Connectware & Azure IoT Hub Integration

by Ragavendra Lingamaneni

Prerequisites

In this lesson, we will send data from the Connectware MQTT Broker to Azure IoT Hub.

It is required to have a working Connectware instance, and also a working Azure IoT Hub setup. We assume you are already familiar with Connectware and its service concept. If not, we recommend reading the articles Connectware Technical Overview and Service Basics for a quick introduction. Furthermore, this lesson requires basic understanding of MQTT and how to publish data on a MQTT topic. If you want to refresh your MQTT knowledge, we recommend reading the lessons MQTT Basics and How to connect an MQTT client to publish and subscribe data.

Introduction

This article is divided into two parts. In the first part, the general background of Azure IoT services and their differences is described. Feel free to skip this section if you are familiar with Azure IoT Hub, in particular with the difference between IoT Edge Runtime and IoT Device SDKs. In the second part, the recommended integration mechanisms between Connectware and Azure IoT Hub are explained in a hands-on approach.

Azure IoT Hub

Azure IoT Hub is a core component of the Azure IoT offerings. It is a fully managed service that enables reliable and secure bi-directional communication between IoT devices and an application back-end. Azure IoT Hub offers reliable device-to-cloud and cloud-to-device hyper-scale messaging, enables secure communications using per-device security credentials and access control, and includes device libraries for most popular programming languages and platforms.

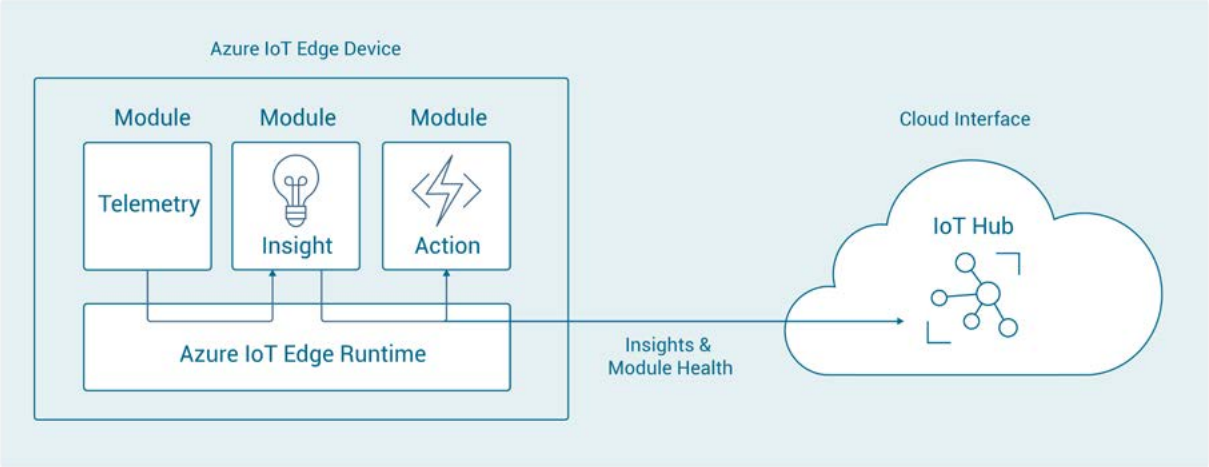
There are two ways you can send data to Azure IoT Hub which are described below.

Azure IoT Hub Edge

Azure IoT Edge lets you offload parts of your IoT workload from your Azure cloud services to your devices. IoT Edge can reduce latency in your solution, reduce the amount of data your devices exchange with the cloud, and enable off-line scenarios. You can manage IoT Edge devices from IoT Central and some solution accelerators.

Azure IoT Edge is made up of three components:

- Edge modules: These are containers which are deployed on the edge devices and run Azure and other third-party services. You can build your own modules to send telemetry to the IoT Hub
- Edge runtime: It runs on each IoT Edge device and manages the modules deployed to each device.
- Web interface: A cloud-based interface is used to remotely monitor and manage IoT Edge devices.



Azure IoT Edge (adapted from Microsoft, 2020)

Azure IoT Hub Device SDKs

IoT Hub Device SDKs enable you to build apps that run on your IoT devices. These apps send telemetry to your IoT Hub, and optionally receive messages, job, method, or twin updates from your IoT Hub. The device SDKs support multiple operating systems, such as Linux, Windows, and real-time operating systems. There are SDKs for multiple programming languages, such as C, Node.js, Java, .NET, and Python.

Connectware & Azure IoT Hub Integration

Before proceeding further, first you need to complete the Azure IoT Hub setup. It is required to obtain the *device connection string*. You can follow the instructions on this page for doing so.

There are multiple possibilities to enable data exchange between Azure IoT Hub and Connectware. These possibilities are described below.

Cybus Azure IoT Hub connector service

Cybus *Azure-IoT-Connector* is a Connectware service provided by Cybus and available as a docker image. It uses the Azure IoT Hub Device Node.js SDK, and MQTT as the transport protocol. This is a simple solution to

send data to the IoT Hub. Most of the complexity is hidden and a simple interface is provided for configuration in the form of "connector configuration". This service is capable of *sending telemetry data and blob data* to the IoT Hub. However, the service does not support *receiving data* from the IoT Hub. As an additional advantage, data buffering is built-in, so that data is stored locally when there is a temporary connection failure to the IoT Hub.

The commissioning file below sends data published on topics '/machine/temperature' and '/machine/vibration' as telemetry data to Azure IoT Hub. Before installing the service, make sure you are publishing data on the Connectware broker on topics '/machine/temperature' and '/machine/vibration'. Further details on setting up the suitable MQTT topics can be found in this article: How to connect an MQTT client to publish and subscribe data.

```
description: >
    Azure IoT Hub connector service

metadata:
  name: Azure IoT Hub connector
 icon: https://www.cybus.io/wp-content/uploads/2019/03/Cybus-logo-Claim-lang.
svg
  provider: cybus
  homepage: https://www.cybus.io
  version: 0.0.1

definitions:
  MQTT_ROOT_TOPIC : !ref Mqtt_Root_Topic

parameters:

  Azure_Iot_Hub_Connection_String:
    type: string

  Mqtt_Root_Topic:
    type: string
    default: machine

resources:
```

```
  #-------------------------------------------------------------------------
--
  # Cybus Azure IoT Hub Connector Service
  #-------------------------------------------------------------------------
--

  azureIoTConnector:
    type: Cybus::Container
    properties:
      image: registry.cybus.io/cybus-services/azure-iot-connector:0.0.5
      environment:
        LOG_LEVEL: 'info'
        CONNECTOR_CONFIG: !sub  |
          {
            "general": {
              "name": "Azure Connector"
            },
            "source": {
              "driver": "azure.iot",
              "connection": {
                "connectionString": "${Azure_Iot_Hub_Connection_String}"
              },
              "defaults": {
                "operation": "write"
              }
            },
            "target": {
              "driver": "mqtt",
              "connection": {
                "protocol": "mqtt",
                "host": ${Cybus::MqttHost},
                "port": ${Cybus::MqttPort},
                "username": ${Cybus::MqttUser},
                "password": ${Cybus::MqttPassword}
              },
              "defaults": {
                "operation": "subscribe",
```

```
            "topicPrefix": "${Mqtt_Root_Topic}"
          }
        },
        "mappings": [
    {
            "source": {
              "name": "temperature",
              "type": "telemetry"
            },
            "target": {
              "topic": "temperature"
            }
          },
          {
            "source": {
              "name": "vibration",
              "type": "telemetry"
            },
            "target": {
              "topic": "vibration"
            }
          }
        ]
      }
```

The resource `azureIoTConnector` describes the Docker container with the connector service. This service is configured by passing the `CONNECTOR_CONFIG` environment variable, which in turn contains a JSON object. This JSON object has three important parts (in addition to the general section with the name):

- Source configuration

  In the source section, the only configurable parameter is the device `connectionString` to connect to a particular device on Azure IoT Hub. In this example, this string is defined as a parameter for the service commissioning file and asked during the installation of the service in the Connectware.

- Target configuration

  In this section, the MQTT broker connection parameters are defined. Theoretically, any arbitrary MQTT broker can be specified, but in this example, the local Connectware broker is used. The connection settings to the Connectware broker are set by using the global parameters of the Connectware for

these values. Optionally, a default MQTT topic prefix can be specified within this section, which is useful when installing multiple services on the same Connectware but which should not interfere with each other.

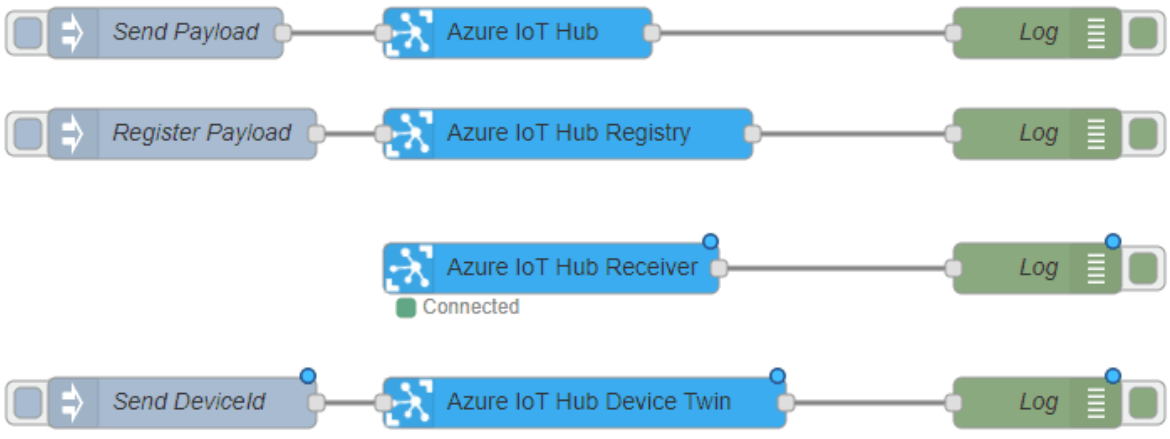
- Mappings

  Every object within the mappings array defines a mapping between source topic which corresponds to IoT Hub device parameter and target MQTT topic (on the Connectware broker). In the above example, we are sending data published on topics `temperature` and `vibration` to IoT Hub as telemetry data with corresponding name.

Cybus workbench service

The *workbench* service is basically a Node-RED instance running securely on the Connectware as a service. This opens up the possibility to install any Node-RED nodes within the service container for quick prototyping as well as for production environment. If your use-case cannot be achieved with the service commissioning file of the previous section, using the workbench might be a suitable choice with higher flexibility and additional tools to prototype your solution using third party Node-RED modules.

node-red-contrib-azure-iot-hub is a Node-RED module that allows you to send messages and register devices with Azure IoT Hub. It contains a total of four Node-RED cloud nodes: Azure IoT Hub, Azure IoT Registry, Azure IoT Hub Receiver and Azure IoT Hub Device Twin. These nodes can be used for configuring more complex data handling.

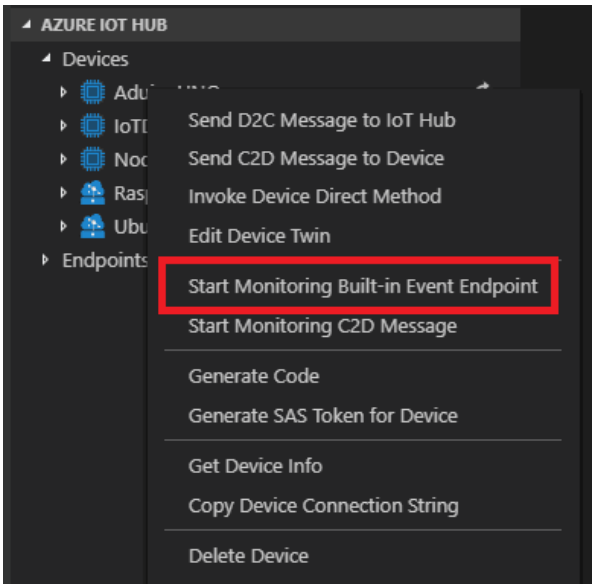


Tools

Now that we are successfully sending the data to the IoT Hub, we can monitor the transmitted data using various tools as described below.

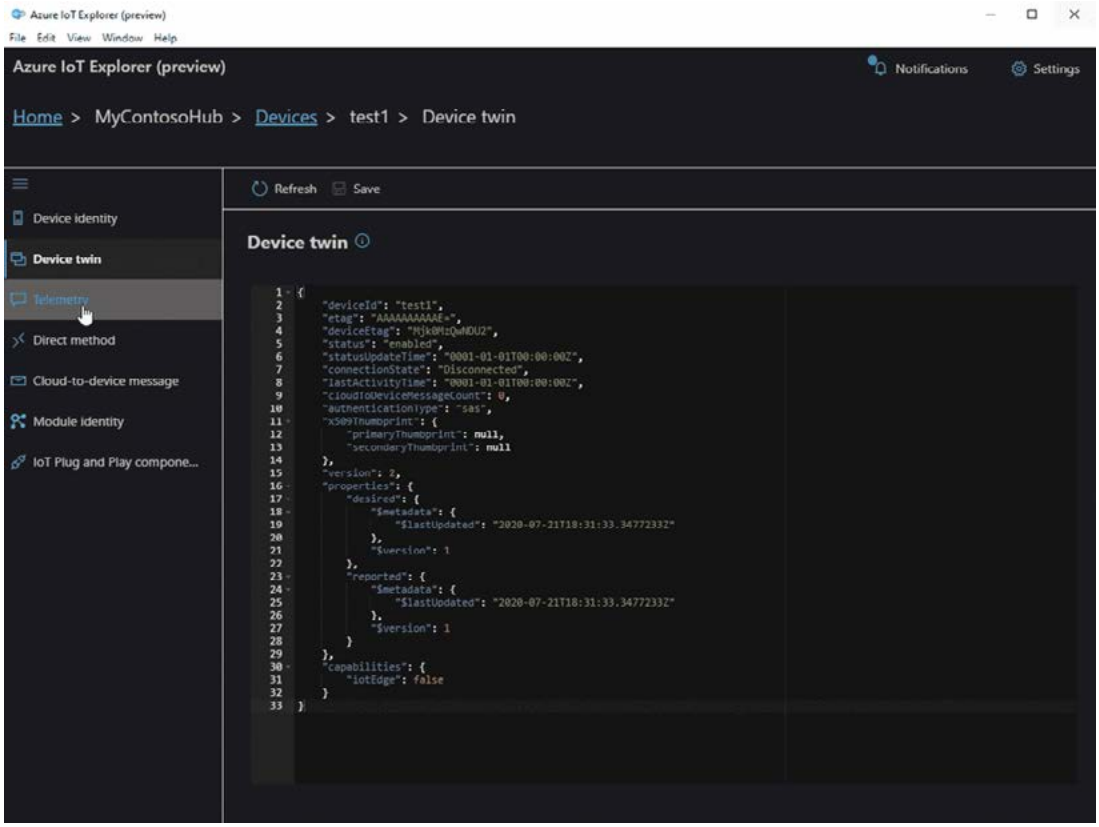
Azure IoT Tools for Visual Studio Code

Azure IoT Tools is a collection of *Visual Studio Code* extensions for working with Azure IoT Hub. Using these extensions, you can interact with an Azure IoT Hub, manage connected devices, and enable distributed tracing for your Azure IoT applications. Further, you can easily subscribe to telemetry messages sent to the IoT Hub for quick testing. Please go to Visual Studio Marketplace to find further documentation regarding installation and usage of the Azure IoT tools.



Azure IoT Explorer

Azure IoT Explorer is an open source cross-platform user interface for interacting with Azure IoT Hub without logging into the Azure portal. This tool can be used to perform tasks like creating, deleting and querying devices within the IoT Hub. Device functionalities such as sending and receiving telemetry and editing device and module twin configuration is also possible with this tool. You can find more information about the explorer on its GitHub page.



Summary

This lesson started with a quick introduction to Azure IoT Hub and its various components and SDKs available for integration with other services. Then an example setup to send data from the Connectware MQTT Broker to Azure IoT Hub was described, using a commissioning file with the azure-iot-connector service. Additionally, the IoT Hub Node-RED modules were described, which can be used with Cybus workbench service for prototyping of complex scenarios. Eventually, some of the monitoring tools were mentioned that can be used to monitor data flow between Azure IoT Hub and Connectware.

Where to go from here

You can take a look at more advanced scenarios involving Azure IoT Hub such as Machine Condition Monitoring in Connectware documentation pages.

References

Microsoft. 2020. *What Is Azure Iot Edge*. [online] Available at: https://docs.microsoft.com/en-us/azure/iot-edge/about-iot-edge [Accessed 27 October 2020].

Cybus is a specialist for secure IIoT Edge software, headquartered in Germany. Cybus Connectware serves smart factories as a universal Edge and DevOps hub. Machine builders and providers of IIoT services use the Cybus Connectware as a software-based gateway. As early as 2017, Cybus published the first secure industrial connector for machine data according to today's DIN SPEC 27070 standard. Industry analyst Gartner named Cybus a worldwide "Cool Vendor". Today, the company counts medium-sized and large companies from numerous industrial sectors such as mechanical engineering, automotive and aviation among its customers.

Cybus GmbH · Osterstraße 124 · 20255 Hamburg · Germany · www.cybus.io · hello@cybus.io · (+49) 40 228 58 68 51