AP NOTE
Ver.3

iGS01S/iGS02E Azure IoT Hub Guide

Introduction

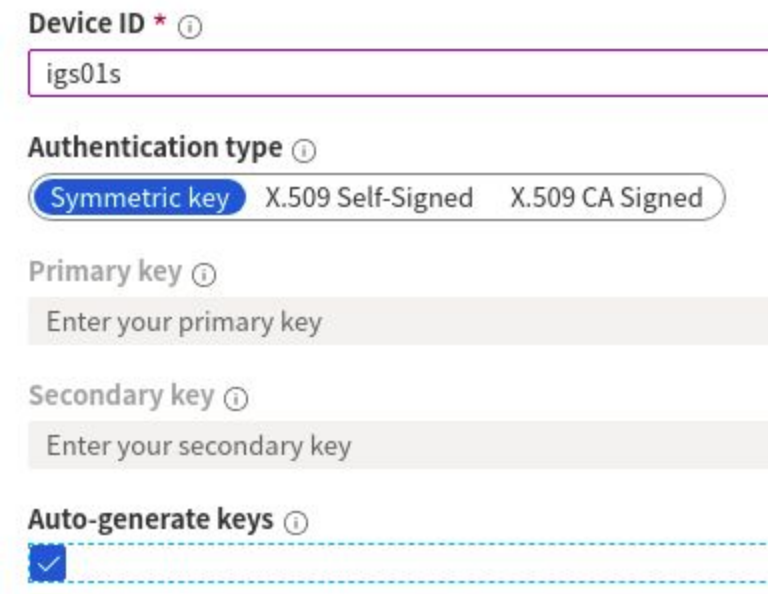
This application note provides a step-by-step setup to connect iGS01S/iGS02E with Azure IoT Hub. Azure-IoT allows Symmetric Key or X.509 Certificates for internal authorization (https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security).
Both should work with IGS01S/iGS02E.

Prerequisites

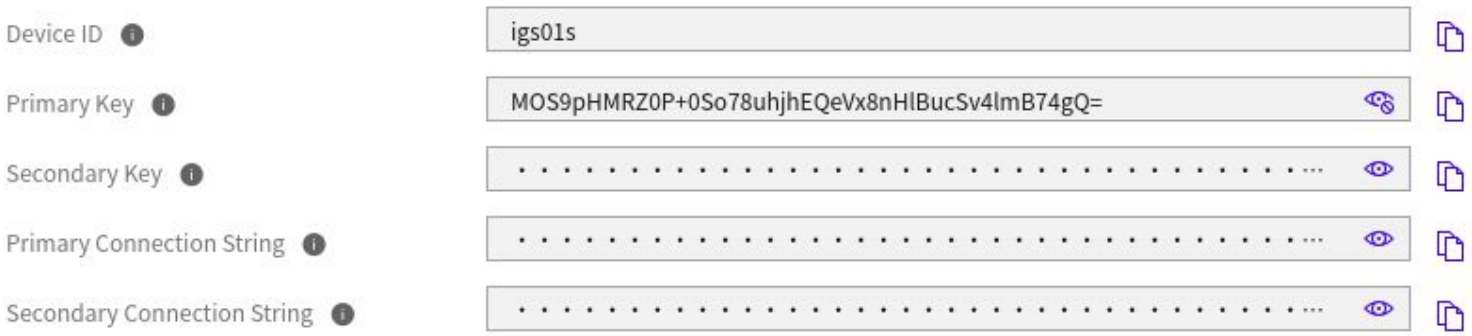
Before using iGS01s/iGS02E with Azure Iot, you have to create an IoT hub with your Azure subscription. See Create an IoT hub for detailed steps.

Example By Using Symmetric Key

IoT Hub can use security tokens to authenticate devices and services to avoid sending keys on the wire. To use a security token, create device with authentication type “Symmetric key”.



After the device is created, you can get the symmetric key in device detail.



Then we need to generate the SAS token from the symmetric key for the MQTT connection. Please refer to Security Token for details about the token structure. And the link alose provides some sample code for how to generate the SAS token for device use.

Or, you can use a tool provided by Intel
https://github.com/intel-iot-devkit/iot-samples-cloud-setup/releases
Usage: sastoken <IoT Hub Name>.azure-devices.net/devices/<Device ID> <Device Primary Key> 1440

Here is the example:

```
$ ./sastoken igs-test.azure-devices.net/devices/igs01s
MOS9pHMRZ0P+0So78uhjhEQeVx8nHlBucSv4lmB74gQ= 1440

SharedAccessSignature
sr=igs-test.azure-devices.net%2Fdevices%2Figs01s&sig=12bF1jJ%2FQWg8XYJPzfne1vWN5HEp02VvIBDEcc1
wx7I%3D&se=1577324521
```

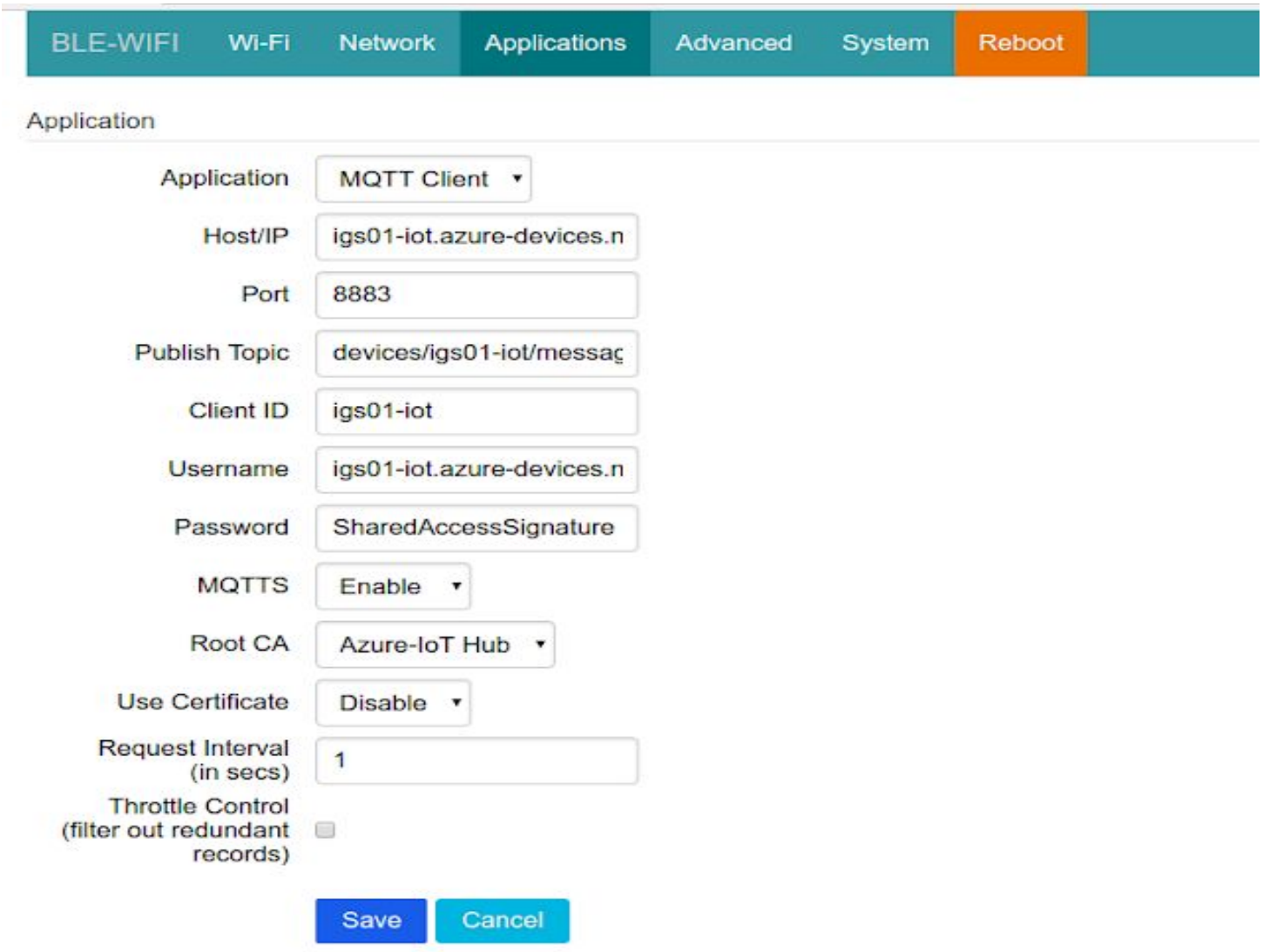
We also suggest users to test your configurations on PC first to confirm your settings are correct.
You can download the mosquitto tool to test connecting Azure IoT Hub with mqtt.
https://mosquitto.org/download/

Below shows an example to publish a “hello” message to Azure IoT using mosquitto command line tool.

```
$ mosquitto_pub -h igs-test.azure-devices.net -p 8883 -i igs01s -t
devices/igs01s/messages/events/ -u igs-test.azure-devices.net/igs01s -P
"SharedAccessSignature
sr=igs-test.azure-devices.net%2Fdevices%2Figs01s&sig=12bF1jJ%2FQWg8XYJPzfne1vWN5HEp02VvIBDEcc1
wx7I%3D&se=1577324521"
--capath /etc/ssl/certs/ --tls-version tlsv1 -d -V mqttv311 -q 0  -m "hello"
```

Then, here is the iGS01S/iGS02E applications configurations:

- MQTT HOST: <IoT Hub Name>.azure-devices.net
- MQTT PORT: 8883
- MQTT PUBTOPIC: devices/<Device ID>/messages/events/
- MQTT CLIENTID: <Device ID>
- MQTT USERNAME: <IoT Hub Name>.azure-devices.net
- MQTT PASSWORD: <SAS Token of the Device>
- Enable MQTTS
- Select Azure-IoT-Hub RootCA
- Disable use certificate



Example By Using X.509 Self-Signed Certificates

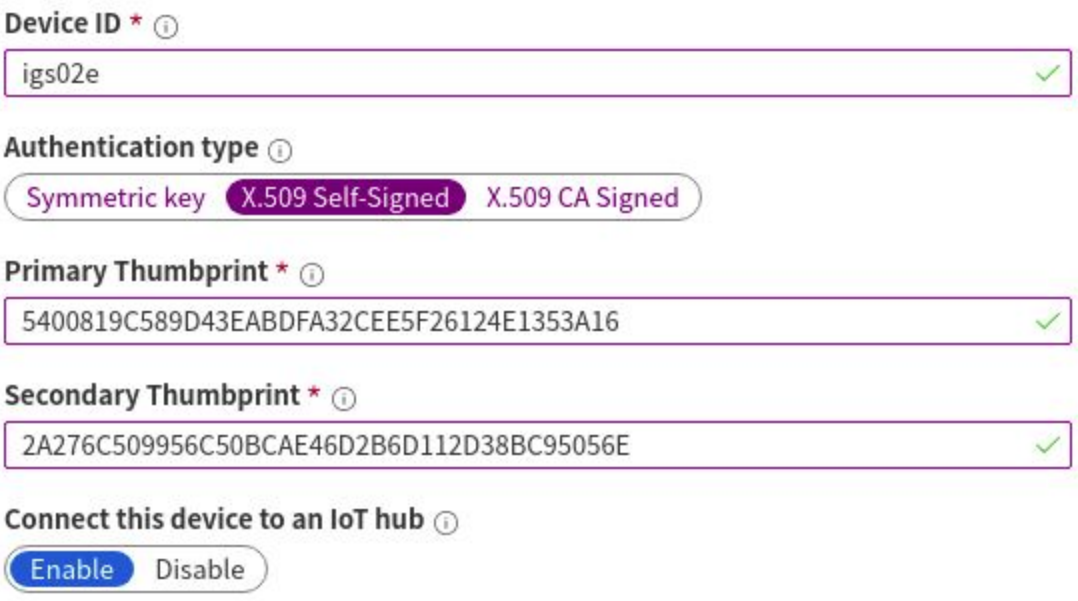
You can create your own self-signed certificates for device authentication.
Azure requires two (primary & secondary) keys for a single IoT self-signed certificate device. Here is the example for creating primary certificate:

```
$ openssl genrsa -out primary.key 2048
$ openssl req -new -key primary.key -sha256 -out primary.csr
$ openssl x509 -req -days 365 -in primary.csr -signkey primary.key -sha256 -out primary.cert
$ openssl x509 -in primary.cert -out primary.cert.pem -outform PEM
$ openssl rsa -in primary.key -out primary.key.pem -outform PEM
```

Repeat the same commands to create the secondary certificates.
And then get the fingerprint of both certificates for creating IoT devices.

```
$ openssl x509 -text -fingerprint -in primary.cert.pem | grep Fingerprint | cut -d '=' -f 2 |
sed 's/[:]//g'
5400819C589D43EABDFA32CEE5F26124E1353A16
$ openssl x509 -text -fingerprint -in secondary.cert.pem | grep Fingerprint | cut -d '=' -f 2
| sed 's/[:]//g'
2A276C509956C50BCAE46D2B6D112D38BC95056E
```

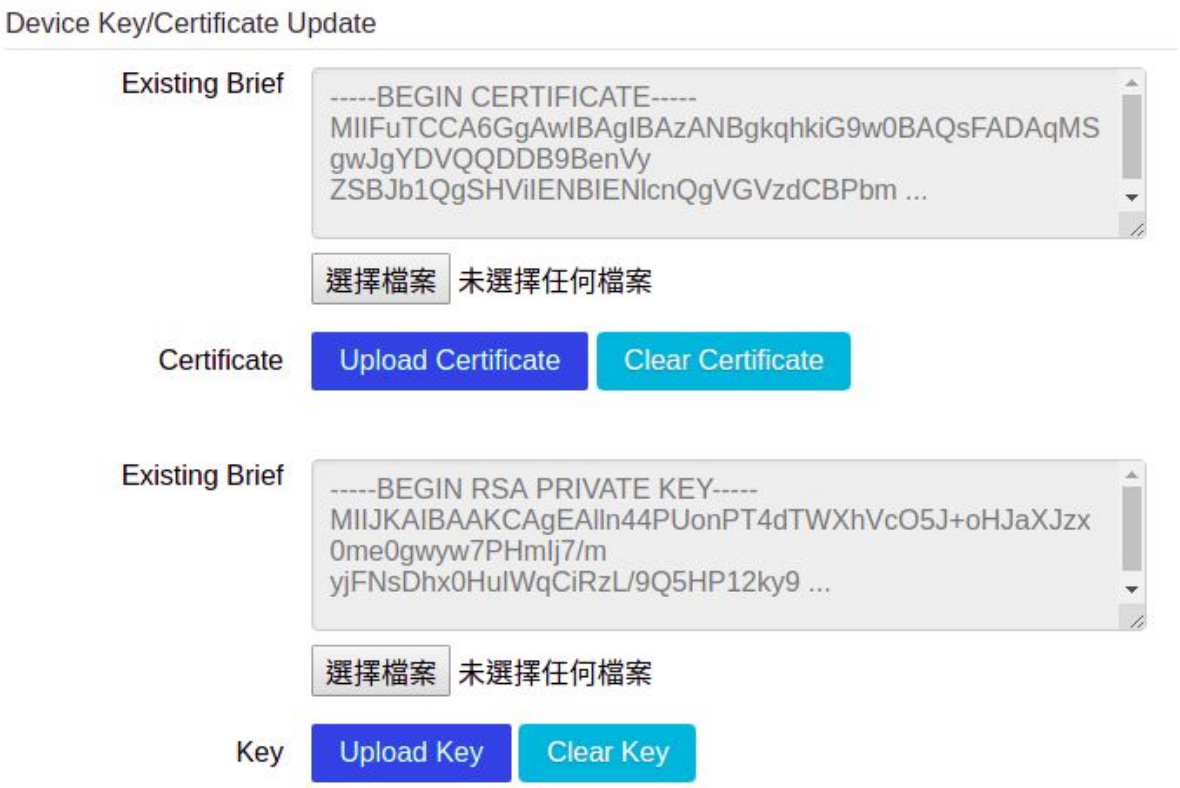
Create an IoT device with the fingerprints.



Use mosquitto tool on PC to verify the Azure configuration. Whatever using the primary key or the secondary key should pass the authentication.

```
$ mosquitto_pub -h <hub_name>.azure-devices.net -p 8883 -t
devices/<device_id>/messages/events/ -i <device_id> --cert primary.cert.pem --key
primary.key.pem -u <hub_name>.azure-devices.net/<device_id> -d -V mqttv311 -q 0 --capath
/etc/ssl/certs/ -m 'hello'
```

Upload the primary.key.pem as key and primary.cert.pem as certificate via webUI advanced page.



The application configurations:

- MQTT HOST: <IoT Hub Name>.azure-devices.net
- MQTT PORT: 8883
- MQTT PUBTOPIC: devices/<Device ID>/messages/events/
- MQTT CLIENTID: <Device ID>
- MQTT USERNAME: <IoT Hub Name>.azure-devices.net
- Enable MQTTS
- Select Azure-IoT-Hub RootCA
- Enable use certificate

Application

Application	MQTT Client
Host/IP	igs-test.azure-devices.net
Port	8883
Publish Topic	devices/igs01s/messages/ev
Client ID	igs01s
Username	igs-test.azure-devices.net/ig
Password	password
MQTTS	Enable
Root CA	Azure-IoT Hub
Use Certificate	Enable
Format Type	plain-text
Request Interval (in secs)	1
Drop reports while cache full	☑
Throttle Control (filter out redundant records)	☑

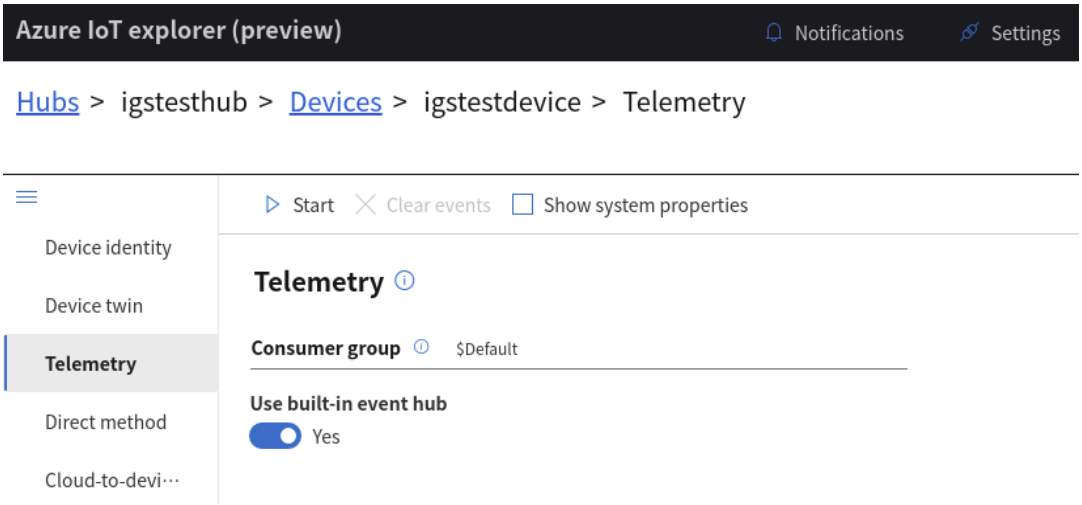
Save Cancel

Verify Configuration

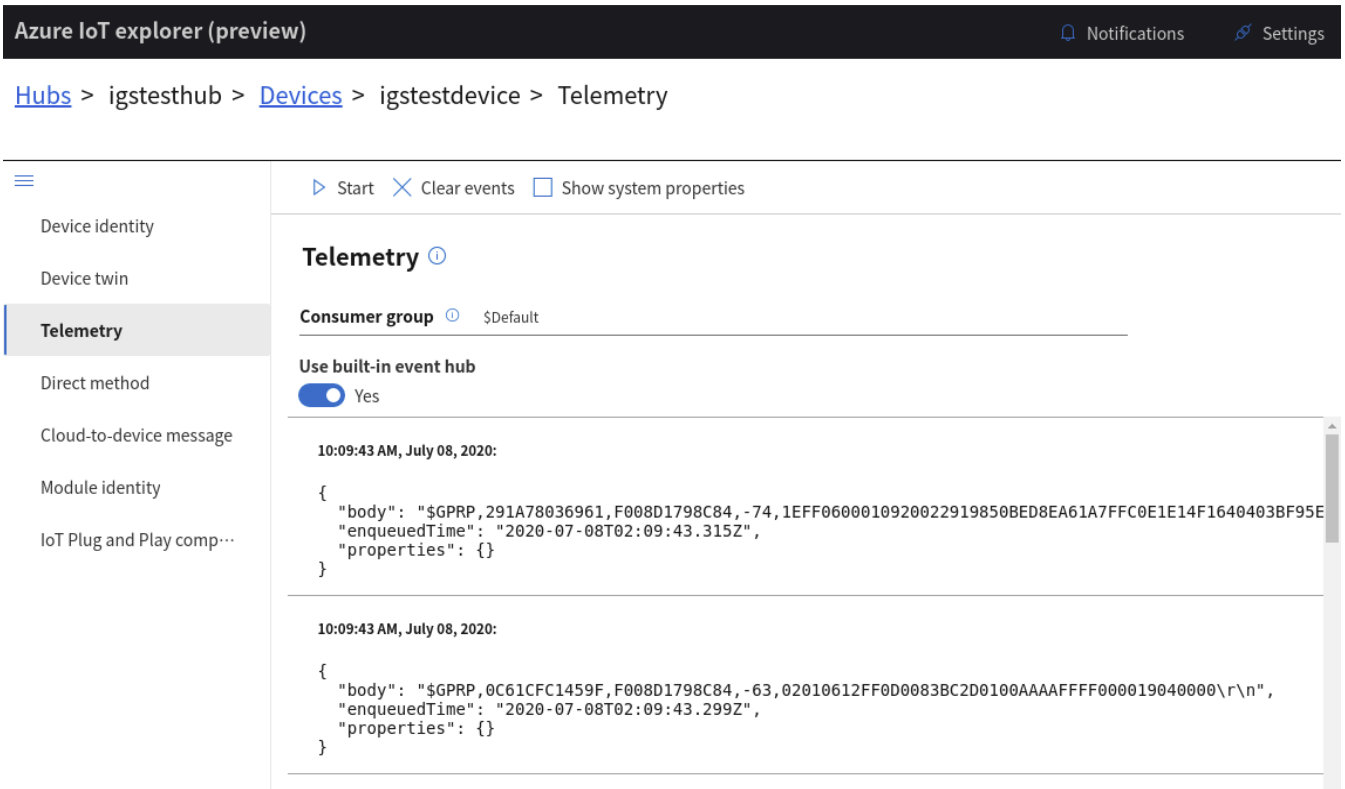
Use Azure Iot Explorer to verify if the configuration works fine.
https://docs.microsoft.com/zh-tw/azure/iot-pnp/howto-install-iot-explorer

Use default EventHub to check the messages from devices to Cloud.

1. Login use IoT Hub connect string
2. Choice your IoT Hub -> IoT device -> Telemetry
   Make sure "Use built-in event hub" is YES, click "Start" button to monitor the events



3. If everything works fine, the message will show up



Revision History

DATE	REVISION	CHANGES
Feb 11, 2019	1	Initial release
Mar 28, 2019	2	Update reference documents
Dec 24, 2019	3	Update example for using X-509 certificates