



Connectivity Guide

KEPServerEX and Microsoft Azure IoT Hub

April, 2020
Ref 1.05

Table of Contents

- 1. Overview.....3
- 2. Create IoT Device and Get SAS Token.....3
- 3. Configure KEPServerEX MQTT Client Agent to connect to Azure IoT Hub.....9
- 4. Verify Successful "Device-to-Cloud" Communications..... 11
- 5. Configure MQTT Client Agent to receive "Cloud-to-Device" messages from Azure IoT Hub 12
- Appendix A: Connecting with self-signed X.509 Certificates..... 14

1. Overview

This Connectivity Guide discusses how to configure the KEPServerEX® IoT Gateway Plug-in to communicate with Microsoft® Azure IoT Hub using MQTT over TLS.

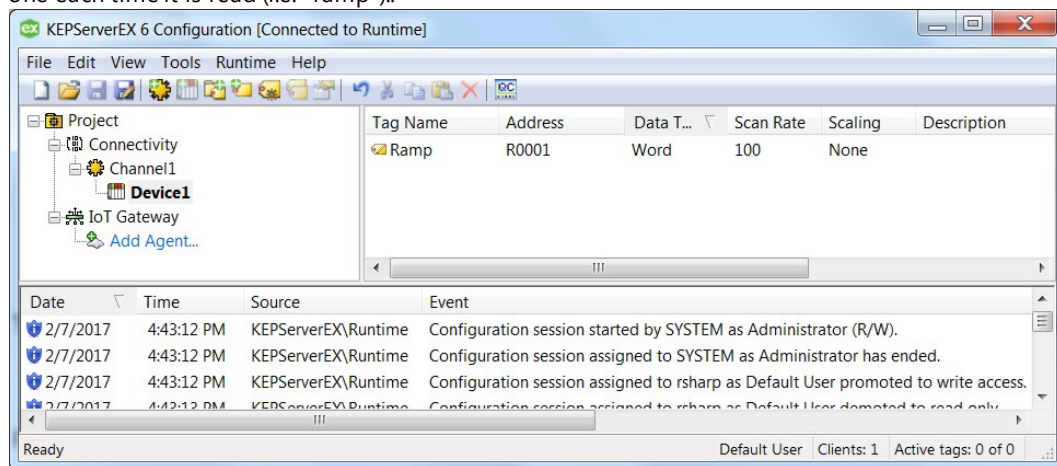
For information on how to connect the KEPServerEX IoT Gateway Plug-in to Azure IoT Hub through Azure IoT Edge, see [KEPServerEX and Microsoft Azure IoT Edge](#).

2. Create IoT Device and Get SAS Token

The Azure IoT Hub acts as a central message hub for bidirectional communication between IoT applications and devices. How “devices” are modeled depends on the specific application, but it’s important to understand that each IoT Device configured in the IoT Hub will be associated with the data stream/connection for an MQTT Client Agent in KEPServerEX IoT Gateway Plug-in.

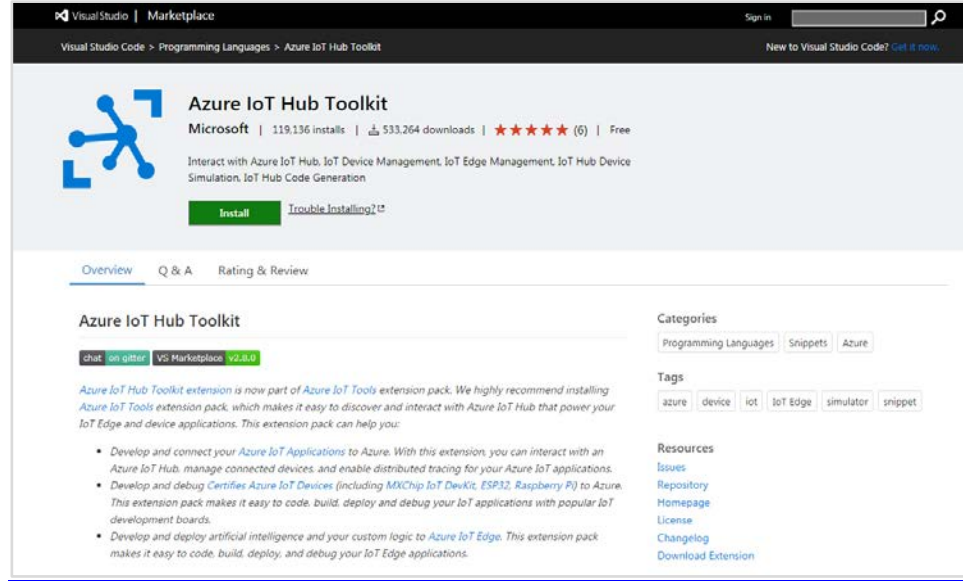
Notes:

- These instructions use Visual Studio Code with an Azure IoT Hub extension to manage the hub, the devices and to monitor or send data to the IoT Devices. IoT Devices can also be managed through Azure CLI commands, through the Azure Portal, and other coded methods using the Azure IoT Hub SDKs.
 - Connection status for a “device” in the IoT Hub is based upon the overall connection state of the MQTT connection. A MQTT Client Agent in Kepware’s IoT Gateway will only connect if the server has data to publish. If an application is designed to monitor the connection status of a “device” in the IoT Hub, it is recommended that you enable the Subscriptions in the Kepware MQTT Client Agent as defined in [Section 5](#).
- Open a KEPServerEX instance with the IoT Gateway Advanced Plug-in. In this example, one channel and device are configured with the Simulator driver, and there is one tag that increases in value by one each time it is read (i.e. “ramp”).



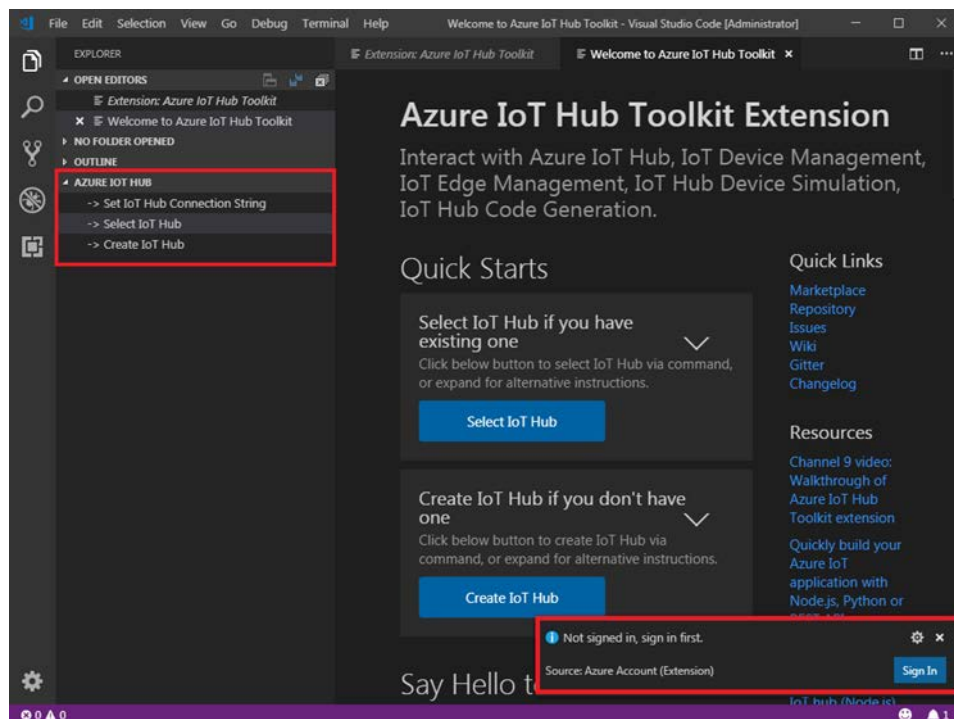
- Download and install Visual Studio Code: <https://code.visualstudio.com/download>

3. Install Azure IoT Hub Extension for Visual Studio Code:
<https://marketplace.visualstudio.com/items?vsciot-vscode.azure-iot-toolkit>

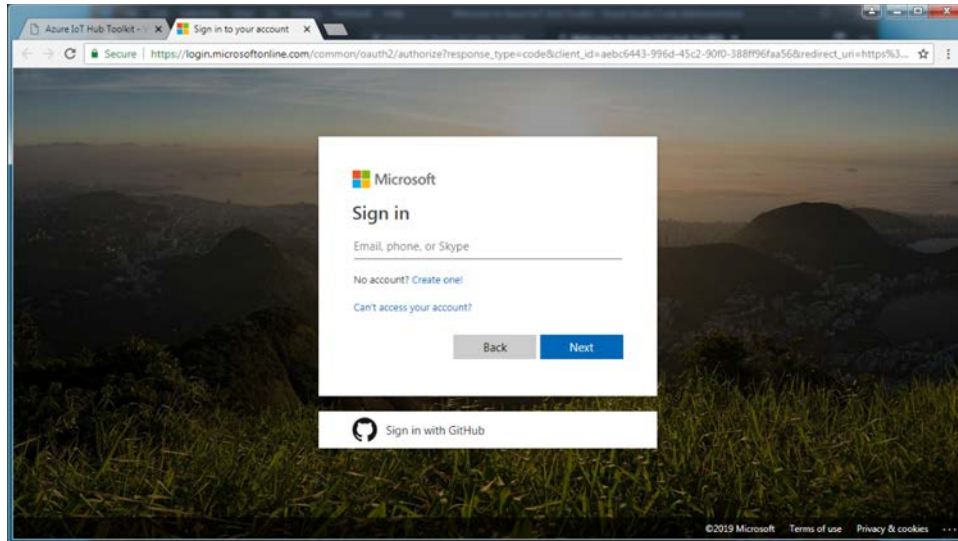


4. Once Azure IoT Hub Extension is installed, click **Select IoT Hub** and select **Sign in** when prompted by the pop-up noted below.

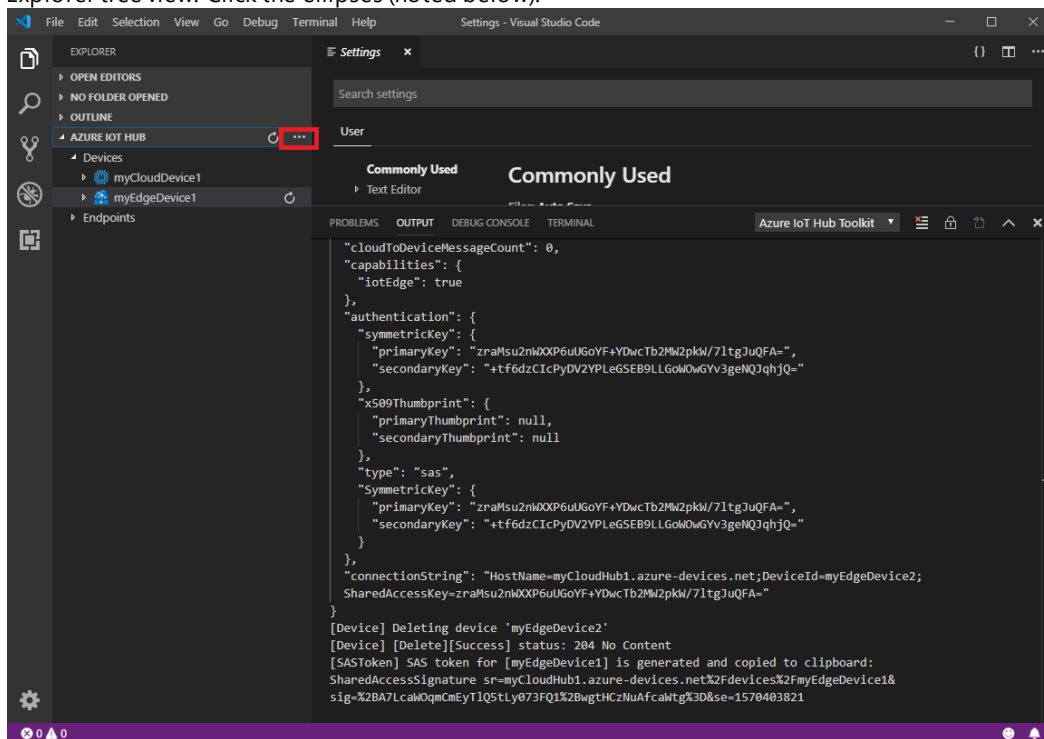
Note: The remainder of this guide assumes that an IoT Hub already exists within the Azure Portal account.



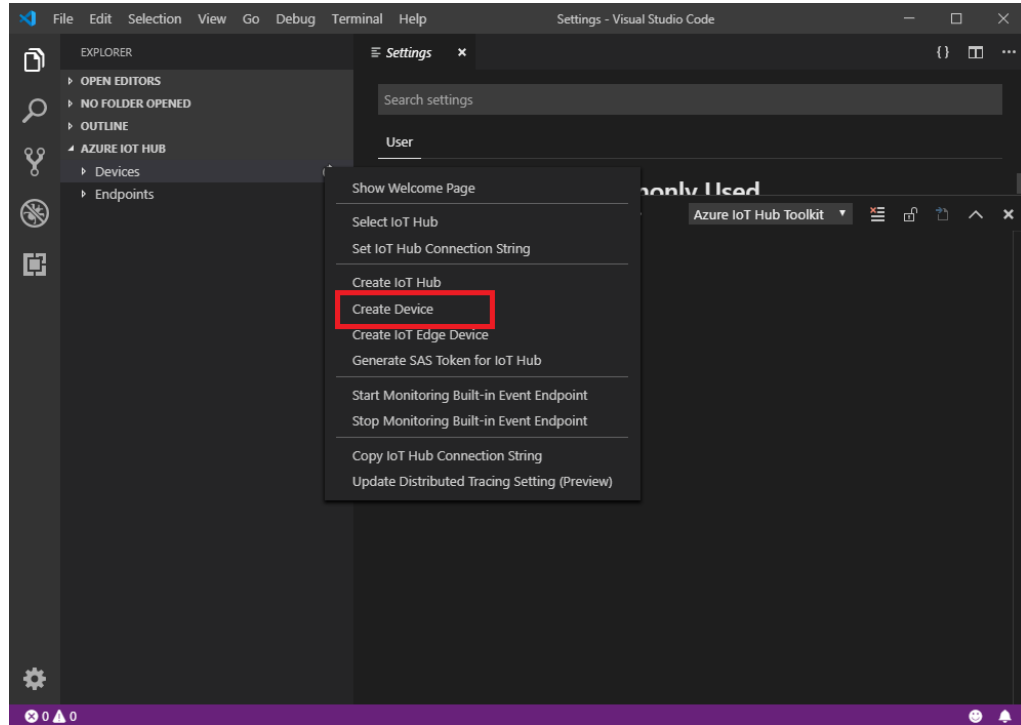
5. The default web browser launches a Microsoft Sign In page. Log into Microsoft using Azure Portal credentials.



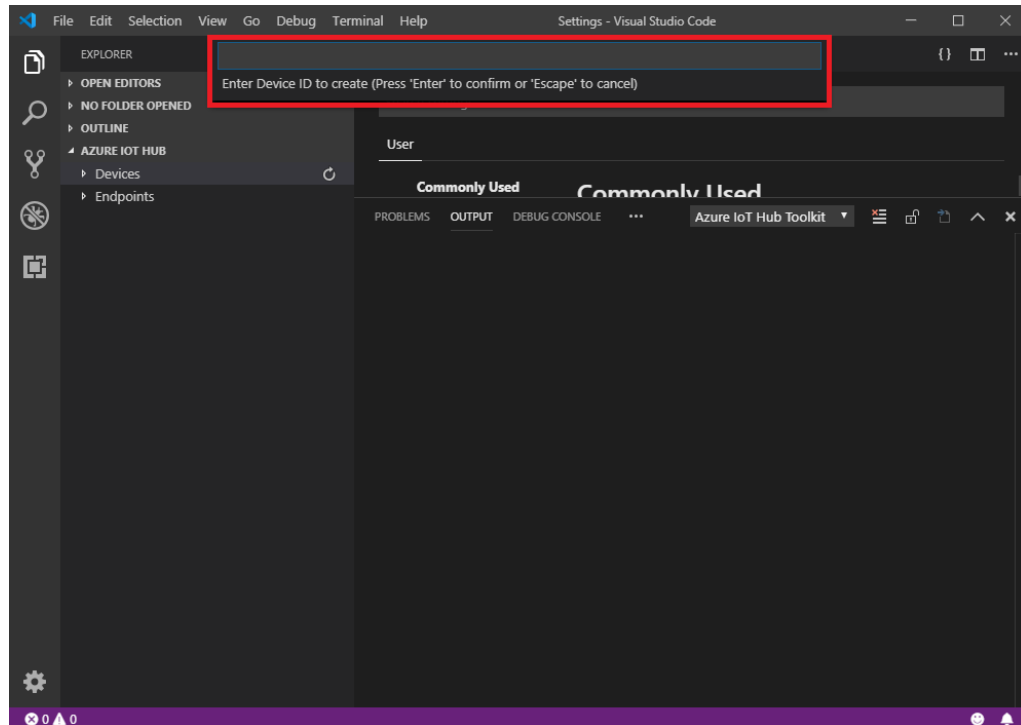
6. Once signed in, return to Visual Studio Code and expand the Azure IoT Hub extension in the Explorer tree view. Click the ellipses (noted below).



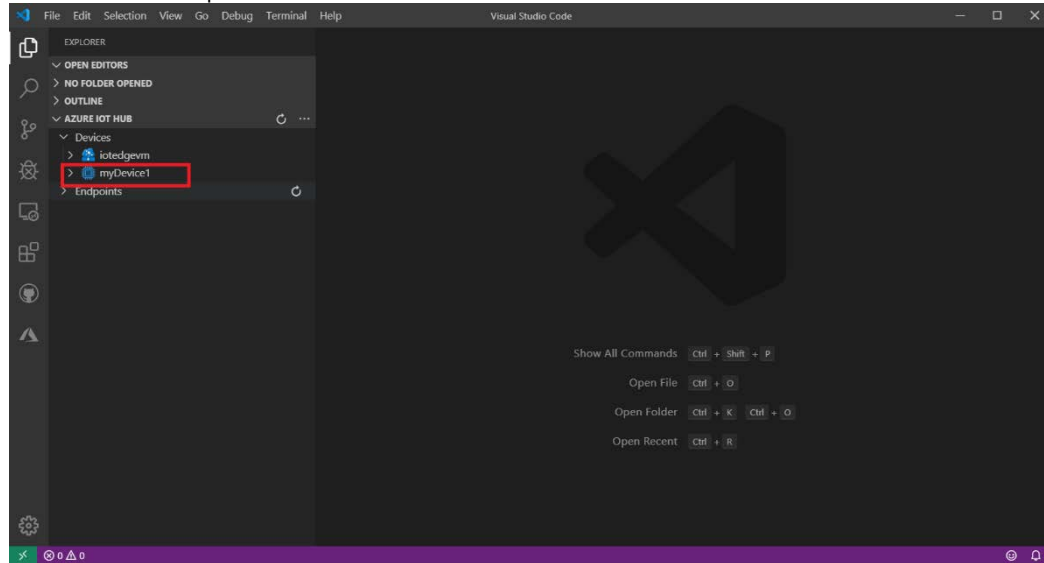
7. From the context menu, click **Create Device**.



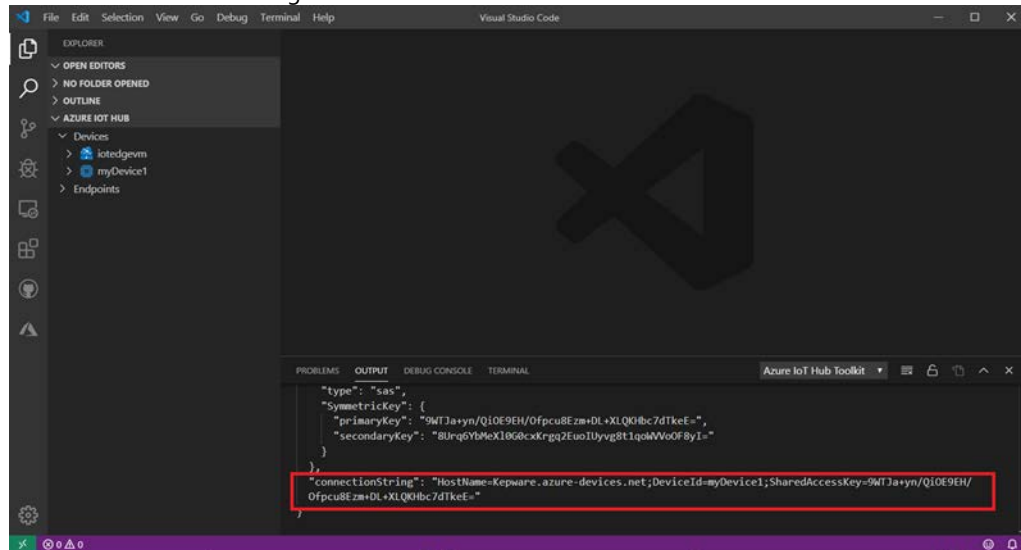
8. Enter a Device ID for the new IoT device.



9. To verify the IoT device has been successfully created, find it listed below the Azure IoT Hub extension in the Explorer tree view.

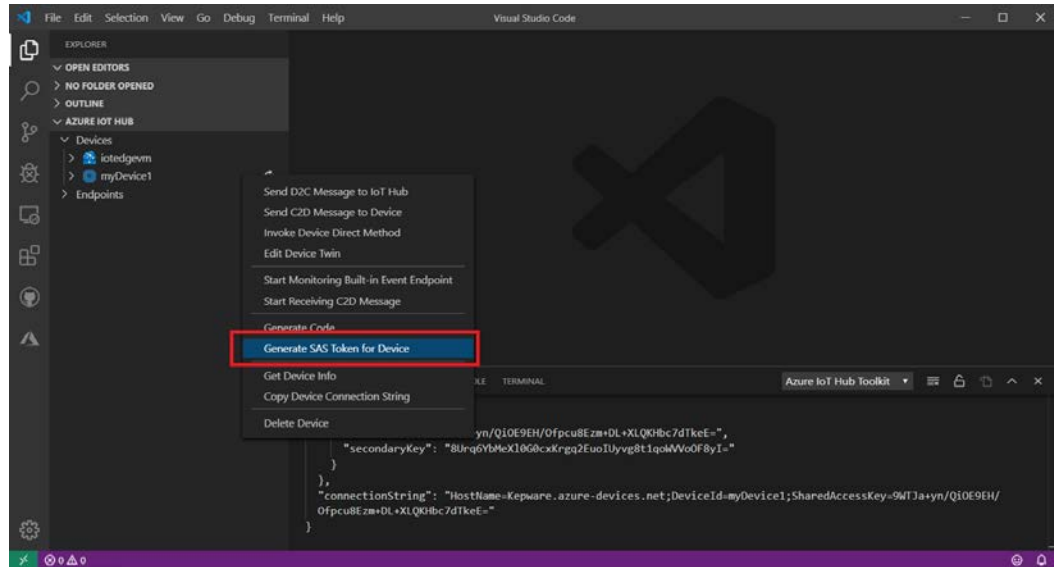


10. The Connection String is part of the Device Info displayed in the Output window in Visual Studio Code. Review the Connection String created for the device and record the following pieces of information from this string:

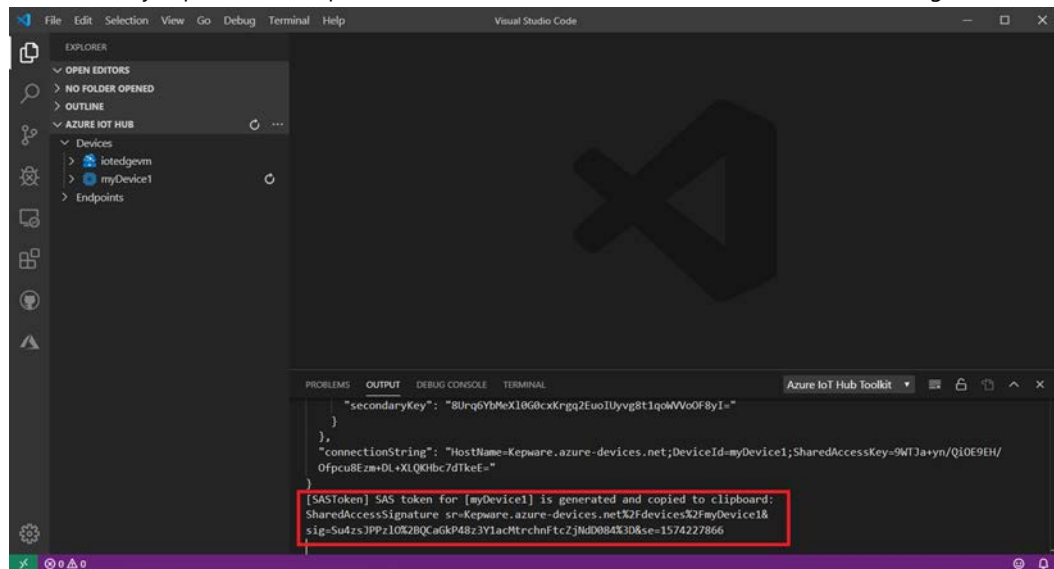


Information	Example
<IoT Hub Hostname>	myCloudHub1.azure-devices.net
<deviceId>	myDevice1

11. Right-click the new device and select **Generate SAS token for device**.



12. Enter a Time To Live (TTL) or expiration for the SAS token
13. The SAS token is created and displayed in the Output window in Visual Studio Code and automatically copied to the clipboard. Paste it into a text document for later use in this guide.

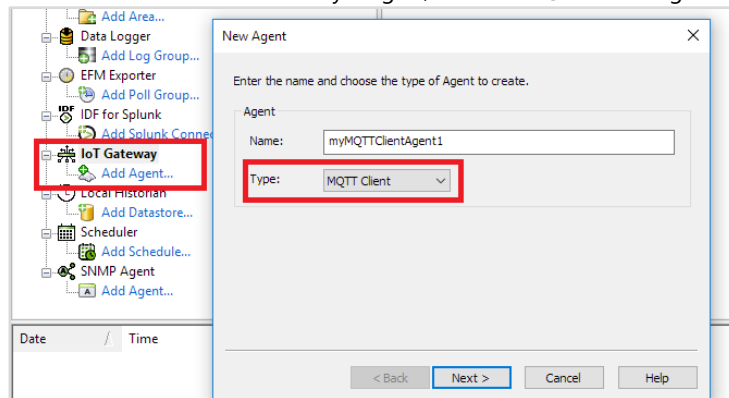


14. Review the SAS token string and record the following pieces of information:

Information	Example
<SAS token>	SharedAccessSignature sr=myCloudHub1.azuredevices.net%2Fdevices%2FmyDevice1&sig=HS%2FfyEVuC Fxm5JYZJ%2BzKKIQZyp1SqfcSVQDzSlgtCg%3D&se=1562872697

3. Configure KEPServerEX MQTT Client Agent to connect to Azure IoT Hub

1. In the KEPServerEX IoT Gateway Plug-in, add an MQTT Client Agent.



2. In the MQTT Client Agent, access **Client** property group to edit the URL and Topic per the following formats:

Property Groups	
General	
Client	
Message	
Security	
Last Will	
Subscriptions	
Licensing	

MQTT Broker	
URL	ssl://myCloudHub1.azure-devices.net:8883
Topic	devices/myCloudDevice1/messages/events/deviceData
Publish	
QoS	1 (At least once)
Rate (ms)	10000
Format	Narrow Format
Max Events per Publish	1000
Transaction timeout in (s)	5

- URL format: ssl://<IoT Hub Hostname>:8883

```
ssl://myCloudHub1.azure-devices.net:8883
```

- Topic format: devices/<deviceId>/messages/events/<property bag>

```
devices/myDevice1/messages/events/location=abcd&id=12345
```

<property bag> (optional) sends each message with additional properties in a url-encoded format. For example:

```
location=abcd&id=12345
```

For more information about the property bag; visit <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support>.

Important: The topic format must include the forward slashes, including the ending forward slash if the <property bag> objects are not included. For example:

```
devices/myDevice1/messages/events/
```

3. Enter security credentials in the following formats:

Property Groups		
General		
Client		
Message		
Security		
Last Will		
Subscriptions		
Licensing		

Credentials		
Client ID	myCloudDevice 1	
Username	myCloudHub 1.azure-devices.net/myCloudDevice 1	
Password	*****	
TLS Configuration		
TLS Version	Default	
Client Certificate	Disable	

Property	Format
Client ID	<deviceID>
Username	<IoT Hub Hostname>/<deviceID>
Password	<SAS token>

- Access the MQTT Client Agent and select **Add IoT items..** to add an IoT item reference (i.e. a KEPServerEX tag reference) to the Agent.

File Edit View Tools Runtime Help

Project

- Connectivity
- Aliases
- Advanced Tags
- Alarms & Events
- Add Area...
- Data Logger
- Add Log Group...
- EFM Exporter
- Add Poll Group...
- IDF for Splunk
- Add Splunk Connection...

Server Tag

Add IoT items...

Server Tag: Channel1.Device1.Ramp

Scan Rate (ms): 1000

Publish

☒ Only on Data Changes

Deadband (%): 0

☐ Every scan

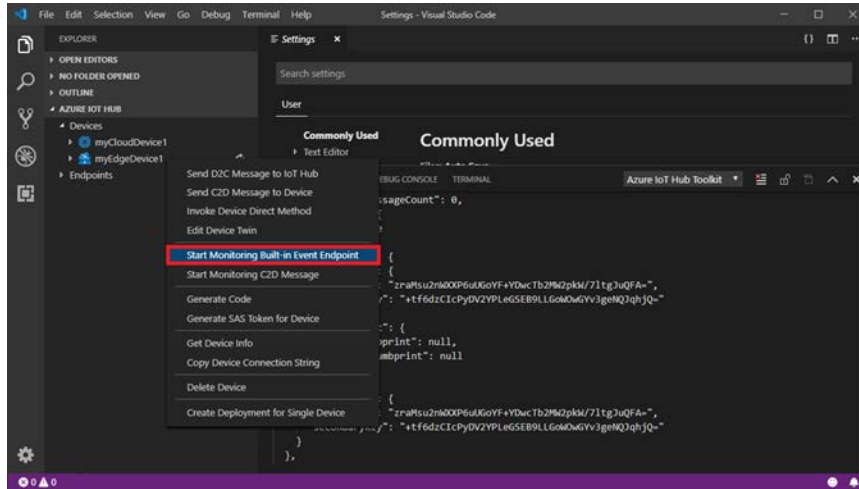
- To verify that the MQTT Client agent has connected to the Azure IoT Hub, find an event log entry similar to the following:

Date	Time	Level	Source	Event
2/10/2017	2:41:35 PM	Information	KEPServerEXRuntime	MQTT agent 'Ramp 2 Cloud' is connected to broker 'ssl://kepwaremqtt.azure-devices.net:8883'

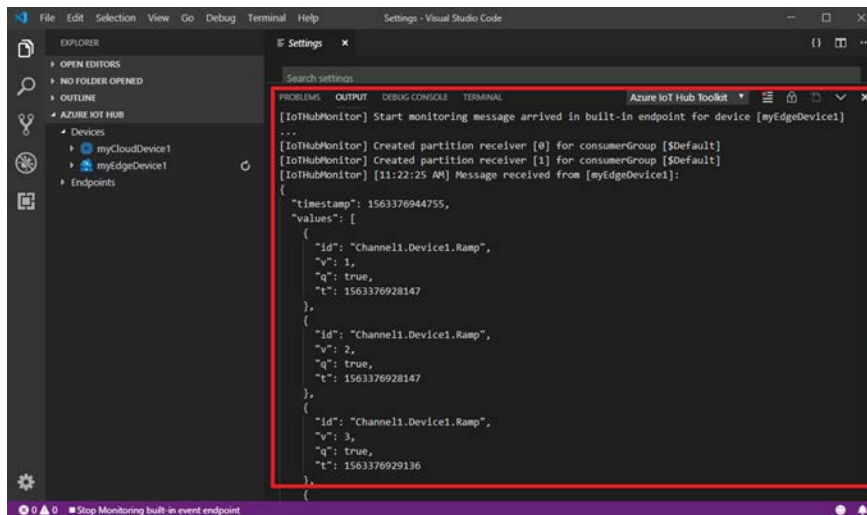
4. Verify Successful “Device-to-Cloud” Communications

Use Visual Studio Code to monitor the device’s built-in event endpoint and verify that data starts flowing from Kepware’s MQTT Client Agent to the cloud-based Azure IoT Hub within a few seconds.

1. In Visual Studio Code, right-click the edge device entry and select **Start Monitoring Built-In Event Endpoint**.



2. KEPServerEX data payloads should display in the Output window of Visual Studio Code.



5. Configure MQTT Client Agent to receive “Cloud-to-Device” messages from Azure IoT Hub

Applications can send write or command messages to Kepware using “Cloud-to-Device” messages. When a “device” is created in the IoT Hub, an endpoint is created to allow this transaction to the “device” or MQTT Client Agent in Kepware. Functionally, to send a command to a PLC, a properly formatted JSON payload will need to be sent as documented in the [IoT Gateway Manual](#).

- **Note:** Connection status for a “device” in the IoT Hub is based upon the overall connection state of the MQTT connection. Setting up the Subscriptions will ensure that the MQTT Agent will always reconnect, even when there is no data from the server to publish.

1. In the MQTT Client Agent, access **Subscriptions** property group. Change the “Listen for Write Requests” to Yes and update Topic per the following formats:

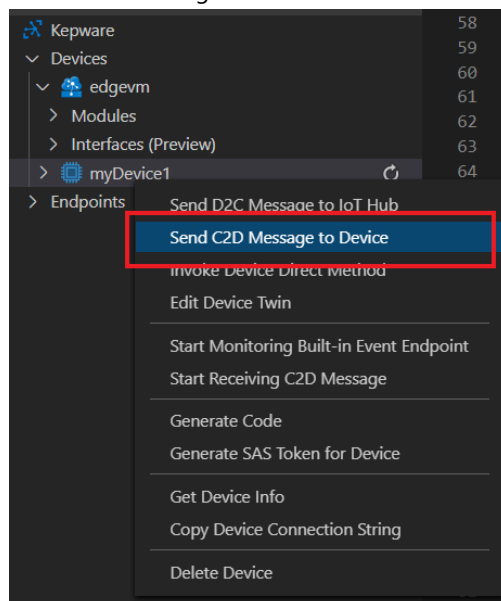
Property Groups	Subscriptions
General	Listen for Write Requests Yes
Client	Topic devices/myDevice1/messages/devicebound/#
Message	
Security	
Last Will	
Subscriptions	
Licensing	

- Topic format: devices/<**deviceID**>/messages/devicebound/#
For example:

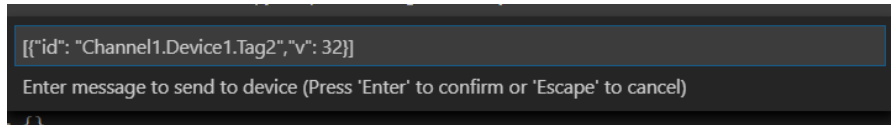
```
devices/myDevice1/messages/devicebound/#
```

● For more information about the cloud to device messages, visit <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support>.

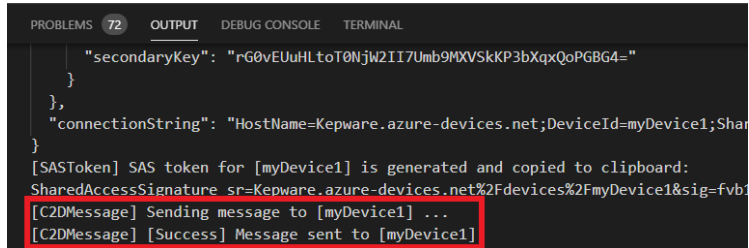
2. To test, you can execute “Cloud-to-Device” messages from the Visual Studio Code IoT Hub extension. To do this, right click on the device you want to send the command to and select the “Send C2D Message to Device”.



3. Enter the JSON payload to send, writing a value to a specific tag or collection of tags per the IoT Gateway manual.



4. If the command was properly received by the MQTT Client Agent, the Visual Studio Code output should indicate that the C2D message was sent successfully.



Appendix A: Connecting with self-signed X.509 Certificates

Microsoft Azure IoT Hub also supports self-signed certificates for authorization, which keeps a certificate and private key on the local machine and stores only a certificate thumbprint on the hub.

- **Note:** This guide uses legacy components (like the Device Manager) to manage the certificate assignment in Azure IoT Hub.

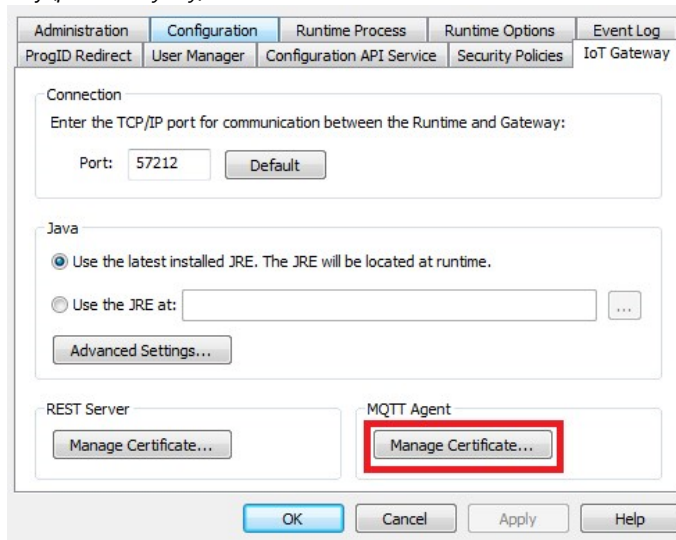
- For more information on how to connect with self-signed certificates refer to <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguidesecurity>.

The following information describes how to connect IoT Gateway to the hub:

1. Generate self-signed certificates. This command outputs a new generated certificate and private key:

```
$ openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout privateKey.key -out certificate.crt
```

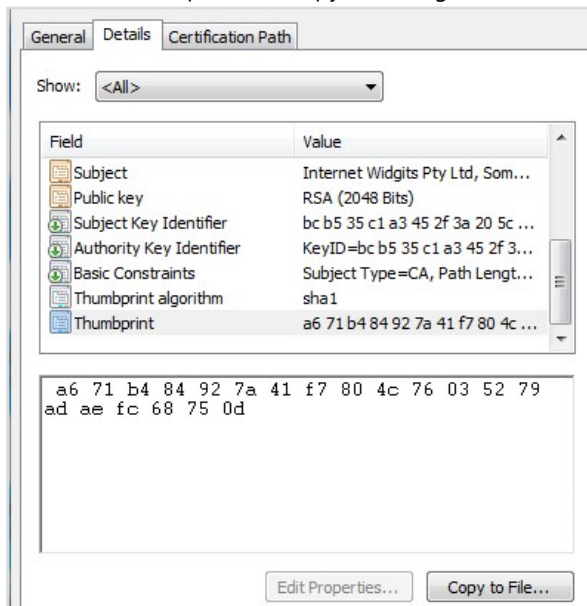
2. To import the certificate and private key to IoT Gateway, access the Administration Menu and select **Settings**. Access the IoT Gateway tab, then click **Manage Certificate...**
3. Click **Import New Certificate...** and browse to the certificate created (*certificate.crt*).
4. Once the certificate is imported, the Import dialogue box immediately reopens to import the private key (*privateKey.key*).



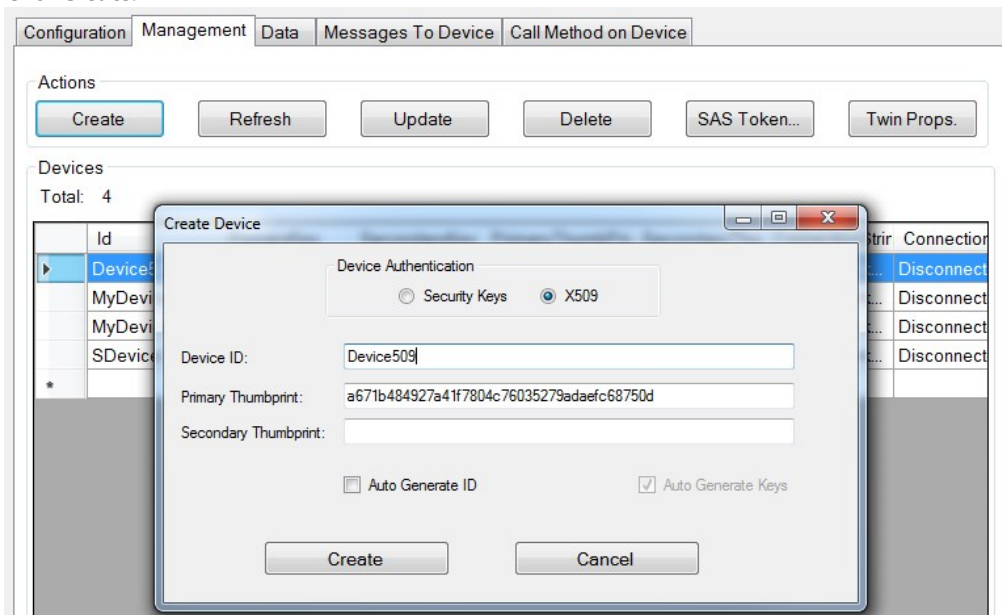
5. Once the private key is imported, a popup dialog will request a private key password.

- **Note:** The files required for this depend on the format and contents of the file(s) being imported. For example, if a PFX file is selected, and it contains both the certificate and private key, no additional files are required. The OpenSSL creation process used in this example creates both a .crt and .key file, and both need to be imported independently (see steps 2-4 above).

6. Obtain the certificate thumbprint:
 - a. Click **View Certificate**.
 - b. Access the **Details** tab, and from the drop-down menu, select **<All>**.
 - c. Select the thumbprint and copy the string.



7. Create device using Device Explorer.
 - a. Open Device Explorer, access the **Management** tab, and select **Create**.
 - b. Select the **X509** option.
 - c. Enter **Device ID** and paste the string into the **Primary Thumbprint** field.
 - d. Click **Create**.



8. Create a new MQTT Client Agent within the KEPServerEX IoT Gateway Plug-in. The following property groups should be configured per the tables below:

Client	
URL	ssl:// Azure IoT Hub Host name :8883
Topic	devices/ Device ID /messages/events/

Property Groups	<div>MQTT Broker</div> <div>URL: ssl://kepiot-azure-devices.net:8883</div> <div>Topic: devices/Device509/messages/events</div> <div>Publish</div> <div>QoS: 1 (At least once)</div> <div>Rate (ms): 10000</div> <div>Format: Narrow Format</div> <div>Max Events per Publish: 1000</div> <div>Transaction timeout in (s): 5</div>
-----------------	---

Security	
Client ID	<deviceID>
Username	<IoT Hub Hostname>/<deviceID>
Password	HostName= <IoT Hub Hostname>; DeviceID = <deviceID>;x509=true
TLS Version	v1.2
Client Certificate	Enabled

Property Groups	<div>Credentials</div> <div>Client ID: Device509</div> <div>Username: kepiot.azure-devices.net/Device509</div> <div>Password: *****</div> <div>TLS Configuration</div> <div>TLS Version: v1.2</div> <div>Client Certificate: Enable</div>
-----------------	---

9. Add tags to the MQTT agent. Event log messages should indicate if the agent was successfully connected to the broker.
10. To monitor messages from IoT Gateway:
 - a. Open Device Explorer and access the **Data** tab.
 - b. Select the device from the drop-down menu and click **Monitor**.

Configuration Management **Data** Messages To Device Call Method on Device

Monitoring

Event Hub: kepiot

Device ID: Device509

Start Time: 02/27/2017 15:29:04

Consumer Group: \$Default ☐ Enable

Monitor Cancel Clear

Event Hub Data

```
{
  "id": "Channel1.Device1.R1",
  "v": 4084,
  "q": true,
  "t": 1488227377340
},
{
  "id": "Channel1.Device1.R1",
  "v": 4085,
  "q": true,
  "t": 1488227378361
},
{
  "id": "Channel1.Device1.R1",
  "v": 4086,
  "q": true,
  "t": 1488227379371
}
```