Azure IoT Hub Device Management Walkthrough

Contents

[Overview 2](#_Toc442458710)

[Setup 2](#_Toc442458711)

[Microsoft Azure IoT Hub 2](#_Toc442458712)

[Service Samples 2](#_Toc442458713)

[Create a Device Identity 3](#_Toc442458714)

[Device Samples 3](#_Toc442458715)

[E2E Sample: Device Property Sync 4](#_Toc442458716)

[E2E Sample: Simulated Firmware Update 5](#_Toc442458717)

[E2E Sample: Service Property Tags 6](#_Toc442458718)

[Learn More 7](#_Toc442458719)

# Overview

This document is a walkthrough of the Azure IoT Device Management capabilities in IoT Hub, which shows how to use both the Service Client and Device Client.

It is intended to take about 1 hour to complete.

Before starting, make sure you have the following

1. **Visual Studio 2015.** You can use the free Community Edition if you meet the licensing requirements. **Be sure to include Visual C++.**
2. **CMake (v2.8 or later).** Install CMake from <https://cmake.org/download/>. For a Windows PC, please choose the Windows Installer (.msi) option. Make sure the check the box to add CMake to your PATH variable.
3. **(Provided) Connection string** for IoT Hub. You should get this from your Microsoft representative.
4. **(Provided) DeviceClient.zip** zip file containing the source code for the Azure IoT Hub Device Management Library for the device. This .zip file also contains sample code for the device.
5. **(Provided) ServiceClient.zip** file containing the compiled unsigned assembly for the Azure IoT Hub Service SDK and a sample solution for the service client.
6. **(Provided) Documentation.zip** file containing all documentation.

# Setup

## Microsoft Azure IoT Hub

For the private preview, Microsoft has setup an Azure IoT Hub which includes device management capabilities. Please contact your Microsoft representative to get the connection string.

## Service Samples

Unzip the ServiceClient.zip folder. Inside the .zip folder, you will find a DeviceManagement folder, which contains

* **DeviceManagement.sln - T**he service client samples
* **Iothubcreds.txt -** Your IoT Hub credentials
* **Microsoft.Azure.Devices.1.0.0-dmpreview-005.nupkg -** NuGet package with compiled service client

Open **DeviceManagement.sln** using Visual Studio 2015. Here, you see 4 projects

* **CreateDeviceIdentity:** used to create and register a device with the service
* **ViewPropertySync:** demonstrates state synchronization between the device and service
* **ExecuteFirmwareUpdate:** used to execute a firmware update on your device
* **ServicePropertyTags:** demonstrates how to add tags as service properties and query based on the tags

Now, you need to configure the solution to use your **IoT Hub connection string** that was provided by your Microsoft representative. In the solution, under Solution Items, open **iothubcreds.txt** and copy your connection string into the empty file.

Note that if you create your own project, the Target Framework must be set to .NET 4.5.1.

For an API reference for the Service SDK see **ServiceAPIDoc.docx**.

## Create a Device Identity

Before connecting your device to the service, you must create a device identity through registration. This sample walks you through how to register a device and obtain a device specific connection string.

Set the **CreateDeviceIdentity** project as the Startup Project. You can do this by right clicking on the project and selecting the menu item **Set As Startup Project**. Then press **F5** to compile and run the project.

The displayed **connection string** and **device id** are stored in the files **devicecreds.txt** and **deviceid.txt** respectively. The device id will be used in a future service client sample. The device connection string will need to be copied into the device client sample (explained in detail later). You don't need to copy anything from the console window, you will copy from devicecreds.txt later in the tutorial.

# Device Samples

Now that you have a device connection string (in devicecreds.txt) you can use that string to connect a device to your IoT Hub.

This tutorial explains how to setup a simulated device on your **Windows** machine. To run it on a **Linux** machine, please see **"How to run the simple\_sample.docx".**

1. Open the **Developer Command Prompt for VS2015** by pressing Windows Key and searching for "developer" and selecting "Developer Command Prompt for VS2015". A command prompt will open. Use this command prompt for the next steps.
2. Create a folder called c:\dm. Extract "DeviceClient.zip" into c:\dm\.

*Note you can extract the files into a different folder location of your choice, but make sure you use the appropriate paths in the below instructions. The following steps assume you extracted the files into the c:\dm folder.*

1. After you extract all the files, make a "projects" folder in the "c:\dm\azure-iot-sdks\c\iotdm\_client" folder and navigate to that folder using the following commands:

mkdir projects

cd projects

1. Run the cmake command in the project folder as follows:

cmake ..

1. Open the "project.sln" file using in **another instance Visual Studio 2015.**
2. Go to the Solution Explorer in VS and in the tree view navigate to iotdm\_simple\_sample -> Source Files -> **iotdm\_simple\_sample.c**
3. Locate the following code in the file:

static const char\* connectionString = "[device connection string]";

1. **Insert your device connection string** between the double quotes. This is the connection string that is in **devicecreds.txt** located in DeviceManagement VS solution directory. The devicecreds.txt file was created when we ran the CreateDeviceIdentity project. Save the changes.
2. Set the **iotdm\_simple\_sample** project as the Startup Project. You can do this by right clicking on the project and selecting the menu item **Set As Startup Project**.
3. Build and run the project by pressing **F5**. When the sample runs, the device is ready to receive and process requests from the service.
4. In the output window, you will see a stream of data. This output shows the construction of outgoing CoAP packets, reception of incoming CoAP packets, and printf statements in the application specific callbacks. This allows you to see incoming and outgoing traffic along with how the sample application is handling the decoded packets.

There is so much traffic because when the device connects to the IoT Hub, the service will automatically start to observe resources on the device. The device libraries will then invoke the device callbacks to retrieve the latest values from the device.

You can use this sample as a starting point for your own applications.

For a list of the device APIs refer to the document “**DeviceAPIDoc.docx”**.

# E2E Sample: Device Property Sync

Device properties are automatically observed by the service and a copy of the properties are stored in IoT Hub. This sample demonstrates how to view the changing battery level of the simulated device.

Since we started the device client Iotdm\_simple\_sample in the previous section, the device client should be running in the background. If you stopped that project, please restart it now.

To run the service sample:

1. Go back to the instance of Visual Studio with **DeviceManagement.sln.**
2. Set the **ViewPropertySync** project as the Startup Project. You can do this by right clicking on the project and selecting the menu item **Set As Startup Project**.
3. Build and run the project by pressing **F5**.
4. In the output window, you will see the battery level of the device.

**NOTE:** The first time you run the sample, you may see a message ‘The battery level is not available’ until the service receives data for the battery level device system property. If you don’t see battery level values in more than 30 seconds, ensure the iotdm\_simple\_sample is running.

On subsequent runs with the same device id, you do not see this error. This is because the service maintains a cache of the last known values for the device.

Note that the service sample uses **deviceid.txt** to fetch the device created by the CreateDeviceIdentity project.

# E2E Sample: Simulated Firmware Update

This sample uses the existing iotdm\_simple\_sample simulated device and the ‘ExecuteFirmwareUpdate’ project to initiate the FirmwareUpdate device job. Since we're running a simulated device, the device will mock a firmware update by logging device job steps to the console.

Since we started the device client Iotdm\_simple\_sample in the previous section, the device client should be running in the background. If you stopped that project, please restart it now.

To run the service sample:

1. Set the **ExecuteFirmwareUpdate** project as the Startup Project. You can do this by right clicking on the project and selecting the menu item **Set As Startup Project**.
2. Build and run the project by pressing **F5.**

In the output from **ExecuteFirmwareUpdate**, you will see:

|  |
| --- |
| Reading IoT Hub Connection String from file  Reading device id from file  Creating job client...  Starting update job...  Job status is Unknown  Job status is Running  Job status is Completed  Done! Hit <Enter> to exit. |

If you copy the output from the **iotdm\_simple\_sample** to notepad and search for "\*\*", you will see the following interesting events indicating the flow of a successful firmware update:

|  |
| --- |
| Info: \*\* URI received. Starting simulated download  Info: \*\* Downloading...  Info: \*\* Downloading...  Info: \*\* Downloading...  Info: \*\* Downloading...  Info: \*\* Download complete  Info: \*\* Simulated update started  Info: \*\* Updating..  Info: \*\* Updating..  Info: \*\* Updating..  Info: \*\* Updating..  Info: \*\* Update complete |

# E2E Sample: Service Property Tags

This sample shows you how to add service property tags to a device and query based on those tags. Once completing execution, the sample deletes the devices that were created in the IoT Hub.

GUIDs are used to ensure tag uniqueness and prevent conflict with your co-workers currently using the same IoT Hub.

To run the service sample:

1. Set the **ServicePropertyTags** project as the Startup Project. You can do this by right clicking on the project and selecting the menu item **Set As Startup Project**.
2. Build and run the project by pressing **F5**.

In this sample, we create 5 devices with different sets of tags. When you run this sample, you will see:

|  |
| --- |
| Reading IoT Hub Connection String from file  Creating devices...  ==== Wait 30s ====  ==== Get tag1 ====  Expected: 4; Found: 4  ==== Get tag1 with maxresults=2 ====  Expected: 2; Found: 2  ==== Get tag1+tag2+tag3 ====  Expected: 1; Found: 1  ==== Get tag2+tag3 ====  Expected: 1; Found: 1  ==== Get tag3 ====  Expected: 3; Found: 3  ==== Get tag3 ====  Expected: 0; Found: 0  ==== Get tag1 after delete ====  Expected: 0; Found: 0  Done! Hit <Enter> to exit. |

If you open Program.cs in the ServicePropertyTags project, you’ll see the use of the **QueryDevicesAsync** method. QueryDeviceAsync returns the list of device where the device query is the union (&&) of the tags parameter.

You can also see that the **maxCount** parameter limits the number of devices that are returned.

# Learn More

You will find a few documents in the Documentation.zip file.

* **Azure IoT Hub device management library for C.docx:** Introduction to the Azure IoT Hub Device Management APIs for the device client
* **AzureIoTHubDeviceManagementPrivatePreview.pptx**: Introduction powerpoint presented by Microsoft representatives
* **DeviceAPIDoc.docx**: API documentation for the device client
* **ServiceAPIDoc.docx**: API documentation for the service client
* **How to run the Intel Edison Sample.docx:** Instructions for using the Intel Edison board instead of the simulated device
* **How to run the simple\_sample.docx:** Instructions for running the iotdm\_simple\_sample on Windows and Linux.