Connectivity Guide

Kepware Server and Microsoft Azure IoT Hub

November 2025
Ref 1.06

Table of Contents

1. Overview

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

For information on how to connect the Kepware Server IoT Gateway Plug-In to Azure IoT Hub through Azure IoT Edge, see Kepware Server and Microsoft Azure IoT Edge.

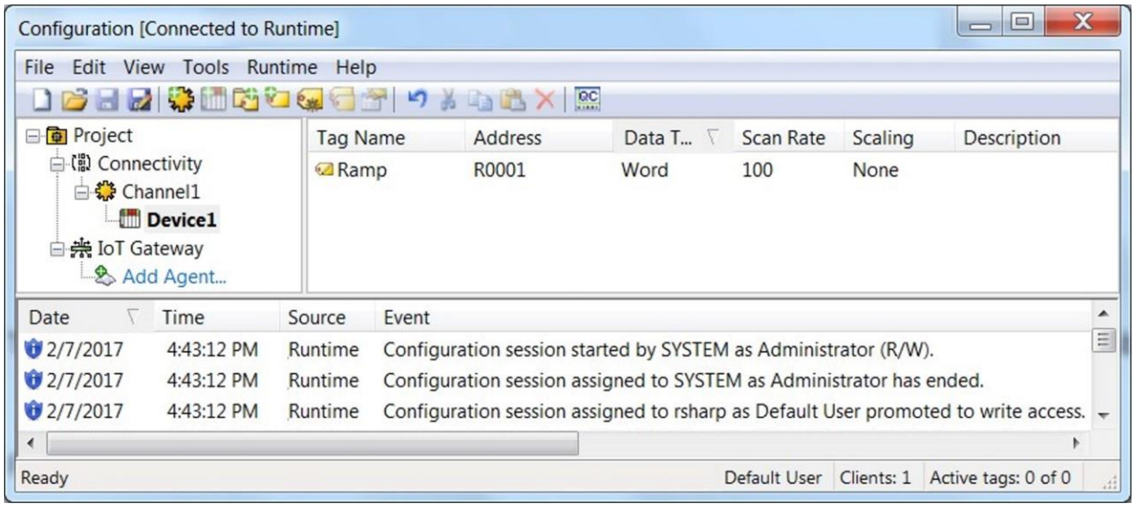
2. Create IoT Device and Get SAS Token

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 Kepware Server 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.

1. Open a Kepware Server 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").

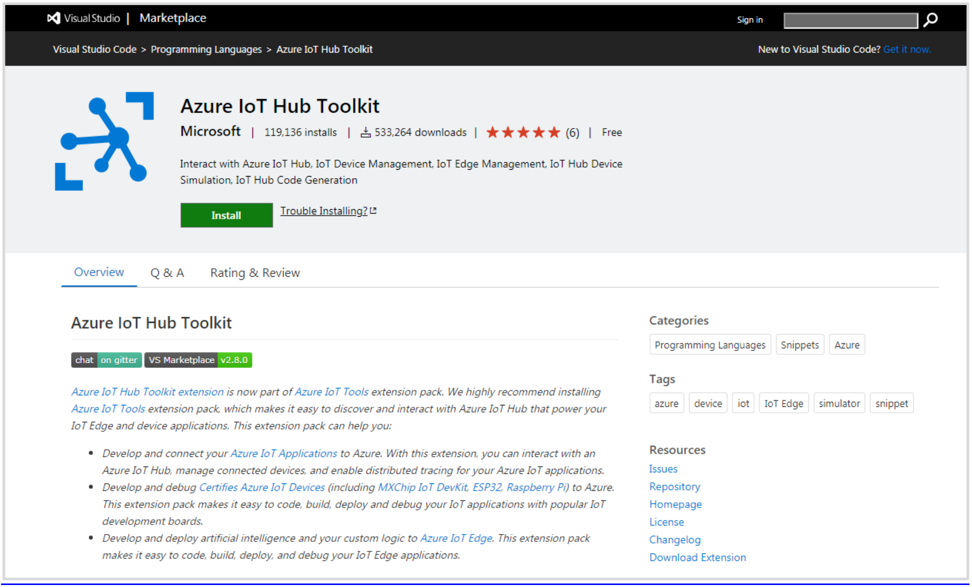


Tag Name	Address	Data T...	Scan Rate	Scaling	Description
Ramp	R0001	Word	100	None	

Date	Time	Source	Event
2/7/2017	4:43:12 PM	Runtime	Configuration session started by SYSTEM as Administrator (R/W).
2/7/2017	4:43:12 PM	Runtime	Configuration session assigned to SYSTEM as Administrator has ended.
2/7/2017	4:43:12 PM	Runtime	Configuration session assigned to rsharp as Default User promoted to write access.

2. 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?itemName=vsciot-vscode.azure-iot-toolkit


Visual Studio | Marketplace
Sign in

Visual Studio Code > Programming Languages > Azure IoT Hub Toolkit
New to Visual Studio Code? Get it now.

Azure IoT Hub Toolkit
Microsoft | 119,136 installs | 533,264 downloads | ★★★★★ (6) | Free

Interact with Azure IoT Hub, IoT Device Management, IoT Edge Management, IoT Hub Device Simulation, IoT Hub Code Generation

Install   Trouble Installing?

Overview   Q & A   Rating & Review

Azure IoT Hub Toolkit

chat on gitter | VS Marketplace v2.8.0

Azure IoT Hub Toolkit extension is now part of Azure IoT Tools extension pack. We highly recommend installing Azure IoT Tools extension pack, which makes it easy to discover and interact with Azure IoT Hub that power your IoT Edge and device applications. This extension pack can help you:

- *Develop and connect your Azure IoT Applications to Azure. With this extension, you can interact with an Azure IoT Hub, manage connected devices, and enable distributed tracing for your Azure IoT applications.*
- *Develop and debug Certifies Azure IoT Devices (including MXChip IoT DevKit, ESP32, Raspberry Pi) to Azure. This extension pack makes it easy to code, build, deploy and debug your IoT applications with popular IoT development boards.*
- *Develop and deploy artificial intelligence and your custom logic to Azure IoT Edge. This extension pack makes it easy to code, build, deploy, and debug your IoT Edge applications.*

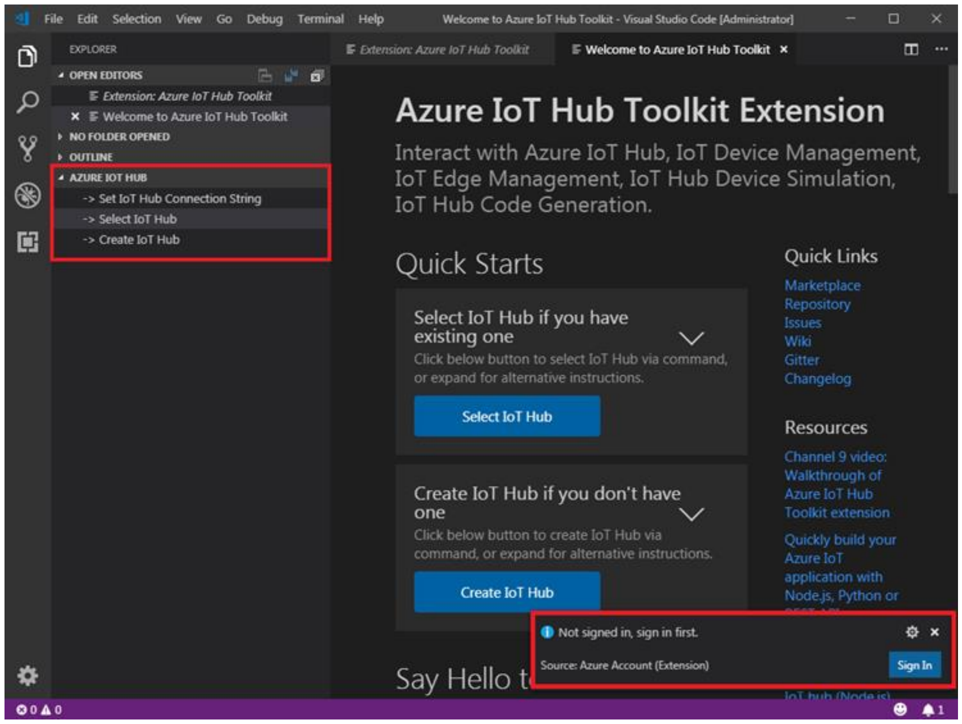
Categories
Programming Languages | Snippets | Azure

Tags
azure | device | iot | IoT Edge | simulator | snippet

Resources
Issues
Repository
Homepage
License
Changelog
Download Extension

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.


File Edit Selection View Go Debug Terminal Help   Welcome to Azure IoT Hub Toolkit - Visual Studio Code [Administrator]

EXPLORER
OPEN EDITORS
Extension: Azure IoT Hub Toolkit
× Welcome to Azure IoT Hub Toolkit
NO FOLDER OPENED
OUTLINE
AZURE IOT HUB
-> Set IoT Hub Connection String
-> Select IoT Hub
-> Create IoT Hub

Extension: Azure IoT Hub Toolkit | Welcome to Azure IoT Hub Toolkit ×

Azure IoT Hub Toolkit Extension
Interact with Azure IoT Hub, IoT Device Management, IoT Edge Management, IoT Hub Device Simulation, IoT Hub Code Generation.

Quick Starts

Select IoT Hub if you have existing one
Click below button to select IoT Hub via command, or expand for alternative instructions.
Select IoT Hub

Create IoT Hub if you don't have one
Click below button to create IoT Hub via command, or expand for alternative instructions.
Create IoT Hub

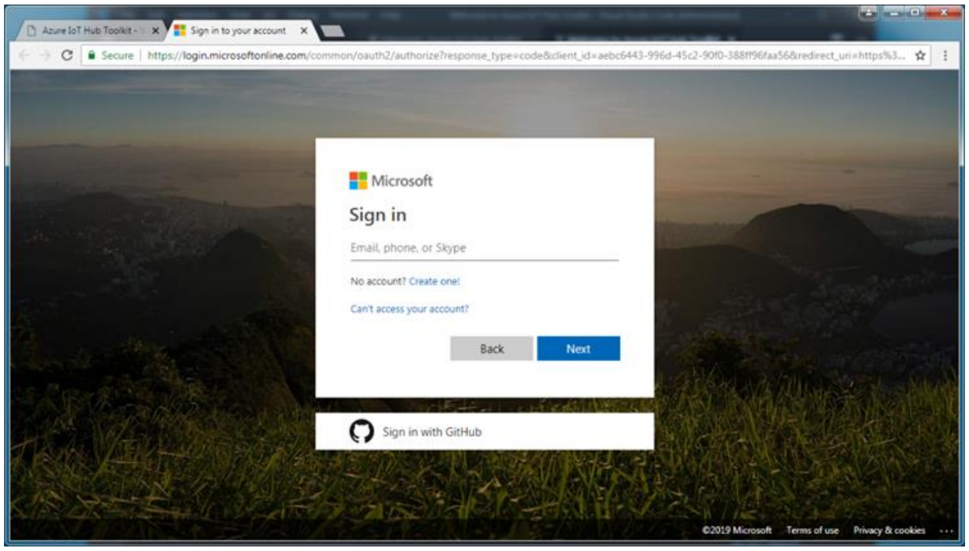
Say Hello t

Quick Links
Marketplace
Repository
Issues
Wiki
Gitter
Changelog

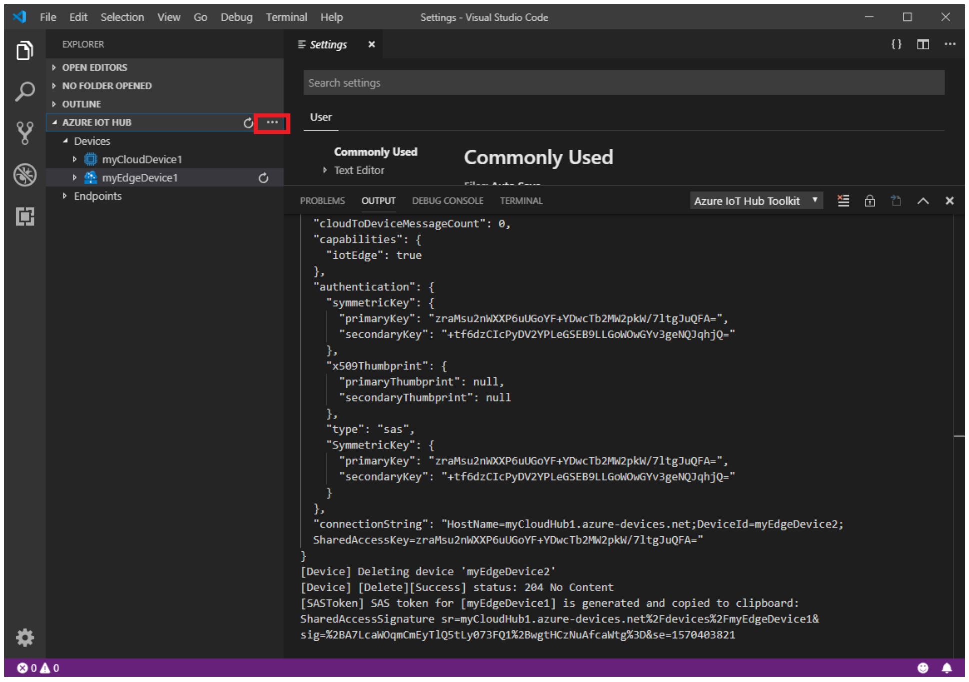
Resources
Channel 9 video: Walkthrough of Azure IoT Hub Toolkit extension
Quickly build your Azure IoT application with Node.js, Python or

Not signed in, sign in first.
Source: Azure Account (Extension)   Sign In

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

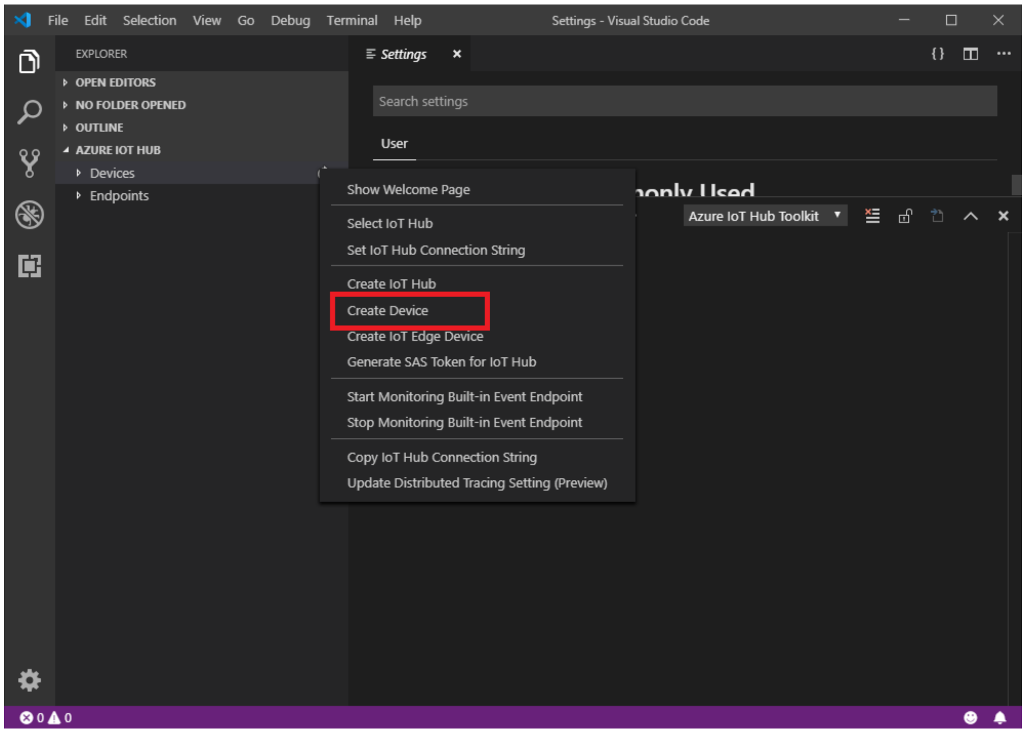

Azure IoT Hub Toolkit
Sign in to your account
Secure https://login.microsoftonline.com/common/oauth2/authorize?response_type=code&client_id=aebc6443-996d-45c2-90f0-388ff96faa56&redirect_uri=https%3...
Microsoft
Sign in
Email, phone, or Skype
No account? Create one!
Can't access your account?
Back
Next
Sign in with GitHub
©2019 Microsoft
Terms of use
Privacy & cookies

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).

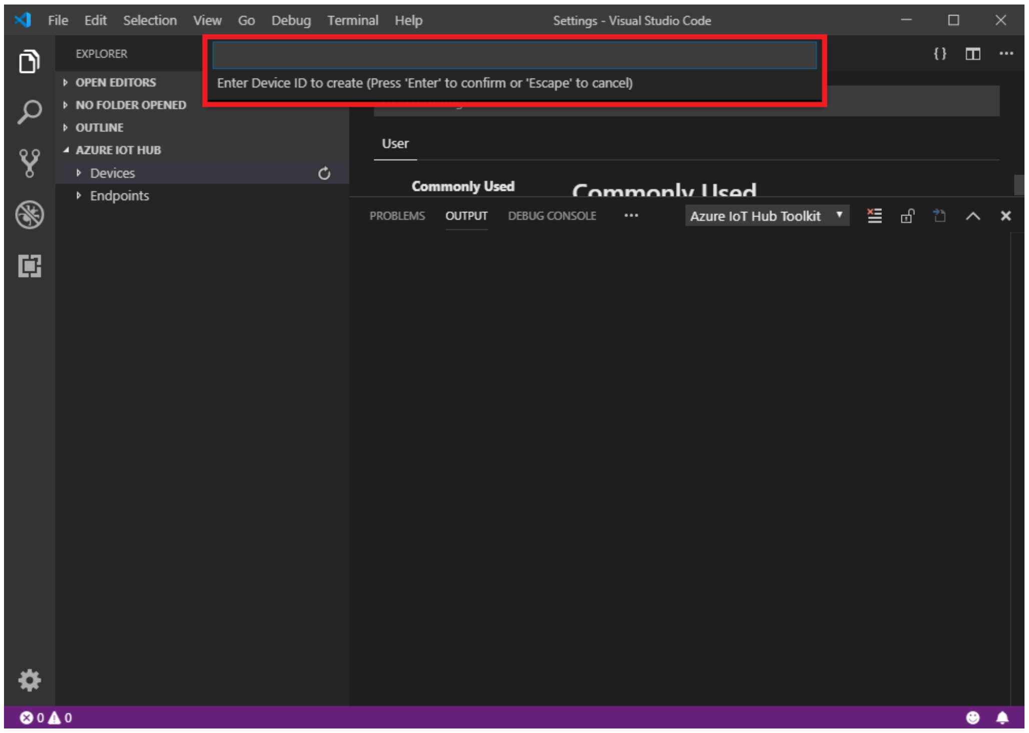


```
  "cloudToDeviceMessageCount": 0,
  "capabilities": {
    "iotEdge": true
  },
  "authentication": {
    "symmetricKey": {
      "primaryKey": "zraMsu2nWXXP6uUGoYF+YDwcTb2MW2pkW/7ltgJuQFA=",
      "secondaryKey": "+tf6dzCIcPyDV2YPLeGSEB9LLGoWOwGYv3geNQJqhjQ="
    },
    "x509Thumbprint": {
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    },
    "type": "sas",
    "SymmetricKey": {
      "primaryKey": "zraMsu2nWXXP6uUGoYF+YDwcTb2MW2pkW/7ltgJuQFA=",
      "secondaryKey": "+tf6dzCIcPyDV2YPLeGSEB9LLGoWOwGYv3geNQJqhjQ="
    }
  },
  "connectionString": "HostName=myCloudHub1.azure-devices.net;DeviceId=myEdgeDevice2;
  SharedAccessKey=zraMsu2nWXXP6uUGoYF+YDwcTb2MW2pkW/7ltgJuQFA="
}
[Device] Deleting device 'myEdgeDevice2'
[Device] [Delete][Success] status: 204 No Content
[SASToken] SAS token for [myEdgeDevice1] is generated and copied to clipboard:
SharedAccessSignature sr=myCloudHub1.azure-devices.net%2Fdevices%2FmyEdgeDevice1&
sig=%2BA7LcaWOqmCmEyTlQ5tLy073FQ1%2BwgtHCzNuAfcaWtg%3D&se=1570403821
```

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

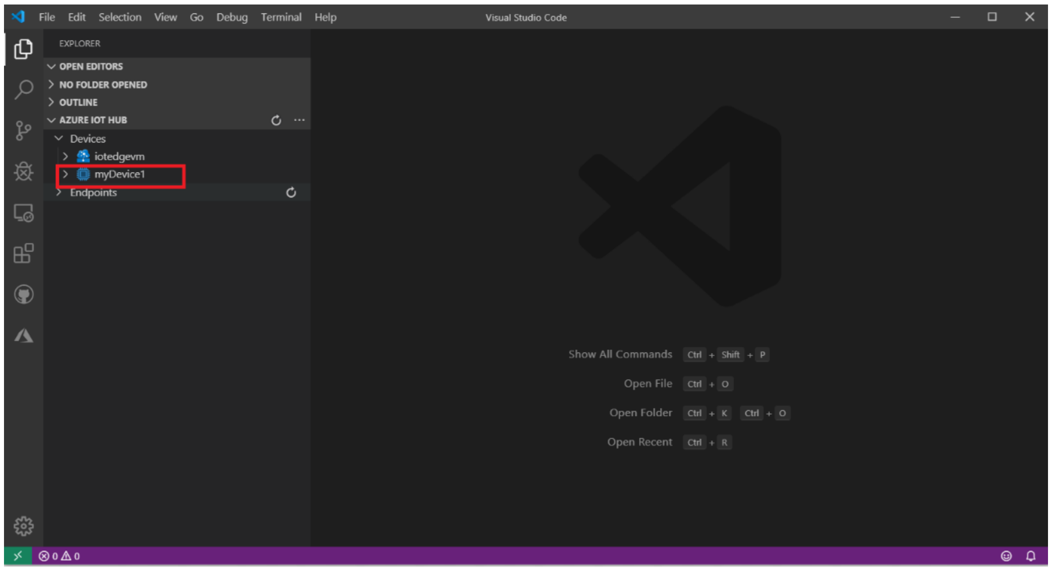

File Edit Selection View Go Debug Terminal Help
Settings - Visual Studio Code
EXPLORER
OPEN EDITORS
NO FOLDER OPENED
OUTLINE
AZURE IOT HUB
Devices
Endpoints
Settings
Search settings
User
Azure IoT Hub Toolkit
Show Welcome Page
Select IoT Hub
Set IoT Hub Connection String
Create IoT Hub
Create Device
Create IoT Edge Device
Generate SAS Token for IoT Hub
Start Monitoring Built-in Event Endpoint
Stop Monitoring Built-in Event Endpoint
Copy IoT Hub Connection String
Update Distributed Tracing Setting (Preview)

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

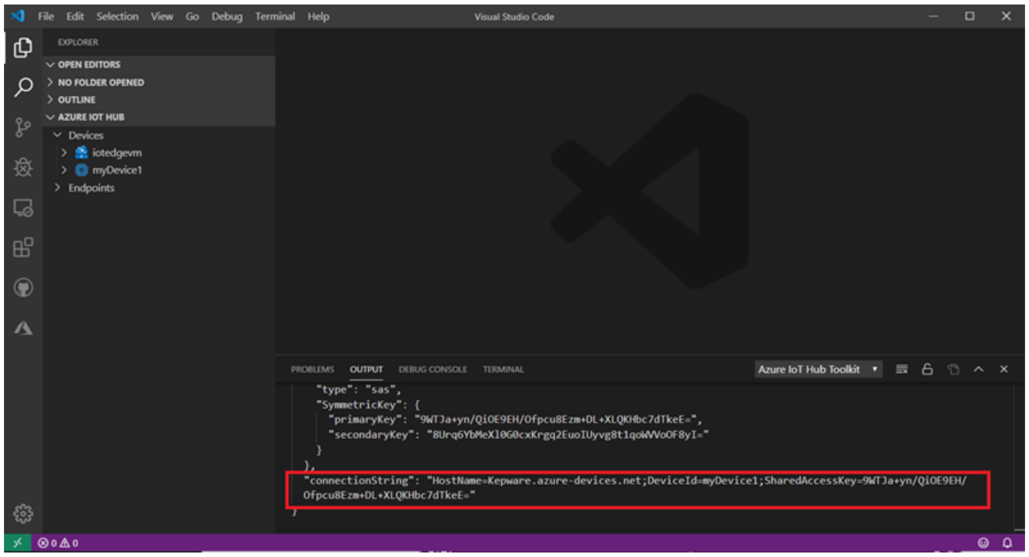


Enter Device ID to create (Press 'Enter' to confirm or 'Escape' to cancel)

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



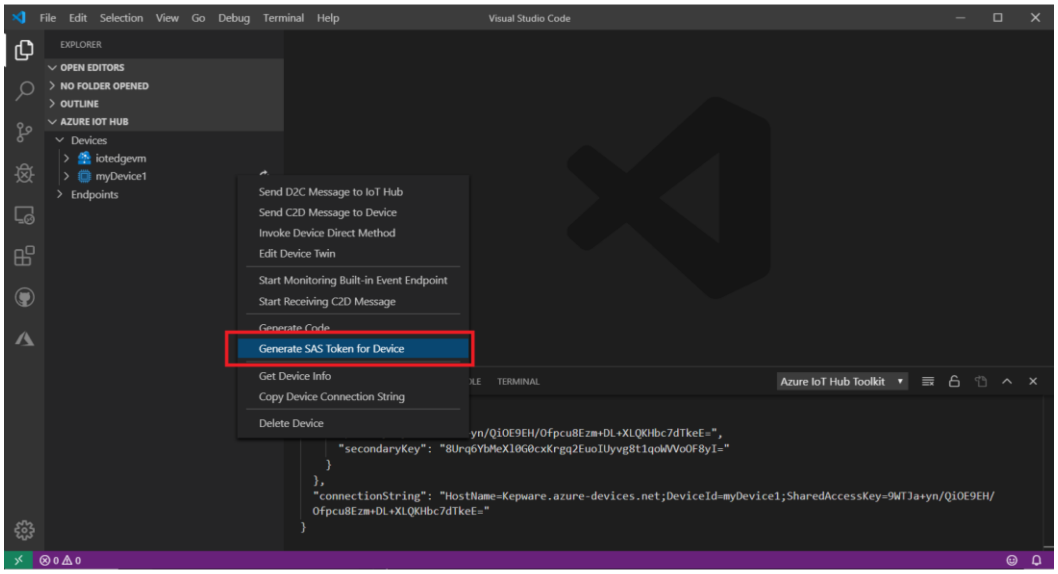
10. The Connection String is part of the Device Info displayed in the Output window in Visual Studio Code. Review the Connection String created for the device and record the following pieces of information from this string:



```
    "type": "sas",
    "SymmetricKey": {
      "primaryKey": "9WTJa+yn/QiOE9EH/Ofpcu8Ezm+DL+XLQKHbc7dTkeE=",
      "secondaryKey": "8Urq6YbMeXl0G0cxKrgq2EuoIUyvg8t1qoWVoOF8yI="
    }
  },
  "connectionString": "HostName=Kepware.azure-devices.net;DeviceId=myDevice1;SharedAccessKey=9WTJa+yn/QiOE9EH/
  Ofpcu8Ezm+DL+XLQKHbc7dTkeE="
}
```

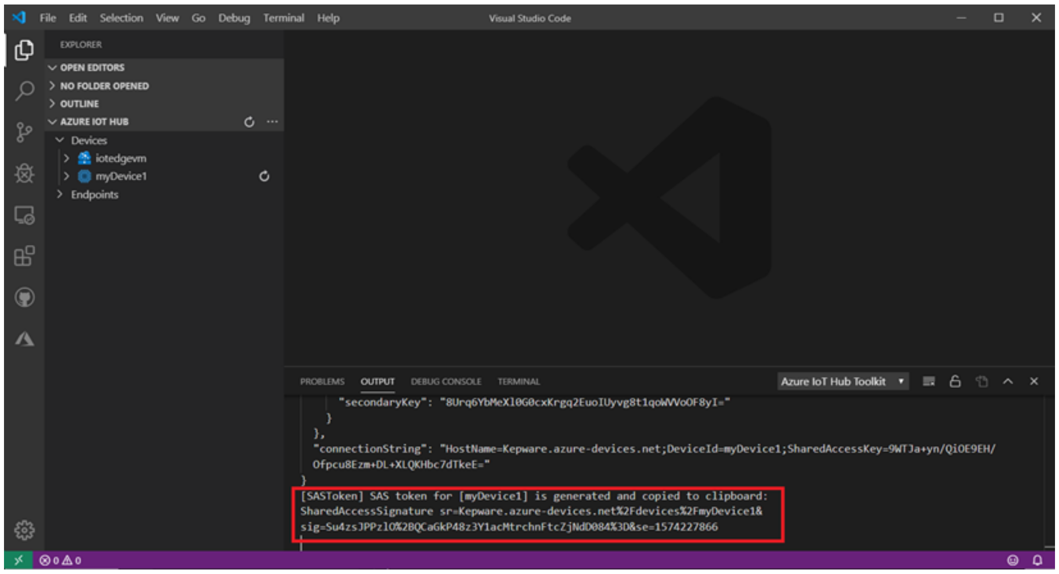
Information	Example
<IoT Hub Hostname>	myCloudHub1.azure-devices.net
<deviceID>	myDevice1

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


File Edit Selection View Go Debug Terminal Help
Visual Studio Code
EXPLORER
OPEN EDITORS
NO FOLDER OPENED
OUTLINE
AZURE IOT HUB
Devices
iotedgevm
myDevice1
Endpoints
Send D2C Message to IoT Hub
Send C2D Message to Device
Invoke Device Direct Method
Edit Device Twin
Start Monitoring Built-in Event Endpoint
Start Receiving C2D Message
Generate Code
Generate SAS Token for Device
Get Device Info
Copy Device Connection String
Delete Device
TERMINAL
Azure IoT Hub Toolkit

```
                    yn/QiOE9EH/Ofpcu8Ezm+DL+XLQKHbc7dTkeE=",
        "secondaryKey": "8Urq6YbMeXl0G0cxKrgq2EuoIUyvg8t1qoWVoOF8yI="
      }
    },
    "connectionString": "HostName=Kepware.azure-devices.net;DeviceId=myDevice1;SharedAccessKey=9WTJa+yn/QiOE9EH/
    Ofpcu8Ezm+DL+XLQKHbc7dTkeE="
  }
```

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.



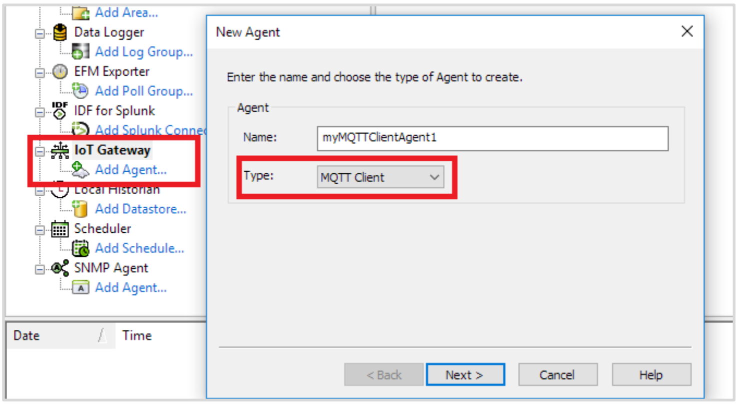
```
        "secondaryKey": "8Urq6YbMeXl0G0cxKrgq2EuoIUyvg8t1qoWVoOF8yI="
      }
    },
    "connectionString": "HostName=Kepware.azure-devices.net;DeviceId=myDevice1;SharedAccessKey=9WTJa+yn/QiOE9EH/
    Ofpcu8Ezm+DL+XLQKHbc7dTkeE="
  }
[SASToken] SAS token for [myDevice1] is generated and copied to clipboard:
SharedAccessSignature sr=Kepware.azure-devices.net%2Fdevices%2FmyDevice1&
sig=Su4zsJPPzlO%2BQCaGkP48z3Y1acMtrchnFtcZjNdD084%3D&se=1574227866
```

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%2FfyEVuCFxem5JYZJ%2BzKKIQZyp1 SqfcSVQDzSlgtCg%3D&se=1562872697

3. Configure Kepware Server MQTT Client Agent to connect to Azure IoT Hub

1. In the Kepware Server IoT Gateway Plug-In, add an MQTT Client Agent.



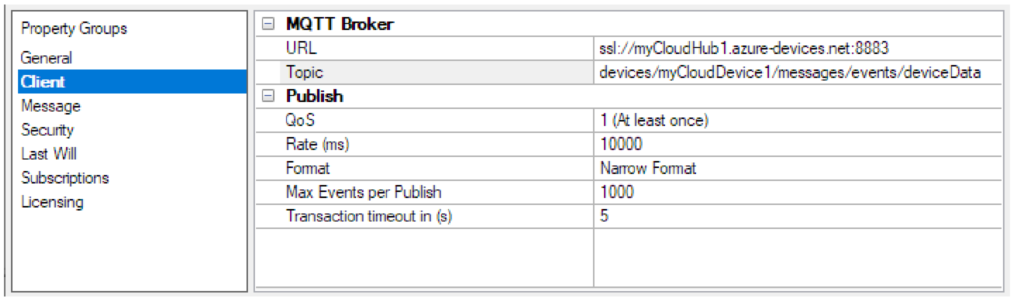
Enter the name and choose the type of Agent to create.

Agent

Name: myMQTTClientAgent1

Type: MQTT Client

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



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
```

- *Please note that when using IoT Hub data with Azure Data Lake, the property bag must be configured with a specific string. This string defines a content type of 'application/json' and a text encoding format of UTF-8:* `devices/{device-id}/messages/events/$.ct=application%2Fjson%3Bcharset%3Dutf-8`

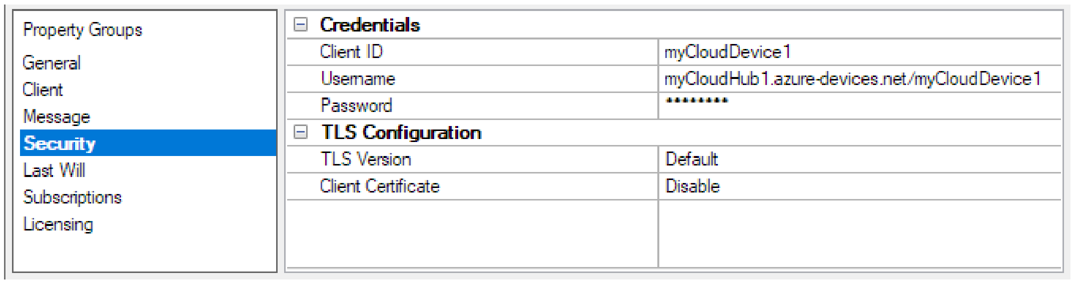
- *For more information about the property bag:, visit* Understand Azure IoT Hub message routing | Azure Storage as a routing endpoint

- *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/
```

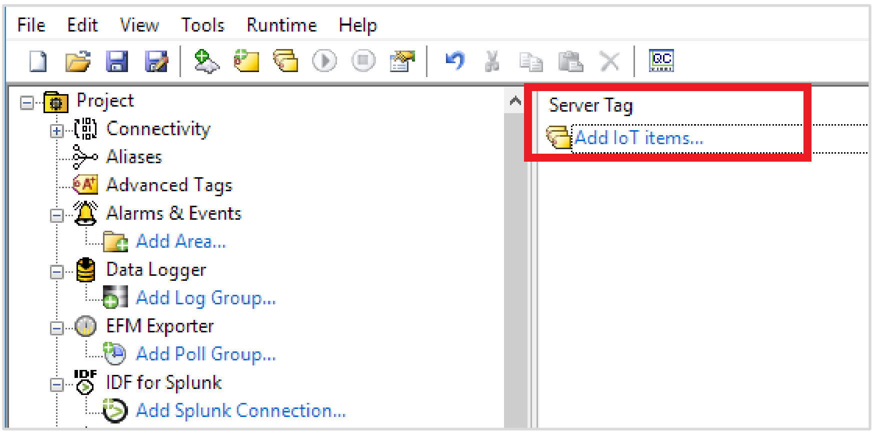
3. Enter security credentials in the following formats:

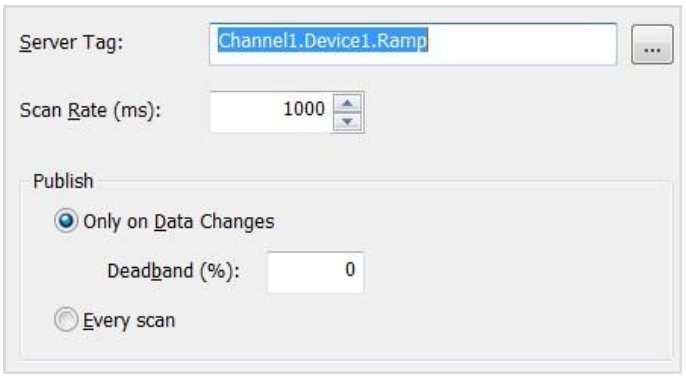


Credentials	
Client ID	myCloudDevice1
Username	myCloudHub1.azure-devices.net/myCloudDevice1
Password	********
TLS Configuration	
TLS Version	Default
Client Certificate	Disable

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

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





Server Tag: Channel1.Device1.Ramp

Scan Rate (ms): 1000

Publish

[✓ Only on Data Changes]

Deadband (%): 0

Every scan

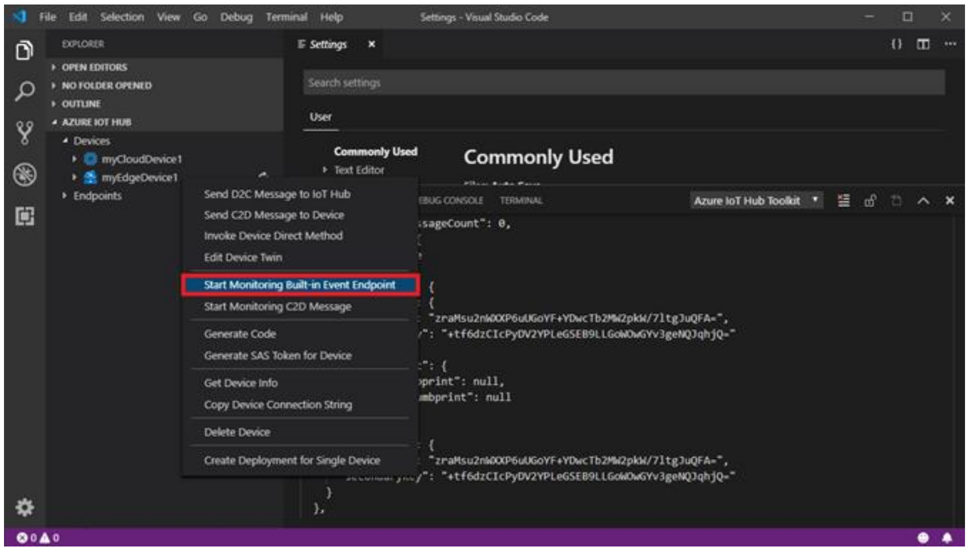
5. To verify that the MQTT Client agent has connected to the Azure IoT Hub, find an event log entry similar to the following:

Date	Time	Level	Event
2/10/2017	2:41:35 PM	Information	MQTT agent 'Ramp_2_Cloud' is connected to broker 'ssl://kepwaremqtt.azure-devices.net:8883'

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

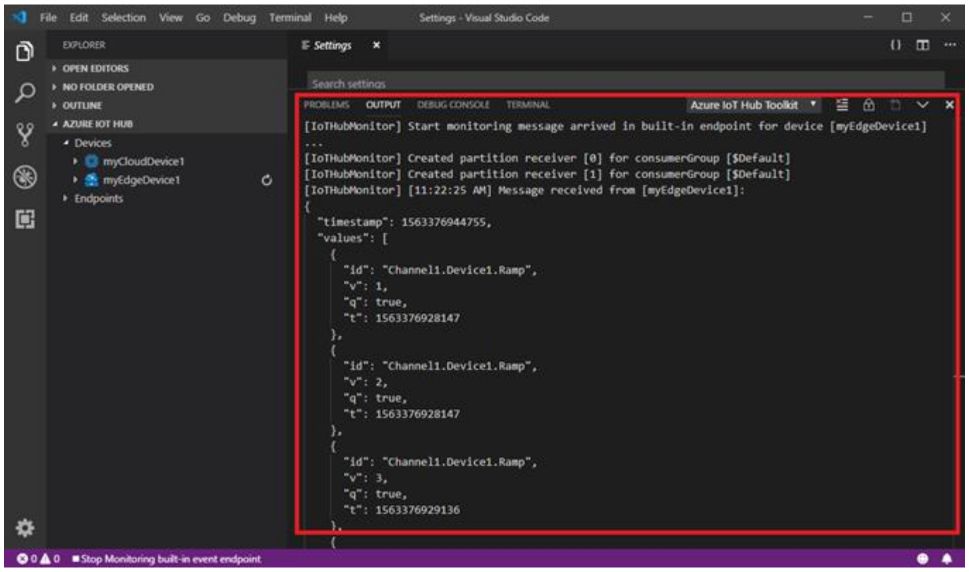
Use Visual Studio Code to monitor the device’s built-in event endpoint and verify that data starts flowing from Kepware’s MQTT Client Agent to the cloud-based Azure IoT Hub within a few seconds.

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


File Edit Selection View Go Debug Terminal Help
Settings - Visual Studio Code
EXPLORER
OPEN EDITORS
NO FOLDER OPENED
OUTLINE
AZURE IOT HUB
Devices
myCloudDevice1
myEdgeDevice1
Endpoints
Settings
Search settings
User
Commonly Used
Text Editor
Commonly Used
Send D2C Message to IoT Hub
Send C2D Message to Device
Invoke Device Direct Method
Edit Device Twin
Start Monitoring Built-in Event Endpoint
Start Monitoring C2D Message
Generate Code
Generate SAS Token for Device
Get Device Info
Copy Device Connection String
Delete Device
Create Deployment for Single Device
DEBUG CONSOLE
TERMINAL
Azure IoT Hub Toolkit

```
sageCount": 0,
{
{
"zraMsu2nWXXP6uUGoYF+YDwcTb2MW2pkW/7ltgJuQFA=",
/": "+tf6dzCIcPyDV2YPLeGSEB9LLGoWOwGYv3geNQJqhjQ="
": {
print": null,
mbprint": null
{
"zraMsu2nWXXP6uUGoYF+YDwcTb2MW2pkW/7ltgJuQFA=",
/": "+tf6dzCIcPyDV2YPLeGSEB9LLGoWOwGYv3geNQJqhjQ="
}
},
```

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



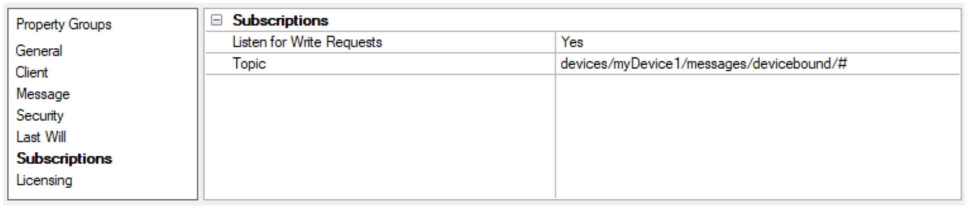
```
[IoTHubMonitor] Start monitoring message arrived in built-in endpoint for device [myEdgeDevice1]
...
[IoTHubMonitor] Created partition receiver [0] for consumerGroup [$Default]
[IoTHubMonitor] Created partition receiver [1] for consumerGroup [$Default]
[IoTHubMonitor] [11:22:25 AM] Message received from [myEdgeDevice1]:
{
  "timestamp": 1563376944755,
  "values": [
    {
      "id": "Channel1.Device1.Ramp",
      "v": 1,
      "q": true,
      "t": 1563376928147
    },
    {
      "id": "Channel1.Device1.Ramp",
      "v": 2,
      "q": true,
      "t": 1563376928147
    },
    {
      "id": "Channel1.Device1.Ramp",
      "v": 3,
      "q": true,
      "t": 1563376929136
    },
    {
```

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:



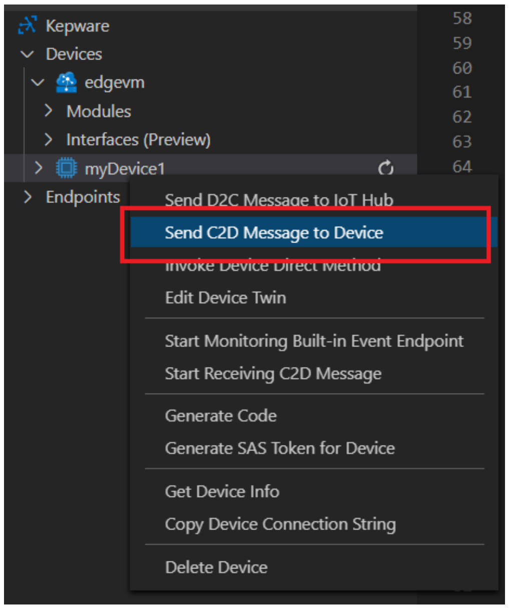
Subscriptions	
Listen for Write Requests	Yes
Topic	devices/myDevice1/messages/devicebound/#

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

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

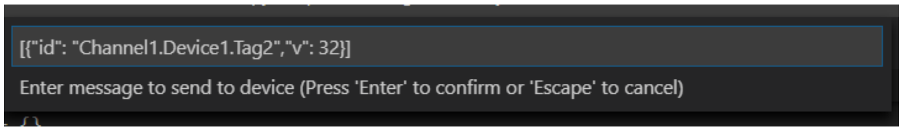
For more information about the cloud to device messages:, visit https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support.

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".



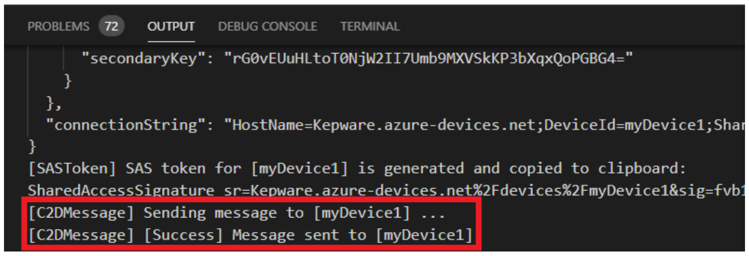
Kepware
Devices
edgevm
Modules
Interfaces (Preview)
myDevice1
Endpoints
Send D2C Message to IoT Hub
Send C2D Message to Device
Invoke Device Direct Method
Edit Device Twin
Start Monitoring Built-in Event Endpoint
Start Receiving C2D Message
Generate Code
Generate SAS Token for Device
Get Device Info
Copy Device Connection String
Delete Device

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



```
[{"id": "Channel1.Device1.Tag2","v": 32}]
Enter message to send to device (Press 'Enter' to confirm or 'Escape' to cancel)
```

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.



```
        "secondaryKey": "rG0vEUuHLtoT0NjW2II7Umb9MXVSkKP3bXqxQoPGBG4="
      }
    },
    "connectionString": "HostName=Kepware.azure-devices.net;DeviceId=myDevice1;Shar
}
[SASToken] SAS token for [myDevice1] is generated and copied to clipboard:
SharedAccessSignature sr=Kepware.azure-devices.net%2Fdevices%2FmyDevice1&sig=fvb1
[C2DMessage] Sending message to [myDevice1] ...
[C2DMessage] [Success] Message sent to [myDevice1]
```

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.

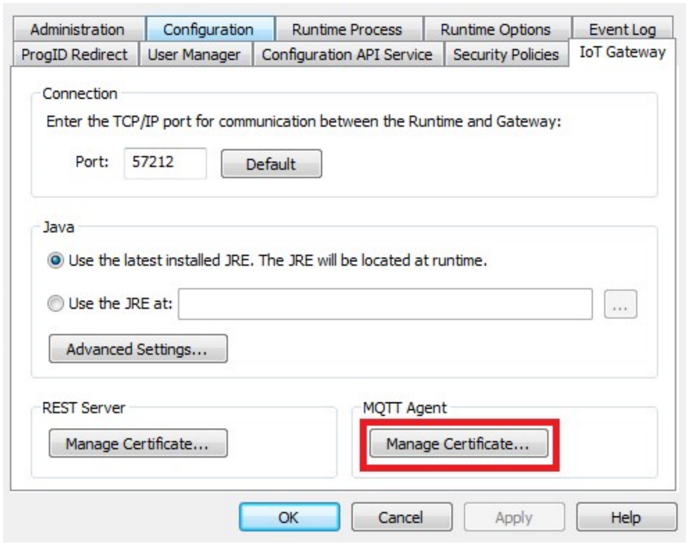
For more information on how to connect with self-signed certificates refer to https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguidesecurity.

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).*



Enter the TCP/IP port for communication between the Runtime and Gateway:

Port: 57212

(●) Use the latest installed JRE. The JRE will be located at runtime.

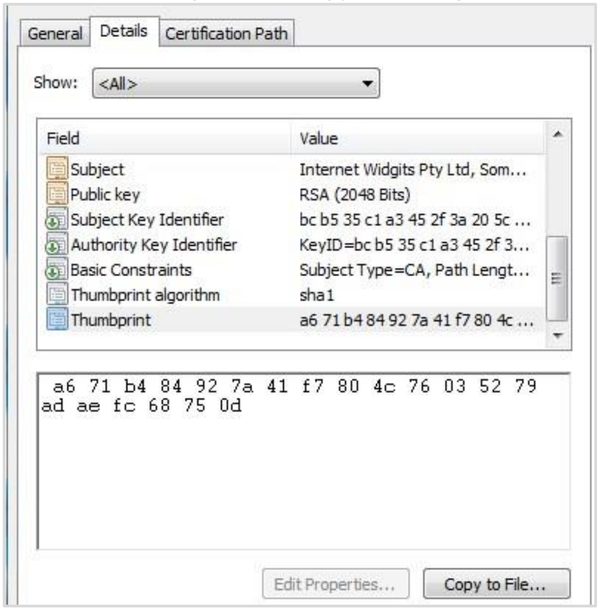
( ) Use the JRE at:

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.



Show: <All>

Field	Value
Subject	Internet Widgits Pty Ltd, Som...
Public key	RSA (2048 Bits)
Subject Key Identifier	bc b5 35 c1 a3 45 2f 3a 20 5c ...
Authority Key Identifier	KeyID=bc b5 35 c1 a3 45 2f 3...
Basic Constraints	Subject Type=CA, Path Lengt...
Thumbprint algorithm	sha1
Thumbprint	a6 71 b4 84 92 7a 41 f7 80 4c ...

```
 a6 71 b4 84 92 7a 41 f7 80 4c 76 03 52 79
ad ae fc 68 75 0d
```

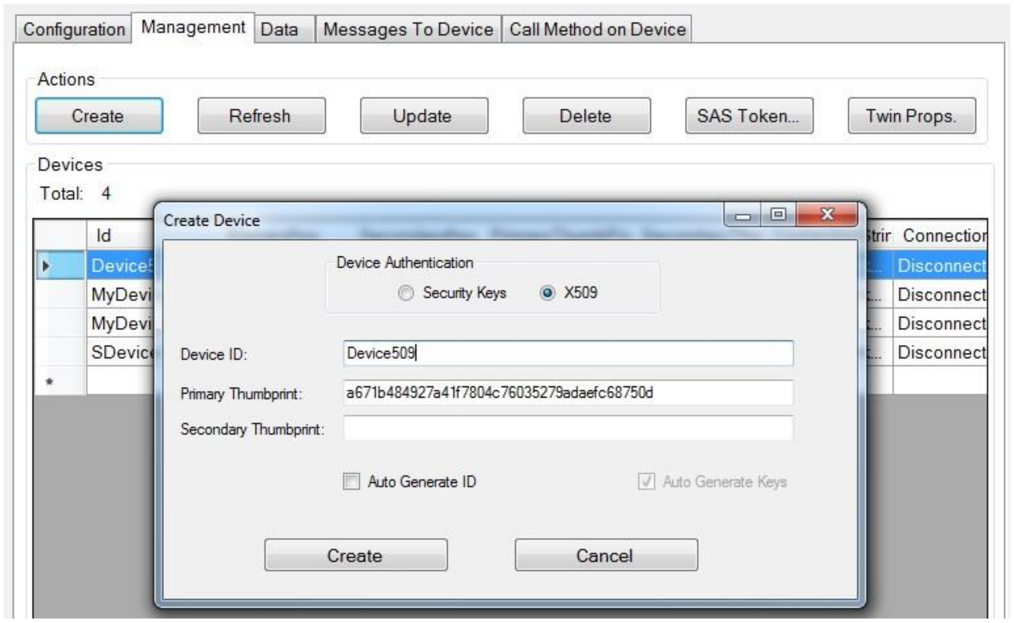
7. Create device using Device Explorer.

   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**.



Create Device

Device Authentication: Security Keys [✓ X509]

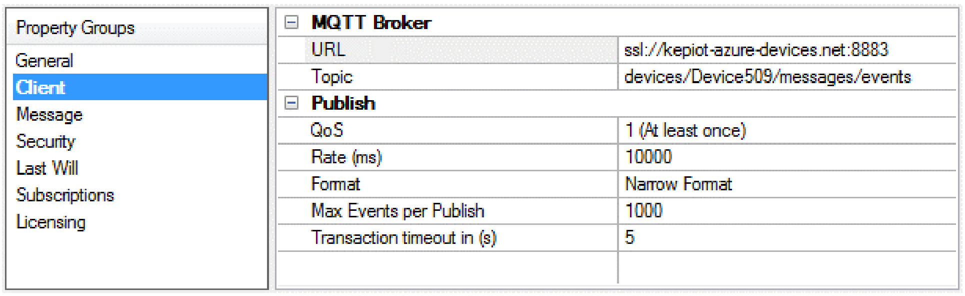
Device ID: Device509

Primary Thumbprint: a671b484927a41f7804c76035279adaefc68750d

Secondary Thumbprint:

Auto Generate ID [ ] Auto Generate Keys [✓]

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



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

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

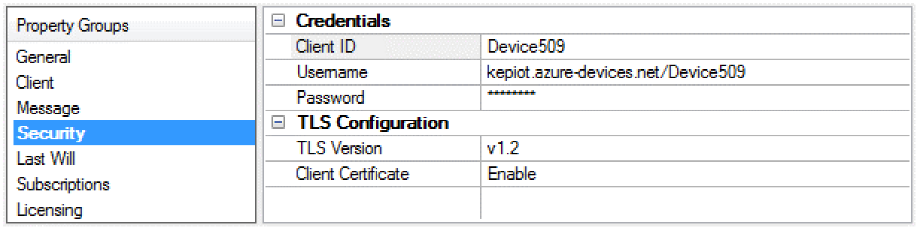
Security	
Client ID	**<deviceID>**
Username	**<IoT Hub Hostname>/<deviceID>**
Password	HostName= **<IoT Hub Hostname>**; DeviceID = **<deviceID>**;x509=true
TLS Version	v1.2
Client Certificate	Enabled

9. Add tags to the MQTT agent. Event log messages should indicate if the agent was successfully connected to the broker.

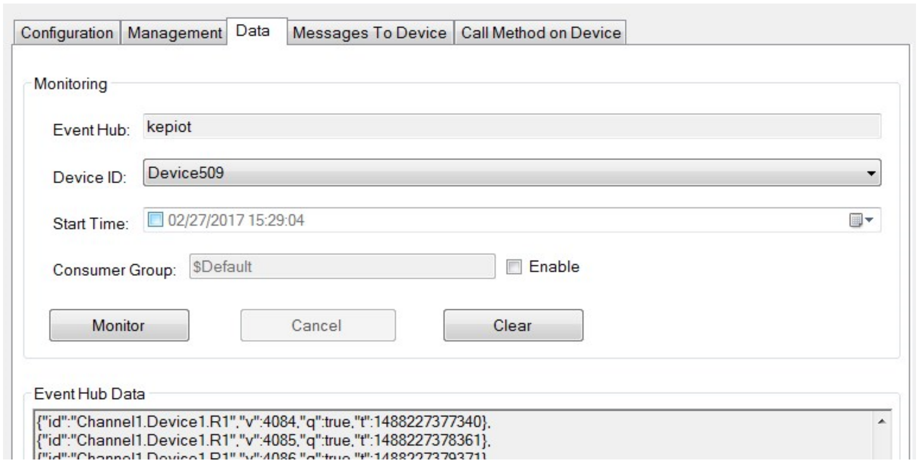
10. To monitor messages from IoT Gateway:

   a Open Device Explorer and access the **Data** tab.

   b Select the device from the drop-down menu and click **Monitor**.



Credentials	
Client ID	Device509
Username	kepiot.azure-devices.net/Device509
Password	********
TLS Configuration	
TLS Version	v1.2
Client Certificate	Enable



Event Hub: kepiot

Device ID: Device509

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

Consumer Group: $Default ☐ Enable

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