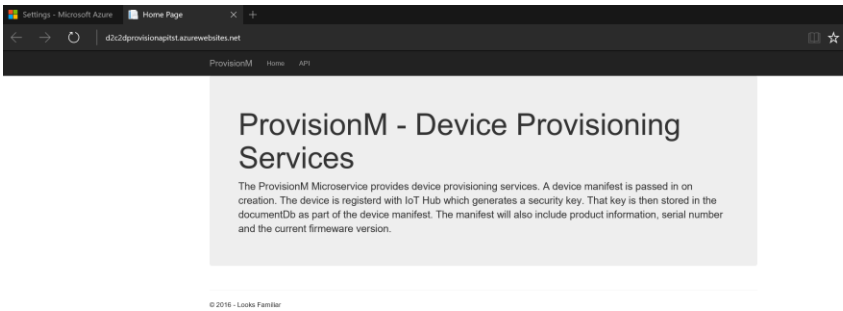
2 Build and Deploy the Provision Microservice

From within the PowerShell console, navigate to the automation folder at the
your source tree and type the following command:

- 03-Provision-Microservices
  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]
  Configuration: [debug | release]

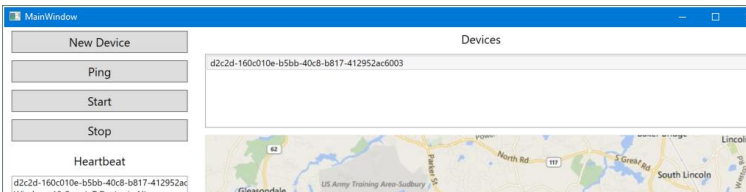
**Note that the parameters are the same as those used in Lab 1 to deploy the
services.** Validate that the deployment process has completed by navigating t
your Azure Web site, and click on the link to the service home page.





3 Dashboard

The Dashboard application will provide a command and control platform for t
environment. From the dashboard you will be able to provision devices, start
telemetry, see the location of the devices, and view the incoming telemetry a
In this step of the lab, you will augment the implementation of the Dashboard



- From within Visual Studio, open the Dashboard solution.
- Open the app.config file and update the values for each of the followi
    - Service Bus Connection String – retrieve from the Azure *Class*
      selecting the Service Bus and clicking the "Connection Informa
      at the footer of the portal.
    - IoT Hub Connection String – retrieved from the Azure Portal a

      1. Select the IoT hub from your list of resources
      2. Select the Settings Icon to open the Settings blade
      3. Select the Shared access policies item in the GENERAL grou
      4. Select iothubowner in the Shared access policies blade
      5. Copy the contents of the "Connection string – primary key

    - IoT Hub Name – full IoT Hub Hostname (e.g., myhub.azure-de
      the Azure Portal
    - Provision API – the URI for the Provision API; this is the link to
      Service you created in the last lab, with the resource name *pr*
      appended:

      `https://<prefix>provisiontapi<suffix>.azurewebsites.net/provis`

```
<appSettings>
    <add key="ServiceBusConnStr" value="" />
    <add key="IoTHubConnStr" value="" />
    <add key="IoTHubName" value="" />
    <add key="ProvisionAPI" value="" />
</appSettings>
```

The Dashboard requires several NuGet package references, some that are spe
and a couple that you have built locally, namely the Message Model and the P

- Open up the NuGet Manager Dashboard in Visual Studio
- With the source set to NuGet.Org, search for Microsoft Azure Devices
  reference to the Microsoft.Azure.Devices NuGet package
- With the source set to NuGet.Org, search for Service Bus. Add a refere
  WindowsAzure.ServiceBus NuGet package
- With the source set to the d2c2d NuGets folder location, add a refere
  MessageModelsNet4 NuGet package
- With the source set to the d2c2d NuGets folder location, add a refere
  ProvisionSDK NuGet package

- Add your Bing Maps Key as the Credential Provider property

Update the code behind for the Main Window:

- Open the MainWindow.xaml.cs file
- Add the following using statements:

```
using Looksfamiliar.d2c2d.MessageModels;
using LooksFamiliar.Microservices.Provision.SDK;
using Microsoft.Azure.Devices;
using Microsoft.ServiceBus.Messaging;
using Newtonsoft.Json;
using Map = Microsoft.Maps.MapControl.WPF;
using System.Windows.Threading;
```

You will need a set of members defined at the class level so that all the metho
class can access them.

- Define the following class members for working with Azure Service Bu
Provision SDK (directly above the line `public MainWindow()`)

```
private readonly QueueClient _messageClient;
private readonly QueueClient _alarmClient;
private ServiceClient _serviceClient;
private ProvisionM _provisionM;
private DeviceManifest _currDevice;
```

- Replace the MainWindow constructor to initialize the member variab
following:

```
public MainWindow()
{
    InitializeComponent();

    _messageClient = QueueClient.CreateFromConnectionString(
              ConfigurationManager.AppSettings["ServiceBusConnStr"], "m
    _alarmClient = QueueClient.CreateFromConnectionString(
              ConfigurationManager.AppSettings["ServiceBusConnStr"], "a
    _serviceClient = ServiceClient.CreateFromConnectionString(
              ConfigurationManager.AppSettings["IoTHubConnStr"], Transp

    _provisionM = new ProvisionM
    {
        ApiUrl = ConfigurationManager.AppSettings["ProvisionAPI"],
        DevKey = ConfigurationManager.AppSettings["DeveloperKey"]
    };
}
```

4 Provision a Device

Before a device can connect to IoT Hub and send telemetry, it must be registe
Additionally, you'll want to be able to easily build a list of registered devices a
metadata, such as model number, firmware revision, longitude, latitude, etc.
will add the code that provisions a new device with IoT Hub and stores the ma
DocumentDb. The ProvisionAPI that you built and deployed in the previous st
capability.

- In the MainWindow.xaml.cs file, locate the ProvisionButton_Click eve
you'll be adding code to this implementation in several phases below

```
private async void ProvisionButton_Click(object sender, RoutedEventArgs
```

- Add a call to obtain you location based on your IP address. The imple
GetLocationAsync is also included in the MainWindow.xaml.cs; it leve
API ip-api.com to identify your location.

```
var location = await GetLocationAsync();
```

- Create an instance of a DeviceManifest and initialize its properties:

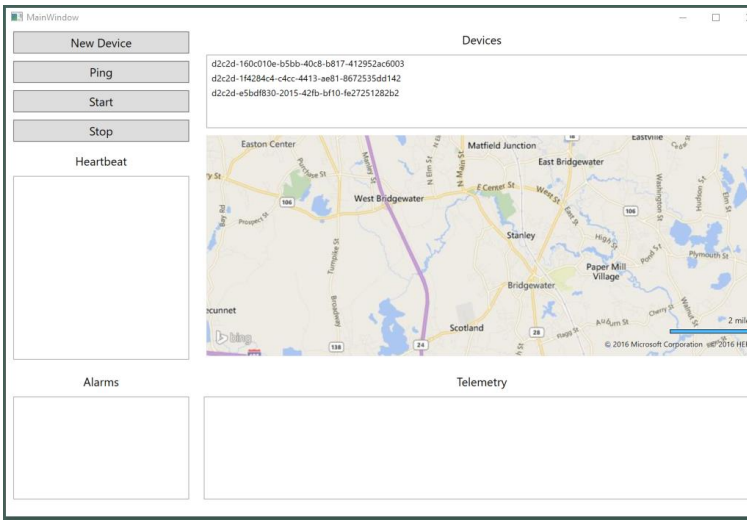
```
// initialize a device manifest
var manifest = new DeviceManifest
{
    latitude = location.lat,
    longitude = location.lon,
    manufacturer = "Looksfamiliar, Inc",
    model = "Weather Station - Win 10 Core IoT",
    firmwareversion = "1.0.0.0",
    version = "1.0.0.0",
    hub = ConfigurationManager.AppSettings["IoTHubName"],
    serialnumber = "d2c2d-" + Guid.NewGuid()
};
manifest.properties.Add(new DeviceProperty("Hardware Platform", "Ra
```

- Call the Create endpoint of the ProvisionM API (a message box will dis
successful)

```
// provision the device in IoT Hub and store the manifest in Docume
manifest = _provisionM.Create(manifest);

MessageBox.Show($"New Device Provisioned: {manifest.serialnumber}",
```

```
        }

        if (DeviceList.Items.Count <= 0) return;

        StartButton.IsEnabled = true;
        StopButton.IsEnabled = true;
        PingButton.IsEnabled = true;
    }
```

- Compile and run the Dashboard. Click the New Device button. You ca
  multiple devices.



5 Congratulations! You have completed Lab 2

Let's review:

- You completed the Provision Microservice implementation
- You built the Common Framework libraries
- You built and deployed the Provision Microservice
- You updated the Dashboard
- You provisioned a Device