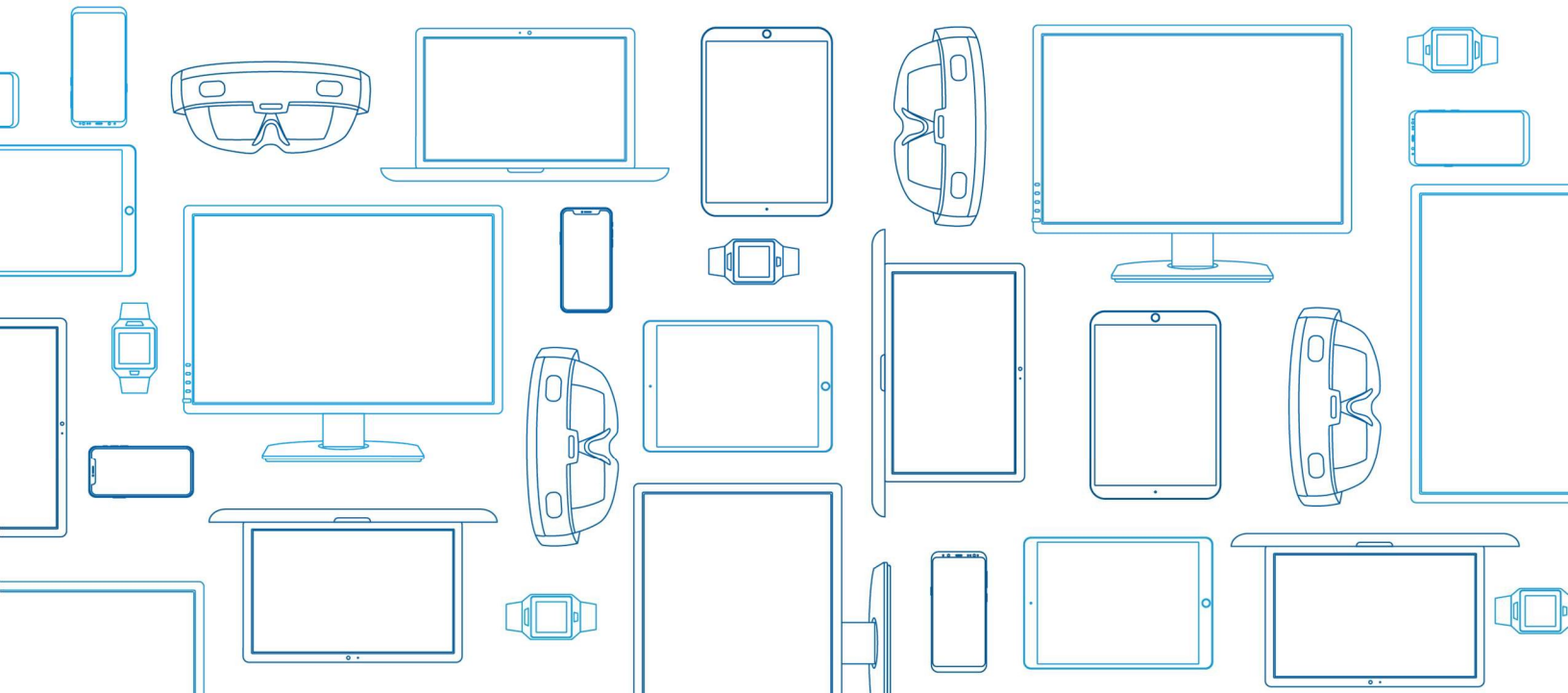
Writing to Devices from Microsoft Azure Using IoTWorX



Contents

1 Installing GENESIS64 and IoT Hub 4
2 Getting the device connection string from the Azure portal 4
3 Configuring data points in GENESIS64 for write access 5
4 Configuring the custom encoder in GENESIS64 5
5 Creating an application to write to a device 6
6 Verifying the write operation in GENESIS64 7
7 Conclusion 9

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

Copyright and Confidentiality

By accessing and using the installation instructions (the “instructions”) you acknowledge and agree, on your behalf and on behalf of the person, entity or other organization on whose behalf you are accessing the instructions, that neither Microsoft or ICONICS, nor any of its service providers, including, without limitation, any system integrator or independent software vendor: (1) makes any representations or warranties of any kind, either express, implied, statuary or otherwise with respect to the instructions, including the accuracy, completeness or usefulness thereof; and (2) shall be liable for damages of any kind, under any legal theory, arising out of or in connection with your election to follow or use, or inability to follow or use, the instructions, including any direct, indirect, incidental, special, punitive or consequential damages, or for loss of use, loss of profits, loss of data, loss of business, or loss of privacy or security, even if foreseeable, arising out of or in connection with your election to follow or use, or inability to follow or use, the instructions. You further acknowledge and agree that your use of the instructions, whether directly or indirectly, is at your own risk and that you expressly assume all risk in connection with your use of the instructions. If you do not agree to the foregoing, you may not access or use the instructions.

Copyright © 2019, Microsoft Corporation and ICONICS, Inc. All rights reserved.

Authors

- Zhi Wei Li, Director of Applications Solutions, ICONICS
- Bhargav Reddy, Solutions Sales Engineer, ICONICS
- Spyros Sakellariadis, Applied Innovation/Enterprise Commercial Business, Microsoft Corporation
- Maksym Mushkin, Applied Innovation/Enterprise Commercial Business, Microsoft Corporation

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

1 Installing GENESIS64 and IoT Hub

Follow the instructions in Using IoTWorX as a gateway for setting up IoTWorX to send data to an Azure IoT Hub.

2 Getting the device connection string from the Azure portal

To write to a device from a cloud application, you send a command to the device as registered in your IoT Hub. To do this, you need the correct endpoint in the IoT Hub, which you can get as follows:

1. Log in to the Azure portal.
2. Select the IoT Hub to which you are sending data to open the IoT Hub configuration blade.
3. Click on Shared Access Policies in the middle pane
4. Click on the iothubowner policy in the right pane
5. Copy the Connection string – primary key to the clipboard. You will use this in the code in Section 5 below, pasting into the string `static string connectionString = "XXXXXXXX"`
6. Now click on IoT Devices from the list of options in the middle pane.
7. Note the Device ID for the Azure IoT Hub device with which IoTWorX is communicating. In the example described in this paper, the Device ID is “ESBLab2019”, as shown in Figure 1 below. You will use this in the code in Section 5 below, pasting it into the string `serviceClient.SendAsync("YYYYYYYY", commandMessage.`

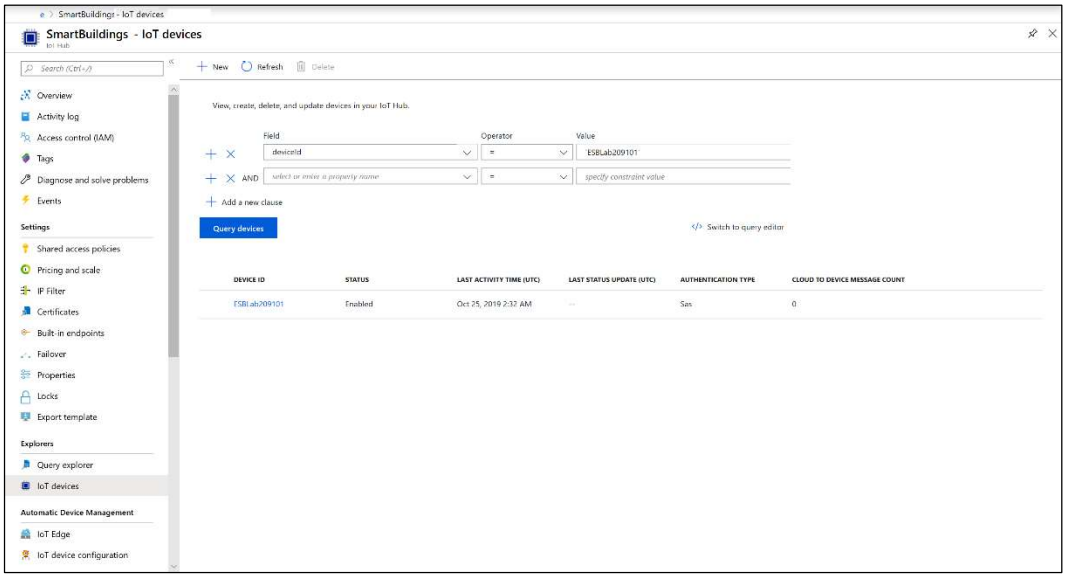


Figure 1 IoT Hub > IoT Devices

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

3 Configuring data points in GENESIS64 for write access

When you created a Publish List in IoTWorX you identified objects from which IoTWorX would read data on a schedule. To write to these objects, you need to first select objects that are writeable (e.g. set points) and mark them as writeable. To do this, follow these steps:

1. Open the ICONICS Workbench.
2. In the Project Explorer, expand your project, then Internet of Things > Publish Lists.
3. Double click on the Publish List containing the point or points to which you want to write to bring up the list in the Project Details pane, as shown in Figure 2 below.

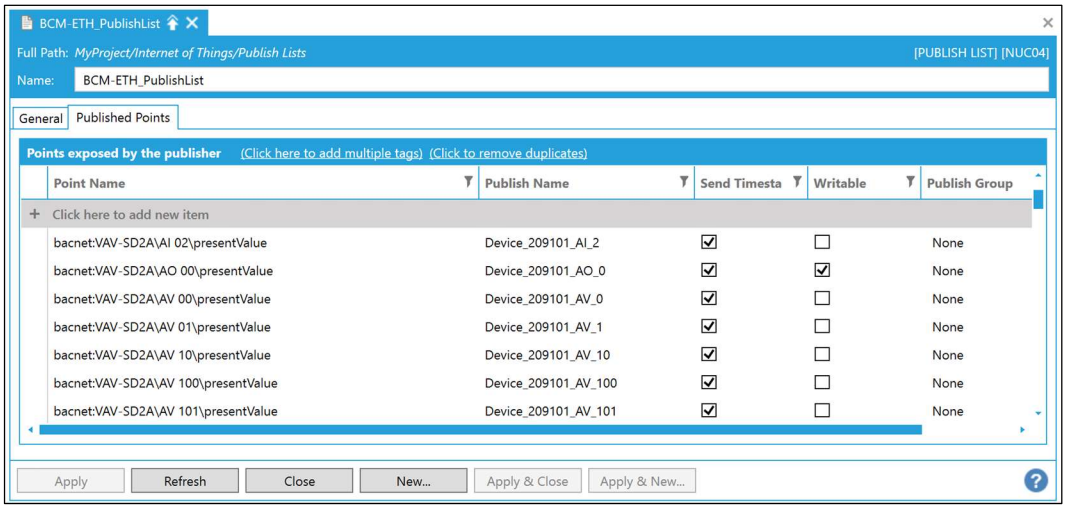


Figure 2 Published Points

4. Check the 'Writable' checkbox for the point or points to which you want to write, for example Point Name bacnet:VAV-SD2A\AO 00, published as "Device_209101_AO_0".
5. Click 'Apply' and then 'Close' to save your edits.

4 Configuring the custom encoder in GENESIS64

Next, you need to specify the format for the write operations by entering some additional information in the custom encoder you created for the connection. This format includes the ID of the targeted device and the value to write.

1. In the Project Explorer, expand your project, then Internet of Things > Custom Encoders.
2. Double click on the encoder you used for sending the data to your Azure IoT Hub to bring up the General Settings window in the Project Details pane. In our example the encoder is "BCM-ETH_encoder".

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

3. Scroll down to the last section.
4. Check the checkbox 'Use different format for writing'.
5. In the Write Format box confirm or change the format to the following.

```
{
 "id": "%PUBLISHNAME%",
 "v": "%VALUE%",
}
```

6. Click 'Apply' and then 'Close' to save your edits.

At this point, you have identified certain objects as writable, and specified how IoTWorX will know which device(s) should be sent write commands. When IoTWorX next connects to your IoT Hub, it will receive any instructions that are addressed to it and forward the write commands to the devices in question.

5 Creating an application to write to a device

The objective is to send a message to your IoT Hub directed at the Device ID representing your IoTWorX gateway. You can do this with third-party applications (for example, IoTWorX itself), or with a custom back-end application. The trick is to send data in a specific format that IoTWorX expects, and further formatted to be compatible with the variables as defined in step 5 of Section 4 above. Here is a C# code snippet to do this (you will need to create your own C# project, of course):

```
using System;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure.Devices;
using Newtonsoft.Json;

namespace CloudToDeviceTest
{
    class Program
    {
        static ServiceClient serviceClient;
        // connection string copied in from Azure portal Section 2 above
        static string connectionString = "XXXXXXXX";
        static void Main(string[] args)
        {
            Console.WriteLine("Send Cloud-to-Device message\n");
            serviceClient = ServiceClient.CreateFromConnectionString(connectionString);
            int i = 1;
            Console.WriteLine("Press any key to send a C2D message. Press ESC to quit.");
            ConsoleKeyInfo input;
            do
            {
                input = Console.ReadKey();
                SendCloudToDeviceMessageAsync(i).Wait();
                i = i + 1;
            } while (input.Key != ConsoleKey.Escape);
            Console.ReadLine();
        }

        private async static Task SendCloudToDeviceMessageAsync(int messageId)
        {
```


© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

```
            dynamic msg = new
            {
                // variable names same as IoTWorX encoder's write format, ID same as IoTWorX Publish Name
                // (not Point Name)
                id = "Device_209101_AO_0",
                v = "3.5"
            };
            string message = JsonConvert.SerializeObject(msg);
            var commandMessage = new Message(Encoding.UTF8.GetBytes(message));
            commandMessage.MessageId = messageId.ToString();
            commandMessage.Properties.Add("cmdType", "Write"); //Type of command performed on the tag
            commandMessage.Properties.Add("iothub-ack", "none"); //No feedback required
            commandMessage.Properties.Add("encoder", " BCM-ETH_encoder"); //Name of encoder in IoTWorX
            await serviceClient.SendAsync("YYYYYYYY", commandMessage); //Device ID of IoTWorX in IoT Hub
            Console.WriteLine();
            Console.WriteLine($"Sent Message: {message}");
        }
    }
}
```

The simple C# code listed above sends a series of strings to the IoTWorX device registered in your IoT Hub, and the next time IoTWorX connects to the IoT Hub, it will pick up this series of strings and understand that it needs to write a value (3.5 in the above example) to one of the devices in the publish list – specifically the fan on the terminal unit (writeable Device_209101_AO_0).

6 Verifying the write operation in GENESIS64

1. In the Project Explorer, expand your project, then Data Connectivity > BACnet > Devices, then the device to which you are writing, for example VAV-SD2A.
2. Double-click on the Point Name (not the Publish Name) for the object to which you wrote data to bring up the object in the Project Details pane.
3. Click on the Runtime tab.
4. In the Basic / Object section, check the Present Value. It should match the value you sent in the SendCloudToDeviceMessageAsync task above. For example, using the code above we wrote 3.5 to the fan:

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

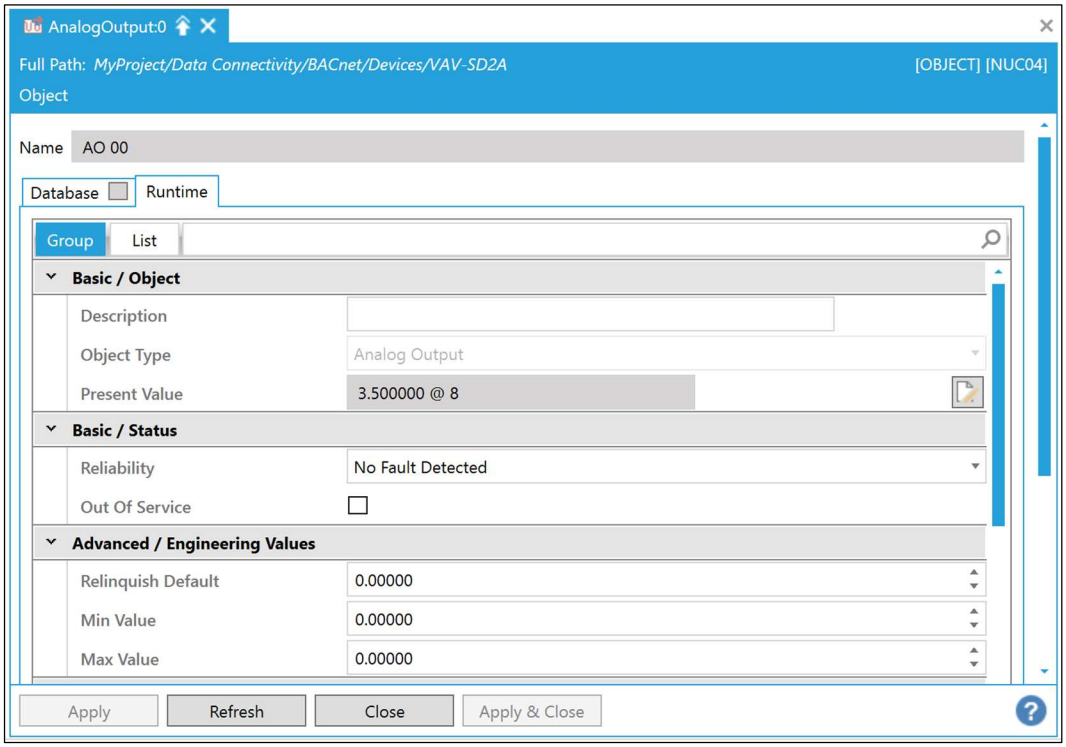


Figure 3 Value 3.5 written to device

If we change the value in the code from 3.5 to 3.0 and re-run the code, we can see the new value of 3.0 written to the fan:

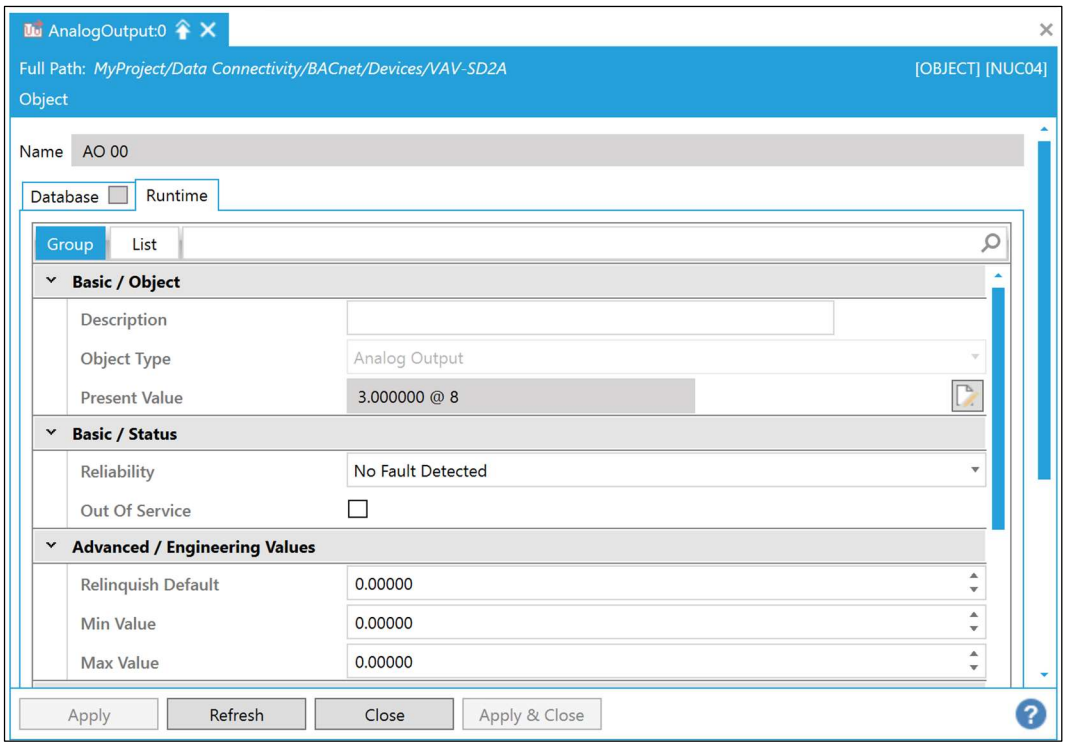


Figure 4 Value 3.0 written to the device

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.

7 Conclusion

In Using IoTWorX as a gateway, we described how to set up IoTWorX to read data from sensors and send to Azure. Here we described a simple way to achieve the reverse; how to set up IoTWorX to write data back to writable sensors or actuators. The steps described are relatively simple – you create an application that connects to Azure IoT Hub and sends the new values to the IoTWorX gateway device. The C# code snippet shown earlier in this document is such an application. The next time IoTWorX connects to the IoT Hub, IoTWorX reads the data from the IoT Hub and writes it back to the targeted device. In this way, you can use IoTWorX to control devices throughout your infrastructure.

© 2019, Microsoft Corporation and ICONICS, Inc.
All rights reserved.