

Device Explorer for IoT Hub

- Getting Device Explorer
- Configure an IoT Hub connection
- Manage devices
- List registered devices
- Create device
- Update device
- Delete device
- Get device connection string or data
- Monitor device-to-cloud events
- Send cloud-to-device messages

Getting Device Explorer

<https://github.com/Azure/azure-iot-sdks/releases>

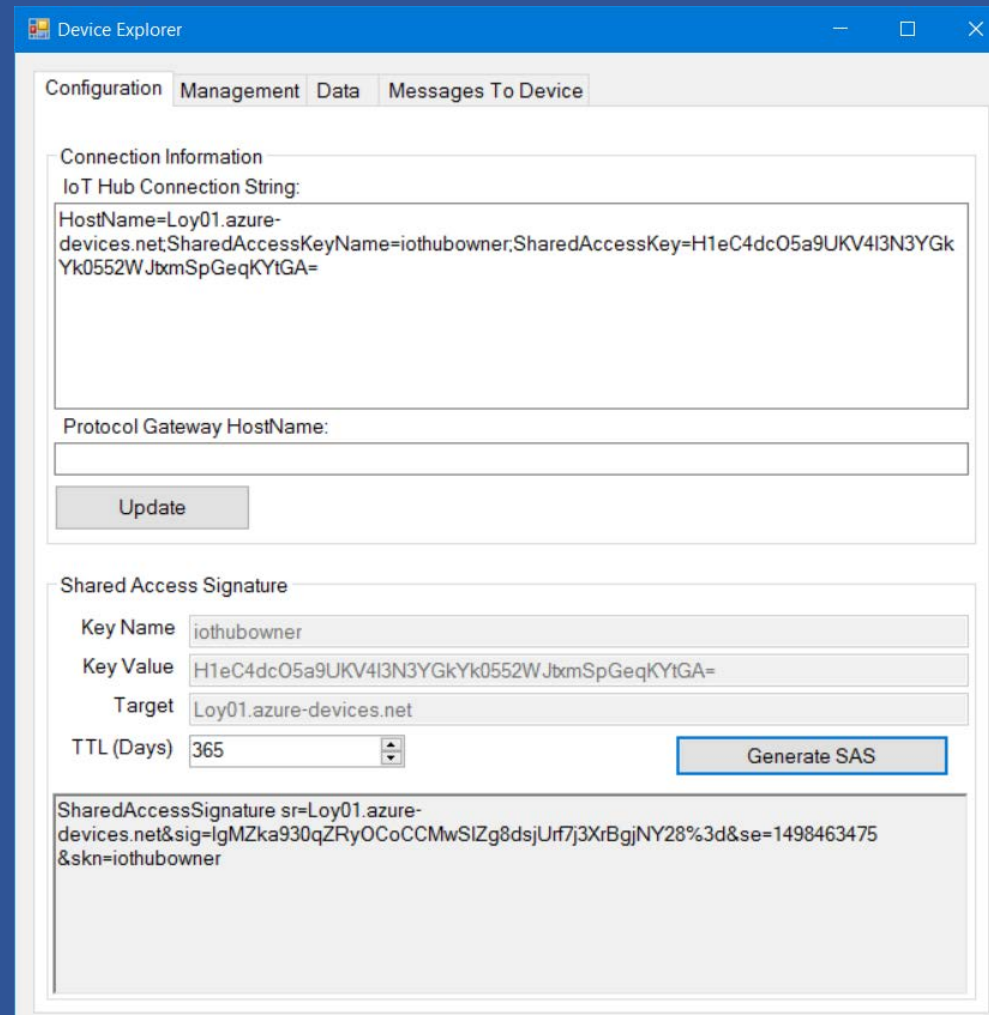
Downloads

 [SetupDeviceExplorer.msi](#)

 [Source code \(zip\)](#)

 [Source code \(tar.gz\)](#)

Configure an IoT Hub connection



The screenshot shows the 'Device Explorer' application window with the 'Configuration' tab selected. The 'IoT Hub Connection String' is displayed in a text box, and the 'Shared Access Signature' section is expanded, showing fields for Key Name, Key Value, Target, and TTL (Days). A 'Generate SAS' button is visible next to the TTL field.

Device Explorer

Configuration Management Data Messages To Device

Connection Information

IoT Hub Connection String:

HostName=Loy01.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=H1eC4dc05a9UKV4I3N3YGkYk0552WJbXmSpGeqKYtGA=

Protocol Gateway HostName:

Update

Shared Access Signature

Key Name: iothubowner

Key Value: H1eC4dc05a9UKV4I3N3YGkYk0552WJbXmSpGeqKYtGA=

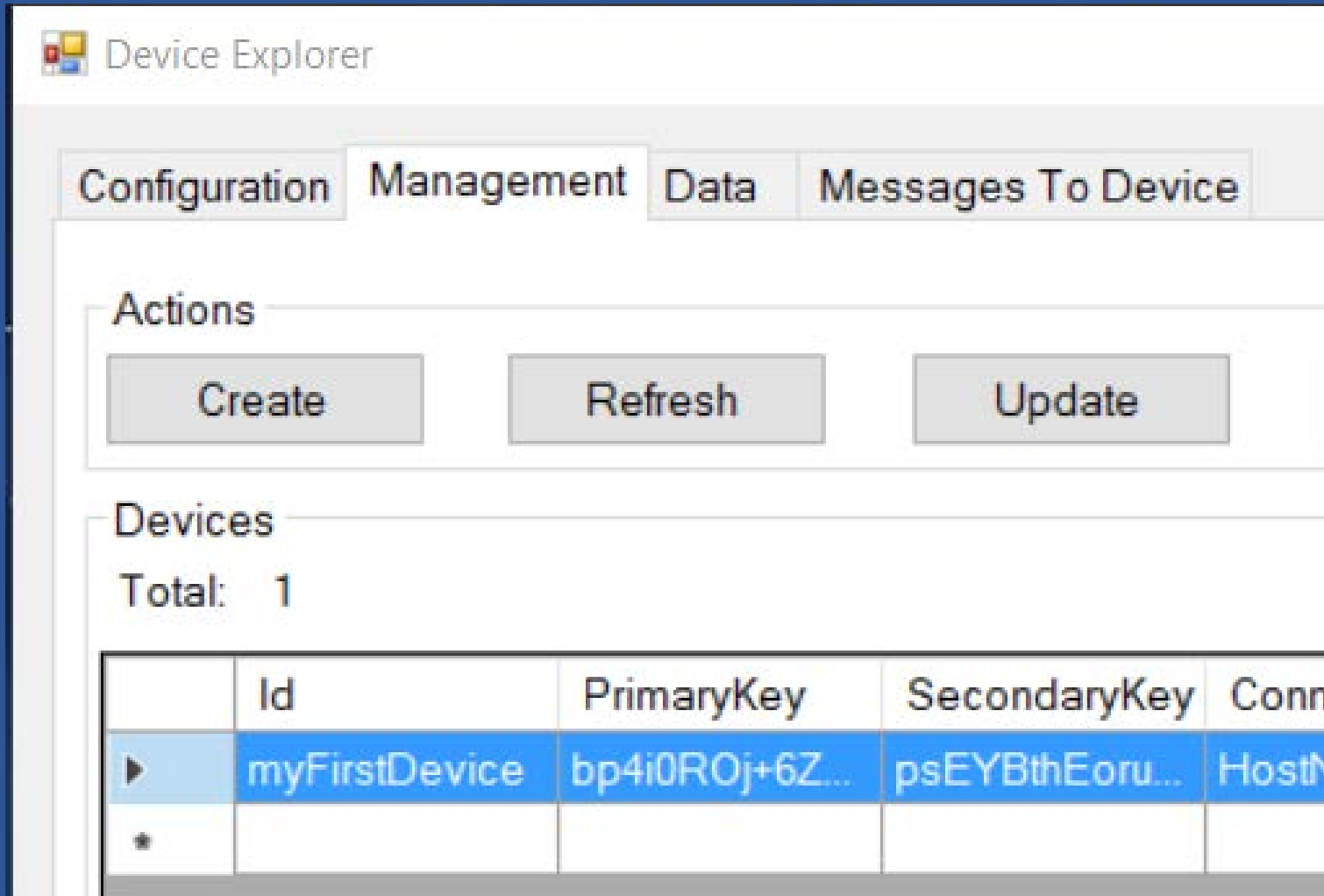
Target: Loy01.azure-devices.net

TTL (Days): 365

Generate SAS

SharedAccessSignature sr=Loy01.azure-devices.net&sig=lgMZka930qZRyOCcCMwSiZg8dsjUrf7j3XrBgjNY28%3d&se=1498463475&skn=iothubowner

Manage devices



The screenshot shows the 'Device Explorer' application window. It has four tabs: 'Configuration', 'Management' (which is selected), 'Data', and 'Messages To Device'. Under the 'Management' tab, there is an 'Actions' section with three buttons: 'Create', 'Refresh', and 'Update'. Below this is a 'Devices' section showing 'Total: 1'. A table lists the device details.

	Id	PrimaryKey	SecondaryKey	Conn
▶	myFirstDevice	bp4i0ROj+6Z...	psEYBthEoru...	HostN
★				

Create device (Add)

The screenshot shows the 'Create Device' dialog box in the Azure IoT Hub interface. The dialog is titled 'Create Device' and has a blue header bar. It contains three text input fields for 'Device ID', 'Primary Key', and 'Secondary Key'. The 'Device ID' field is empty. The 'Primary Key' field contains the value 'luzyyMFR7NkJkYP/RMmXEK15exg9meS4aY4Gjy2Xfz|='. The 'Secondary Key' field contains the value 'EG+W68D/4zil3dxVRWQeL1OrULEvBhbK2p+bmRVv3y0='. Below the input fields, there are two checkboxes: 'Auto Generate ID' (unchecked) and 'Auto Generate Keys' (checked). At the bottom of the dialog, there are two buttons: 'Create' and 'Cancel'.

Configuration Management Data Messages To Device

Actions

Create Refresh Update Delete

Create Device

Device ID:

Primary Key:

Secondary Key:

☐ Auto Generate ID ☒ Auto Generate Keys

Create Cancel

Update device

Configuration Management Data Messages To Device

Actions

Create Refresh **Update** Delete SAS T

Devices

Total: 2

	Id
▶	loyDev00
	myFirstDe
*	

DeviceUpdateForm

DeviceID

loyDev009

Primary Key

1AzYE0d3yuck3EX++HTRnWo2aghwkZ/5wCicv39Rk6Q=

Secondary Key

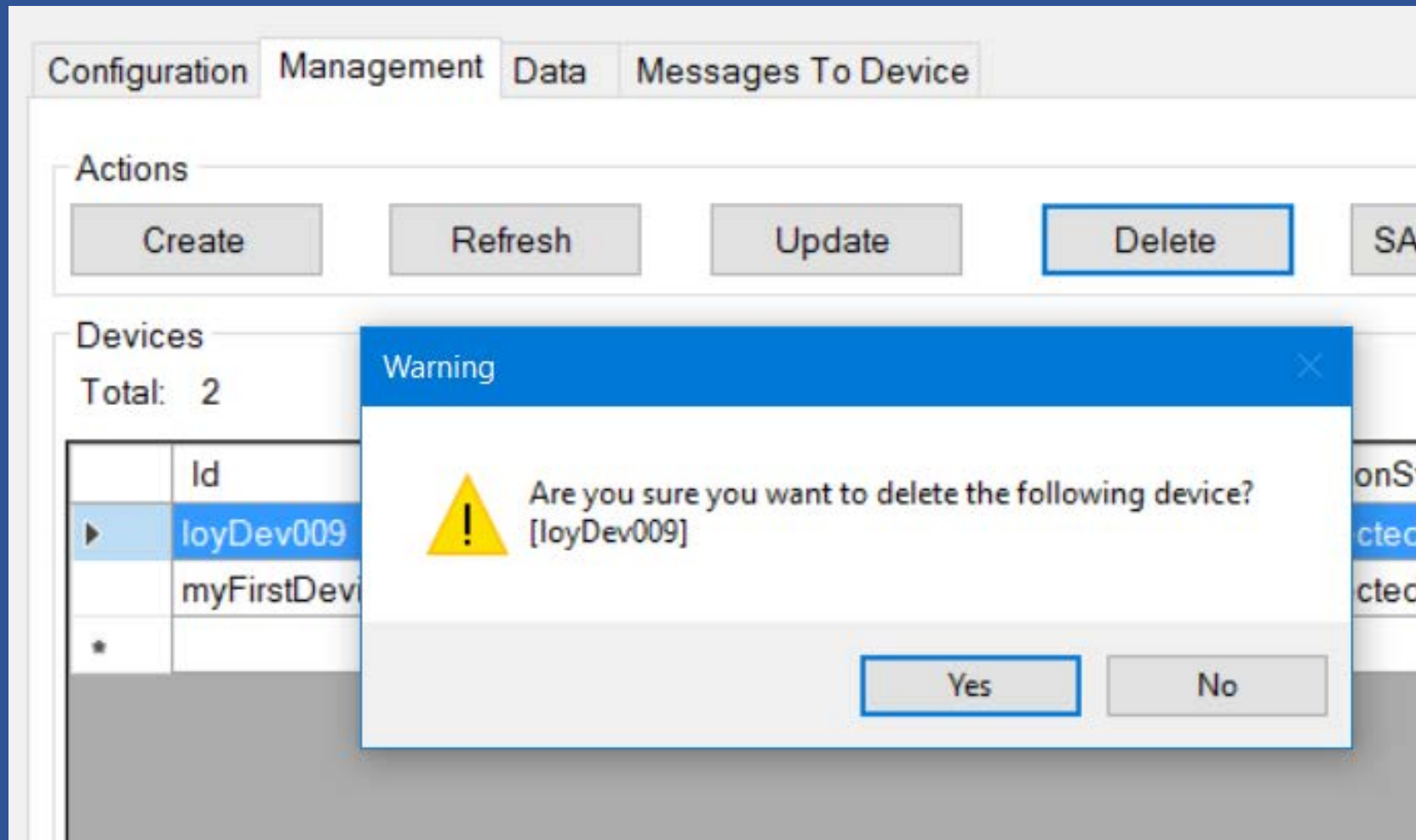
1t07UK1t0J8E+w5/x8zEv7Ydd+HdwQo38sbZSDq9QmA=

Update

Restore

Cancel

Delete device



Get device connection string or data

The screenshot shows the 'Data' tab in the IoT Hub Management console. The 'Actions' section contains buttons for 'Create', 'Refresh', 'Update', and 'Delete'. The 'Devices' section displays a table with columns: Id, PrimaryKey, SecondaryKey, and ConnectionString. The device 'MyNewDevice' is selected, and a context menu is open over it, offering three options: 'Copy data for all device', 'Copy data for selected device', and 'Copy connection string for selected device'.

	Id	PrimaryKey	SecondaryKey	ConnectionString
▶	MyNewDevice

- Copy data for all device
- Copy data for selected device
- Copy connection string for selected device

Monitor device-to-cloud events

Configuration Management Data Messages To Device

Monitoring

Event Hub: Loy01

Device ID: IoyDev009

Start Time: ☐ 06/26/2016 15:10:17

Consumer Group: \$Default ☐ Enable

Monitor Cancel Clear

Event Hub Data

Receiving events...

Send cloud-to-device messages

The screenshot shows a web application interface with four tabs: Configuration, Management, Data, and Messages To Device. The 'Messages To Device' tab is active. Below the tabs, the heading 'Send Message to Device:' is followed by three input fields: 'IoT Hub:' with the value 'Loy01', 'Device ID:' with the value 'loyDev009', and 'Message:' which is empty. Below these fields are two checkboxes: 'Add Time Stamp' and 'Monitor Feedback Endpoint', both of which are unchecked. At the bottom of the form are two buttons: 'Send' and 'Clear'. Below the buttons is an 'Output' section, which is currently empty.

Creating Device Simulator

- AMQP
- JSON
- Create Device Simulator
- Send Device to Cloud Message
- Receive Cloud to Device Message

AMQP



- The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware.
- The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security.

JSON

For embedded devices& low bandwidth, an another format is preferred JSON (JavaScript Object Notation)



JSON

```
{
  "id": 123,
  "title": "Object Thinking",
  "author": "David West",
  "published": {
    "by": "Microsoft Press",
    "year": 2004
  }
}
```

XML

```
<?xml version="1.0"?>
<book id="123">
  <title>Object Thinking</title>
  <author>David West</author>
  <published>
    <by>Microsoft Press</by>
    <year>2004</year>
  </published>
</book>
```

- JSON parser needs to be implemented to analyze server response
- JSON serializer to publish some data on the cloud

Create Device to Cloud Simulator


Device Simulator 1

Device to Cloud Data

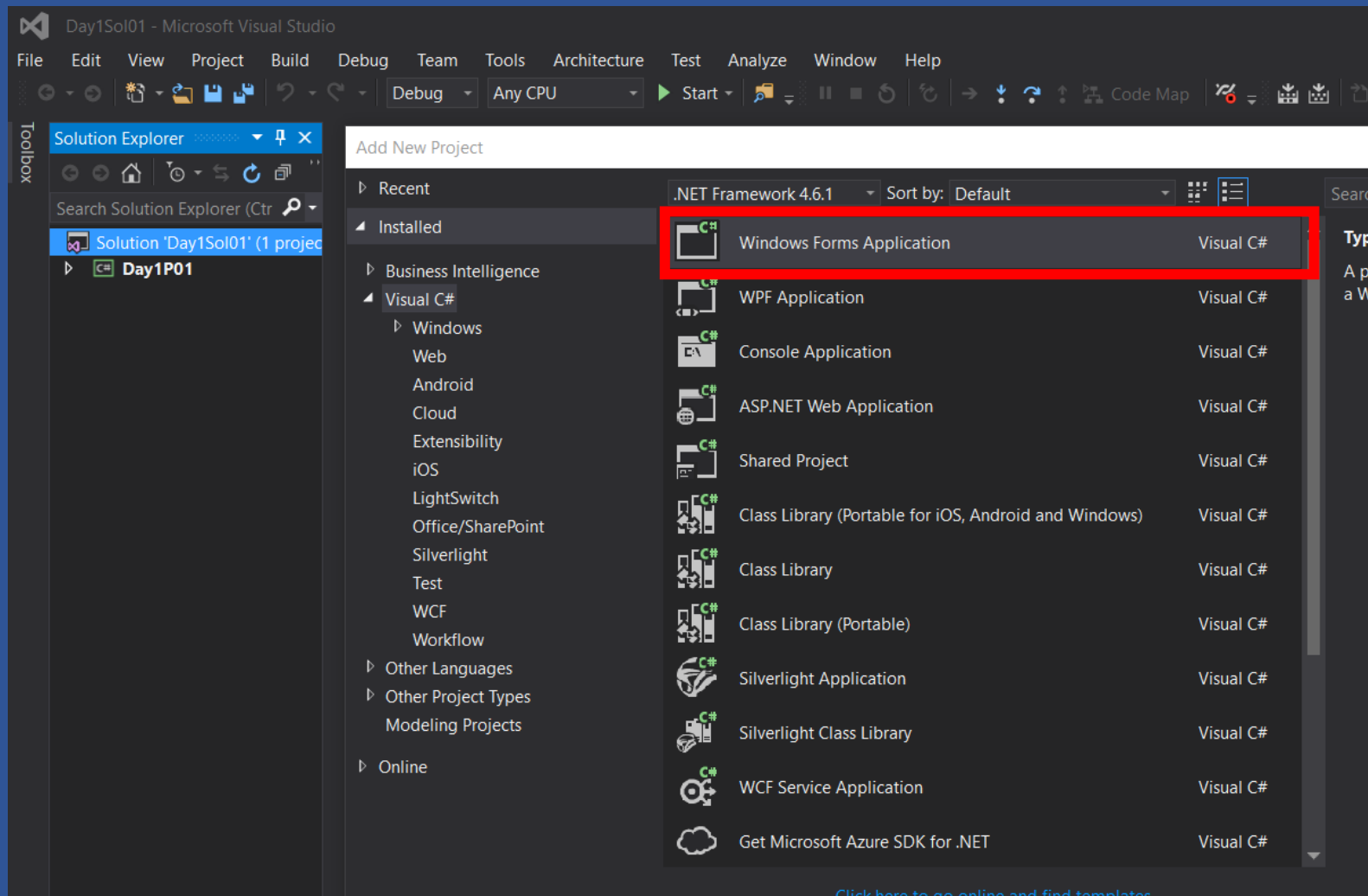
```
6/27/2016 1:52:18 PM{"deviceId":"myFirstDevice","Temperature":10.7422229218028}  
6/27/2016 1:52:26 PM{"deviceId":"myFirstDevice","Temperature":8.35227697405604}  
6/27/2016 1:52:34 PM{"deviceId":"myFirstDevice","Temperature":9.96233102630932}  
6/27/2016 1:52:42 PM{"deviceId":"myFirstDevice","Temperature":7.5723850785626}  
6/27/2016 1:52:50 PM{"deviceId":"myFirstDevice","Temperature":9.18243913081588}  
6/27/2016 1:52:58 PM{"deviceId":"myFirstDevice","Temperature":10.7924931830692}  
6/27/2016 1:53:06 PM{"deviceId":"myFirstDevice","Temperature":8.40254723532244}  
6/27/2016 1:53:14 PM{"deviceId":"myFirstDevice","Temperature":10.0126012875757}  
6/27/2016 1:53:22 PM{"deviceId":"myFirstDevice","Temperature":7.622655339829}  
6/27/2016 1:53:30 PM{"deviceId":"myFirstDevice","Temperature":9.23270939208228}
```

Sensor Data

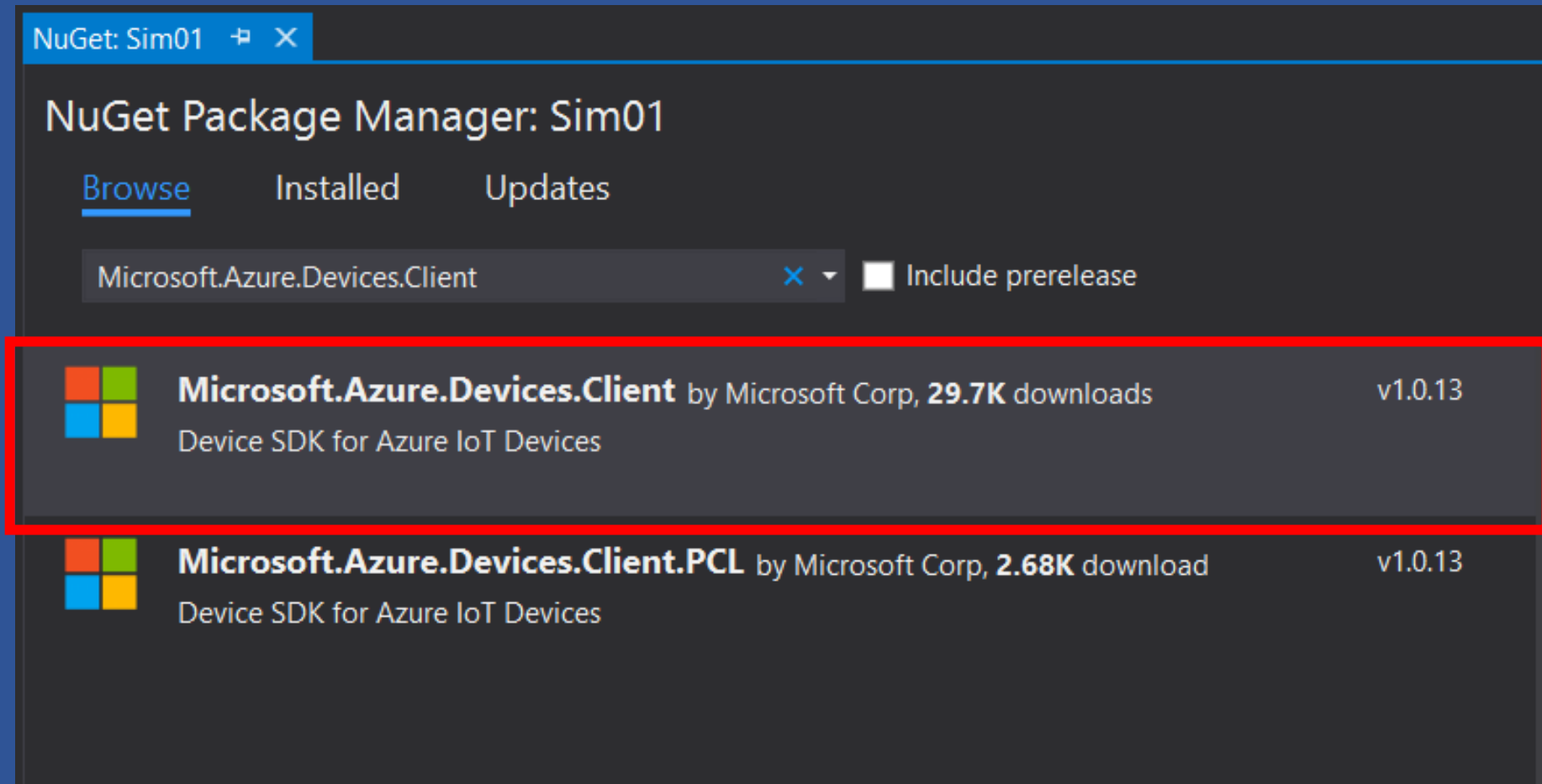
```
10.7422229218028  
8.35227697405604  
9.96233102630932  
7.5723850785626  
9.18243913081588  
10.7924931830692  
8.40254723532244  
10.0126012875757  
7.622655339829  
9.23270939208228
```



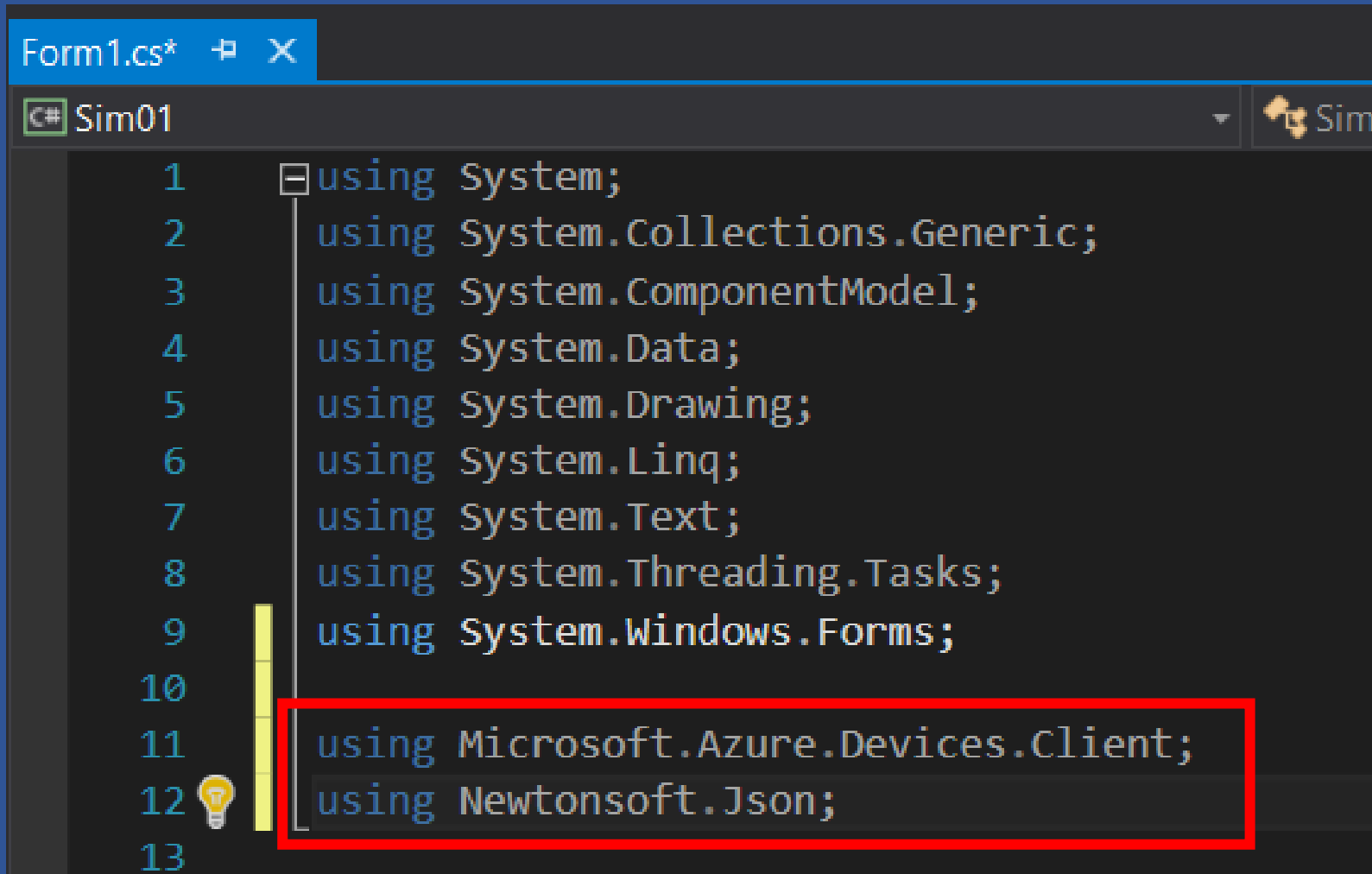
1. Create New Windows Forms Application “Sim01”



2. Add NuGet Package: Microsoft.Azure.Devices.Client



3. Add the following using statement at the top of the Form1.cs



```
Form1.cs*  [icon] [X]
C# Sim01
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11  using Microsoft.Azure.Devices.Client;
12  using Newtonsoft.Json;
13
```

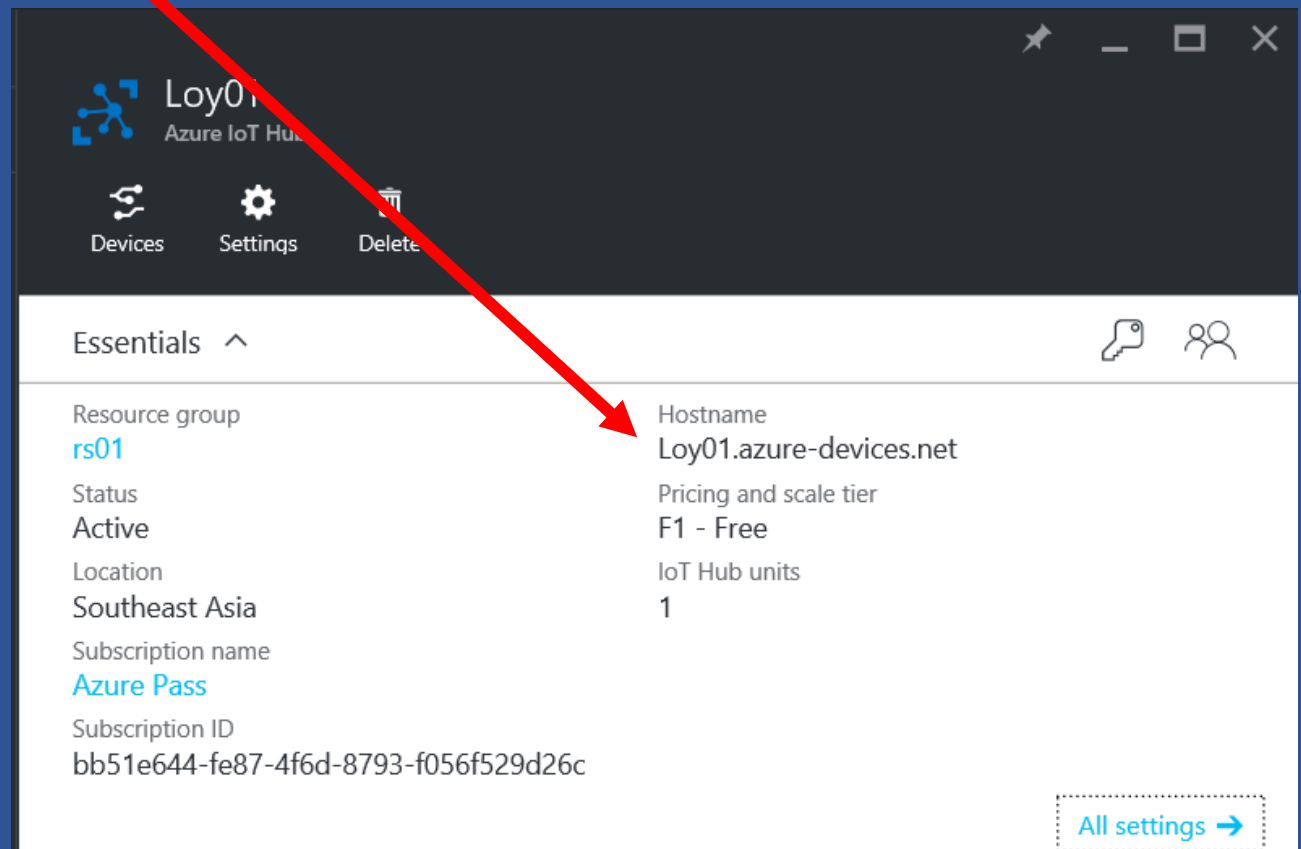
4. Add the following fields to the Form1 class.

- Change iotHubUri
- Change deviceKey

```
public partial class Form1 : Form
{
    DeviceClient deviceClient;
    string deviceId = "myFirstDevice";
    string iotHubUri = "loyHub2.azure-devices.net";
    string deviceKey = "T8m7nbGcHr1JL56AMKT/va3ybdh52BiJ/6pCpnXHMcc=";
    1 reference
    public Form1()
    {
        InitializeComponent();
    }
}
```

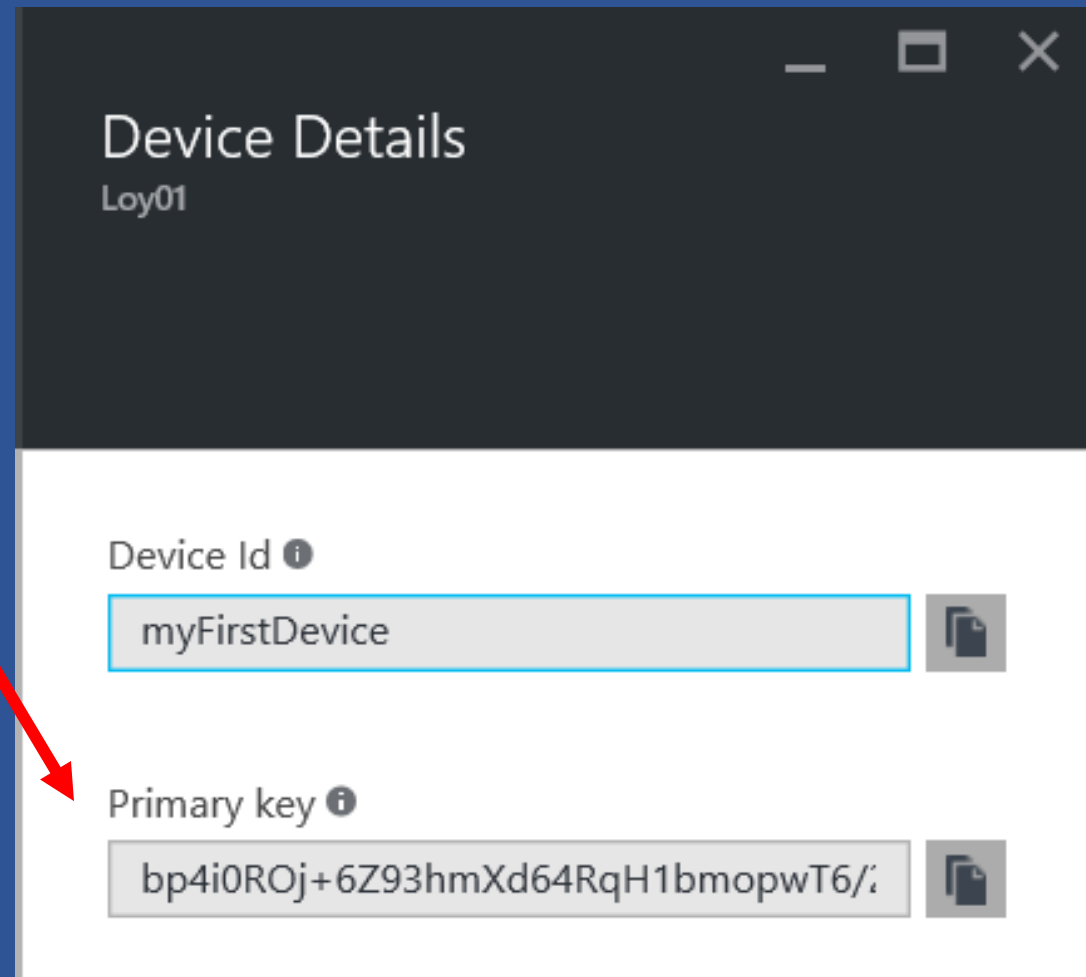
5. Your IoT Hub Home page

IoT Hub Uri



6. Azure IoT Hub/Devices/myFirstDevice/Device Details

device key

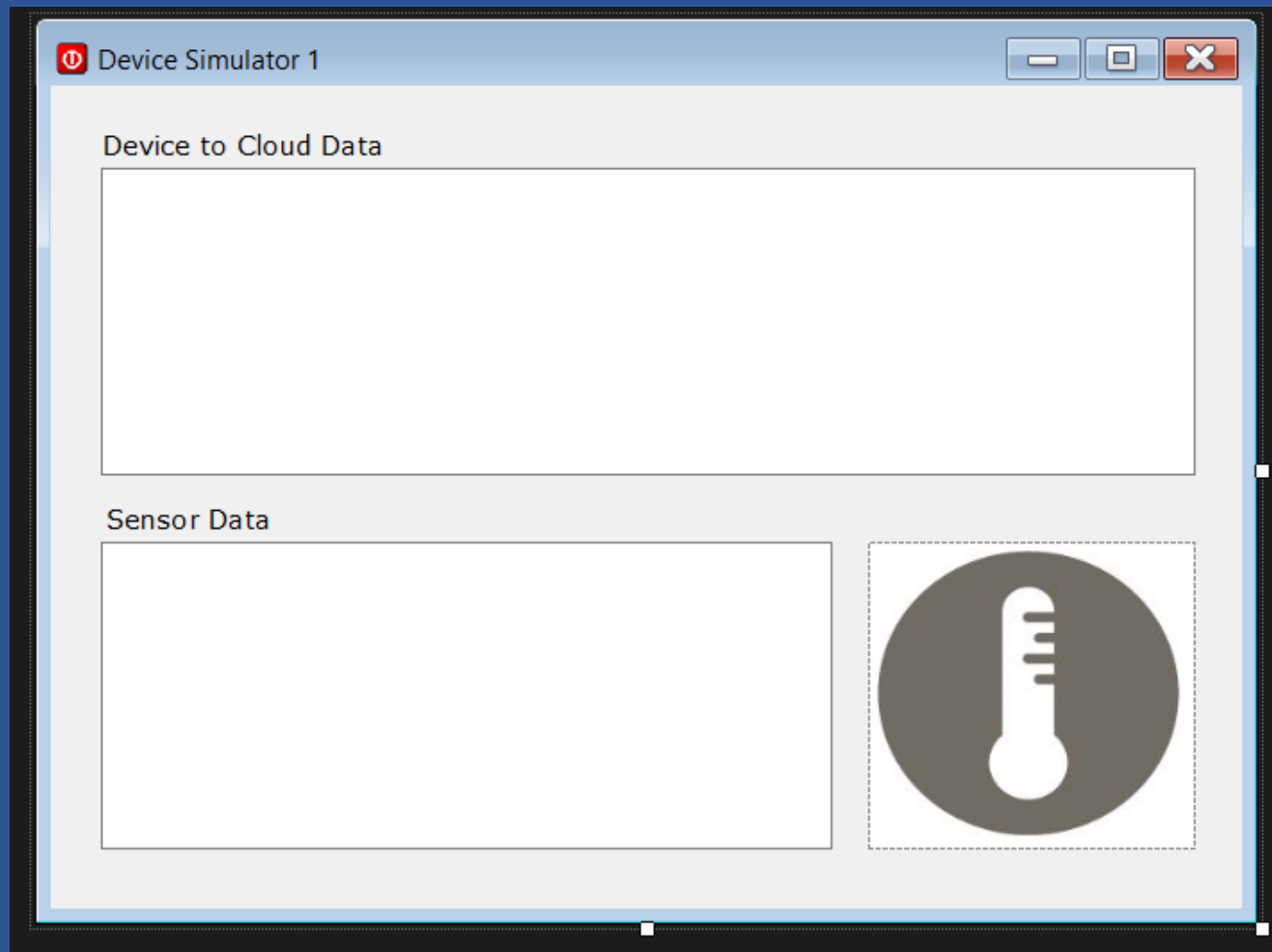


Device Details
Loy01

Device Id ⓘ
myFirstDevice

Primary key ⓘ
bp4i0ROj+6Z93hmXd64RqH1bmopwT6/;

7. Add 2 TextBox “textBoxD2C” and “textBoxSensorData”



8. Add below code to Form1 constructor method

```
public Form1()
{
    InitializeComponent();
    deviceClient =
        DeviceClient.Create(iotHubUri,
            new DeviceAuthenticationWithRegistrySymmetricKey
                (deviceId, deviceKey));
    Timer myTimer = new Timer();
    myTimer.Enabled = true;
    myTimer.Interval = 1000;
    myTimer.Tick += MyTimer_Tick;
}
```

9. Add below code to Form1 Class

```
private void MyTimer_Tick(object sender, EventArgs e)
{
    double temper = 10; // m/s
    Random rand = new Random();
    double currentTemper = temper + rand.NextDouble() * 4 - 3;
    string temperStr = currentTemper.ToString();
    textBoxSensorData.Invoke(new Action(() =>
        { textBoxSensorData.AppendText(temperStr + "\r\n"); }));
}
```

10. Test program. Observe the output.

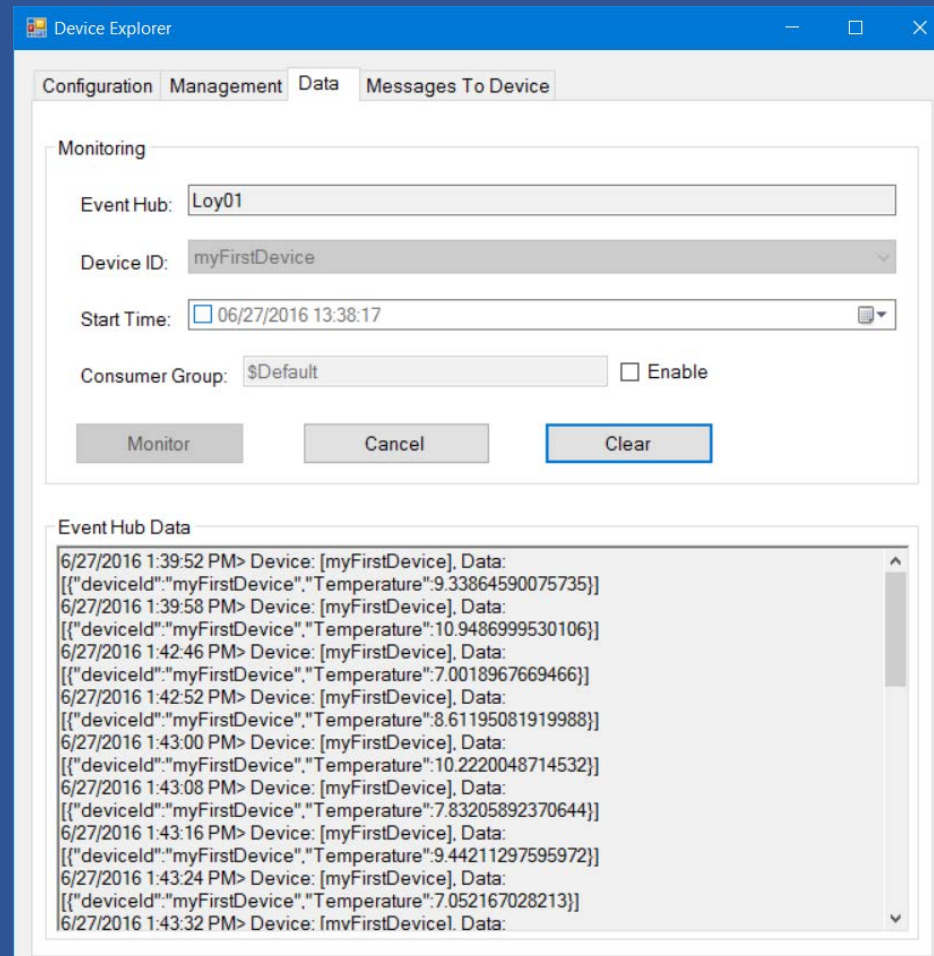
11. Add below code to Form1 Class

```
private async void SendDeviceToCloudMessagesAsync(string data)
{
    var telemetryDataPoint = new
    {
        deviceId = deviceId,
        Temperature = Convert.ToDouble(data)
    };
    var messageString = JsonConvert.SerializeObject(telemetryDataPoint);
    var message = new Microsoft.Azure.Devices.Client.Message
        (Encoding.ASCII.GetBytes(messageString));
    await deviceClient.SendEventAsync(message);
    textBoxD2C.Invoke(new Action(() =>
        { textBoxD2C.AppendText(DateTime.Now + messageString + "\r\n"); }));
}
```

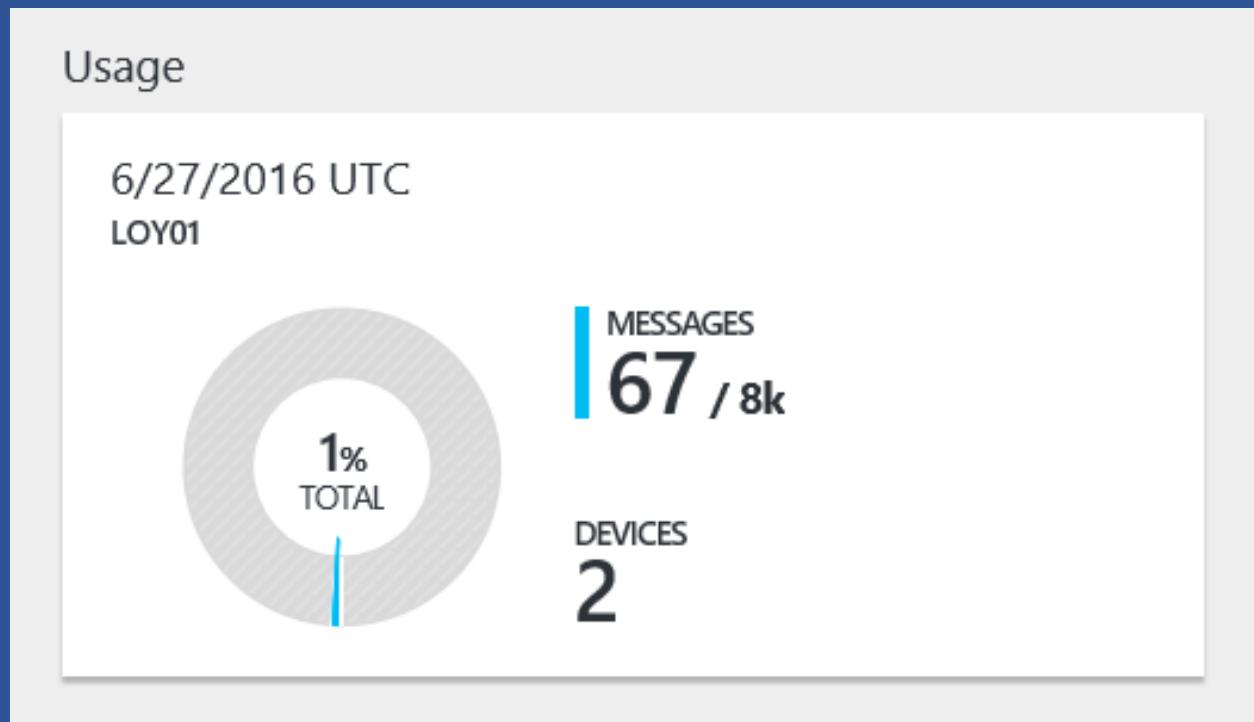

12. Modify MyTimer_Tick method

```
private void MyTimer_Tick(object sender, EventArgs e)
{
    double temper = 10; // m/s
    Random rand = new Random();
    double currentTemper = temper + rand.NextDouble() * 4 - 3;
    string temperStr = currentTemper.ToString();
    textBoxSensorData.Invoke(new Action(() =>
        { textBoxSensorData.AppendText(temperStr + "\r\n"); }));
    SendDeviceToCloudMessagesAsync(temperStr);
}
```

13. Use Device Explorer to inspect Device to Cloud Message



14. Open IoT Hub Home to inspect the Usage



More on Create Simulated Device for Device to Cloud

- Get started with Azure IoT Hub for .NET

<https://azure.microsoft.com/en-us/documentation/articles/iot-hub-csharp-csharp-getstarted/#create-a-simulated-device-app>

- ADD A CUSTOM DEVICE TO THE AZURE IOT SUITE REMOTE MONITORING SOLUTION

<http://rickrainey.com/2016/06/22/add-custom-device-azure-iot-suite-remote-monitoring-solution/>

Receive message from Cloud


Device Simulator 1

Device to Cloud Message

```
6/27/2016 5:23:45 PM{"deviceId":"myFirstDevice","Temperature":9.76216730976578}
6/27/2016 5:23:51 PM{"deviceId":"myFirstDevice","Temperature":7.37222136201906}
6/27/2016 5:23:59 PM{"deviceId":"myFirstDevice","Temperature":8.98227541427234}
6/27/2016 5:24:07 PM{"deviceId":"myFirstDevice","Temperature":10.5923294665256}
6/27/2016 5:24:15 PM{"deviceId":"myFirstDevice","Temperature":8.2023835187789}
6/27/2016 5:24:23 PM{"deviceId":"myFirstDevice","Temperature":9.81243757103218}
6/27/2016 5:24:31 PM{"deviceId":"myFirstDevice","Temperature":7.42249162328546}
6/27/2016 5:24:39 PM{"deviceId":"myFirstDevice","Temperature":9.03254567553874}
6/27/2016 5:24:47 PM{"deviceId":"myFirstDevice","Temperature":10.642599727792}
6/27/2016 5:24:55 PM{"deviceId":"myFirstDevice","Temperature":8.25265378004529}
```

Sensor Data

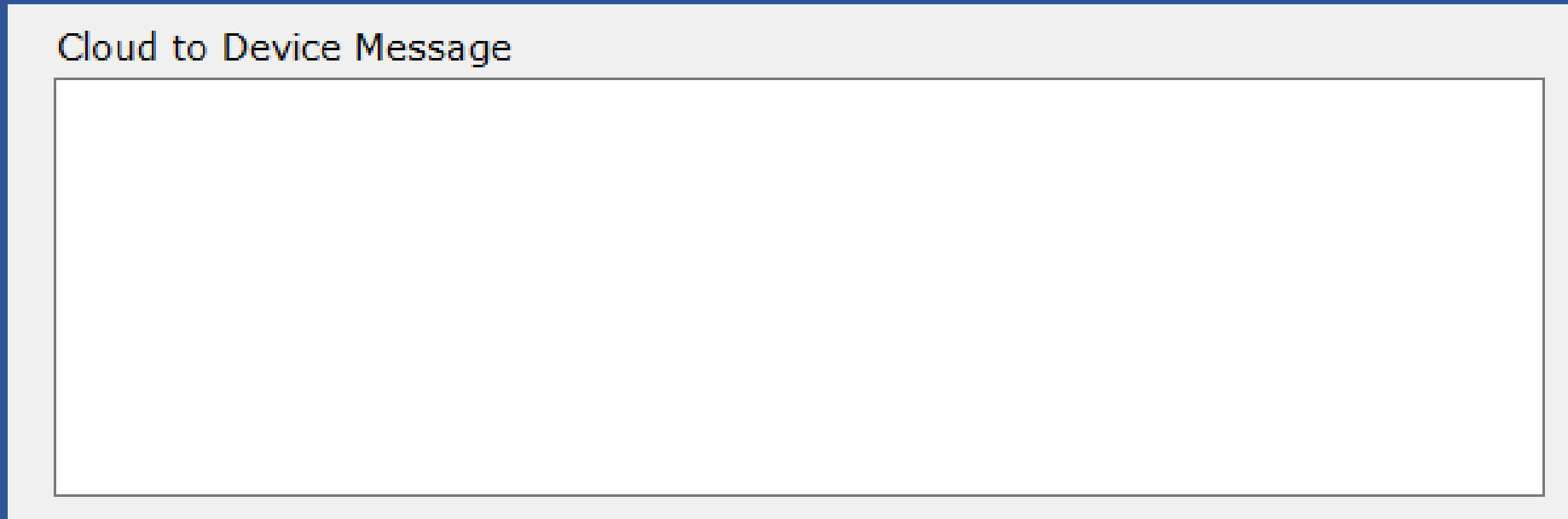
9.76216730976578
7.37222136201906
8.98227541427234
10.5923294665256
8.2023835187789
9.81243757103218
7.42249162328546
9.03254567553874
10.642599727792
8.25265378004529



Cloud to Device Message

```
6/27/2016 5:24:45 PM: Message from Cloud 1
6/27/2016 5:24:53 PM: Message from Cloud 2
```

1. Add a TextBox to Form1. Name = "TextBoxC2D"

A screenshot of a Windows Form. The form has a title bar that says "Cloud to Device Message". Below the title bar is a large, empty rectangular text box with a thin black border. The form itself has a light gray background.

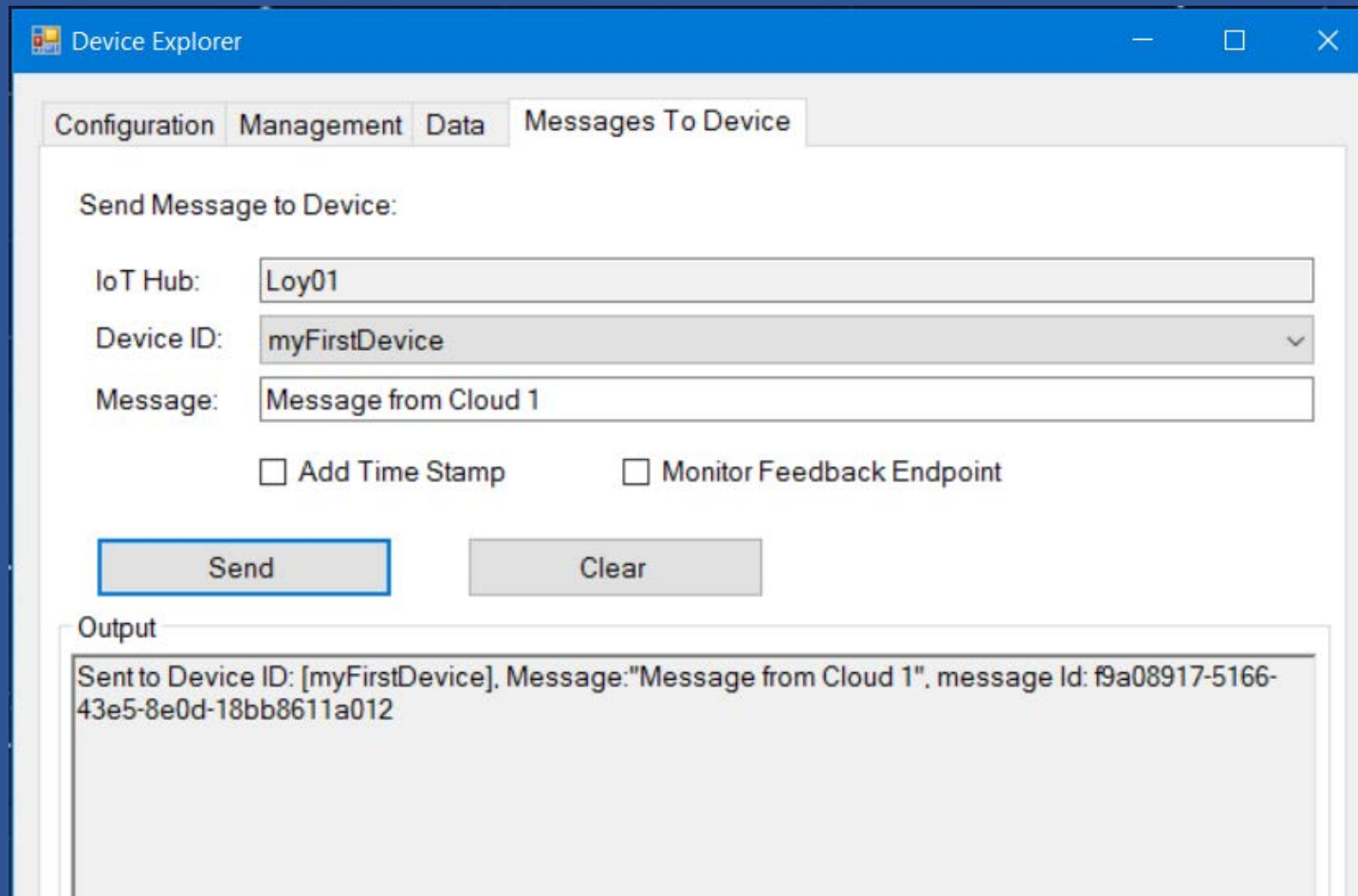
2. Add below method to Form1

```
private async void ReceiveC2dAsync()
{
    string msg = string.Empty;
    while (true)
    {
        Microsoft.Azure.Devices.Client.Message
            receivedMessage = await deviceClient.ReceiveAsync();
        if (receivedMessage == null) continue;
        msg = Encoding.ASCII.GetString(receivedMessage.GetBytes());
        textBoxC2D.Invoke(new Action(() =>
            { textBoxC2D.AppendText(DateTime.Now + ": " + msg + "\r\n"); }));
        await deviceClient.CompleteAsync(receivedMessage);
    }
}
```

3. Modify Form1 Class Constructor

```
public Form1()
{
    InitializeComponent();
    deviceClient =
        DeviceClient.Create(iotHubUri,
            new DeviceAuthenticationWithRegistrySymmetricKey
                (deviceId, deviceKey));
    Timer myTimer = new Timer();
    myTimer.Enabled = true;
    myTimer.Interval = 8000;
    myTimer.Tick += MyTimer_Tick;
    ReceiveC2dAsync();
}
```


4. Test program: Use Device Explorer to send message



More on Cloud to Device Message

- Tutorial: How to send cloud-to-device messages with IoT Hub

<https://azure.microsoft.com/en-us/documentation/articles/iot-hub-csharp-csharp-c2d/>

- CONNECTING TO THE AZURE IOT HUB USING AN AMQP STACK

<https://paolopatierno.wordpress.com/2015/10/24/connecting-to-the-azure-iot-hub-using-an-the-amqp-stack/>