BC66&BC66-NA MQTT Application Note

NB-IoT Module Series

Rev. BC66&BC66-NA_MQTT_Application_Note_V2.0

Date: 2020-03-20

Status: Released

www.quectel.com

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

THE INFORMATION CONTAINED HERE IS PROPRIETARY TECHNICAL INFORMATION OF QUECTEL WIRELESS SOLUTIONS CO., LTD. TRANSMITTING, REPRODUCTION, DISSEMINATION AND EDITING OF THIS DOCUMENT AS WELL AS UTILIZATION OF THE CONTENT ARE FORBIDDEN WITHOUT PERMISSION. OFFENDERS WILL BE HELD LIABLE FOR PAYMENT OF DAMAGES. ALL RIGHTS ARE RESERVED IN THE EVENT OF A PATENT GRANT OR REGISTRATION OF A UTILITY MODEL OR DESIGN.

Copyright © Quectel Wireless Solutions Co., Ltd. 2020. All rights reserved.

About the Document

Revision History

Version	Date	Author	Description
1.0	2018-08-28	Louis GU	Initial
2.0	2020-03-20	Taber JIANG	1. Added AT+QMTCFG="ssl" command. 2. Added descriptions to illustrate how to establish connections with Azure IoT Hub and AWS IoT Core (Chapter 3 and Chapter 4).

Contents

Table Index

Figure Index

1 Introduction

MQTT is a broker-based publish/subscribe messaging protocol designed to be open, simple, lightweight and easy to implement. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited.

This document mainly introduces how to use the MQTT function of Quectel BC66 and BC66-NA modules through AT commands.

1.1. MQTT Data Interaction

This chapter gives the data interaction mechanism of MQTT function.

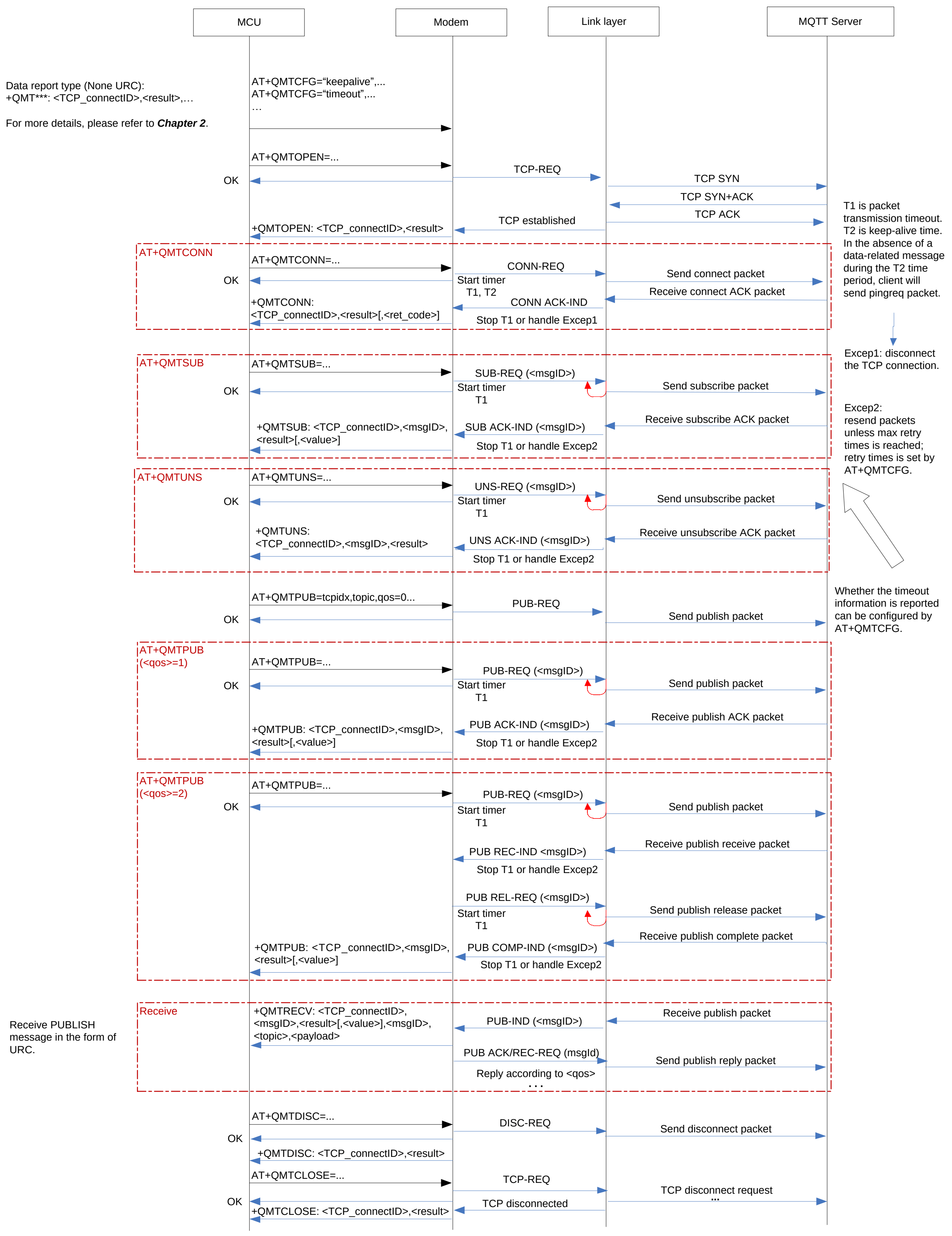

MCU
Modem
Link layer
MQTT Server
Data report type (None URC):
+QMT***: <TCP_connectID>,<result>,…
For more details, please refer to **Chapter 2**.
AT+QMTCFG="keepalive",...
AT+QMTCFG="timeout",...
…
AT+QMTOPEN=...
OK
TCP-REQ
TCP SYN
TCP SYN+ACK
TCP ACK
+QMTOPEN: <TCP_connectID>,<result>
TCP established
T1 is packet transmission timeout. T2 is keep-alive time. In the absence of a data-related message during the T2 time period, client will send pingreq packet.
AT+QMTCONN
AT+QMTCONN=...
OK
CONN-REQ
Start timer
T1, T2
Send connect packet
Receive connect ACK packet
+QMTCONN:
<TCP_connectID>,<result>[,<ret_code>]
CONN ACK-IND
Stop T1 or handle Excep1
Excep1: disconnect the TCP connection.
AT+QMTSUB
AT+QMTSUB=...
OK
SUB-REQ (<msgID>)
Start timer
T1
Send subscribe packet
+QMTSUB: <TCP_connectID>,<msgID>,
<result>[,<value>]
SUB ACK-IND (<msgID>)
Receive subscribe ACK packet
Stop T1 or handle Excep2
Excep2: resend packets unless max retry times is reached; retry times is set by AT+QMTCFG.
AT+QMTUNS
AT+QMTUNS=...
OK
UNS-REQ (<msgID>)
Start timer
T1
Send unsubscribe packet
+QMTUNS:
<TCP_connectID>,<msgID>,<result>
UNS ACK-IND (<msgID>)
Receive unsubscribe ACK packet
Stop T1 or handle Excep2
AT+QMTPUB=tcpidx,topic,qos=0...
OK
PUB-REQ
Send publish packet
Whether the timeout information is reported can be configured by AT+QMTCFG.
AT+QMTPUB
(<qos>=1)
AT+QMTPUB=...
OK
PUB-REQ (<msgID>)
Start timer
T1
Send publish packet
Receive publish ACK packet
+QMTPUB: <TCP_connectID>,<msgID>,
<result>[,<value>]
PUB ACK-IND (<msgID>)
Stop T1 or handle Excep2
AT+QMTPUB
(<qos>=2)
AT+QMTPUB=...
OK
PUB-REQ (<msgID>)
Start timer
T1
Send publish packet
PUB REC-IND <msgID>)
Receive publish receive packet
Stop T1 or handle Excep2
PUB REL-REQ (<msgID>)
Start timer
T1
Send publish release packet
Receive publish complete packet
+QMTPUB: <TCP_connectID>,<msgID>,
<result>[,<value>]
PUB COMP-IND (<msgID>)
Stop T1 or handle Excep2
Receive
Receive PUBLISH message in the form of URC.
+QMTRECV: <TCP_connectID>,
<msgID>,<result>[,<value>],<msgID>,
<topic>,<payload>
PUB-IND (<msgID>)
Receive publish packet
PUB ACK/REC-REQ (msgId)
Send publish reply packet
Reply according to <qos>
…
AT+QMTDISC=...
OK
DISC-REQ
Send disconnect packet
+QMTDISC: <TCP_connectID>,<result>
AT+QMTCLOSE=...
OK
TCP-REQ
TCP disconnect request
…
+QMTCLOSE: <TCP_connectID>,<result>
TCP disconnected

Figure 1: MQTT Data Interaction Mechanism

2 MQTT Related AT Commands

2.1. Definitions

● **<...>**	Parameter name. Angle brackets do not appear on command line.
● **[...]**	Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on command line. When an optional parameter is not given, the new value equals to its previous value or its default setting, unless otherwise specified.
● **Underline**	Default setting of a parameter.

2.2. Types of AT Commands

Table 1: Types of AT Commands and Responses

Test Command	**AT+<cmd>=?**	This command returns the list of parameters and value ranges set by the corresponding Write Command or internal processes.
Read Command	**AT+<cmd>?**	This command returns the currently set value of the parameter or parameters.
Write Command	**AT+<cmd>=<p1>[,<p2>[,<p3>[...]]]**	This command sets the user-definable parameter values.
Execution Command	**AT+<cmd>**	This command reads non-variable parameters affected by internal processes in the module.

2.3. Description of MQTT AT Commands

2.3.1. AT+QMTCFG Configure Optional Parameters of MQTT

This command is used to configure optional parameters of MQTT.

AT+QMTCFG Configure Optional Parameters of MQTT	
Test Command **AT+QMTCFG=?**	Response **+QMTCFG: "dataformat",(**range of supported **<TCP_connectID>**s**),(**list of supported **<send_format>**s**),(**list of supported **<recv_format>**s) **+QMTCFG: "keepalive",(**range of supported **<TCP_connectID>**s**),(**range of supported **<keep-alive_time>**s) **+QMTCFG: "session",(**range of supported **<TCP_connectID>**s**),(**list of supported **<clean_session>**s) **+QMTCFG: "timeout",(**range of supported **<TCP_connectID>**s**),(**range of supported **<pkt_timeout>**s**),(**range of supported **<retry_times>**s**),(**list of supported **<timeout_notice>**s**)** **+QMTCFG: "will",(**range of supported **<TCP_connectID>**s**),(**list of supported **<will_fg>**s**),(**range of supported **<will_qos>**s**),(**list of supported **<will_retain>**s**),"will_topic","will_msg"** **+QMTCFG: "version",(**range of supported **<TCP_connectID>**s**),(**list of supported **<version>**s) **+QMTCFG: "aliauth",(**range of supported **<TCP_connectID>**s**),"productkey","devicename","devicesecret"** **+QMTCFG: "echomode",(**range of supported **<TCP_connectID>**s**),(**list of supported **<echo_mode>**s) **+QMTCFG: "ssl",(**range of supported **<TCP_connectID>s)[,(**list of supported **<SSL_enable>s)[,(**range of supported **<SSL_contextID>s),(**range of supported **<SSL_connectID>s)]]** **OK**
Write Command Configure the format of sent/received data **AT+QMTCFG="dataformat",<TCP_connectID>,[<send_format>,<recv_format>]**	Response If the parameters **<send_format>** and **<recv_format>** are both omitted, query the format of sent/received data: **+QMTCFG: "dataformat",<send_format>,<recv_format>** **OK** If the parameters **<send_format>** and **<recv_format>** are both specified, configure the format of data to be sent/received: **OK** If there is any error: **ERROR**

Write Command Configure the keep-alive time **AT+QMTCFG="keepalive",<TCP_connectID>[,<keep-alive_time>]**	Response If the parameter **<keep-alive_time>** is omitted, query the keep-alive time: **+QMTCFG: "keepalive",<keep-alive_time>** **OK** If the parameter **<keep-alive_time>** is specified, configure the keep-alive time: **OK** If there is any error: **ERROR**
Write Command Configure the session type **AT+QMTCFG="session",<TCP_connectID>[,<clean_session>]**	Response If the parameter **<clean_session>** is omitted, query the session type: **+QMTCFG: "session",<clean_session>** **OK** If the parameter **<clean_session>** is specified, configure the session type: **OK** If there is any error: **ERROR**
Write Command Configure timeout of message delivery **AT+QMTCFG="timeout",<TCP_connectID>[,<pkt_timeout>,<retry_times>,<timeout_notice>]**	Response If the parameters **<pkt_timeout>**, **<retry_times>** and **<timeout_notice>** are omitted, query the timeout value of message delivery: **+QMTCFG: "timeout",<pkt_timeout>,<retry_times>,<timeout_notice>** **OK** If the parameters **<pkt_timeout>**, **<retry_times>** and **<timeout_notice>** are entered, configure the timeout of message delivery: **OK** If there is any error: **ERROR**
Write Command Configure Will information **AT+QMTCFG="will",<TCP_connectI**	Response If the parameters **<will_fg>**, **<will_qos>**, **<will_retain>**, **"<will_topic>"** and **"<will_msg>"** are omitted, query the Will

D>[,<will_fg>,<will_qos>,<will_retain>,"<will_topic>","<will_msg>"]	information: **+QMTCFG: "will",<will_fg>[,<will_qos>,<will_retain>,"<will_topic>","<will_msg>"]** **OK** If the parameters **<will_fg>**, **<will_qos>**, **<will_retain>**, **"<will_topic>"** and **"<will_msg>"** are specified, configure the Will information: **OK** If there is any error: **ERROR**
Write Command Configure the MQTT protocol version to be used **AT+QMTCFG="version",<TCP_connectID>[,<version>]**	Response If the parameter **<version>** is omitted, query the MQTT protocol version: **+QMTCFG: "version",<version>** **OK** If the parameter **<version>** is specified, configure the MQTT protocol version: **OK** If there is any error: **ERROR**
Write Command Configure the device information for AliCloud **AT+QMTCFG="aliauth",<TCP_connectID>[,"<product_key>","<device_name>","<device_secret>"]**	Response If the parameters **"<product_key>"**, **"<device_name>"** and **"<device_secret>"** are omitted, query the device information: **[+QMTCFG: "aliauth","<product_key>","<device_name>","<device_secret>"]** **OK** If the parameters **"<product_key>"**, **"<device_name>"** and **"<device_secret>"** are specified, configure the device information for AliCould: **OK** If there is any error: **ERROR**
Write Command Configure whether to echo the input data to UART in data mode	Response If the parameter **<echo_mode>** is omitted, query the data echo mode:

AT+QMTCFG="echomode",<TCP_connectID>[,<echo_mode>]	**+QMTCFG: "echomode",<echo_mode>** **OK** If the parameter **<echo_mode>** is specified, configure whether to echo the input data to UART in data mode: **OK** If there is any error: **ERROR**
Write Command Configure whether to use SSL secure connection **AT+QMTCFG="ssl",<TCP_connectID>[,<SSL_enable>,<SSL_contextID>,<SSL_connectID>]**	Response If the parameters **<SSL_enable>**, **<SSL_contextID>** and **<SSL_connectID>** are omitted, query the current configuration: **+QMTCFG: "ssl",<SSL_enable>[,<SSL_contextID>,<SSL_connectID>]** **OK** If the parameters **<SSL_enable>**, **<SSL_contextID>** and **<SSL_connectID>** are specified, configure whether to use SSL secure connection: **OK** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	The command takes effect immediately. Invalid after deep-sleep wakeup. The configurations will not be saved to NVRAM.

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<will_fg>	Integer type. The Will flag. 0 Ignore the Will flag configuration 1 Require the Will flag configuration
<will_qos>	Integer type. Quality of service for message delivery 0 At most once 1 At least once 2 Exactly once
<will_retain>	Integer type. The Will retain flag is only used on PUBLISH messages. 0 When a client sends a PUBLISH message to a server, the server will not hold

	on to the message after it has been delivered to the current subscribers.
	1 When a client sends a PUBLISH message to a server, the server should hold on to the message after it has been delivered to the current subscribers.
<will_topic>	String type. Will topic string. The maximum size is 255 bytes.
<will_msg>	String type. The Will message defines the content of the message that is published to the will topic if the client is unexpectedly disconnected. It can be a zero-length message. The maximum size is 255 bytes.
<pkt_timeout>	Integer type. Timeout of the packet delivery. The range is 1-60. The default value is 10. Unit: second.
<retry_times>	Integer type. Retry times when packet delivery times out. The range is 0-10. The default value is 3.
<timeout_notice>	Integer type. Whether to report timeout message when transmitting packet.
	0 Not report
	1 Report
<clean_session>	Integer type. The session type.
	0 The server must store the subscriptions of the client after it is disconnected.
	1 The server must discard any previously maintained information about the client and treat the connection as "clean".
<keep-alive_time>	Integer type. Keep-alive time. The range is 0-3600. The default value is 120. Unit: second. It defines the maximum time interval between messages received from a client. If the server does not receive a message from the client within 1.5 times of the keep-alive time period, it disconnects the client as if the client has sent a DISCONNECT message.
	0 The client is not disconnected
<product_key>	String type. Product key issued by AliCloud. The maximum size is 64 bytes.
<device_name>	String type. Device name issued by AliCloud. The maximum size is 64 bytes.
<device_secret>	String type. Device secret key issued by AliCloud. The maximum size is 64 bytes.
<version>	Integer type. The version of MQTT protocol.
	3 MQTT v3.1
	4 MQTT v3.1.1
<send_format>	Integer type. The format of sent data.
	0 Text format
	1 Hex format
<recv_format>	Integer type. The format of received data.
	0 Text format
	1 Hex format
<echo_mode>	Integer type. Whether to echo the input data to UART in data mode.
	0 Do not echo the input data to UART
	1 Echo the input data to UART
<SSL_enable>	Integer type. Configure whether to use SSL secure connection for MQTT.
	0 Use normal TCP connection for MQTT
	1 Use SSL/TLS TCP secure connection for MQTT
<SSL_contextID>	Integer type. SSL context index. The range is 1-3.

<SSL_connectID>	Integer type. SSL connect index. The range is 0-5.

NOTES

1. **<clean_session>**=0 is effective only when the server supports the "clean" operation.
2. Care must be taken to ensure message delivery does not time out while it is still being sent.
3. **AT+QMTCFG="aliauth"** command is only used for AliCloud. If it is configured, the parameters **<username>** and **<password>** in command **AT+QMTCONN** can be omitted.
4. **<echo_mode>** is only valid in data mode transferring.
5. When **<SSL_enable>**=1, parameters **<SSL_contextID>** and **<SSL_connectID>** must be specified, and then the SSL/TLS connection configurations must be set by **AT+QSSLCFG**. For details of the command, please refer to *Quectel_BC66&BC66-NA_SSL_Application_Note*.
6. The settings of **"will"**, **"session"**, **"keepalive"**, **"aliauth"**, **"version"** and **"ssl"** have to be configured before the executing of **AT+QMTOPEN**.

2.3.2. AT+QMTOPEN Open a Network for MQTT Client

This command is used to open a network for MQTT client.

AT+QMTOPEN Open a Network for MQTT Client	
Test Command **AT+QMTOPEN=?**	Response **+QMTOPEN: (**range of supported **<TCP_connectID>**s**),"<host_name>",<port>** **OK**
Read Command **AT+QMTOPEN?**	Response **[+QMTOPEN: <TCP_connectID>,"<host_name>",<port>]** **OK**
Write Command **AT+QMTOPEN=<TCP_connectID>,"<host_name>",<port>**	Response **OK** **+QMTOPEN: <TCP_connectID>,<result>** If there is any error: **ERROR**
Maximum Response Time	75 s, determined by network
Characteristics	/

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<host_name>	String type. The address of the server. It could be an IP address or a domain name. The maximum size is 150 bytes.
<port>	Integer type. The port number of the server. The range is 1-65535.
<result>	Integer type. Result of the command execution. -1 Failed to open network 0 Network opened successfully

2.3.3. AT+QMTCLOSE Close a Network for MQTT Client

This command is used to close a network for MQTT client.

AT+QMTCLOSE Close a Network for MQTT Client	
Test Command **AT+QMTCLOSE=?**	Response **+QMTCLOSE: (**range of supported **<TCP_connectID>**s**)** **OK**
Write Command **AT+QMTCLOSE=<TCP_connectID>**	Response **OK** **+QMTCLOSE: <TCP_connectID>,<result>** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<result>	Integer type. Result of the command execution. -1 Failed to close the network 0 Network closed successfully

2.3.4. AT+QMTCONN Connect a Client to MQTT Server

This command is used when a client requests a connection to MQTT server. When a TCP/IP socket connection is established from a client to a server, a protocol level session must be created using a CONNECT flow.

AT+QMTCONN Connect a Client to MQTT Server	
Test Command **AT+QMTCONN=?**	Response **+QMTCONN: (**range of supported **<TCP_connectID>**s)**,"<clientID>"[,"<username>"[,"<password>"]]** **OK**
Read Command **AT+QMTCONN?**	Response **[+QMTCONN: <TCP_connectID>,<state>]** **OK**
Write Command **AT+QMTCONN=<TCP_connectID>,"<clientID>"[,"<username>"[,"<password>"]]**	Response **OK** **+QMTCONN: <TCP_connectID>,<result>[,<ret_code>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** (default 10 s), determined by network
Characteristics	**/**

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<clientID>	String type. The client identifier. The max length is 128 bytes.
<username>	String type. User name of the client. It can be used for authentication. The max length is 256 bytes.
<password>	String type. Password corresponding to the user name of the client. It can be used for authentication. The max length is 256 bytes.
<result>	Integer type. Result of the command execution 0 Sent the packet successfully and received ACK from server 1 Packet retransmission 2 Failed to send packet
<state>	Integer type. The MQTT connection state. 1 MQTT is initial 2 MQTT is connecting 3 MQTT is connected 4 MQTT is disconnecting
<ret_code>	Integer type. Connection return code. 0 Connection Accepted 1 Connection Refused: Unacceptable Protocol Version 2 Connection Refused: Identifier Rejected

	3 Connection Refused: Server Unavailable 4 Connection Refused: Bad User Name or Password 5 Connection Refused: Not Authorized
<pkt_timeout>	Integer type. Timeout of the packet delivery. The range is 1-60. The default value is 10. Unit: second.

NOTE

If a client with the same Client ID is already connected to the server, the "older" client will be disconnected by the server automatically after completing the CONNECT flow of the new client.

2.3.5. AT+QMTDISC Disconnect a Client from MQTT Server

This command is used when a client requests a disconnection from MQTT server. A DISCONNECT message is sent from the client to the server to indicate that it is about to close its TCP/IP connection.

AT+QMTDISC Disconnect a Client from MQTT Server	
Test Command **AT+QMTDISC=?**	Response **+QMTDISC: (**range of supported **<TCP_connectID>**s**)** **OK**
Write Command **AT+QMTDISC=<TCP_connectID>**	Response **OK** **+QMTDISC: <TCP_connectID>,<result>** If there is any error: **ERROR**
Maximum Response Time	300 ms
Characteristics	/

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<result>	Integer type. Result of the command execution. -1 Failed to disconnect the client 0 Client disconnected successfully

2.3.6. AT+QMTSUB Subscribe to Topics

This command is used to subscribe to one or more topics. A SUBSCRIBE message is sent by a client to register an interest in one or more topic names with the server. Messages published to these topics are delivered from the server to the client as PUBLISH messages.

AT+QMTSUB Subscribe to Topics	
Test Command AT+QMTSUB=?	Response **+QMTSUB: (**range of supported **<TCP_connectID>s),<msgID>,"<topic>",<qos>[,"<topic>",qos>...]** **OK**
Write Command **AT+QMTSUB=<TCP_connectID>,<msgID>,"<topic1>",<qos1>[,"<topic2>",<qos2>…]**	Response **OK** **+QMTSUB: <TCP_connectID>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** + **<pkt_timeout>** × **<retry_times>** (default 40 s), determined by network
Characteristics	/

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<msgID>	Integer type. Message identifier of packet. The range is 1-65535.
<topic>	String type. Topic that the client wants to subscribe to or unsubscribe from. The maximum length is 255 bytes.
<qos>	Integer type. The QoS level at which the client wants to publish the messages. 0 At most once 1 At least once 2 Exactly once
<result>	Integer type. Result of the command execution. 0 Sent the packet successfully and received ACK from server 1 Packet retransmission 2 Failed to send packet
<value>	If **<result>** is 0, it is a vector of granted QoS levels. The value 128 indicates that the subscription has been rejected by the server. If **<result>** is 1, it means the times of packet retransmission. If **<result>** is 2, it will not be presented.
<pkt_timeout>	Integer type. Timeout of the packet delivery. The range is 1-60. The default value is 10. Unit: second.

<retry_times>	Integer type. Retry times when packet delivery times out. The range is 0-10. The default value is 3.

NOTE

The **<msgID>** is only present in messages where the QoS bits in the fixed header indicate QoS levels 1 or 2. It must be unique amongst the set of "in flight" messages in a particular direction of communication. It typically increases by exactly one from one message to the next, but is not required to do so.

2.3.7. AT+QMTUNS Unsubscribe from Topics

This command is used to unsubscribe from one or more topics. An UNSUBSCRIBE message is sent by the client to the server to unsubscribe from named topics.

AT+QMTUNS Unsubscribe from Topics	
Test Command **AT+QMTUNS=?**	Response **+QMTUNS: (**range of supported **<TCP_connectID>s),<msgID>,"<topic>"[,"<topic>"...]** **OK**
Write Command **AT+QMTUNS=<TCP_connectID>,<msgID>,"<topic1>"[,"<topic2>"…]**	Response **OK** **+QMTUNS: <TCP_connectID>,<msgID>,<result>** If there is any error: **ERROR**
Maximum Response Time	**<pkt_timeout>** + **<pkt_timeout>** × **<retry_times>** (default 40 s), determined by network
Characteristics	/

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<msgID>	Integer type. Message identifier of packet. The range is 1-65535.
<topic>	String type. Topic that the client wants to subscribe to or unsubscribe from. The maximum length is 255 bytes.
<result>	Integer type. Result of the command execution. 0 Sent the packet successfully and received ACK from server 1 Packet retransmission 2 Failed to send packet
<pkt_timeout>	Integer type. Timeout of the packet delivery. The range is 1-60. The default value

	is 10. Unit: second.
<retry_times>	Integer type. Retry times when packet delivery times out. The range is 0-10. The default value is 3.

2.3.8. AT+QMTPUB Publish Messages

This command is used to publish messages by a client to a server for distribution to interested subscribers. Each PUBLISH message is associated with a topic name. If a client subscribes to one or more topics, any message published to those topics are sent by the server to the client as a PUBLISH message.

AT+QMTPUB Publish Messages	
Test Command **AT+QMTPUB=?**	Response **+QMTPUB: (**range of supported **<TCP_connectID>s),<msgID>,<qos>,<retain>,"<topic>","<msg>"** **OK**
Write Command **AT+QMTPUB=<TCP_connectID>,<msgID>,<qos>,<retain>,"<topic>"**	Response **>** After the above response, the module enters into data mode and then please input the data to be sent. Tap **CTRL+Z** to send, and tap **Esc** to cancel the operation **OK** **+QMTPUB: <TCP_connectID>,<msgID>,<result>[,<value>]** If there is any error: **ERROR**
Write Command **AT+QMTPUB=<TCP_connectID>,<msgID>,<qos>,<retain>,"<topic>","<msg>"**	Response **OK** **+QMTPUB: <TCP_connectID>,<msgID>,<result>[,<value>]** If there is an error: **ERROR**
Maximum Response Time	**<pkt_timeout> + <pkt_timeout> × <retry_times>** (default 40 s), determined by network
Characteristics	/

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier. The range is 0-5.
<msgID>	Integer type. Message identifier of packet. The range is 0-65535. It will be 0 only when **<qos>**=0.
<qos>	Integer type. The QoS level at which the client wants to publish the messages. 0 At most once 1 At least once 2 Exactly once
<retain>	Integer type. Whether or not the server will retain the message after it has been delivered to the current subscribers. 0 The server will not retain the message after it has been delivered to the current subscribers 1 The server will retain the message after it has been delivered to the current subscribers
<topic>	String type. Topic that the client wants to subscribe to or unsubscribe from. The maximum length is 255 bytes.
<msg>	String type. The message that needs to be published. The maximum length is 700 bytes. If in data mode (after **>** is responded), the maximum length is 1024 bytes.
<result>	Integer type. Result of the command execution 0 Sent the packet successfully and received ACK from server (message published when **<qos>**=0 does not require ACK) 1 Packet retransmission 2 Failed to send packet
<value>	Integer type. If **<result>** is 1, it means the times of packet retransmission. If **<result>** is 0 or 2, it will not be presented.
<pkt_timeout>	Integer type. Timeout of the packet delivery. The range is 1-60. The default value is 10. Unit: second.
<retry_times>	Integer type. Retry times when packet delivery times out. The range is 0-10. The default value is 3.

NOTES

1. PUBLISH messages can be sent either from a publisher to the server, or from the server to a subscriber. When a server publishes messages to a subscriber, the following URC will be returned to notify the host to read the received data that is reported by MQTT server:
   +QMTRECV: <TCP_connectID>,<msgID>,<topic>,<payload>
   For more details about the URC, please refer to ***Chapter 2.3.2***.
2. **<msg>** must be enclosed in double quotation marks when it is a string in special formats such as JSON. Currently, it does not support special characters such as a semicolon (;).

2.4. MQTT Related URCs

This chapter gives MQTT related URCs and their descriptions.

Table 2: MQTT Related URCs

Index	URC Format	Description
[1]	**+QMTSTAT: <TCP_connectID>,<err_code>**	When the state of MQTT link layer is changed, the client will close the MQTT connection and report the URC.
[2]	**+QMTRECV: <TCP_connectID>,<msgID>,<topic>,<payload>**	Reported when the client has received the packet data from MQTT server.

2.4.1. +QMTSTAT URC to Indicate State Change in MQTT Link Layer

The URC begins with **+QMTSTAT**. It will be reported when there is a change in the state of MQTT link layer.

+QMTSTAT URC to Indicate State Change in MQTT Link Layer	
+QMTSTAT: <TCP_connectID>,<err_code>	When the state of MQTT link layer is changed, the client will close the MQTT connection and report the URC.
Reference	

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier.
<err_code>	Integer type. An error code. Please refer to the table below for details.

Table 3: Error Codes of +QMTSTAT URC

Code of <err>	Description	How to do
1	Connection is closed or reset by peer.	Execute AT+QMTOPEN command and reopen MQTT connection.
2	Sending PINGREQ packet timed out or failed.	Deactivate PDP first, and then active PDP and reopen MQTT connection.
3	Sending CONNECT packet timed out or failed	1. Check whether the inputted user name and password are correct or not. 2. Make sure the client ID is not used. 3. Reopen MQTT connection and try to send

		CONNECT packet to server again.
4	Receiving CONNACK packet timed out or failed	1. Check whether the inputted user name and password are correct. 2. Make sure the client ID is not used. 3. Reopen MQTT connection and try to send CONNECT packet to server again.
5	The client sends DISCONNECT packet to sever but the server is initiative to close MQTT connection.	This is a normal process.
6	The client is initiative to close MQTT connection due to packet sending failure all the time.	1. Make sure the data is correct. 2. Try to reopen MQTT connection since there may be network congestion or an error.
7	The link is not alive or the server is unavailable.	Make sure the link is alive or the server is available currently.
8-255	Reserved for future use	

2.4.2. +QMTRECV URC to Inform the Host to Read MQTT Packet Data

The URC begins with **+QMTRECV**. It is mainly used to inform the host to read the received MQTT packet data that is reported from MQTT server.

+QMTRECV URC to Inform the Host to Read MQTT Packet Data	
+QMTRECV: <TCP_connectID>,<msgID>,<topic>,<payload>	Inform the host to read the received data that is reported from MQTT server.
Reference	

Parameter

<TCP_connectID>	Integer type. MQTT socket identifier.
<msgID>	Integer type. The message identifier of packet.
<topic>	String type. The topic that received from MQTT server.
<payload>	String type. The payload that relates to the topic name.

2.5. Examples

2.5.1. Use Normal TCP Connection for MQTT

```
AT+QMTOPEN=0,"220.180.239.212",8401  //Open a network for MQTT client.
OK

+QMTOPEN: 0,0                        //Opened the MQTT client network successfully.
AT+QMTCONN=0,"clientExample"
OK

+QMTCONN: 0,0,0                      //Connected the client to MQTT server successfully.
AT+QMTSUB=0,1,"topic/example",2      //Subscribe to the topic.

OK

+QMTSUB: 0,1,0,2
AT+QMTSUB=0,1,"topic/pub",0
OK

+QMTSUB: 0,1,0,0

//If a client subscribes to a topic and other devices publish the same topic to the server, the module will
report the following information.
+QMTRECV: 0,0,"topic/example","This is the payload related to topic"

AT+QMTUNS=0,2,"topic/example"     //Unsubscribe from the topic.
OK

+QMTUNS: 0,2,0
AT+QMTPUB=0,0,0,0,"topic/pub","hello MQTT."     //Publish the message.
OK

+QMTPUB: 0,0,0

//If a client subscribes to a topic named "topic/pub" and other devices publish the same topic to the server,
the module will report the following information.
+QMTRECV: 0,0,"topic/pub","hello MQTT."

AT+QMTPUB=0,0,0,0,"topic/pub"     //Publish the message in data mode.
>
This is test data, hello MQTT.    //Input the data to be published and then tap CTRL+Z to send.
OK
```

```
+QMTPUB: 0,0,0

//If a client subscribes to a topic named "topic/pub" and other devices publish the same topic to the server,
the module will report the following information.
+QMTRECV: 0,0,"topic/pub","This is test data, hello MQTT."

AT+QMTDISC=0                      //Disconnect the client from MQTT server.
OK

+QMTDISC: 0,0                     //Connection closed successfully.
```

2.5.2. Use SSL/TLS TCP Secure Connection for MQTT

```
AT+QSCLK=0                          //Disable sleep mode
OK

//Configure certificates and keys
AT+QSSLCFG=1,5,"seclevel",2         //Manage server and client authentication
OK
AT+QSSLCFG=1,5,"cacert"             //Configure CA certificate
>                                   //Input the content of the trusted CA certificate in PEM
                                    format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"cacert",1216

OK
AT+QSSLCFG=1,5,"clientcert"         //Configure client certificate
>                                   //Input the content of the client certificate in PEM format.
                                    Tap CTRL+Z to send.
+QSSLCFG: 1,5,"clientcert",1224

OK
AT+QSSLCFG=1,5,"clientkey"          //Configure client private  key
>                                   //Input the content of the client private key in PEM
                                    format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"clientkey",1679

OK
AT+QSCLK=1                          //Enable light sleep and deep sleep, and wakeup by
                                    PSM_EINT (falling edge).
OK

AT+QMTCFG="ssl",3,1,1,5             //Enable SSL and configure SSL context/connect index.
OK
AT+QMTOPEN=3,"hf.quectel.com",8164  //Open a network for MQTT client.
```

```
OK

+QMTOPEN: 3,0                               //Opened the MQTT client network successfully.
AT+QMTCONN=3,"clientExample"
OK

+QMTCONN: 3,0,0                             //Connected the client to MQTT server successfully.
AT+QMTSUB=3,1,"topic/example/tls",1         //Subscribe to the topic.
OK

+QMTSUB: 3,1,0,1
AT+QMTPUB=3,0,0,0,"topic/example/tls","hello MQTT."      //Publish the message.

OK

+QMTPUB: 3,0,0

//If a client subscribes to a topic named "topic/example/tls" and other devices publish the same topic to
the server, the module will report the following information.
+QMTRECV: 3,0,"topic/example/tls","hello MQTT."

AT+QMTPUB=3,0,0,0,"topic/example/tls"       //Publish the messages in data mode.
>
This is test data, hello MQTT.         //Input the data to be published and then tap CTRL+Z to send.
OK

+QMTPUB: 3,0,0

//If a client subscribes to a topic named "topic/pub" and other devices publish the same topic to the
server, the module will report the following information.
+QMTRECV: 3,0,"topic/pub","This is test data, hello MQTT."

AT+QMTDISC=3                     //Disconnect the client from MQTT server.
OK

+QMTDISC: 3,0                    //Connection closed successfully.
```

3 Connection with Azure IoT Hub

This chapter introduces the steps to establish connection with Azure IoT hub through TLS/SSL secured MQTT.

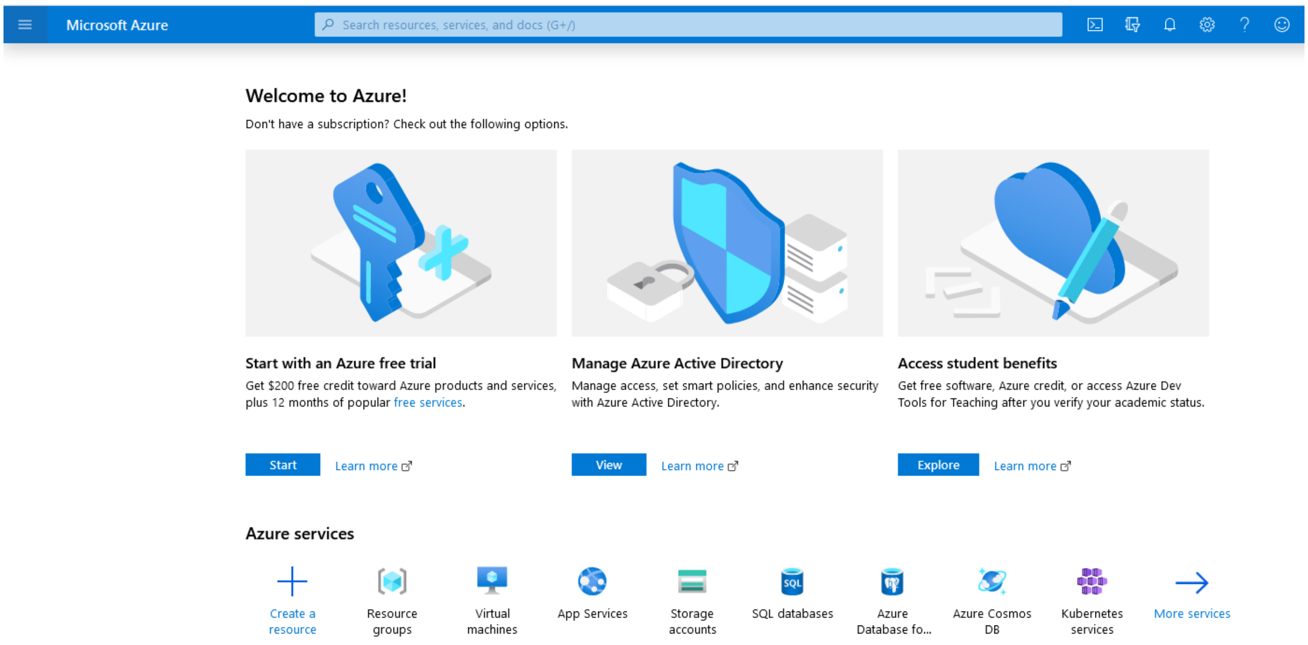
3.1. Create Azure Account

3.1.1. Create a Free Account

Create a free Azure account in https://azure.microsoft.com/en-gb/free/.

3.1.2. Enter Azure Portal

Enter Azure Portal via link https://portal.azure.com/#home.



Welcome to Azure!

Don't have a subscription? Check out the following options.

Start with an Azure free trial

Get $200 free credit toward Azure products and services, plus 12 months of popular free services.

Manage Azure Active Directory

Manage access, set smart policies, and enhance security with Azure Active Directory.

Access student benefits

Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.

Azure services

Figure 2: Azure Portal

3.1.3. Create a Resource Group

a) In the left navigation bar, move your mouse cursor to "**Resource groups**", and then click "**Create**" to create a resource group.
b) Fill a resource group name in the box after "**Resource group**".
c) Click "**Review + create**".
d) Click "**Create**", and then check whether the resource group has been created successfully through clicking "**Resource groups**" button in the homepage again.

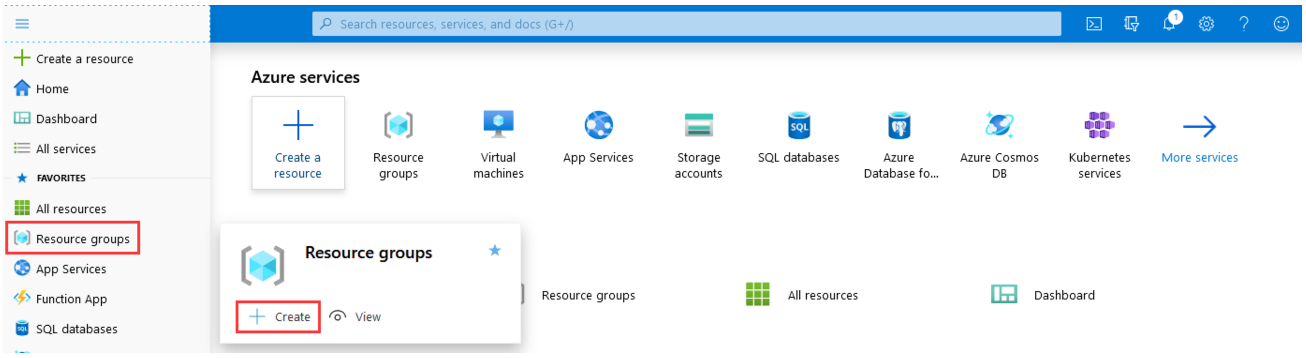
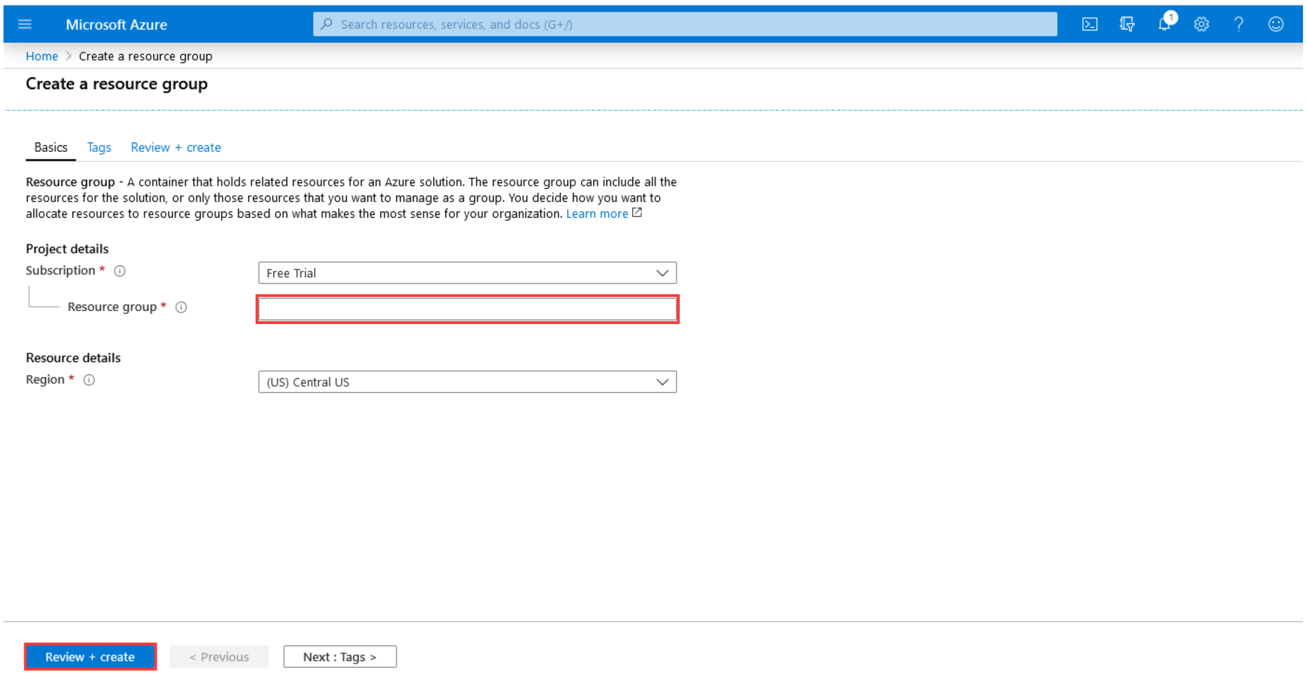


Figure 3: Create a Resource Group (Step a)


Microsoft Azure
Search resources, services, and docs (G+/)
Home > Create a resource group
Create a resource group
Basics
Tags
Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. Learn more

Project details

Subscription * : Free Trial

Resource group * :

Resource details

Region * : (US) Central US

Figure 4: Create a Resource Group (Step b and c)

3.1.4. Create an IoT Hub Resource

a) Click the name of the newly created resource group, and then click "**Add**".
b) Afterwards, there will be a search box indicating "**Search the Marketplace**". Input "IoT Hub" in the search box to enter IoT Hub page.
c) Click "**Create**" in IoT Hub page.
d) Name your IoT Hub in the box after "**IoT Hub Name**", and then click "**Review + create**".
e) Finally click "**Create**" to finish the operation.

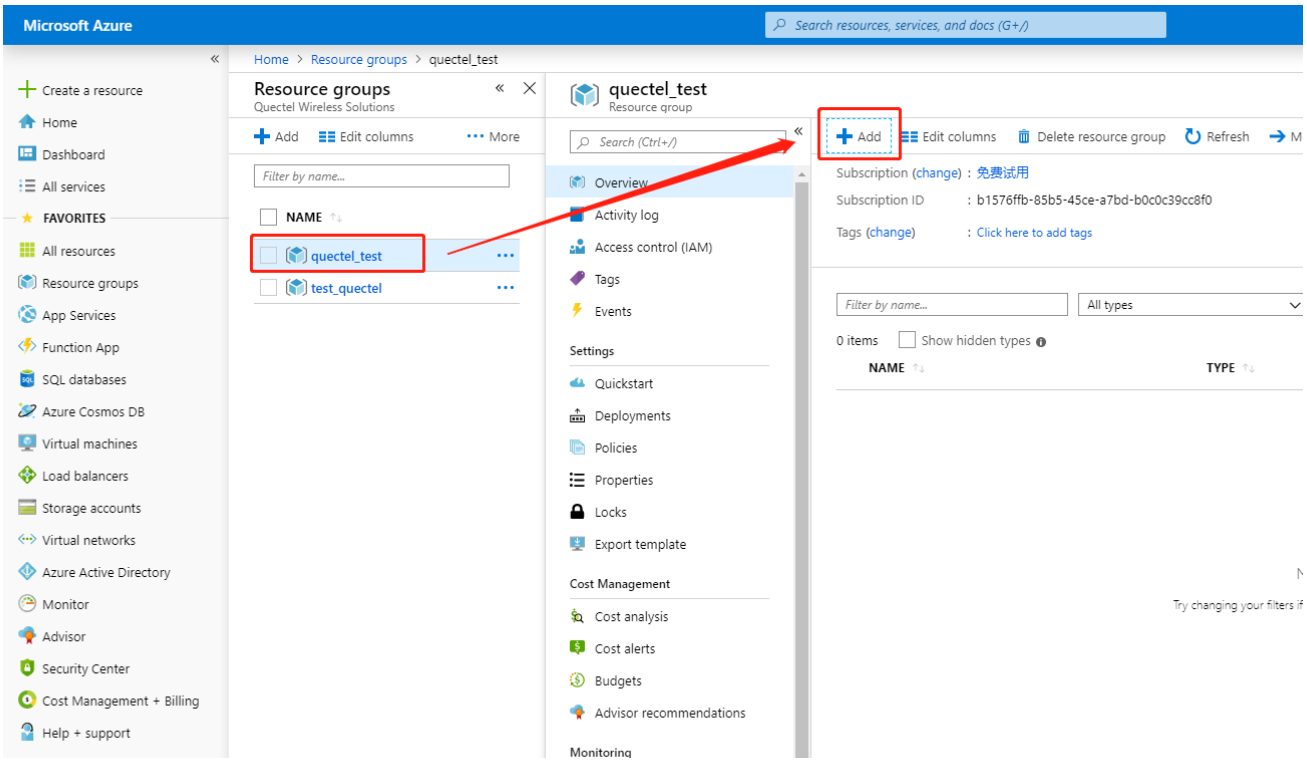
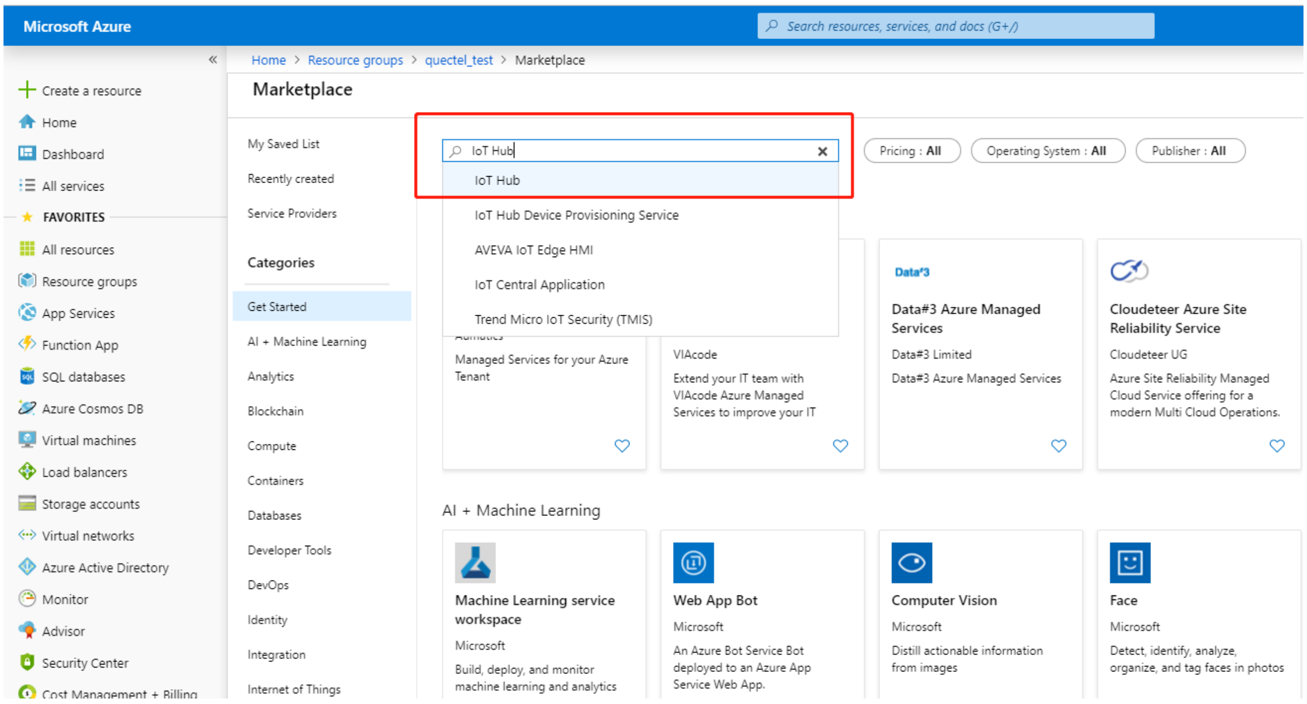


Figure 5: Create an IoT Hub Resource (Step a)


Microsoft Azure

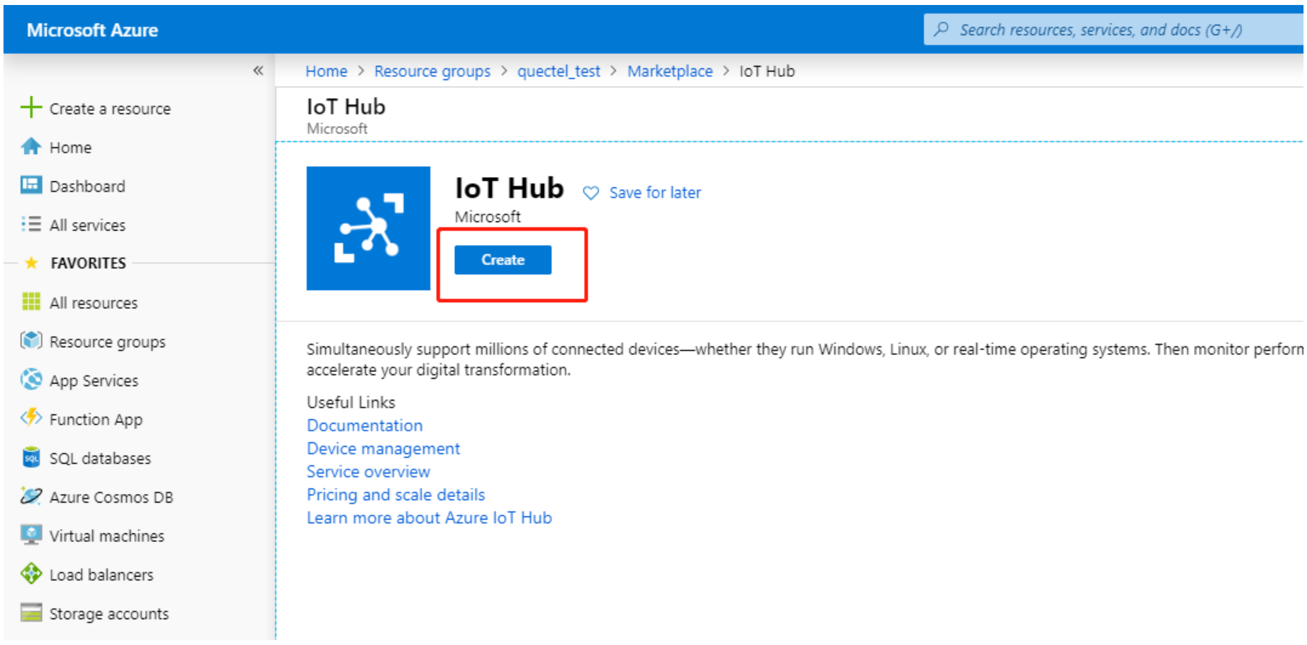
Home > Resource groups > quectel_test > Marketplace

Marketplace

IoT Hub

IoT Hub
IoT Hub Device Provisioning Service
AVEVA IoT Edge HMI
IoT Central Application
Trend Micro IoT Security (TMIS)

Figure 6: Create an IoT Hub Resource (Step b)


Microsoft Azure

Home > Resource groups > quectel_test > Marketplace > IoT Hub

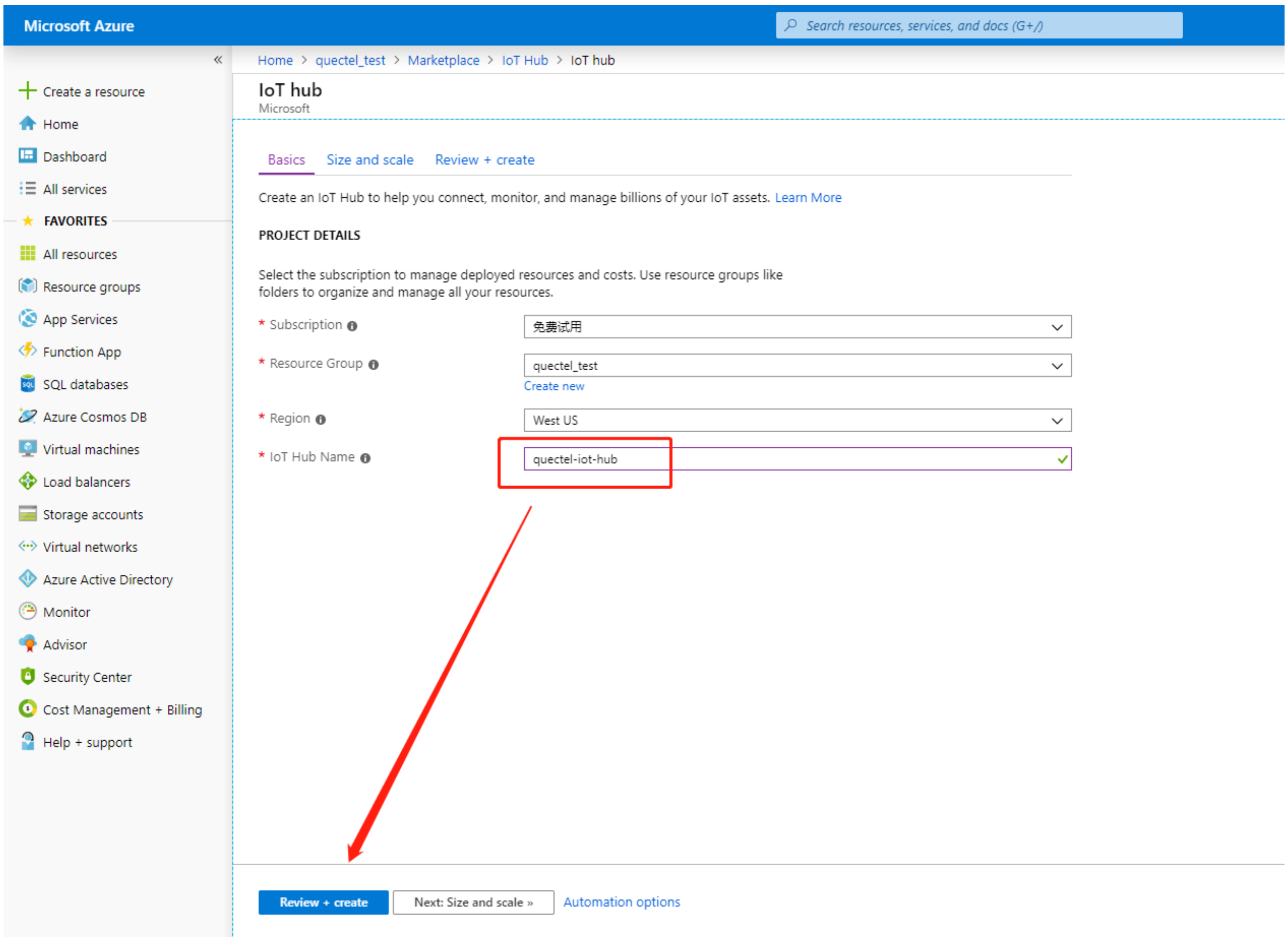
IoT Hub
Microsoft

Create

Simultaneously support millions of connected devices—whether they run Windows, Linux, or real-time operating systems. Then monitor perform
accelerate your digital transformation.

Useful Links
Documentation
Device management
Service overview
Pricing and scale details
Learn more about Azure IoT Hub

Figure 7: Create an IoT Hub Resource (Step c)



Create an IoT Hub to help you connect, monitor, and manage billions of your IoT assets. Learn More

PROJECT DETAILS

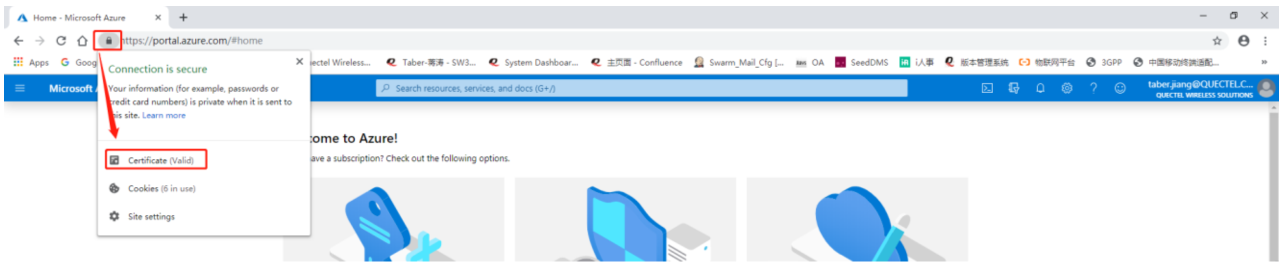
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription	免费试用
* Resource Group	quectel_test
* Region	West US
* IoT Hub Name	quectel-iot-hub

Figure 8: Create an IoT Hub Resource (Step d)

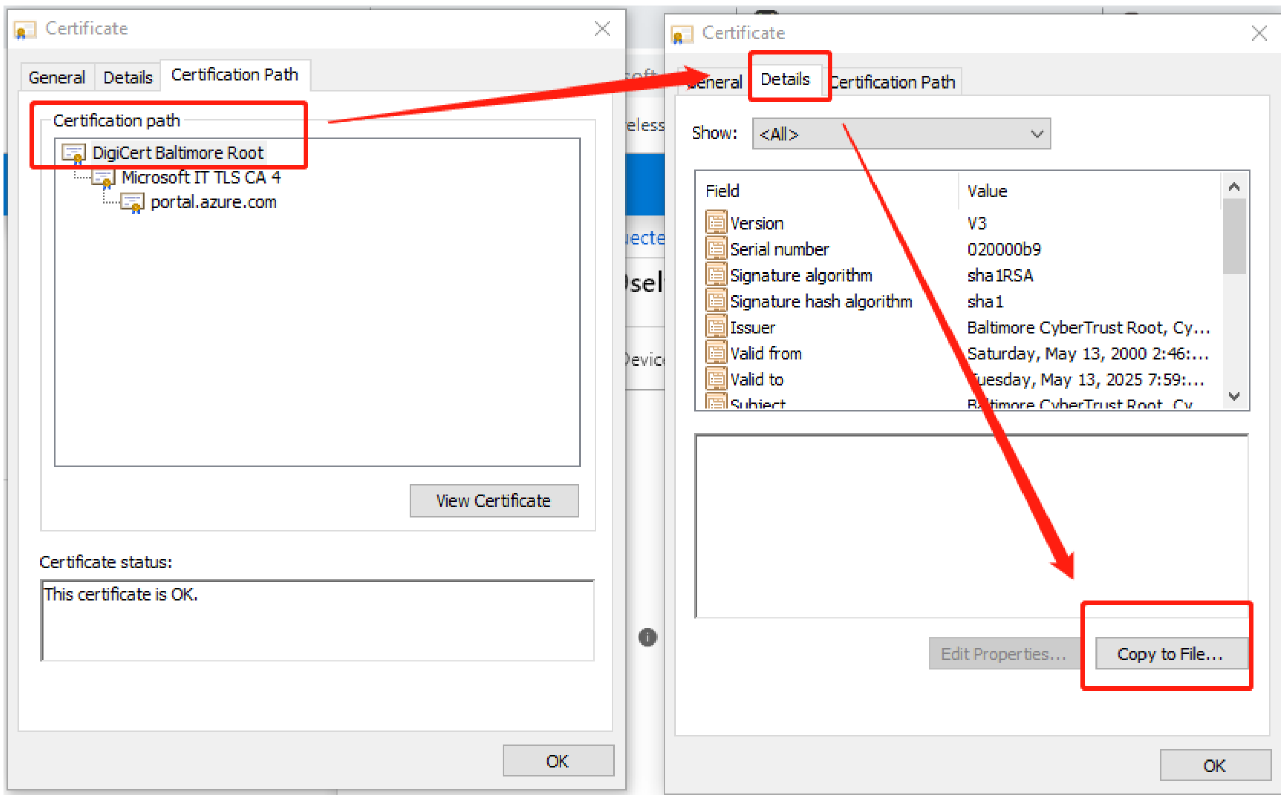
3.1.5. Get Azure Root CA certificate

- The Azure root CA certificates can be got with Google Chrome as follows:



Your information (for example, passwords or credit card numbers) is private when it is sent to this site. Learn more

Figure 9: Set Azure Root CA Certificate with Google Chrome

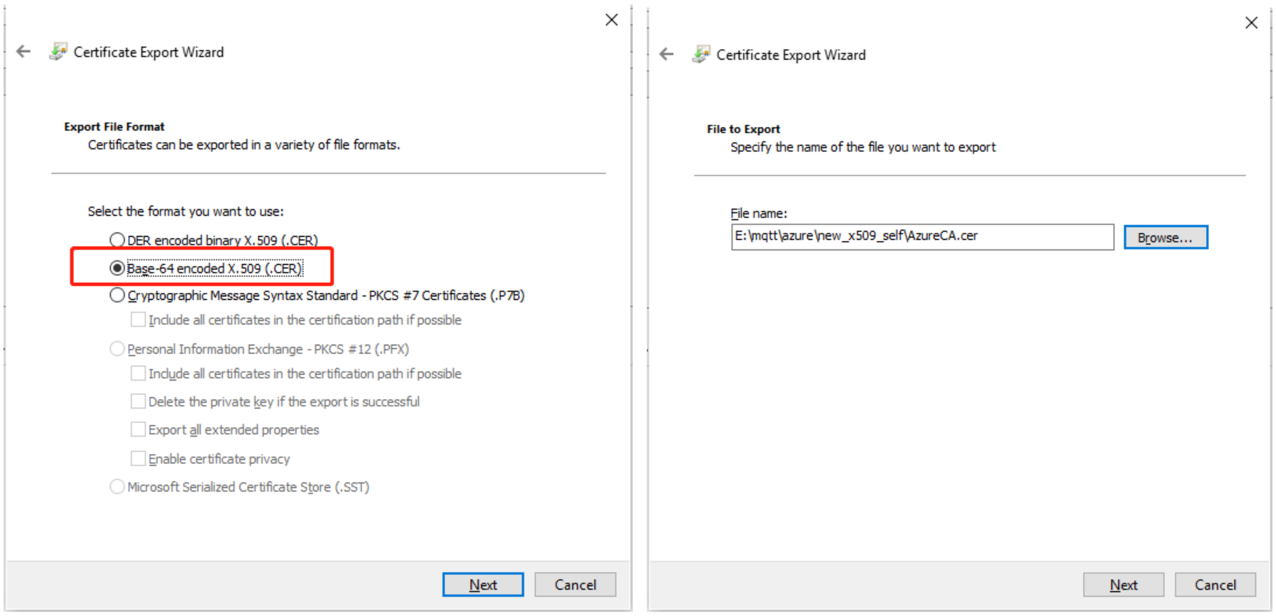


Certification path
DigiCert Baltimore Root
Microsoft IT TLS CA 4
portal.azure.com

Certificate status:
This certificate is OK.

Field	Value
Version	V3
Serial number	020000b9
Signature algorithm	sha1RSA
Signature hash algorithm	sha1
Issuer	Baltimore CyberTrust Root, Cy...
Valid from	Saturday, May 13, 2000 2:46:...
Valid to	Tuesday, May 13, 2025 7:59:...
Subject	Baltimore CyberTrust Root, Cy...

Figure 10: Copy Certification Path to File



Export File Format
Certificates can be exported in a variety of file formats.

Select the format you want to use:
- ◯ DER encoded binary X.509 (.CER)
- ● Base-64 encoded X.509 (.CER)
- ◯ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  - ☐ Include all certificates in the certification path if possible
- ◯ Personal Information Exchange - PKCS #12 (.PFX)
  - ☐ Include all certificates in the certification path if possible
  - ☐ Delete the private key if the export is successful
  - ☐ Export all extended properties
  - ☐ Enable certificate privacy
- ◯ Microsoft Serialized Certificate Store (.SST)

File to Export
Specify the name of the file you want to export

File name: E:\mqtt\azure\new_x509_self\AzureCA.cer

Figure 11: Export File

- Or please refer to https://github.com/Azure/azure-iot-sdk-c/blob/master/certs/certs.c.

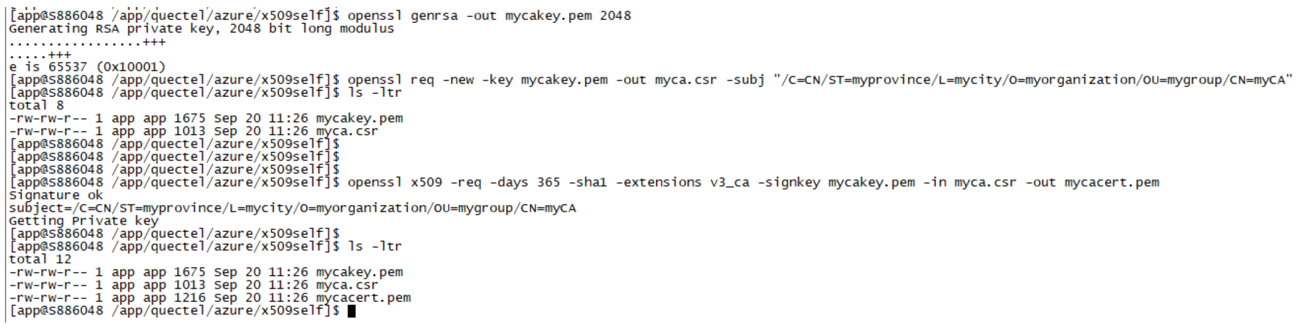
3.2. Communicate with Azure via X.509 Self-Signed Certificate

This document takes X.509 certificate as an example. Create customized X.509 certificates using a third-party tool such as OpenSSL. This technique is ideal for test and development purposes.

3.2.1. Generate X.509 Self-Signed CA Certificate

The X.509 self-signed CA certificate can be generated with OpenSSL.

```
#Generate ca certificate private key (pem file)
openssl genrsa -out mycakey.pem 2048

#Generate the ca certificate sign application file (csr file)
openssl req -new -key mycakey.pem -out myca.csr -subj
"/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myCA"

#Self-signed ca certificate
openssl x509 -req -days 365 -sha1 -extensions v3_ca -signkey mycakey.pem -in myca.csr -out
mycacert.pem
```

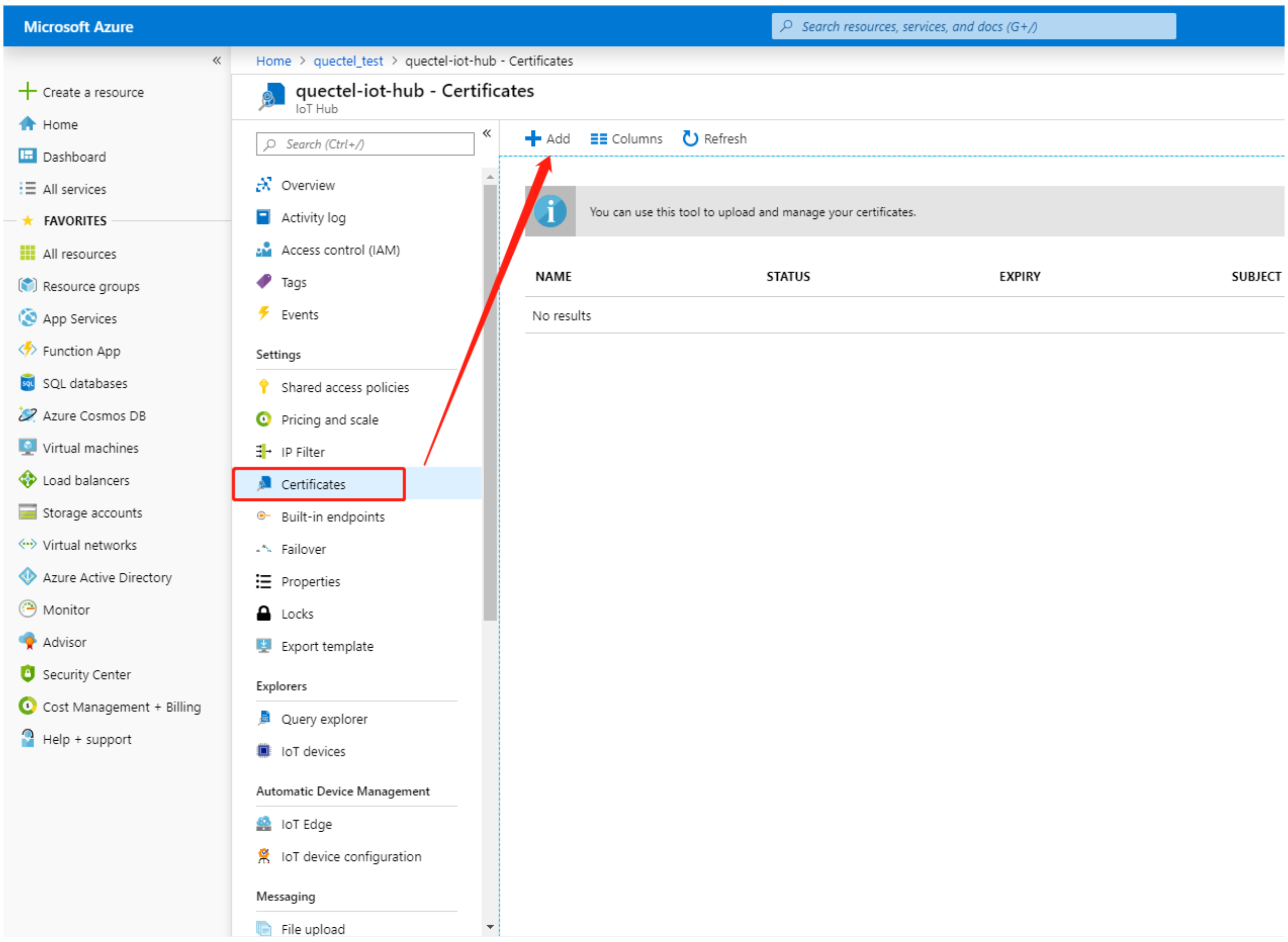


```
[app@S886048 /app/quectel/azure/x509self]$ openssl genrsa -out mycakey.pem 2048
Generating RSA private key, 2048 bit long modulus
................+++
.....+++
e is 65537 (0x10001)
[app@S886048 /app/quectel/azure/x509self]$ openssl req -new -key mycakey.pem -out myca.csr -subj "/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myCA"
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 8
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$ openssl x509 -req -days 365 -sha1 -extensions v3_ca -signkey mycakey.pem -in myca.csr -out mycacert.pem
Signature ok
subject=/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myCA
Getting Private key
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 12
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
-rw-rw-r-- 1 app app 1216 Sep 20 11:26 mycacert.pem
[app@S886048 /app/quectel/azure/x509self]$
```

3.2.2. Add X.509 Self-Signed CA Certificate

Follow the steps below to add the generated X.509 self-signed CA certificates into Azure IoT Hub.

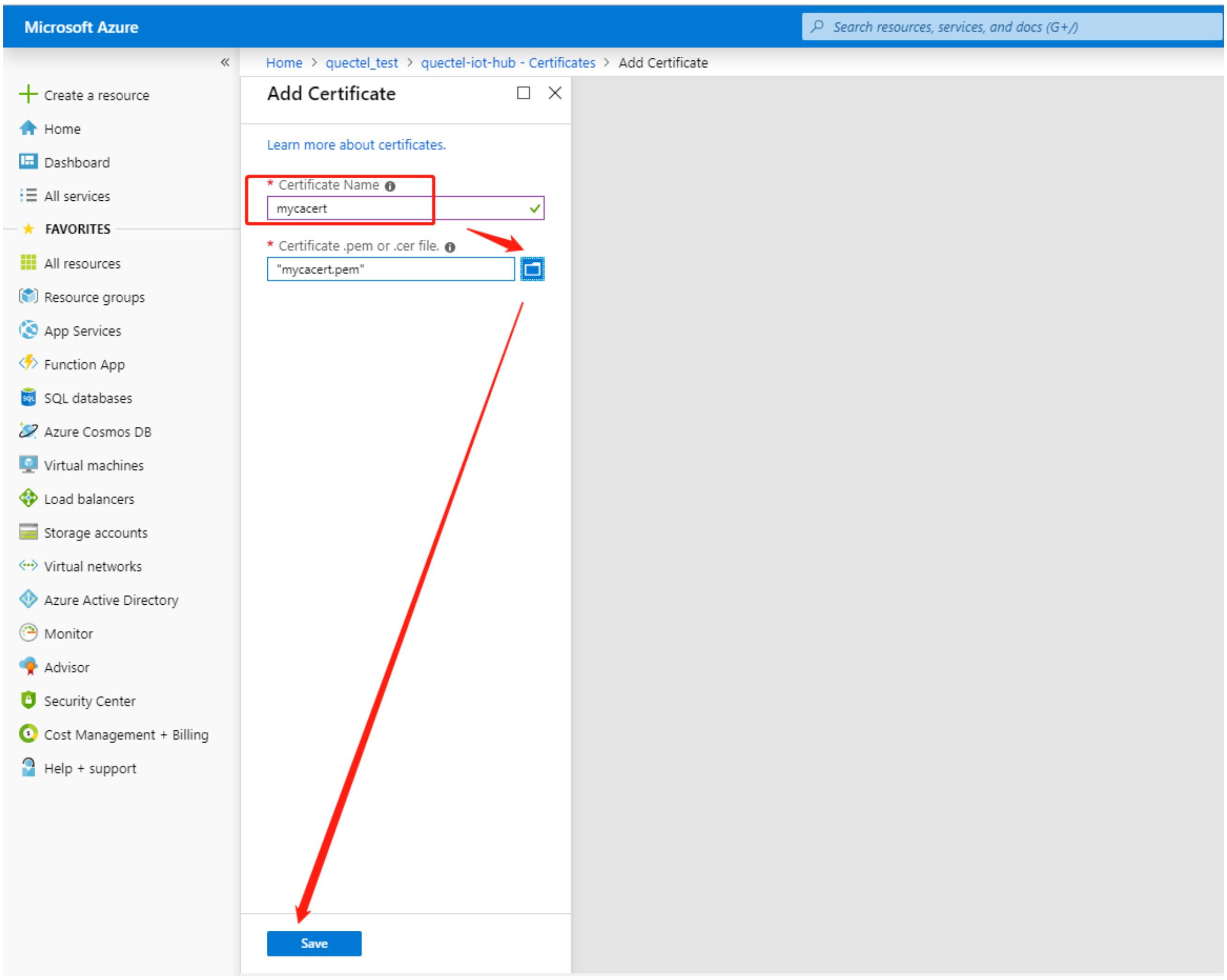
a) Click the name of "**quectel-iot-hub**" resource.
b) Click "**Certificates**", and then click "**Add**" to add a certificate.
c) Fill the certificate name (*mycacert* for instance) and then upload the certificate *mycacert.pem*.
d) Click "**Save**" to save the operations.



You can use this tool to upload and manage your certificates.

NAME	STATUS	EXPIRY	SUBJECT
No results			

Figure 12: Add a Certificate (Step b)



* Certificate Name: mycacert

* Certificate .pem or .cer file.: "mycacert.pem"

Figure 13: Add a Certificate (Step c and d)

3.2.3. Verify X.509 Self-Signed CA Certificate

Follow the steps illustrated in the chapters below to verify the X.509 Self-signed CA certificate.

3.2.3.1. Generate Verification Code

a) Select "**mycacert**" certificate.
b) Click "**Generate Verification Code**".
c) After the "**Verification Code**" is generated successfully, click "**Copy**" button to copy the code to clipboard.

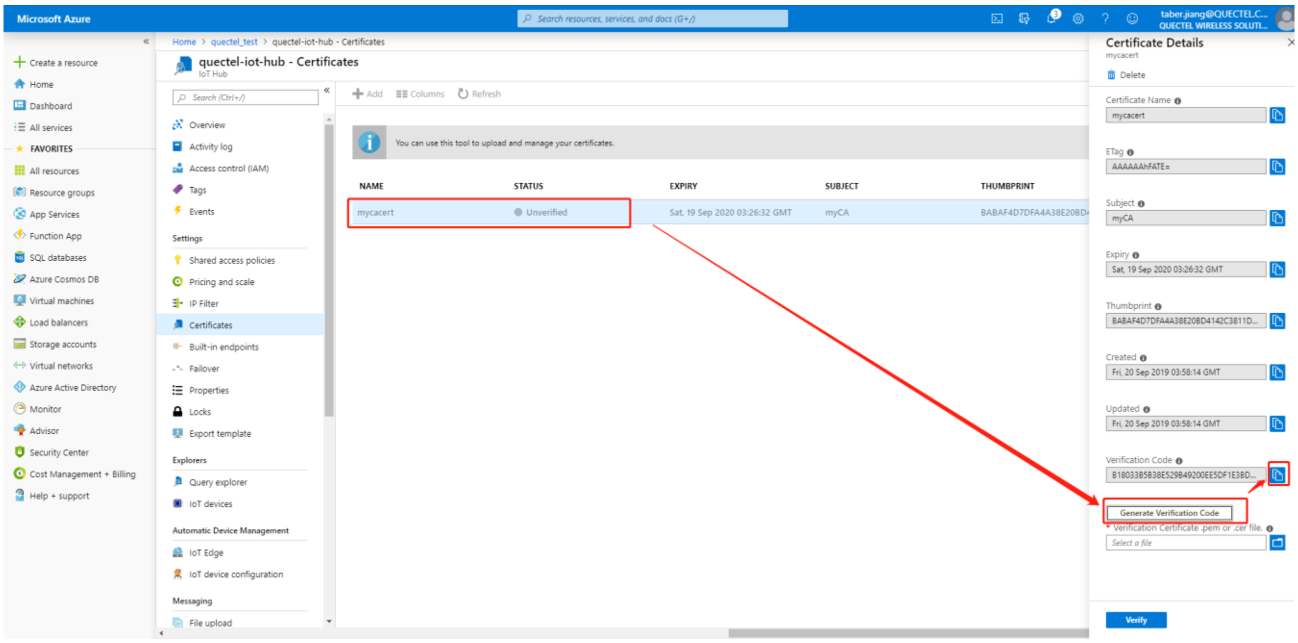
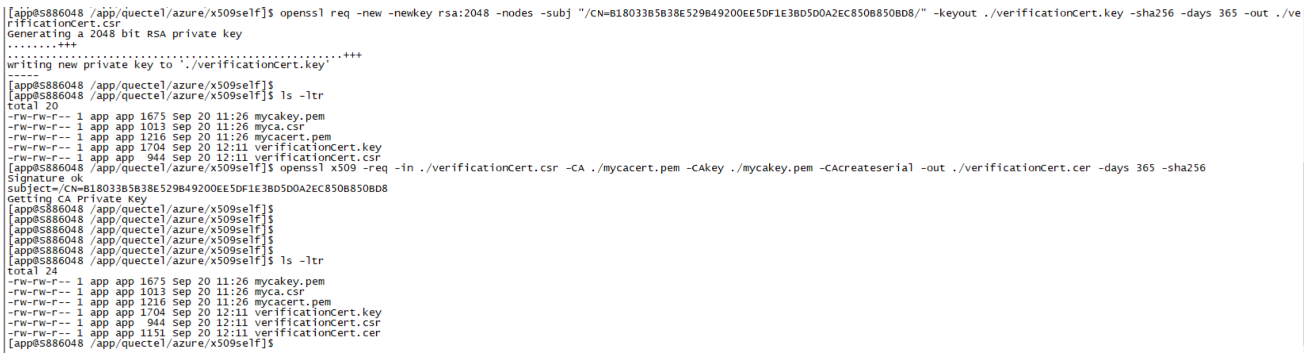

Microsoft Azure
Home > quectel_test > quectel-iot-hub - Certificates
quectel-iot-hub - Certificates
IoT Hub
Add
Columns
Refresh
You can use this tool to upload and manage your certificates.
NAME
STATUS
EXPIRY
SUBJECT
THUMBPRINT
mycacert
Unverified
Sat, 19 Sep 2020 03:26:32 GMT
myCA
Certificate Details
Delete
Certificate Name
mycacert
ETag
AAAAAAhFATE=
Subject
myCA
Expiry
Sat, 19 Sep 2020 03:26:32 GMT
Created
Fri, 20 Sep 2019 03:58:14 GMT
Updated
Fri, 20 Sep 2019 03:58:14 GMT
Verification Code
Generate Verification Code
Verification Certificate .pem or .cer file.
Select a file
Verify

Figure 14: Generate Verification Code

3.2.3.2. Generate Verification Certificate

Here are the details about how to generate verification certificate.

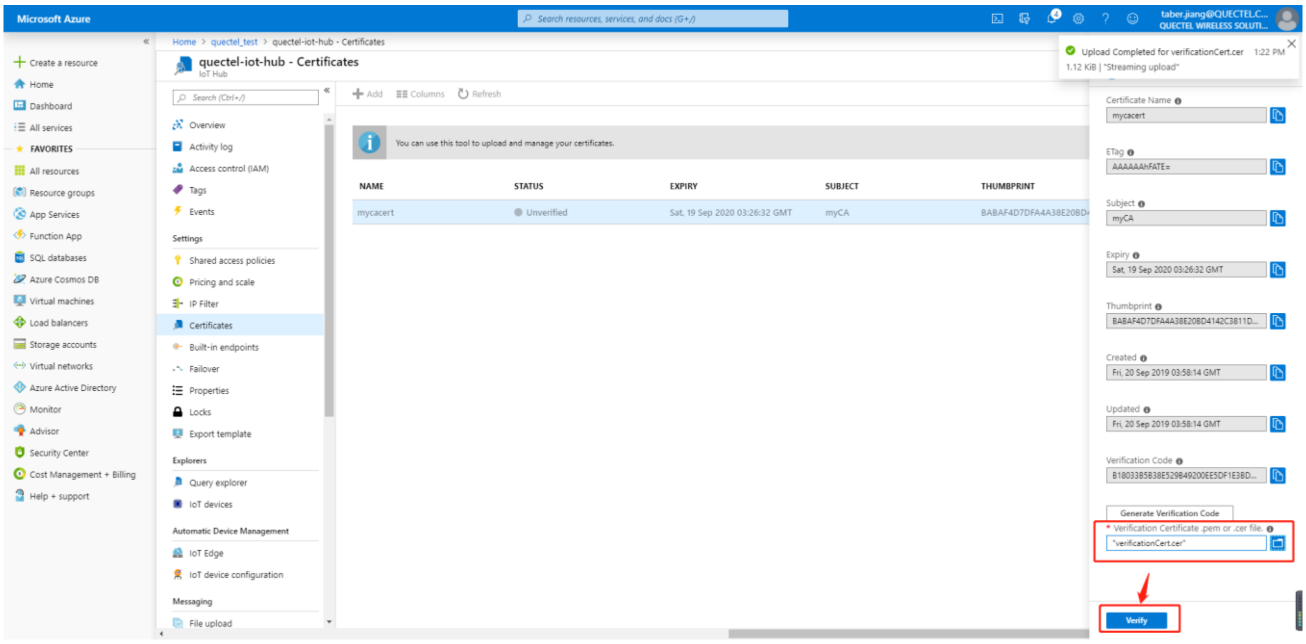
```
#Generate verificationCert.csr
openssl req -new -newkey rsa:2048 -nodes -subj "/CN=B18033B5B38E529B49200EE5DF1E3BD5D0A2EC850B850BD8/" -keyout ./verificationCert.key -sha256 -days 365 -out ./verificationCert.csr

#Self-signed verificationCert
openssl x509 -req -in ./verificationCert.csr -CA ./mycacert.pem -CAkey ./mycakey.pem -CAcreateserial -out ./verificationCert.cer -days 365 -sha256
```



```
[app@S886048 /app/quectel/azure/x509self]$ openssl req -new -newkey rsa:2048 -nodes -subj "/CN=B18033B5B38E529B49200EE5DF1E3BD5D0A2EC850B850BD8/" -keyout ./verificationCert.key -sha256 -days 365 -out ./ve
rificationCert.csr
Generating a 2048 bit RSA private key
........+++
.....................................................+++
writing new private key to './verificationCert.key'
-----
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 20
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
-rw-rw-r-- 1 app app 1216 Sep 20 11:26 mycacert.pem
-rw-rw-r-- 1 app app 1704 Sep 20 12:11 verificationCert.key
-rw-rw-r-- 1 app app  944 Sep 20 12:11 verificationCert.csr
[app@S886048 /app/quectel/azure/x509self]$ openssl x509 -req -in ./verificationCert.csr -CA ./mycacert.pem -CAkey ./mycakey.pem -CAcreateserial -out ./verificationCert.cer -days 365 -sha256
Signature ok
subject=/CN=B18033B5B38E529B49200EE5DF1E3BD5D0A2EC850B850BD8
Getting CA Private Key
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 24
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
-rw-rw-r-- 1 app app 1216 Sep 20 11:26 mycacert.pem
-rw-rw-r-- 1 app app 1704 Sep 20 12:11 verificationCert.key
-rw-rw-r-- 1 app app  944 Sep 20 12:11 verificationCert.csr
-rw-rw-r-- 1 app app 1151 Sep 20 12:11 verificationCert.cer
[app@S886048 /app/quectel/azure/x509self]$
```

3.2.3.3. Verify Verification Certificate

a) Click "**Upload**" button to upload file *verificationCert.cer*.
b) Click "**Verify**" to verify the certificate.
c) Click "**Refresh**" to check whether the verification certificate has been **Verified**.


Microsoft Azure
Search resources, services, and docs (G+/)
taber.jiang@QUECTEL.C...
QUECTEL WIRELESS SOLUTI...
Home > quectel_test > quectel-iot-hub - Certificates
quectel-iot-hub - Certificates
IoT Hub
Add
Columns
Refresh
Generate Verification Code
Verify

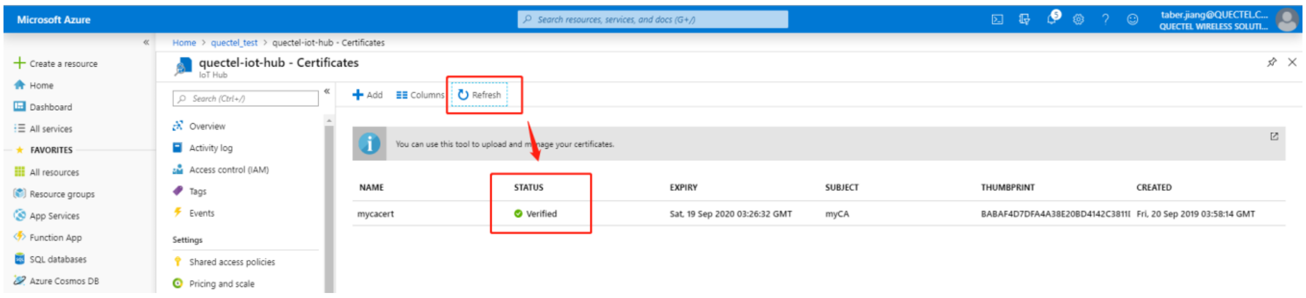
Upload Completed for verificationCert.cer 1:22 PM
1.12 KiB | "Streaming upload"

You can use this tool to upload and manage your certificates.

NAME	STATUS	EXPIRY	SUBJECT	THUMBPRINT
mycacert	Unverified	Sat, 19 Sep 2020 03:26:32 GMT	myCA	BABAF4D7DFA4A38E20BD

Certificate Name: mycacert
ETag: AAAAAAhFATE=
Subject: myCA
Expiry: Sat, 19 Sep 2020 03:26:32 GMT
Thumbprint: BABAF4D7DFA4A38E20BD4142C3811D...
Created: Fri, 20 Sep 2019 03:58:14 GMT
Updated: Fri, 20 Sep 2019 03:58:14 GMT
Verification Code: B18033B5B38E529B49200EE5DF1E3BD...
* Verification Certificate .pem or .cer file.: "verificationCert.cer"

Figure 15: Upload and Verify Verification Certificate


Microsoft Azure
Search resources, services, and docs (G+/)
taber.jiang@QUECTEL.C...
QUECTEL WIRELESS SOLUTI...
Home > quectel_test > quectel-iot-hub - Certificates
quectel-iot-hub - Certificates
IoT Hub
Add
Columns
Refresh

You can use this tool to upload and manage your certificates.

NAME	STATUS	EXPIRY	SUBJECT	THUMBPRINT	CREATED
mycacert	Verified	Sat, 19 Sep 2020 03:26:32 GMT	myCA	BABAF4D7DFA4A38E20BD4142C3811I	Fri, 20 Sep 2019 03:58:14 GMT

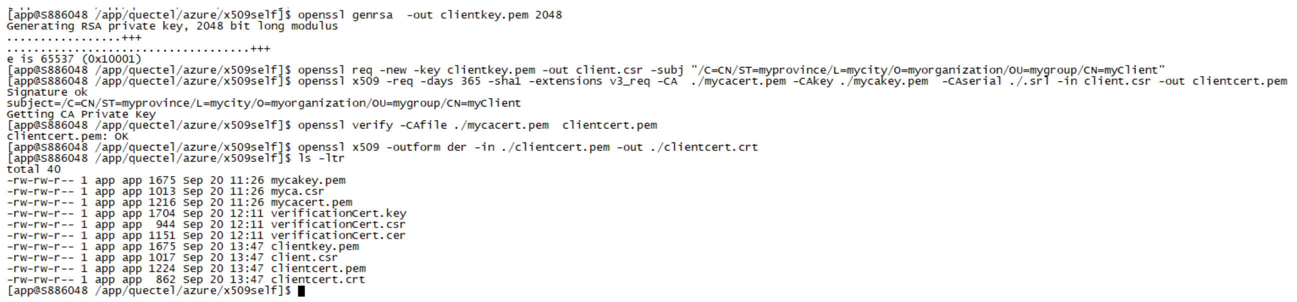
Figure 16: Check the Verification Result

3.2.4. Generate X.509 Self-Signed Client Certificate

The X.509 self-signed client certificate can be generated with OpenSSL.

```
#Generate client certificate private key (pem file)
openssl genrsa   -out clientkey.pem 2048
```

```
#Generate client certificate sign application file (csr file)
Openssl req -new -key clientkey.pem -out client.csr -subj "/C=CN/ST=myprovince/L=mycity/O=myorg
anization/OU=mygroup/CN=myClient"

#Self-signed client certificate
openssl x509 -req -days 365 -sha1 -extensions v3_req -CA  ./mycacert.pem -Cakey ./mycakey.pe
 m -CAserial ./.srl -in client.csr -out clientcert.pem

#verify
Openssl verify -CAfile ./mycacert.pem   clientcert.pem

#convert PEM to CRT format
openssl x509 -outform der -in ./clientcert.pem -out ./clientcert.crt
```



```
[app@S886048 /app/quectel/azure/x509self]$ openssl genrsa  -out clientkey.pem 2048
Generating RSA private key, 2048 bit long modulus
.................+++
......................................+++
e is 65537 (0x10001)
[app@S886048 /app/quectel/azure/x509self]$ openssl req -new -key clientkey.pem -out client.csr -subj "/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myClient"
[app@S886048 /app/quectel/azure/x509self]$ openssl x509 -req -days 365 -sha1 -extensions v3_req -CA  ./mycacert.pem -CAkey ./mycakey.pem  -CAserial ./.srl -in client.csr -out clientcert.pem
Signature ok
subject=/C=CN/ST=myprovince/L=mycity/O=myorganization/OU=mygroup/CN=myClient
Getting CA Private Key
[app@S886048 /app/quectel/azure/x509self]$ openssl verify -CAfile ./mycacert.pem  clientcert.pem
clientcert.pem: OK
[app@S886048 /app/quectel/azure/x509self]$ openssl x509 -outform der -in ./clientcert.pem -out ./clientcert.crt
[app@S886048 /app/quectel/azure/x509self]$ ls -ltr
total 40
-rw-rw-r-- 1 app app 1675 Sep 20 11:26 mycakey.pem
-rw-rw-r-- 1 app app 1013 Sep 20 11:26 myca.csr
-rw-rw-r-- 1 app app 1216 Sep 20 11:26 mycacert.pem
-rw-rw-r-- 1 app app 1704 Sep 20 12:11 verificationCert.key
-rw-rw-r-- 1 app app  944 Sep 20 12:11 verificationCert.csr
-rw-rw-r-- 1 app app 1151 Sep 20 12:11 verificationCert.cer
-rw-rw-r-- 1 app app 1675 Sep 20 13:47 clientkey.pem
-rw-rw-r-- 1 app app 1017 Sep 20 13:47 client.csr
-rw-rw-r-- 1 app app 1224 Sep 20 13:47 clientcert.pem
-rw-rw-r-- 1 app app  862 Sep 20 13:47 clientcert.crt
[app@S886048 /app/quectel/azure/x509self]$ 
```

3.2.5. Create X.509 Self-Signed Device

Please create X.509 self-signed device in Azure IoT Hub as follows:

a) Click "**IoT devices**" and then click "**New**".
b) Select "**X.509 Self-Signed**".
c) Open *clientcert.crt* to get "Thumbprint".
d) Input "Device ID", "Primary Thumbprint" and "Secondary Thumbprint".
e) Click "**Save**".
f) Check the creation result.

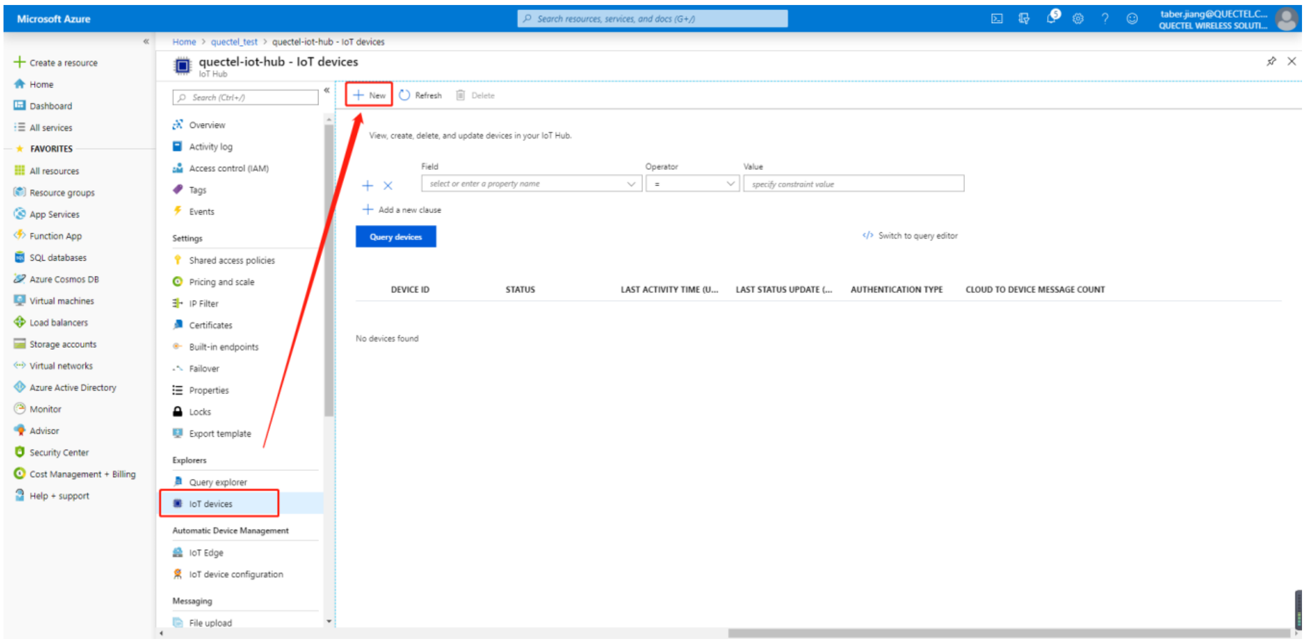
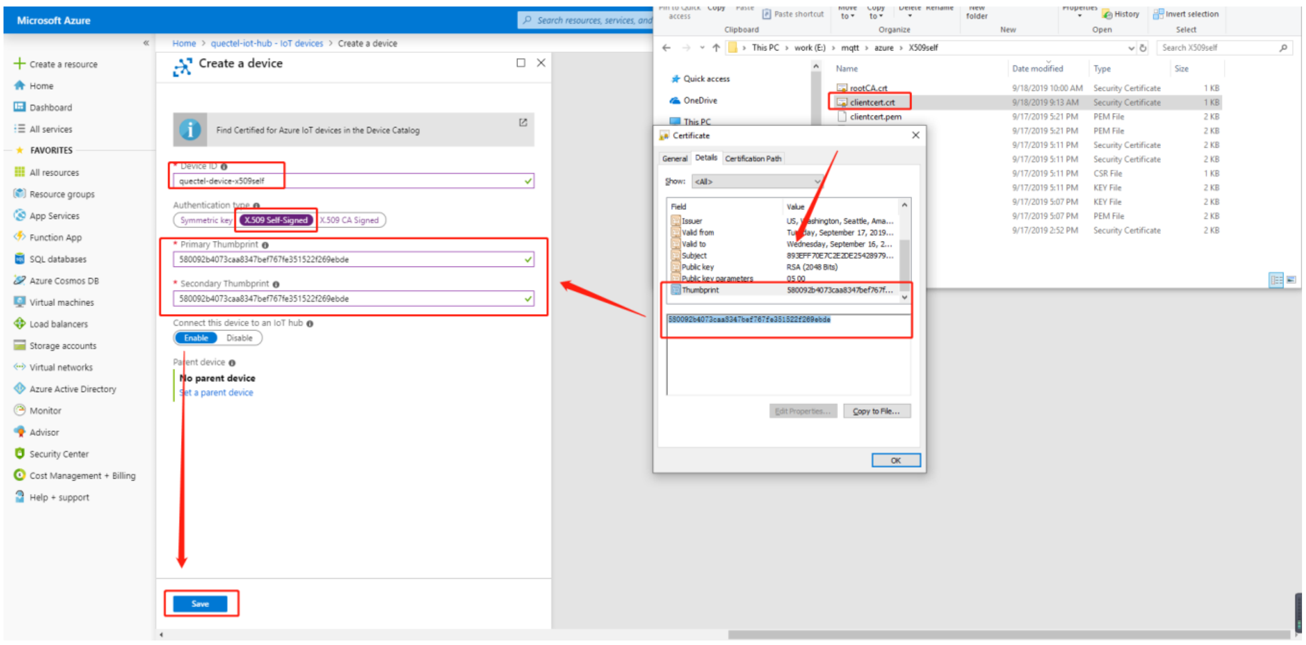


Figure 17: Create X.509 Self-Signed Device (Step a)


Home > quectel-iot-hub - IoT devices > Create a device

Create a device

Find Certified for Azure IoT devices in the Device Catalog

* Device ID: quectel-device-x509self

Authentication type: Symmetric key [✓ X.509 Self-Signed] X.509 CA Signed

* Primary Thumbprint: 580092b4073caa8347bef767fe351522f269ebde

* Secondary Thumbprint: 580092b4073caa8347bef767fe351522f269ebde

Connect this device to an IoT hub: [✓ Enable] Disable

Parent device: No parent device

Set a parent device

Save

This PC > work (E:) > mqtt > azure > X509self

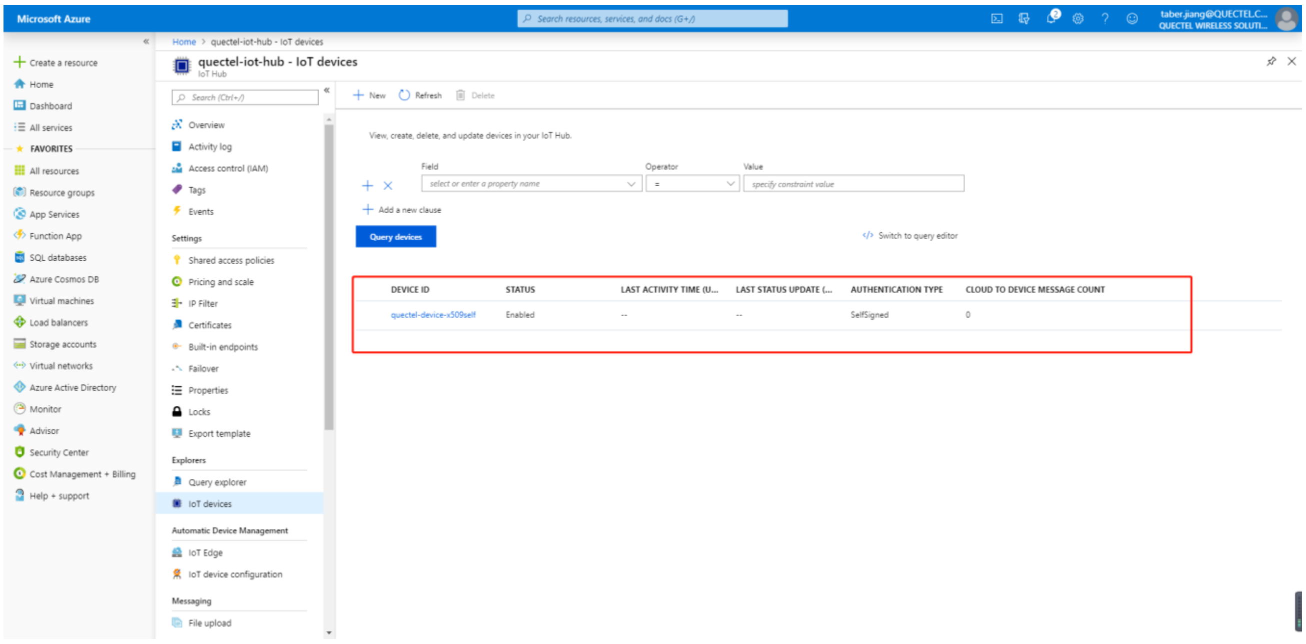
Name	Date modified	Type	Size
rootCA.crt	9/18/2019 10:00 AM	Security Certificate	1 KB
clientcert.crt	9/18/2019 9:13 AM	Security Certificate	1 KB
clientcert.pem	9/17/2019 5:21 PM	PEM File	2 KB

Certificate — Details

Field	Value
Issuer	US, Washington, Seattle, Ama...
Valid from	Tuesday, September 17, 2019...
Valid to	Wednesday, September 16, 2...
Subject	893EFF70E7C2E2DE25428979...
Public key	RSA (2048 Bits)
Public key parameters	05 00
Thumbprint	580092b4073caa8347bef767f...

580092b4073caa8347bef767fe351522f269ebde

Figure 18: Create X.509 Self-Signed Device (Step b to e)



DEVICE ID	STATUS	LAST ACTIVITY TIME (U...	LAST STATUS UPDATE (...	AUTHENTICATION TYPE	CLOUD TO DEVICE MESSAGE COUNT
quectel-device-x509self	Enabled	--	--	SelfSigned	0

Figure 19: Check the Creation Result

3.3. Usage of Device Explorer Tool

The Device Explorer tool can be used to manage devices connecting to customer's IoT hub, for example, registering a device with customer's IoT hub, monitoring messages from the devices, and sending messages to the devices.

This chapter describes the usage of the Device Explorer tool which will be used as Azure server tool.

3.3.1. Download/Install

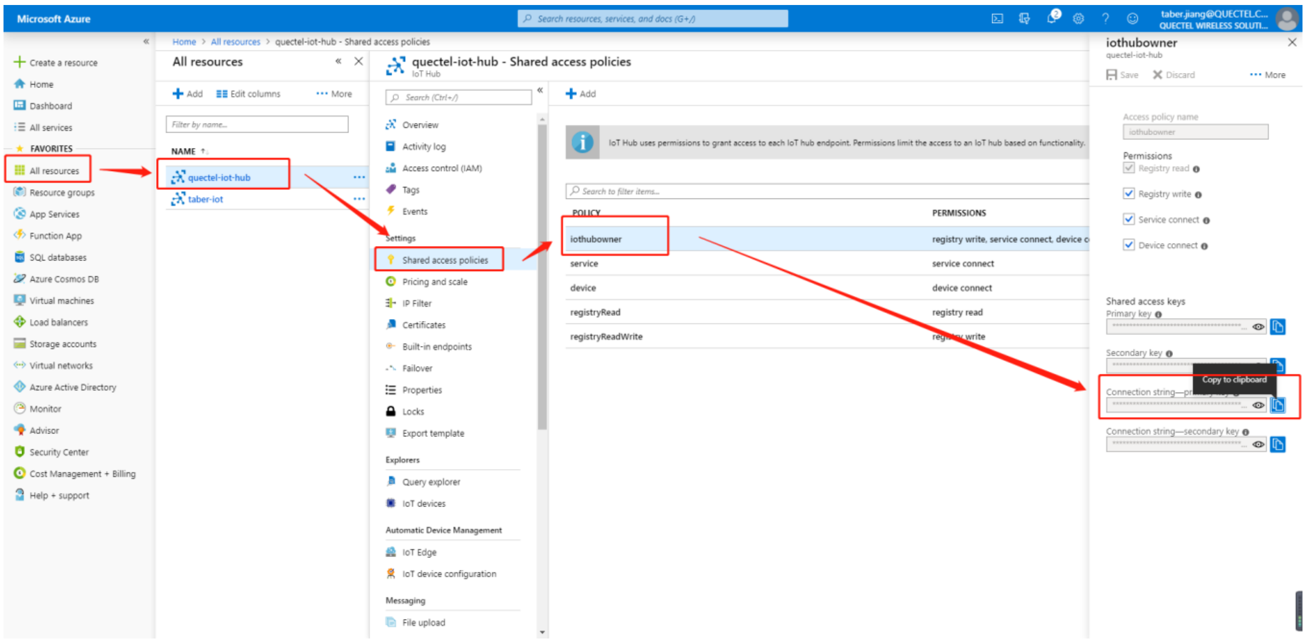
The Device Explorer tool can be downloaded from link: https://github.com/Azure/azure-iot-sdk-csharp/releases/download/2019-1-4/SetupDeviceExplorer.msi.

More details about downloading and installation can be reached from link: https://github.com/Azure/azure-iot-sdk-csharp/tree/master/tools/DeviceExplorer.

3.3.2. Configuration

a) Get "Connection string—primary key" from Azure iothubowner, and copy the connection string to clipboard.

b) Click the "**Configuration**" tab in the Device Explorer Twin window, and then paste the connection string to the input box of "IoT Hub Connection String". After that, click "**Update**" → "**OK**" to finish the operation.


Microsoft Azure
Home > All resources > quectel-iot-hub - Shared access policies
All resources
quectel-iot-hub
Shared access policies
Copy to clipboard
Connection string—primary key

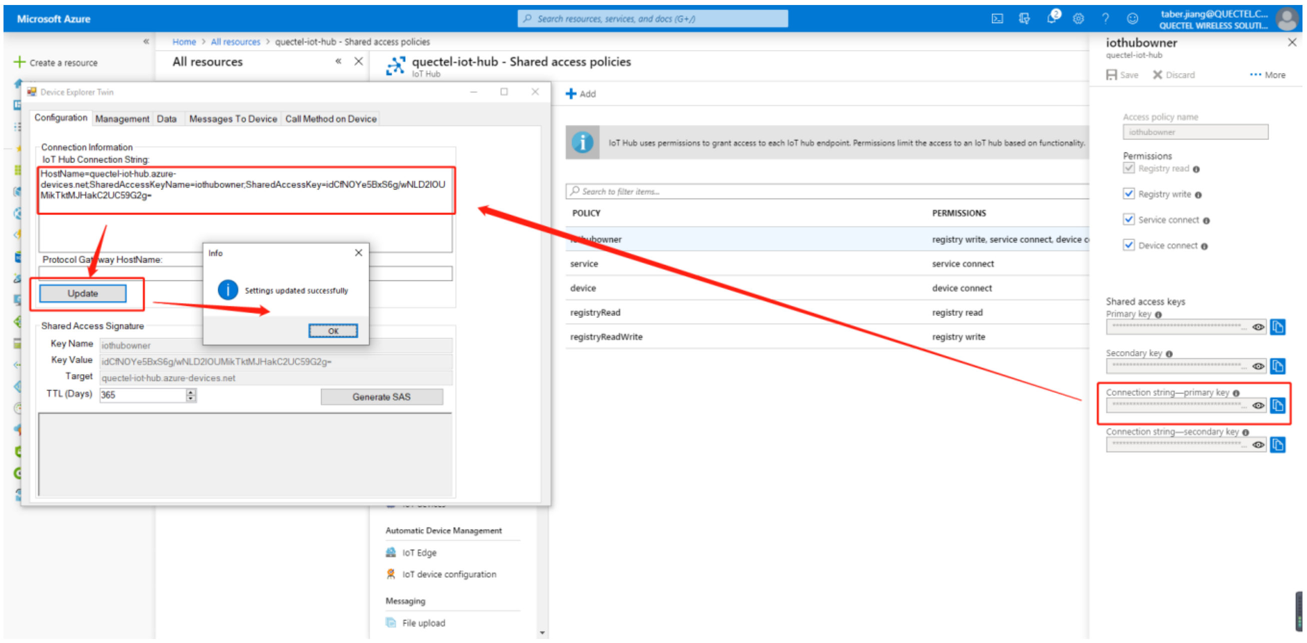
IoT Hub uses permissions to grant access to each IoT hub endpoint. Permissions limit the access to an IoT hub based on functionality.

POLICY	PERMISSIONS
iothubowner	registry write, service connect, device c
service	service connect
device	device connect
registryRead	registry read
registryReadWrite	registry write

Access policy name: iothubowner

Permissions: ✓ Registry read; ✓ Registry write; ✓ Service connect; ✓ Device connect

Figure 20: Device Explorer Configuration (Step a)



Connection Information

IoT Hub Connection String:

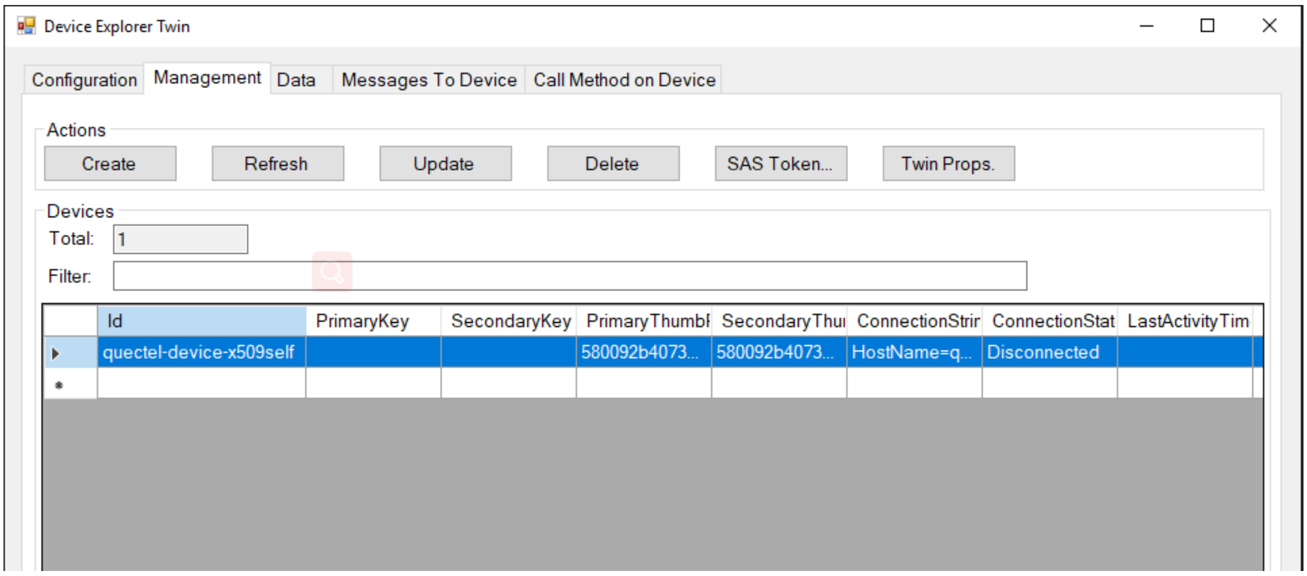
```
HostName=quectel-iot-hub.azure-
devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=idCfNOYe5BxS6g/wNLD2IOU
MikTktMJHakC2UC59G2g=
```

Settings updated successfully

Shared Access Signature

Key Name	iothubowner
Key Value	idCfNOYe5BxS6g/wNLD2IOUMikTktMJHakC2UC59G2g=
Target	quectel-iot-hub.azure-devices.net
TTL (Days)	365

Figure 21: Device Explorer Configuration (Step b)



	Id	PrimaryKey	SecondaryKey	PrimaryThumbl	SecondaryThu	ConnectionStrir	ConnectionStat	LastActivityTim
▸	quectel-device-x509self			580092b4073...	580092b4073...	HostName=q...	Disconnected	
*								

Figure 22: Use Device Explorer

3.4. Usage of MQTT.fx Tool

MQTT.fx is a MQTT Client written in Java based on Eclipse Paho. This chapter describes the usage of the MQTT.fx tool which will be used as a MQTT client.

3.4.1. Download/Install

MQTT.fx tool can be downloaded from http://mqttfx.jensd.de/index.php/download.

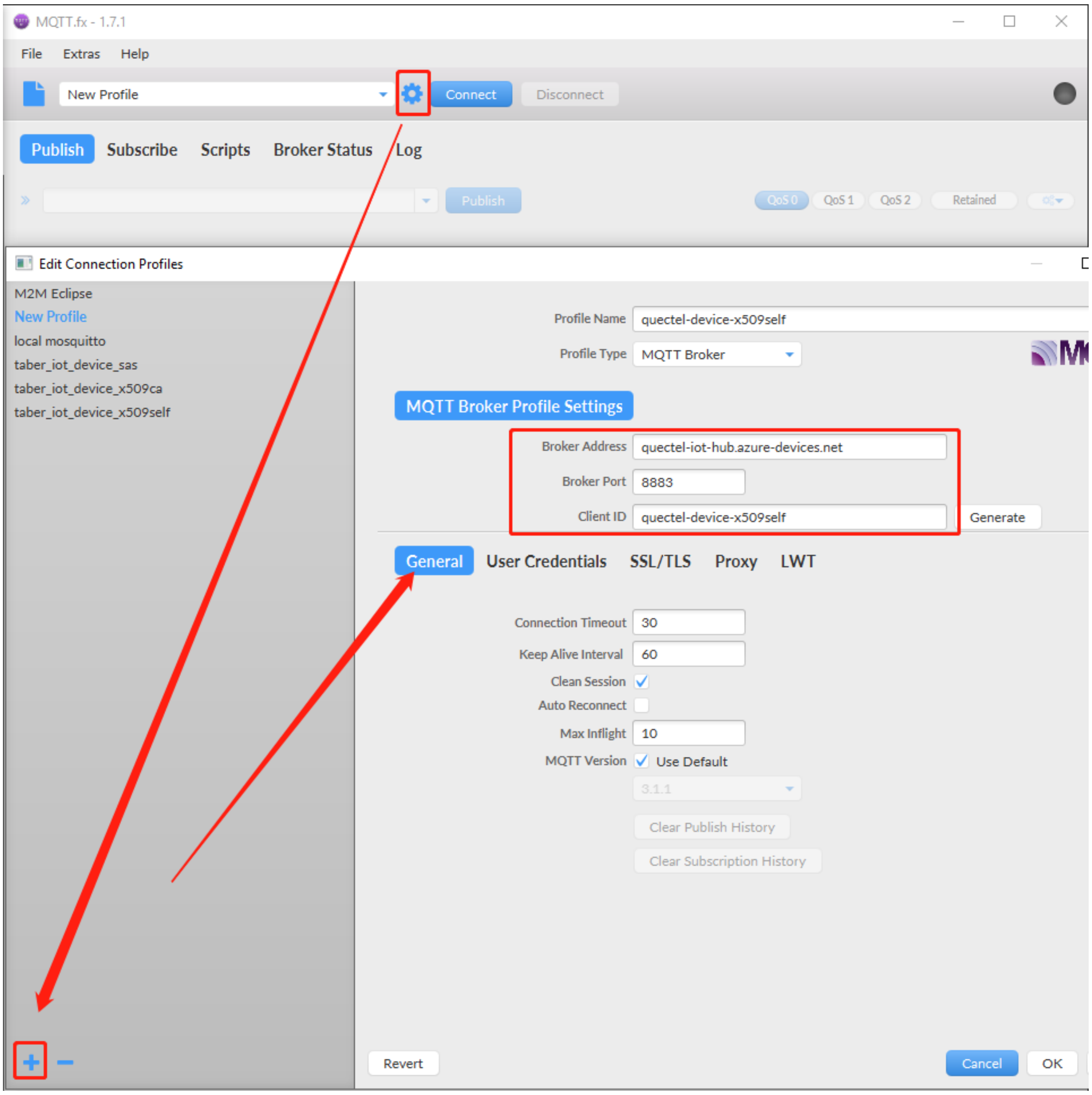
3.4.2. Edit Connection Profiles

- **MQTT Broker Profile Settings**

  Broker Address: {customized hub name}.azure-devices.net
  Broker Port: 8883
  Client ID: {device_id}

- **General**

  Use the default settings.

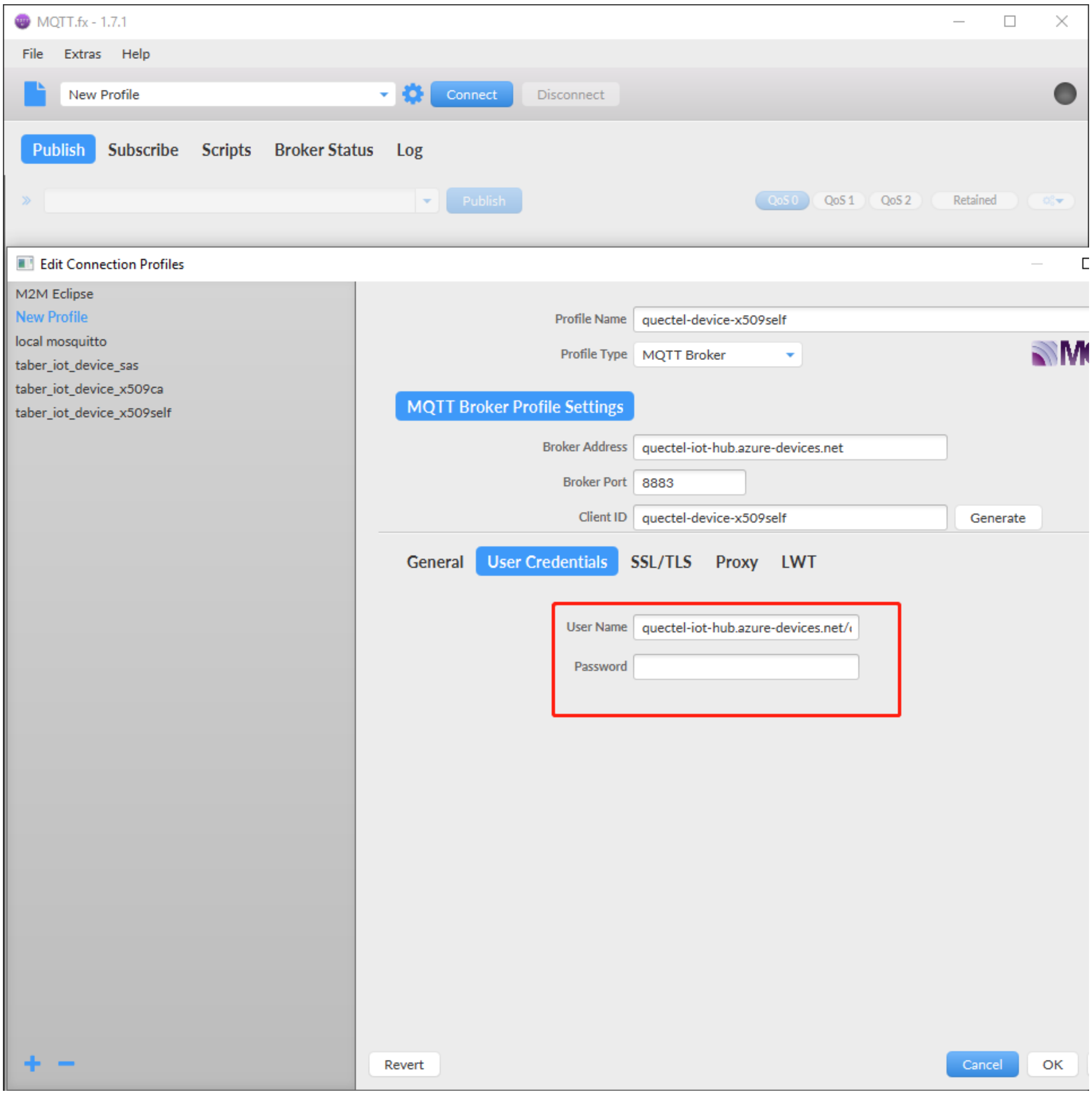


Profile Name: quectel-device-x509self
Profile Type: MQTT Broker
Broker Address: quectel-iot-hub.azure-devices.net
Broker Port: 8883
Client ID: quectel-device-x509self
Connection Timeout: 30
Keep Alive Interval: 60
Clean Session: ✓
Auto Reconnect:
Max Inflight: 10
MQTT Version: ✓ Use Default
3.1.1

Figure 23: MQTT Broker Profile Settings and General Configuration

- **User Credentials**

  User Name: *{customized hub name}.azure-devices.net/{device_id}/?api-version=2018-06-30*
  Password: no password required.

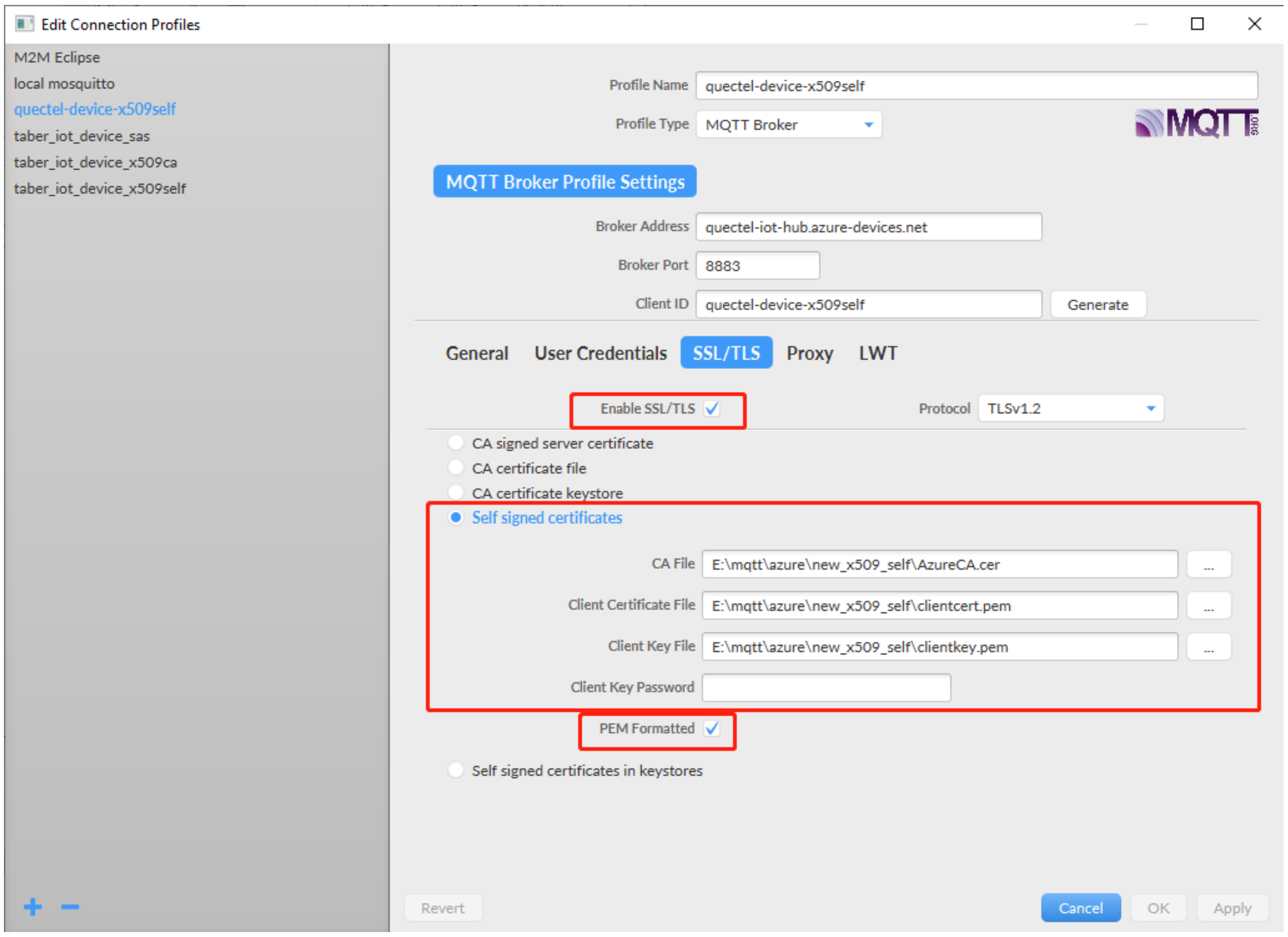


Profile Name: quectel-device-x509self
Profile Type: MQTT Broker
Broker Address: quectel-iot-hub.azure-devices.net
Broker Port: 8883
Client ID: quectel-device-x509self
User Name: quectel-iot-hub.azure-devices.net/
Password:

Figure 24: User Credentials Configuration

- **SSL/TLS**

  Select "**Enable SSL/TLS**" option
  Check "**Self signed certificates**"
  CA File: *AzureCA.cer*
  Client Certificate File: *clientcert.pem*
  Client Key File: *clientkey.pem*


Edit Connection Profiles

M2M Eclipse
local mosquitto
quectel-device-x509self
taber_iot_device_sas
taber_iot_device_x509ca
taber_iot_device_x509self

Profile Name: quectel-device-x509self
Profile Type: MQTT Broker

MQTT Broker Profile Settings
Broker Address: quectel-iot-hub.azure-devices.net
Broker Port: 8883
Client ID: quectel-device-x509self

General | User Credentials | SSL/TLS | Proxy | LWT

Enable SSL/TLS ✓ Protocol: TLSv1.2

CA signed server certificate
CA certificate file
CA certificate keystore
● Self signed certificates

CA File: E:\mqtt\azure\new_x509_self\AzureCA.cer
Client Certificate File: E:\mqtt\azure\new_x509_self\clientcert.pem
Client Key File: E:\mqtt\azure\new_x509_self\clientkey.pem
Client Key Password:

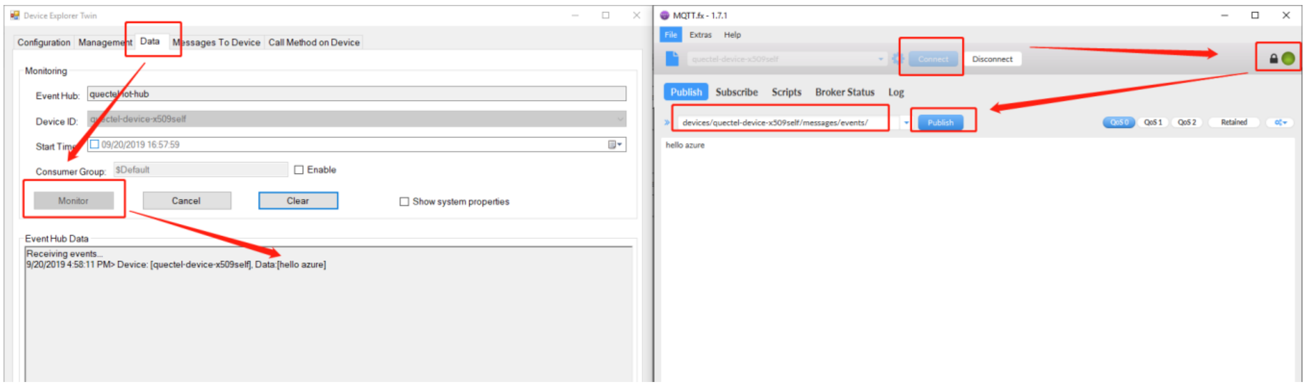
PEM Formatted ✓

Self signed certificates in keystores

Figure 25: SSL/TLS Configuration

3.4.3. Send Device-to-Cloud Messages

a) Click "**Data**" of Device Explorer tool, and then click "**Monitor**".
b) Click "**Connect**" of MQTT.fx tool and then input the topic *devices/{device_id}/messages/events/.* Finally click "**Publish**".



Monitoring

Event Hub: quectel-iot-hub

Device ID: quectel-device-x509self

Start Time: 09/20/2019 16:57:59

Consumer Group: $Default

Event Hub Data

```
Receiving events...
9/20/2019 4:58:11 PM> Device: [quectel-device-x509self], Data:[hello azure]
```

quectel-device-x509self

devices/quectel-device-x509self/messages/events/

hello azure

Figure 26: Send Messages with MQTT.fx

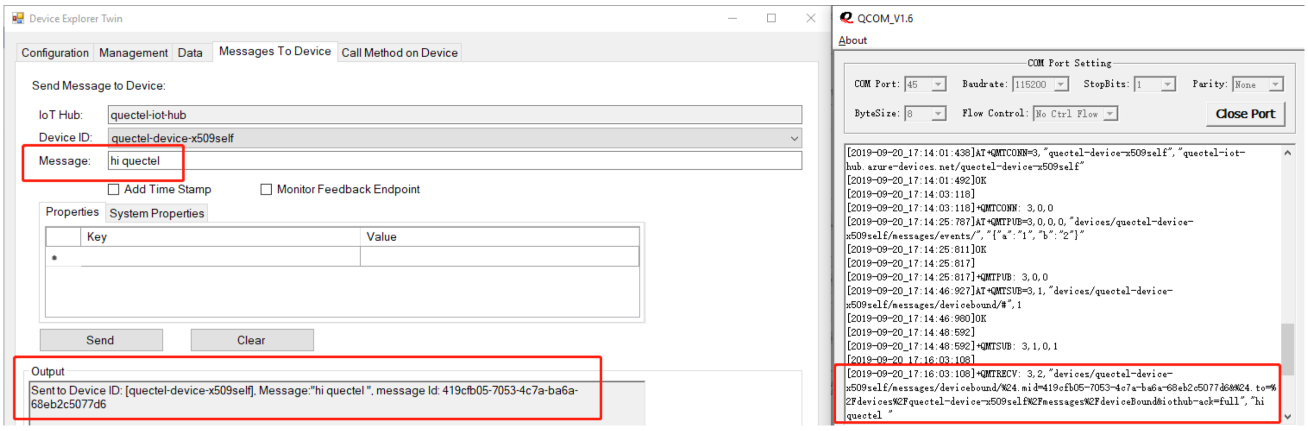
3.5. Use BC66/BC66-NA as MQTT Client

3.5.1. AT Command Example

```
AT+QSCLK=0                          //Disable sleep mode.
OK

//Configure certificates and keys
AT+QSSLCFG=1,5,"seclevel",2         //Manage server and client authentication.
OK
AT+QSSLCFG=1,5,"cacert"             //Configure CA certificate.
>                                   //Input the content of trusted CA certificate <AzureCA.cer> in
                                    PEM format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"cacert",1282

OK
AT+QSSLCFG=1,5,"clientcert"         //Configure client certificate.
>                                   //Input the content of the client certificate <clientcert.pem> in
                                    PEM format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"clientcert ",1216

OK
AT+QSSLCFG=1,5,"clientkey"          //Configure client private key.
>                                   //Input the content of the client private key <clientkey.pem> in
                                    PEM format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"clientkey",1679

OK
AT+QSCLK=1                          //Enable light sleep and deep sleep, and wakeup by PSM_EINT
```

```
                                        (falling edge).
OK
AT+QMTCFG="ssl",3,1,1,5                 //Enable SSL and configure SSL context/connect index.
OK
AT+QMTCFG="version",3,4                 //Configure the MQTT version. Azure IoT Hub supports MQTT
                                        v3.1.1 only.
OK
AT+QMTOPEN=3,"quectel-iot-hub.azure-devices.net",8883        //Open a network for Azure MQTT
                                                              client with TLS 1.2.
OK

+QMTOPEN: 3,0                           //Opened the MQTT client network successfully.
AT+QMTCONN=3,"quectel-device-x509self","quectel-iot-hub.azure-devices.net/quectel-device-x5
09self"
OK

+QMTCONN: 3,0,0                         //Connected the client to MQTT server successfully.
AT+QMTSUB=3,1,"devices/quectel-device-x509self/messages/devicebound/#",1
OK

+QMTSUB: 3,1,0,1

+QMTRECV: 3,2,"devices/quectel-device-x509self/messages/devicebound/%24.mid=419cfb05-70
53-4c7a-ba6a-68eb2c5077d6&%24.to=%2Fdevices%2Fquectel-device-x509self%2Fmessages%2F
deviceBound&iothub-ack=full","hi quectel"          //Received cloud-to-device messages.

AT+QMTPUB=3,0,0,0,"devices/quectel-device-x509self/messages/events/","{"a":"1","b":"2"}"
OK

+QMTPUB: 3,0,0
AT+QMTPUB=3,0,0,0,"devices/quectel-device-x509self/messages/events/"   //Publish the message
                                                                       in data mode.
>
hello azure iot hub                     //Input the data to be published and then tap CTRL+Z to send.
OK

+QMTPUB: 3,0,0
AT+QMTDISC=3                            //Disconnect the client from MQTT server.
OK

+QMTDISC: 3,0                           //Connection closed successfully.
```

3.5.2. Receive Cloud-to-Device Messages



Send Message to Device:

IoT Hub: quectel-iot-hub

Device ID: quectel-device-x509self

Message: hi quectel

Output

```
Sent to Device ID: [quectel-device-x509self], Message:"hi quectel ", message Id: 419cfb05-7053-4c7a-ba6a-
68eb2c5077d6
```

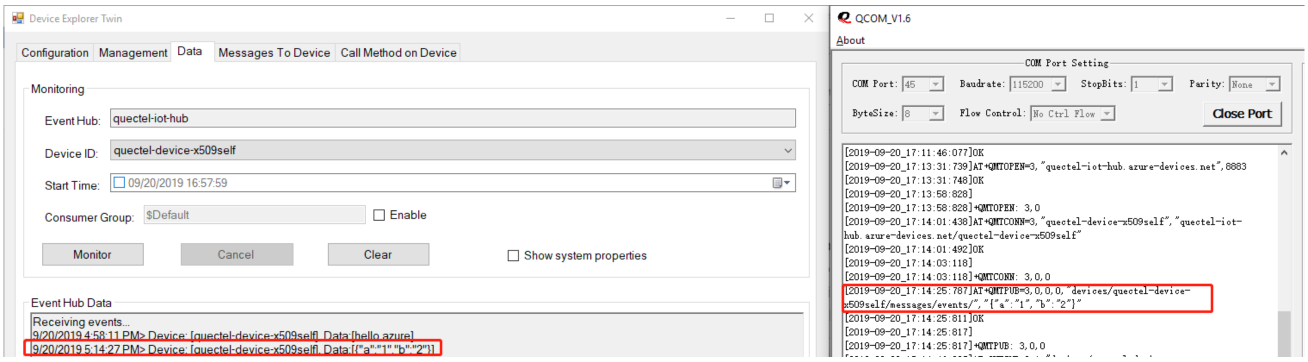
COM Port: 45 Baudrate: 115200 StopBits: 1 Parity: None

ByteSize: 8 Flow Control: No Ctrl Flow

```
[2019-09-20_17:14:01:438]AT+QMTCONN=3,"quectel-device-x509self","quectel-iot-
hub.azure-devices.net/quectel-device-x509self"
[2019-09-20_17:14:01:492]OK
[2019-09-20_17:14:03:118]
[2019-09-20_17:14:03:118]+QMTCONN: 3,0,0
[2019-09-20_17:14:25:787]AT+QMTPUB=3,0,0,0,"devices/quectel-device-
x509self/messages/events/","{"a":"1","b":"2"}"
[2019-09-20_17:14:25:811]OK
[2019-09-20_17:14:25:817]
[2019-09-20_17:14:25:817]+QMTPUB: 3,0,0
[2019-09-20_17:14:46:927]AT+QMTSUB=3,1,"devices/quectel-device-
x509self/messages/devicebound/#",1
[2019-09-20_17:14:46:980]OK
[2019-09-20_17:14:48:592]
[2019-09-20_17:14:48:592]+QMTSUB: 3,1,0,1
[2019-09-20_17:16:03:108]
[2019-09-20_17:16:03:108]+QMTRECV: 3,2,"devices/quectel-device-
x509self/messages/devicebound/%24.mid=419cfb05-7053-4c7a-ba6a-68eb2c5077d6&%24.to=%
2Fdevices%2Fquectel-device-x509self%2Fmessages%2FdeviceBound&iothub-ack=full","hi
quectel "
```

Figure 27: Module Receives Messages from IoT Cloud Platform

3.5.3. Send Device-to-Cloud Messages



Monitoring

Event Hub: quectel-iot-hub

Device ID: quectel-device-x509self

Start Time: 09/20/2019 16:57:59

Consumer Group: $Default

Event Hub Data

```
Receiving events...
9/20/2019 4:58:11 PM> Device: [quectel-device-x509self], Data:[hello azure]
9/20/2019 5:14:27 PM> Device: [quectel-device-x509self], Data:[{"a":"1","b":"2"}]
```

COM Port: 45 Baudrate: 115200 StopBits: 1 Parity: None

ByteSize: 8 Flow Control: No Ctrl Flow

```
[2019-09-20_17:11:46:077]OK
[2019-09-20_17:13:31:739]AT+QMTOPEN=3,"quectel-iot-hub.azure-devices.net",8883
[2019-09-20_17:13:31:748]OK
[2019-09-20_17:13:58:828]
[2019-09-20_17:13:58:828]+QMTOPEN: 3,0
[2019-09-20_17:14:01:438]AT+QMTCONN=3,"quectel-device-x509self","quectel-iot-
hub.azure-devices.net/quectel-device-x509self"
[2019-09-20_17:14:01:492]OK
[2019-09-20_17:14:03:118]
[2019-09-20_17:14:03:118]+QMTCONN: 3,0,0
[2019-09-20_17:14:25:787]AT+QMTPUB=3,0,0,0,"devices/quectel-device-
x509self/messages/events/","{"a":"1","b":"2"}"
[2019-09-20_17:14:25:811]OK
[2019-09-20_17:14:25:817]
[2019-09-20_17:14:25:817]+QMTPUB: 3,0,0
```

Figure 28: Module Sends Messages to the IoT Cloud Platform

4 Connection with AWS IoT Core

This chapter introduces the steps to establish connection with AWS IoT core through TLS/SSL secured MQTT.

4.1. Create AWS Account

4.1.1. Create a Free Account

Create a free AWS account in https://aws.amazon.com/.

4.1.2. Enter AWS Console

Enter the AWS Console via link https://console.aws.amazon.com/console/home. Then click "**My Account**" → "**AWS Management Console**".

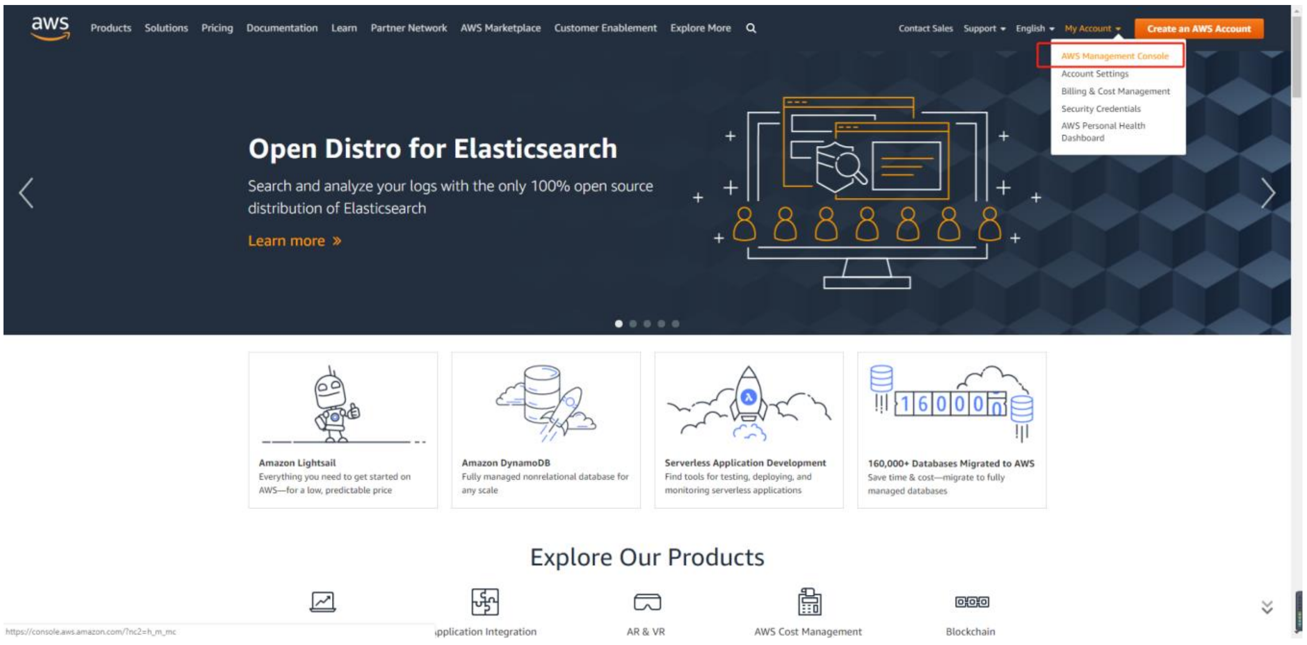
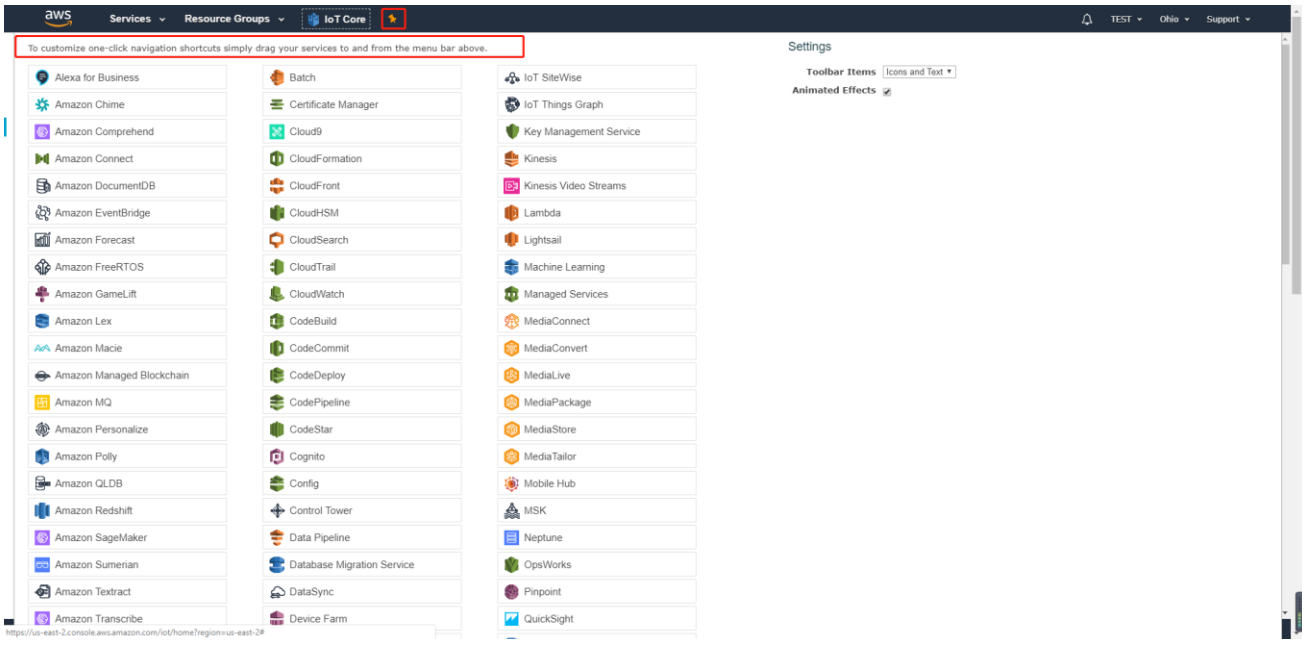

aws
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Marketplace
Customer Enablement
Explore More
Contact Sales
Support
English
My Account
Create an AWS Account
AWS Management Console
Account Settings
Billing & Cost Management
Security Credentials
AWS Personal Health Dashboard
Open Distro for Elasticsearch
Search and analyze your logs with the only 100% open source distribution of Elasticsearch
Learn more »
Amazon Lightsail
Everything you need to get started on AWS—for a low, predictable price
Amazon DynamoDB
Fully managed nonrelational database for any scale
Serverless Application Development
Find tools for testing, deploying, and monitoring serverless applications
160,000+ Databases Migrated to AWS
Save time & cost—migrate to fully managed databases
Explore Our Products
Application Integration
AR & VR
AWS Cost Management
Blockchain

Figure 29: AWS Console

4.1.3. Create a Shortcut for "IoT Core" Resource

In the "**AWS Management Console**" as shown below, a shortcut for "**IoT Core**" can be created simply by dragging it to the menu bar.


To customize one-click navigation shortcuts simply drag your services to and from the menu bar above.

Settings
Toolbar Items: Icons and Text
Animated Effects: ✓

Figure 30: Create a Shortcut for "IoT Core" Resource

4.1.4. Create Things

a) Click "**IoT Core**" menu.
b) Click "**Manage**"→"**Things**" in the left navigation bar.
c) Click "**Create**".
d) Click "**Create a single thing**".
   STEP 1: Fill "**Name**", and then click "**Next**".
   STEP 2: Click "**Create certificate**", and then click "**Download**"→"**Activate**".
e) Click "**Done**".

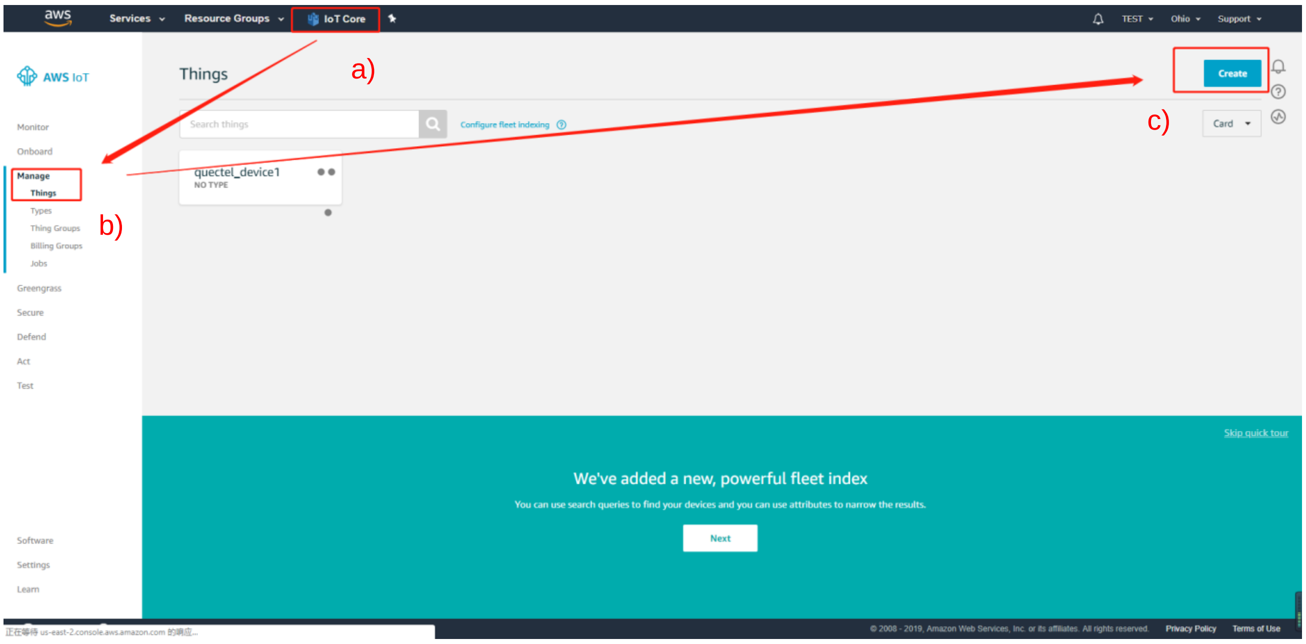
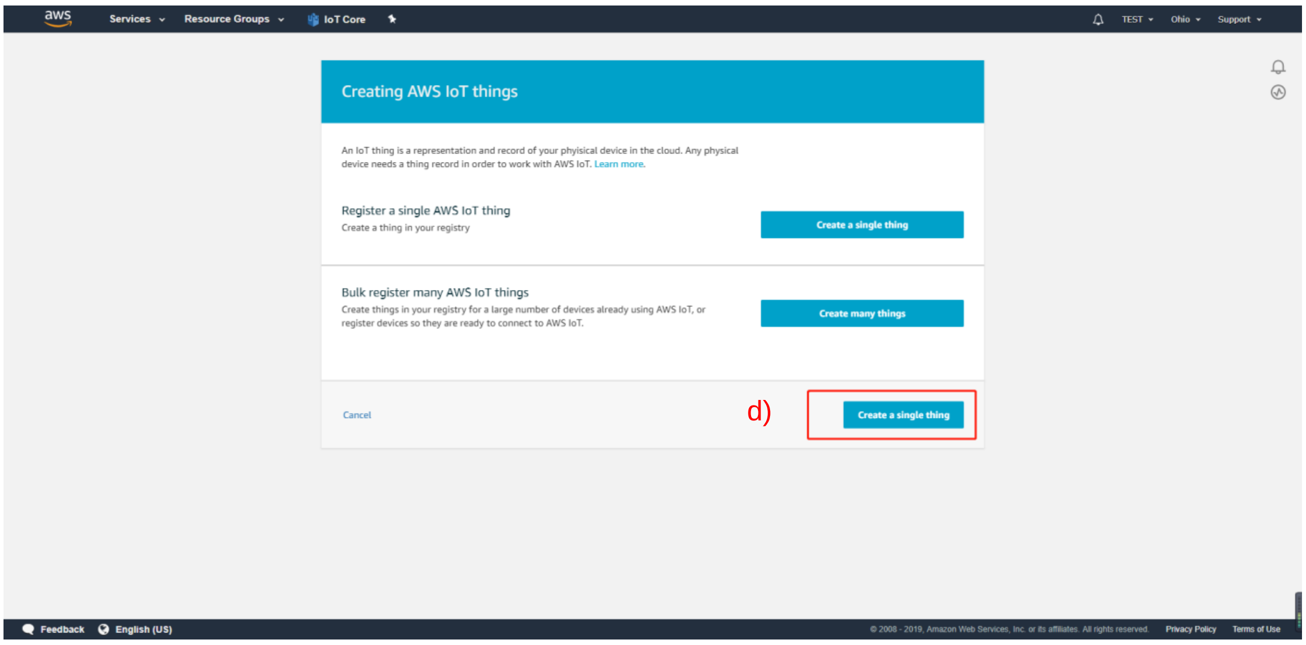


Figure 31: Create Things (Step a to c)

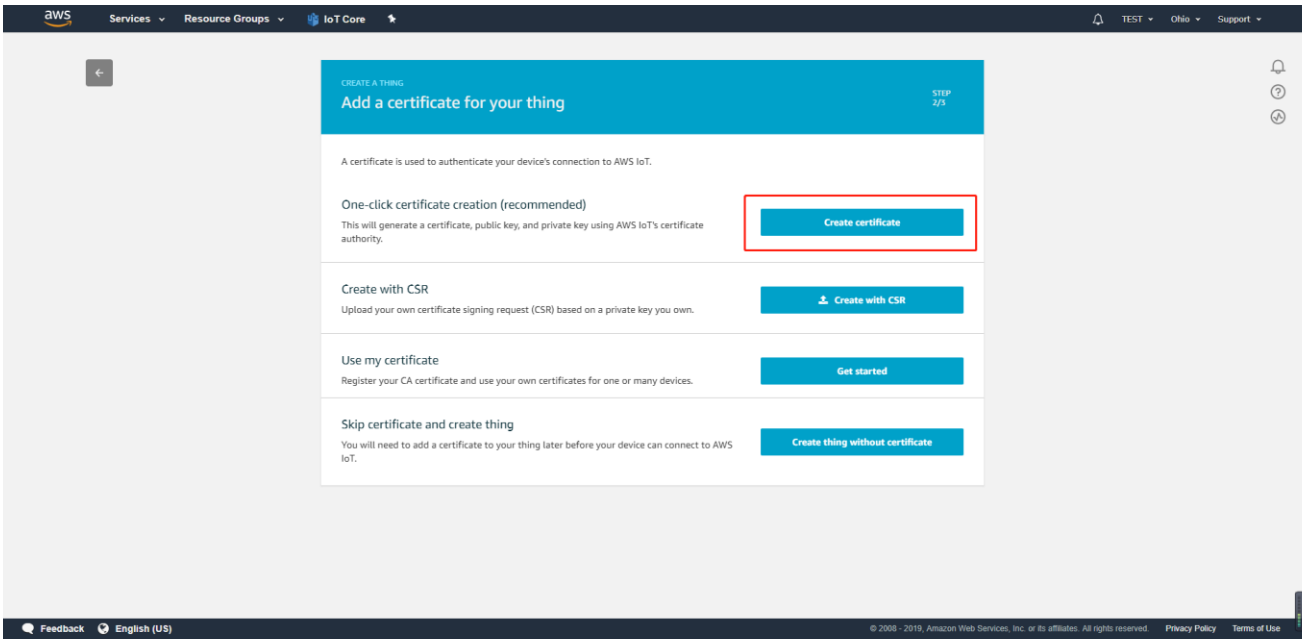


An IoT thing is a representation and record of your phyisical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

Register a single AWS IoT thing
Create a thing in your registry
Create a single thing

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.
Create many things

Figure 32: Create Things (Step d-1)



CREATE A THING

Add a certificate for your thing

STEP 2/3

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.

Create with CSR

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.

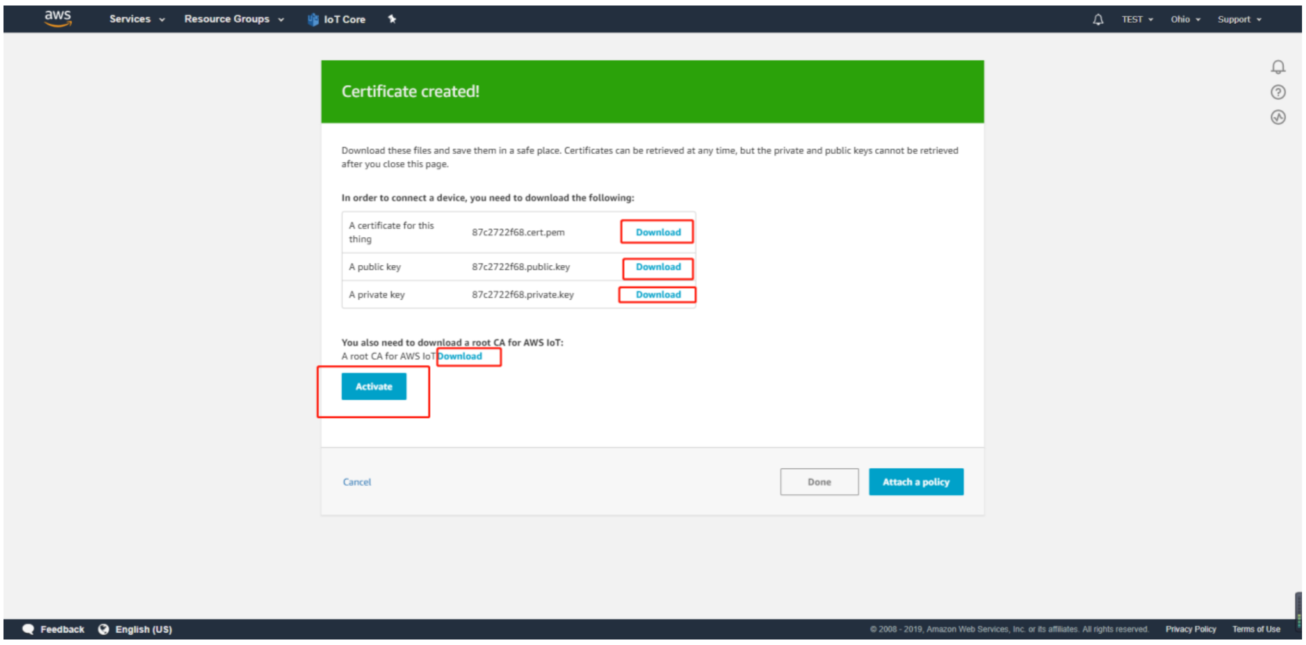
Get started

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT.

Create thing without certificate

Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Figure 33: Create Things (Step d-2)



Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	87c2722f68.cert.pem	Download
A public key	87c2722f68.public.key	Download
A private key	87c2722f68.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT Download

Activate

Cancel   Done   Attach a policy

Feedback
English (US)
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Figure 34: Create Things (Step e)

4.1.5. Create Policies

a) Click "**IoT Core**" menu.

b) Click "**Secure**"→"**Policies**" in the left navigation bar.

c) Click "**Create**".

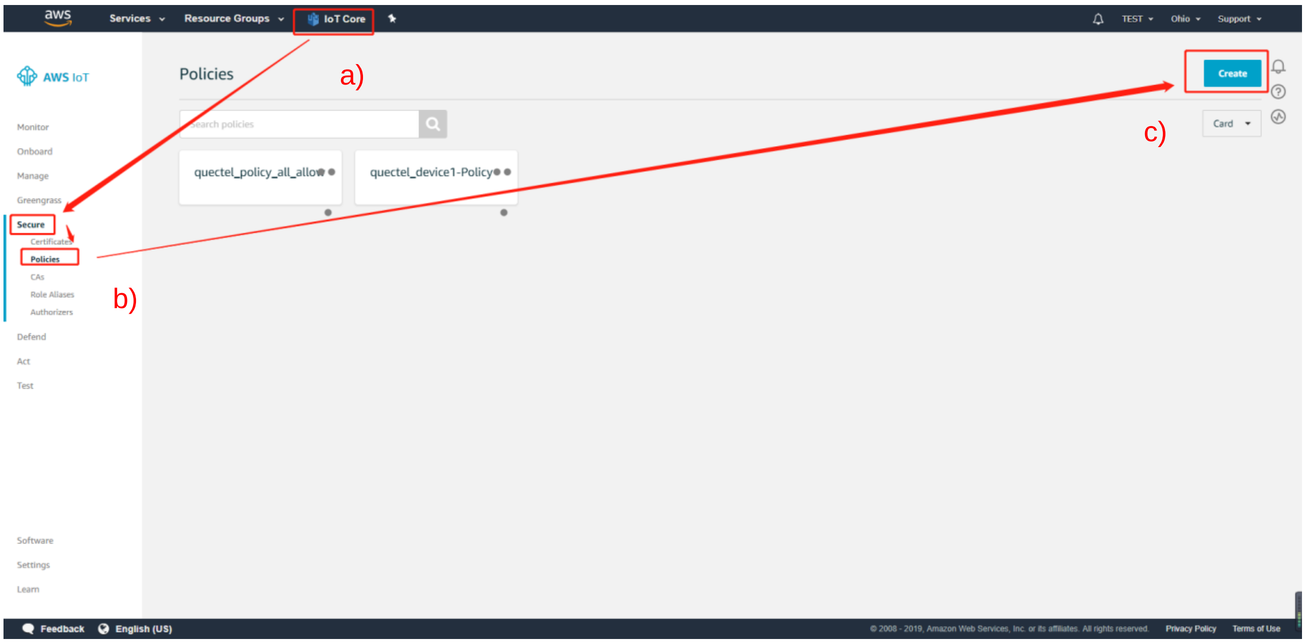
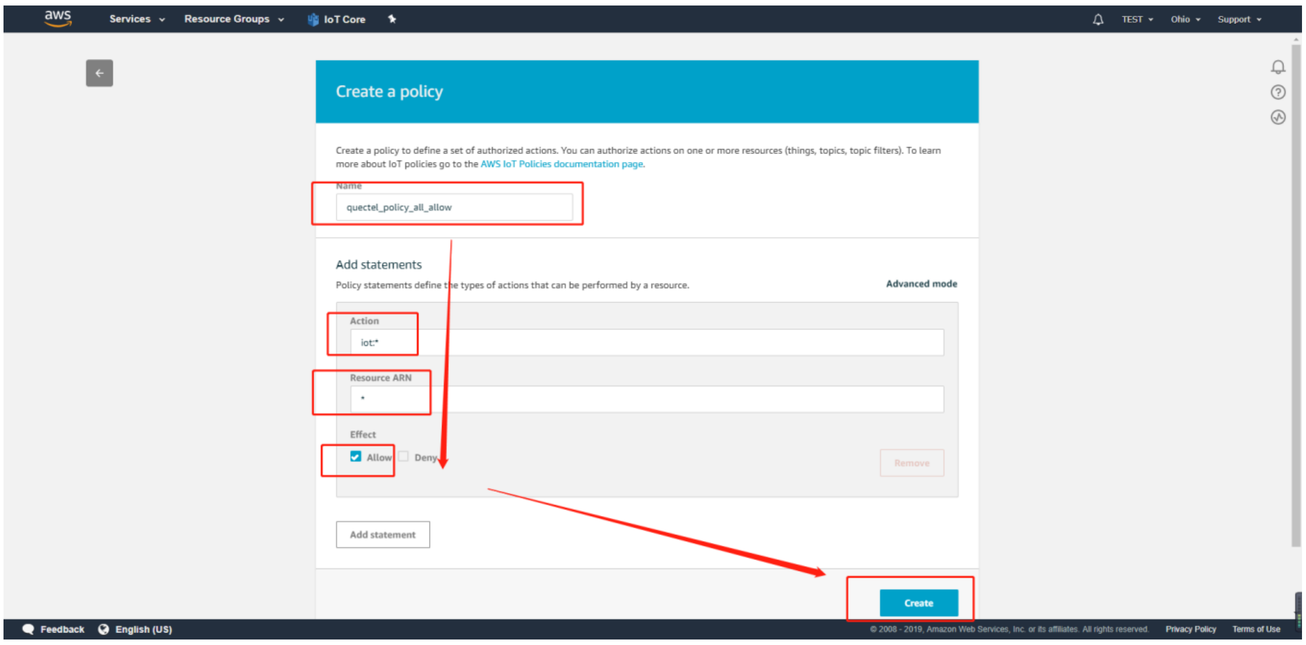


Figure 35: Create Policies (Step a to c)



Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: quectel_policy_all_allow

Add statements

Policy statements define the types of actions that can be performed by a resource.

Action: iot:*

Resource ARN: *

Effect: [✓ Allow] Deny

Figure 36: Policy Created

4.1.6. Attach Policies to Certificate(s)

a) Click "**IoT Core**" menu, click "**Secure**"→"**Certificates**" in the left navigation bar, and then click a certificate to show details.

b) Click "**Actions**"→"**Attach policy**".

c) Select a policy and click "**Attach**".

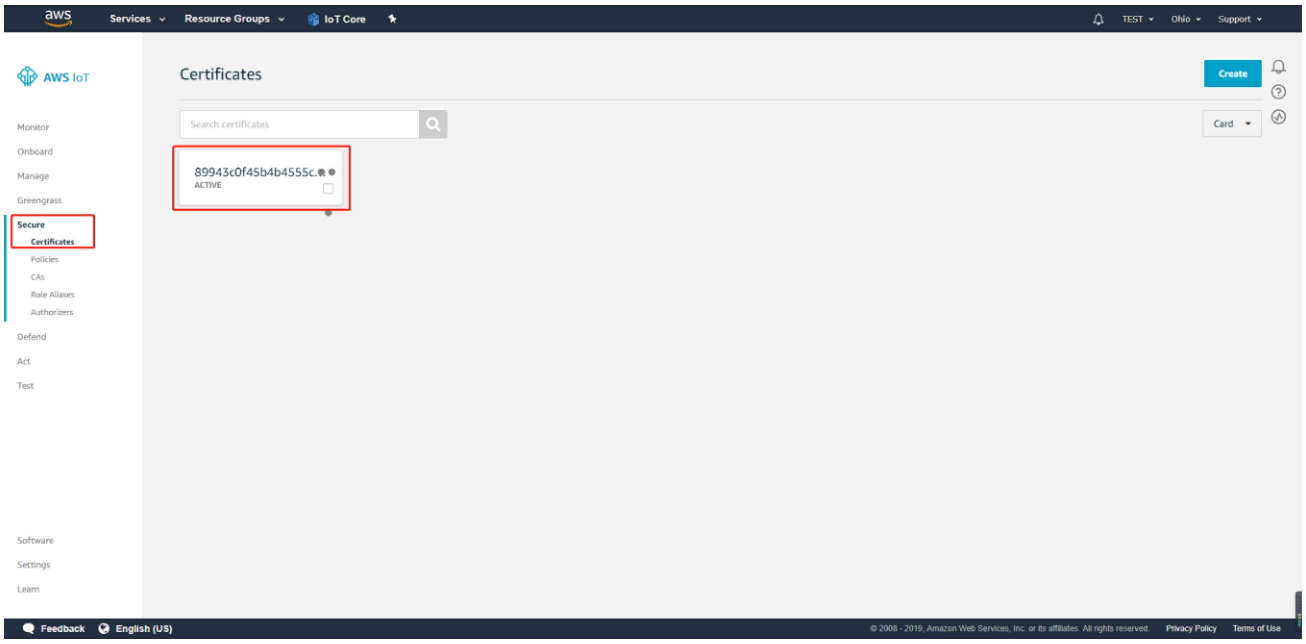
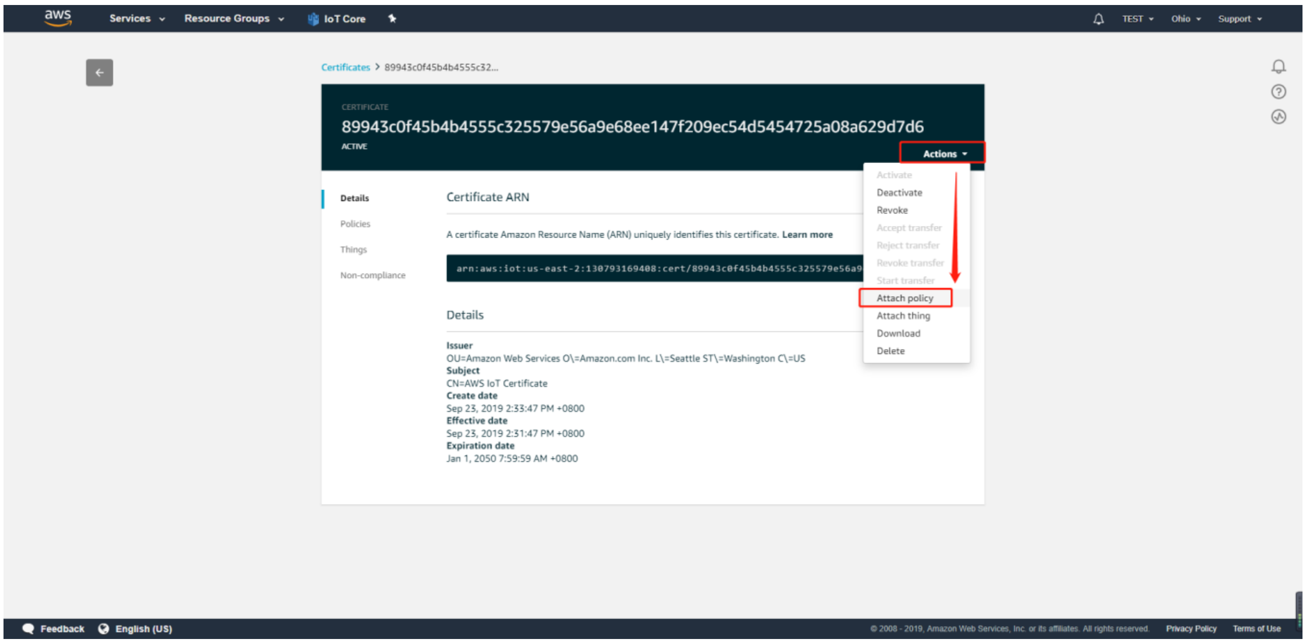


Figure 37: Attach Policies to Certificate(s) (Step a)



89943c0f45b4b4555c325579e56a9e68ee147f209ec54d5454725a08a629d7d6
ACTIVE

Certificate ARN

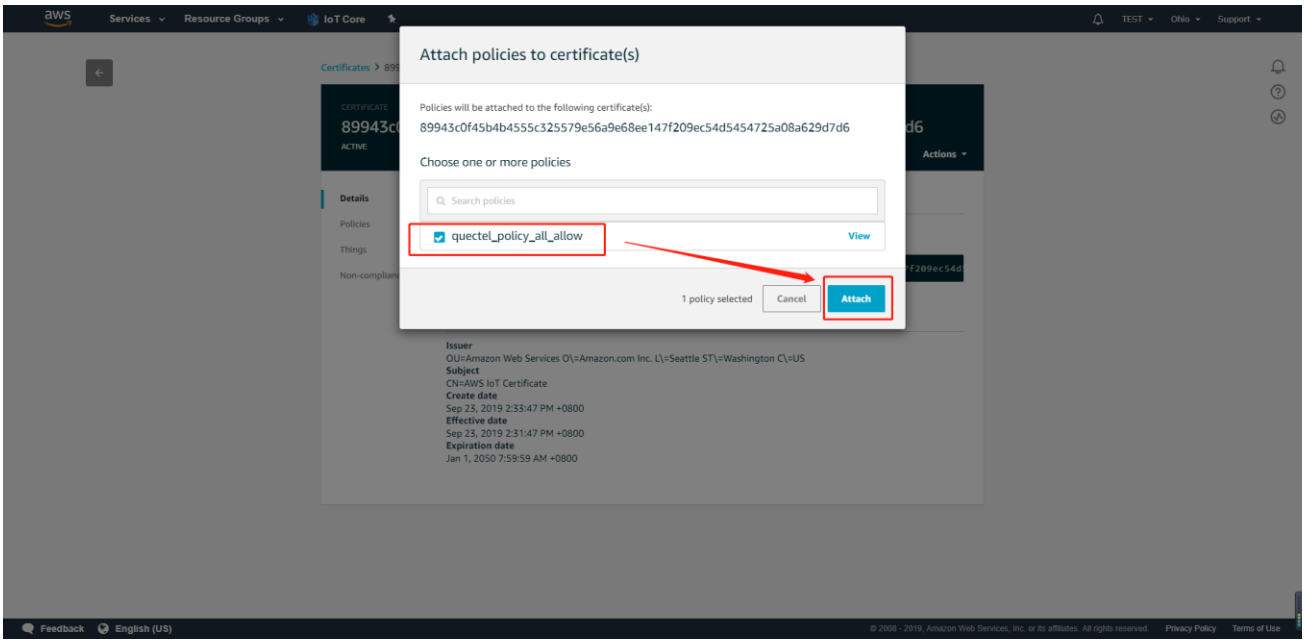
A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:us-east-2:130793169408:cert/89943c0f45b4b4555c325579e56a9
```

Details

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Sep 23, 2019 2:33:47 PM +0800
Effective date
Sep 23, 2019 2:31:47 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800

Figure 38: Attach Policies to Certificate(s) (Step b)


Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
89943c0f45b4b4555c325579e56a9e68ee147f209ec54d5454725a08a629d7d6

Choose one or more policies

quectel_policy_all_allow

1 policy selected

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Sep 23, 2019 2:33:47 PM +0800
Effective date
Sep 23, 2019 2:31:47 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800

Figure 39: Attach Policies to Certificate(s) (Step c)

4.1.7. Attach Things to Certificate(s)

a) Click "**IoT Core**" menu, click "**Secure**"→"**Certificates**" in the left navigation bar, and then click a certificate to show details.
b) Click "**Actions**"→"**Attach thing**".
c) Select a thing and click "**Attach**".

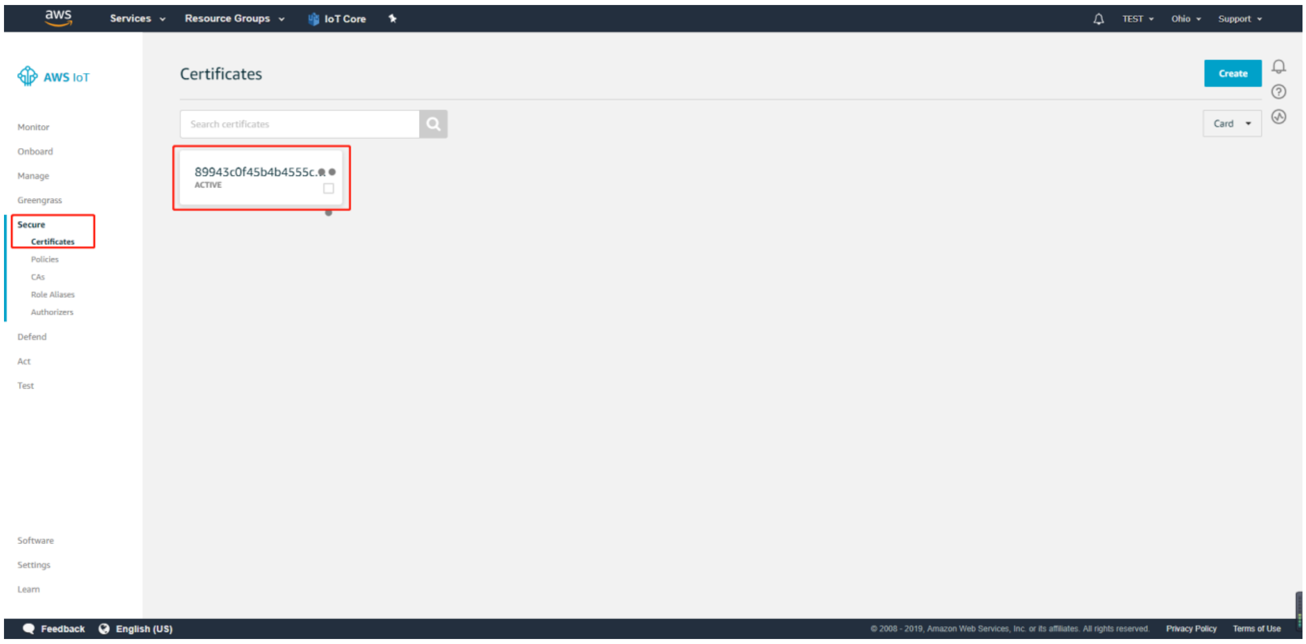
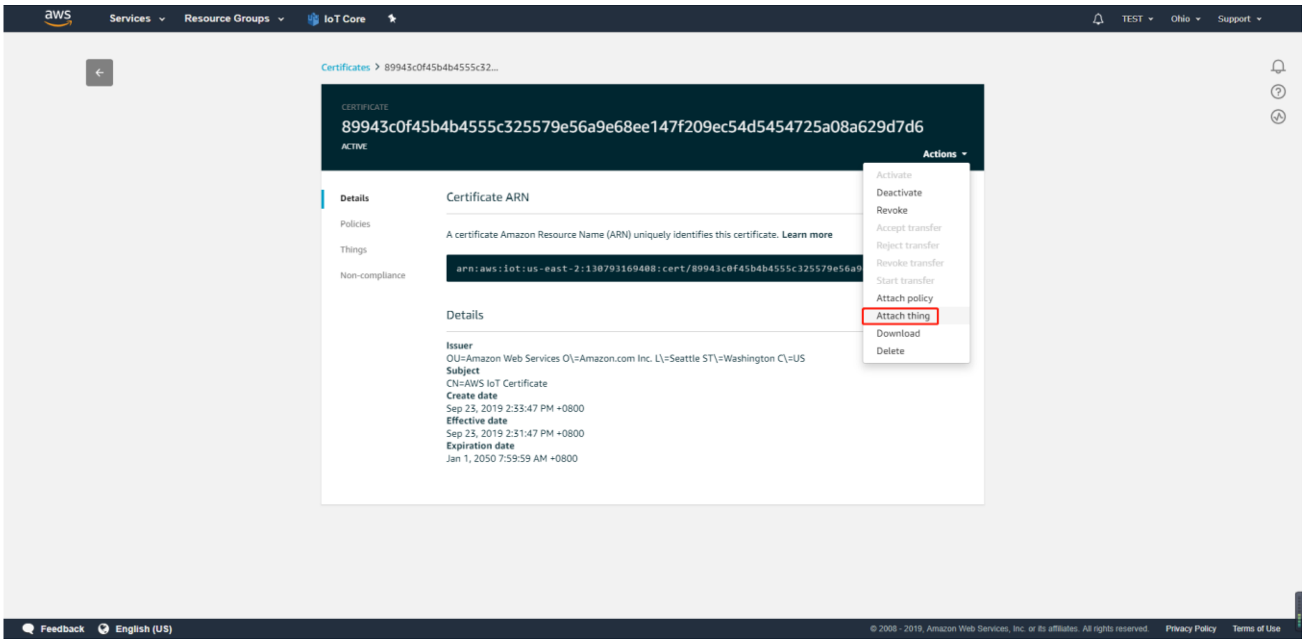


Figure 40: Attach Things to Certificate(s) (Step a)



CERTIFICATE
89943c0f45b4b4555c325579e56a9e68ee147f209ec54d5454725a08a629d7d6
ACTIVE

Certificate ARN

A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. Learn more

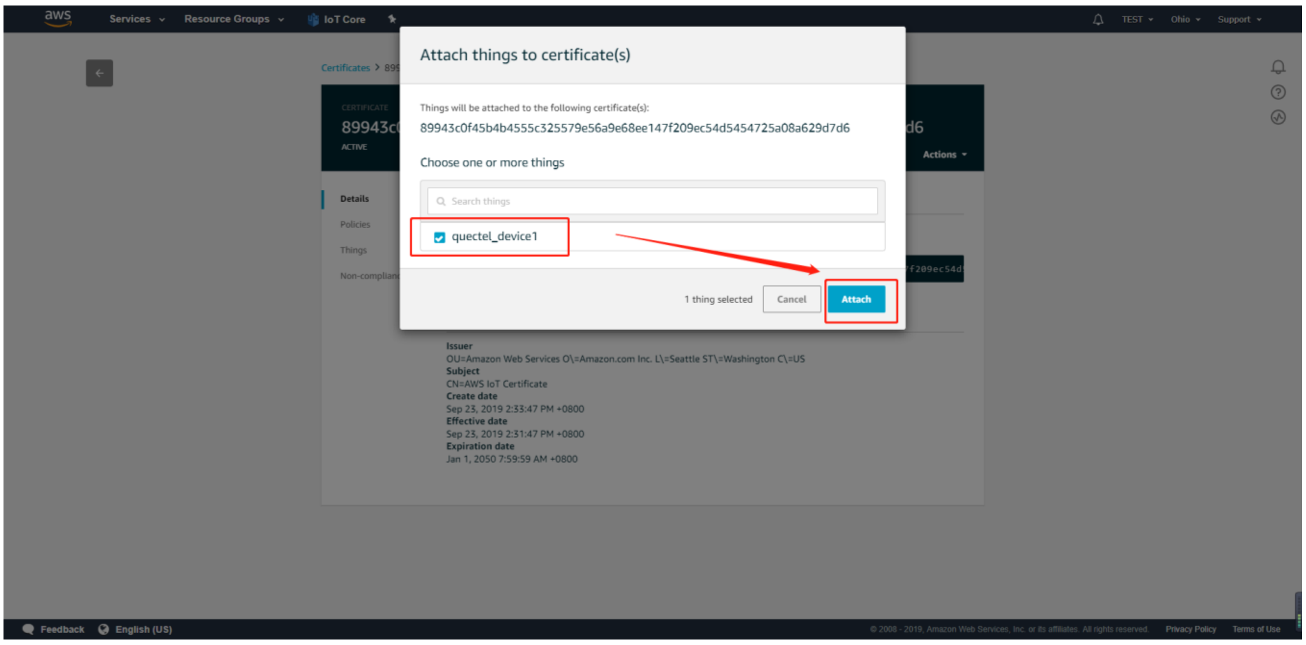
```
arn:aws:iot:us-east-2:130793169408:cert/89943c0f45b4b4555c325579e56a9
```

Details

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Sep 23, 2019 2:33:47 PM +0800
Effective date
Sep 23, 2019 2:31:47 PM +0800
Expiration date
Jan 1, 2050 7:59:59 AM +0800

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 41: Attach Things to Certificate(s) (Step b)



Attach things to certificate(s)

Things will be attached to the following certificate(s):
89943c0f45b4b4555c325579e56a9e68ee147f209ec54d5454725a08a629d7d6

Choose one or more things

Figure 42: Attach Things to Certificate(s) (Step c)

4.2. Communicate with AWS IoT Core

4.2.1. Related Resource Information

4.2.1.1. Endpoint

The endpoint of thing "**quectel_device1**" is a3pupxb4was62j-ats.iot.us-east-2.amazonaws.com.

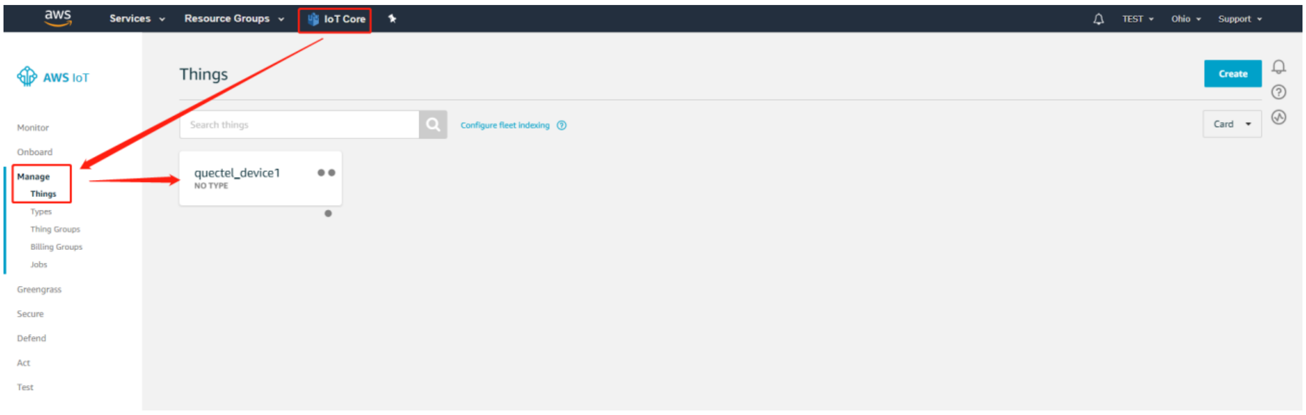
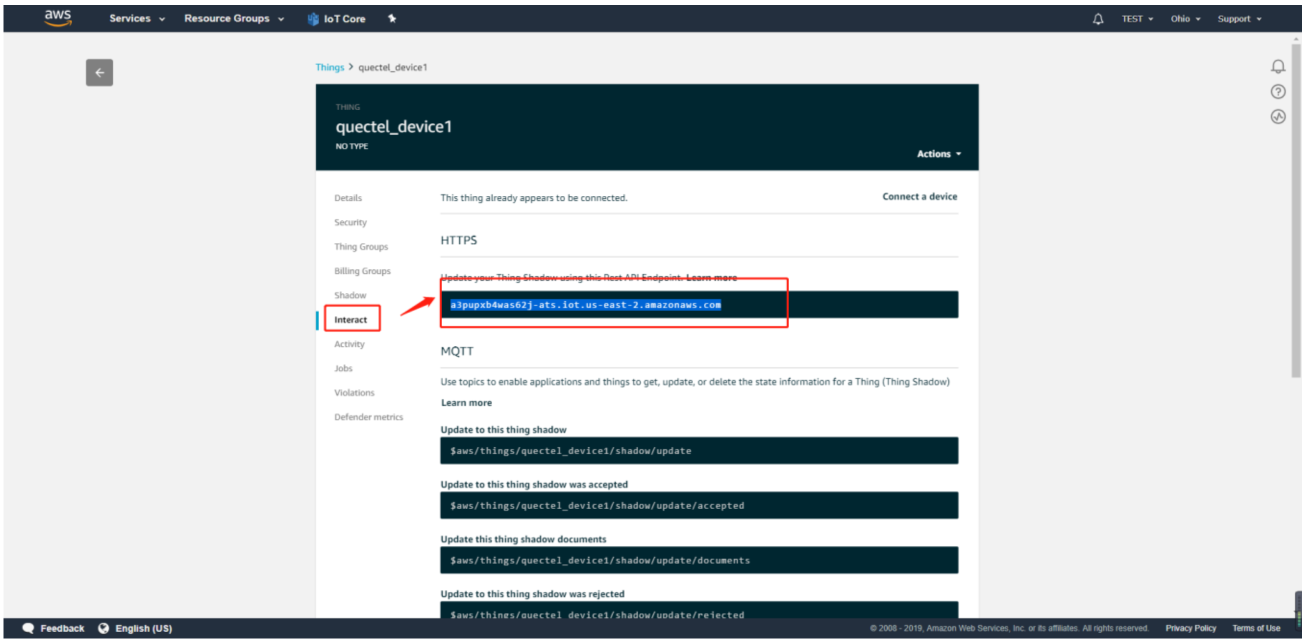


Figure 43: Check Endpoint - A



This thing already appears to be connected.

HTTPS

Update your Thing Shadow using this Rest API Endpoint. Learn more

```
a3pupxb4was62j-ats.iot.us-east-2.amazonaws.com
```

MQTT

Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)

Learn more

Update to this thing shadow

```
$aws/things/quectel_device1/shadow/update
```

Update to this thing shadow was accepted

```
$aws/things/quectel_device1/shadow/update/accepted
```

Update this thing shadow documents

```
$aws/things/quectel_device1/shadow/update/documents
```

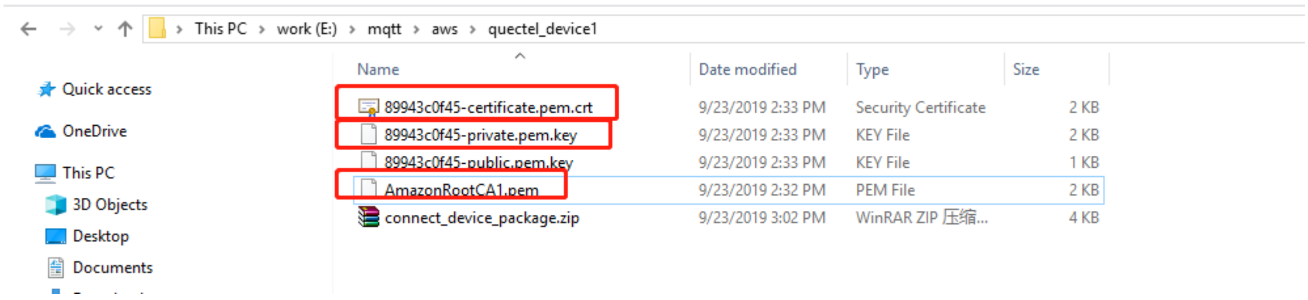
Update to this thing shadow was rejected

```
$aws/things/quectel_device1/shadow/update/rejected
```

Figure 44: Check Endpoint - B

4.2.1.2. Certificates

Please refer to Step d - 2 in ***Chapter 4.1.4***.


This PC › work (E:) › mqtt › aws › quectel_device1

Name	Date modified	Type	Size
89943c0f45-certificate.pem.crt	9/23/2019 2:33 PM	Security Certificate	2 KB
89943c0f45-private.pem.key	9/23/2019 2:33 PM	KEY File	2 KB
89943c0f45-public.pem.key	9/23/2019 2:33 PM	KEY File	1 KB
AmazonRootCA1.pem	9/23/2019 2:32 PM	PEM File	2 KB
connect_device_package.zip	9/23/2019 3:02 PM	WinRAR ZIP 压缩...	4 KB

Figure 45: Check Certificates

4.2.2. Usage of MQTT.fx Tool

MQTT.fx is a MQTT Client written in Java based on Eclipse Paho. This chapter describes the usage of the MQTT.fx tool which will be used as a MQTT client.

4.2.2.1. Download/Install

MQTT.fx tool can be downloaded from http://mqttfx.jensd.de/index.php/download.

4.2.2.2. Edit Connection Profiles

- **MQTT Broker Profile Settings**

  Broker Address: {the endpoint of your thing}
  Broker Port: 8883
  Client ID: self-defined or click "**Generate**" to create a client ID.

- **General**

  Use the defalut settings.

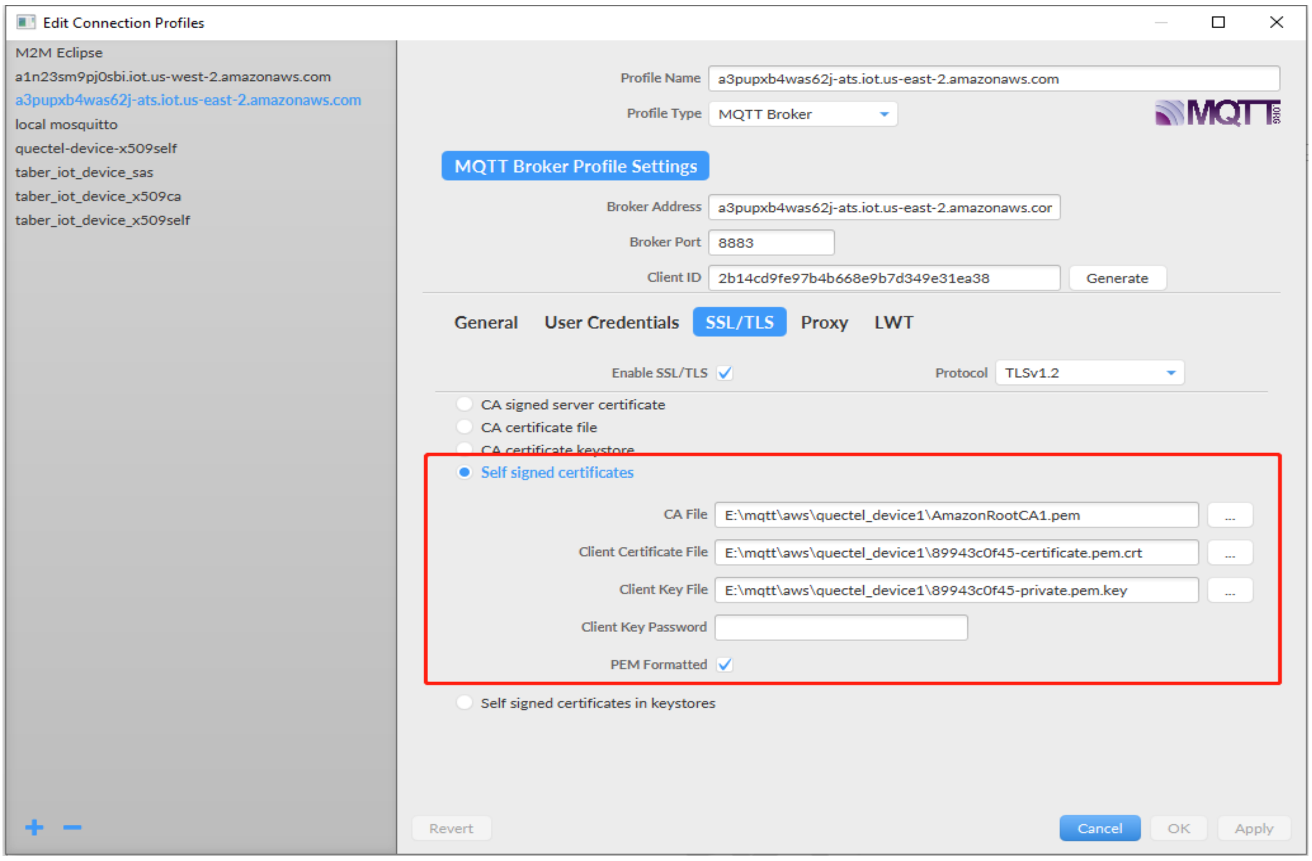
- **User Credentials**

  User Name: no password required.
  Password: no password required.

- **SSL/TLS**

  Select "**Enable SSL/TLS**" option.
  Select "**Self signed certificates**" option.

CA File: *AmazonRootCA1.pem*
Client Certificate File: *89943c0f45-certificate.pem.crt*
Client Key File: *89943c0f45-private.pem.key*


Edit Connection Profiles

M2M Eclipse
a1n23sm9pj0sbi.iot.us-west-2.amazonaws.com
a3pupxb4was62j-ats.iot.us-east-2.amazonaws.com
local mosquitto
quectel-device-x509self
taber_iot_device_sas
taber_iot_device_x509ca
taber_iot_device_x509self

Profile Name: a3pupxb4was62j-ats.iot.us-east-2.amazonaws.com
Profile Type: MQTT Broker

MQTT Broker Profile Settings

Broker Address: a3pupxb4was62j-ats.iot.us-east-2.amazonaws.cor
Broker Port: 8883
Client ID: 2b14cd9fe97b4b668e9b7d349e31ea38

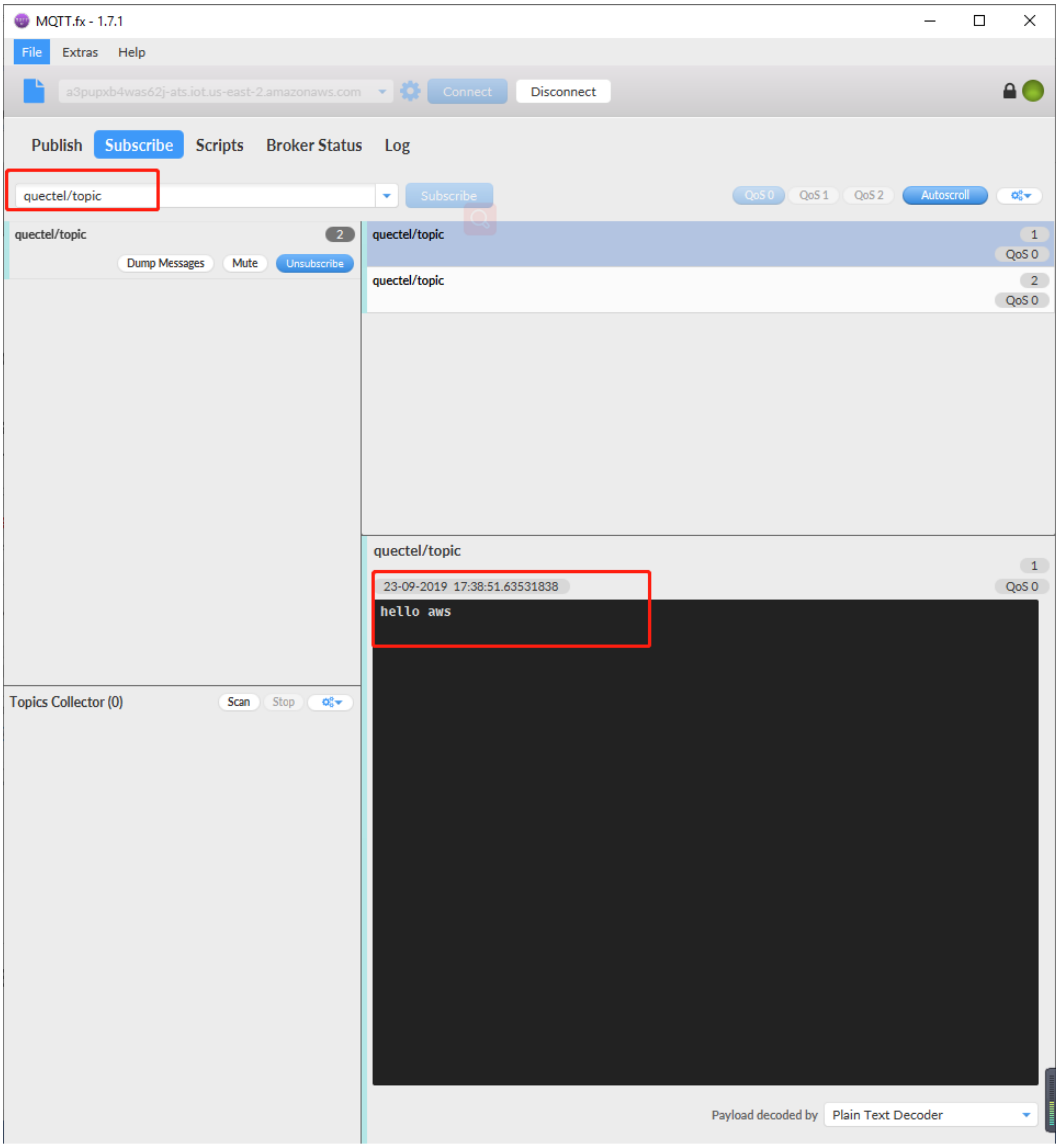
Enable SSL/TLS: ✓
Protocol: TLSv1.2

Self signed certificates: ●
CA File: E:\mqtt\aws\quectel_device1\AmazonRootCA1.pem
Client Certificate File: E:\mqtt\aws\quectel_device1\89943c0f45-certificate.pem.crt
Client Key File: E:\mqtt\aws\quectel_device1\89943c0f45-private.pem.key
Client Key Password:
PEM Formatted: ✓

Figure 46: Edit Connection Profiles

4.2.2.3. Subscribe and Publish

a) Click "**Connect**" of MQTT.fx tool.
b) Input the topic to be subscribed, *quectel/topic* for instance.
c) Click "**Subscribe**".
d) Click "**Publish**".



quectel/topic 2

quectel/topic 1 QoS 0

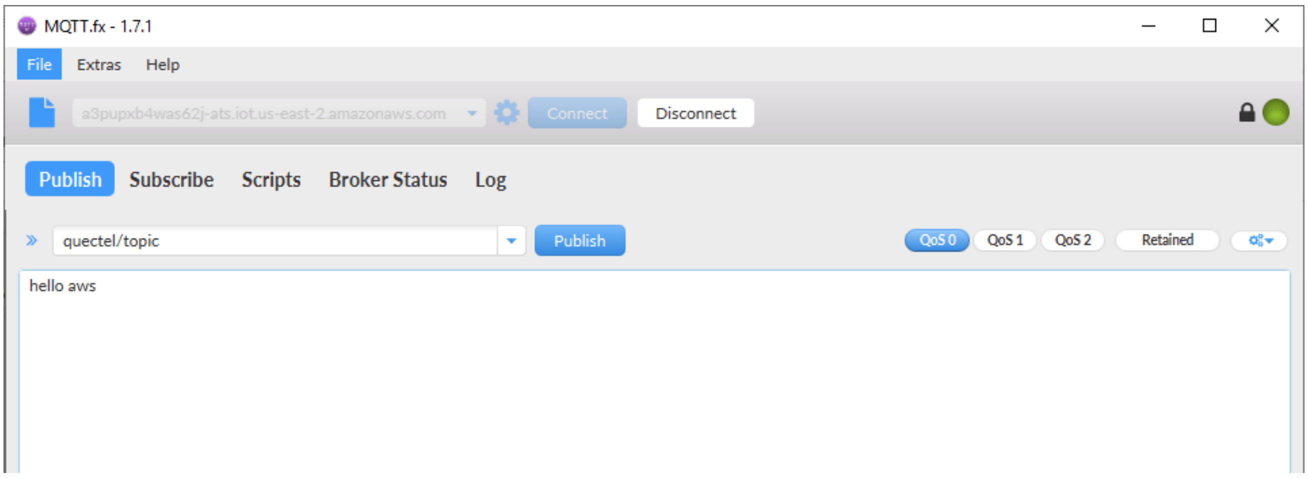
quectel/topic 2 QoS 0

quectel/topic 1 QoS 0

23-09-2019 17:38:51.63531838

```
hello aws
```

Figure 47: Subscribe to a Topic



hello aws

Figure 48: Publish the Message

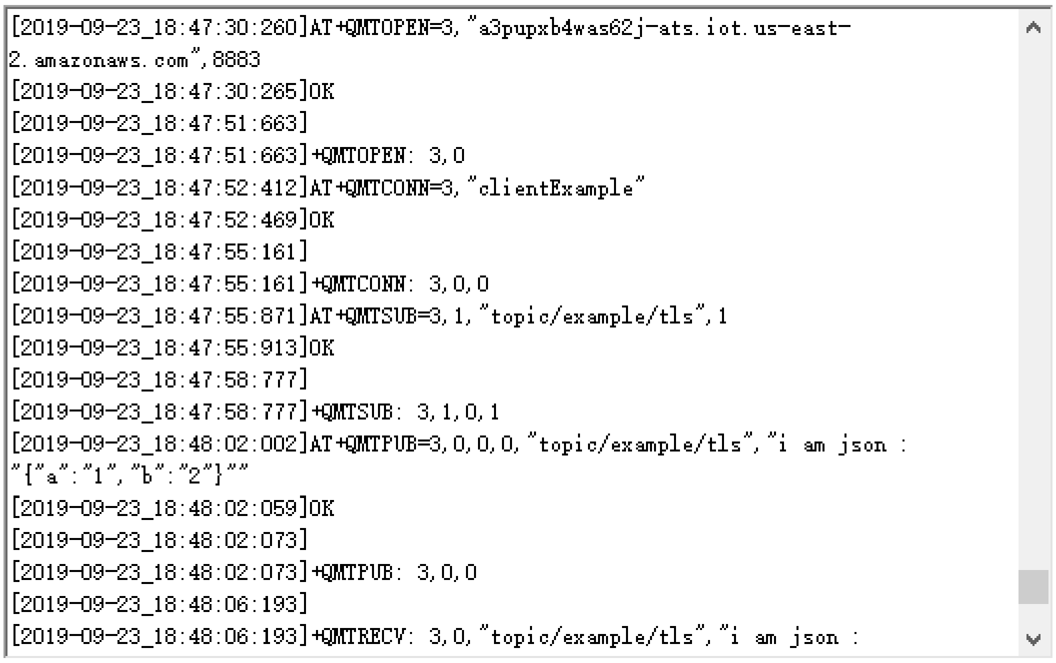
4.2.3. Use BC66/BC66-NA as MQTT Client

4.2.3.1. AT Command Example

```
AT+QSCLK=0                          //Disable sleep mode.
OK

//Configure certificates and keys
AT+QSSLCFG=1,5,"seclevel",2         //Manage server and client authentication.
OK
AT+QSSLCFG=1,5,"cacert"             //Configure CA certificate.
>                                   //Input the content of the trusted CA certificate <AmazonRo
                                    otCA1.pem> in PEM format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"cacert",1220

OK
AT+QSSLCFG=1,5,"clientcert"         //Configure client certificate.
>                                   //Input the content of client certificate <89943c0f45-certificat
                                    e.pem.crt> in PEM format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"clientcert",1679

OK
AT+QSSLCFG=1,5,"clientkey"          //Configure client private key.
>                                   //Input the content of the client private key <89943c0f45-pri
                                    vate.pem.key> in PEM format. Tap CTRL+Z to send.
+QSSLCFG: 1,5,"clientkey",1451

OK
```

```
AT+QSCLK=1                       //Enable light sleep and deep sleep, and wakeup by PSM_EINT
                                 (falling edge).
OK
AT+QMTCFG="ssl",3,1,1,5          //Enable SSL and configure SSL context/connect index.
OK
AT+QMTCFG="version",3,4          //Configure the MQTT version. AWS IoT Core supports MQTT
                                 v3.1.1.
OK
AT+QMTOPEN=3,"a3pupxb4was62j-ats.iot.us-east-2.amazonaws.com",8883 //Open a network for
                                                                   AWS MQTT client with
                                                                   TLS 1.2.
OK

+QMTOPEN: 3,0                    //Opened the MQTT client network successfully.
AT+QMTCONN=3,"clientExample"
OK

+QMTCONN: 3,0,0                  //Connected the client to MQTT server successfully.
AT+QMTSUB=3,1,"topic/example/tls",1    //Subscribe to the topic.
OK

+QMTSUB: 3,1,0,1

AT+QMTPUB=3,0,0,0,"topic/example/tls","i am json : "{"a":"1","b":"2"}"" //Send device-to-cloud
                                                                   messages.
OK

+QMTPUB: 3,0,0

+QMTRECV: 3,0,"topic/example/tls","i am json : "{"a":"1","b":"2"}""  //Received cloud-to-device
                                                                   messages

AT+QMTPUB=3,0,0,0,"topic/example/tls"  //Publish the message in data mode.
>
hello aws iot core               //Input the data to be published and then tap CTRL+Z to send.
OK

+QMTPUB: 3,0,0

+QMTRECV: 3,0,"topic/example/tls","hello aws iot core"    //Received cloud-to-device messages
AT+QMTDISC=3                     //Disconnect the client from MQTT server.
OK

+QMTDISC: 3,0                    //Connection closed successfully.
```

4.2.3.2. Subscribe and Publish



```
[2019-09-23_18:47:30:260]AT+QMTOPEN=3,"a3pupxb4was62j-ats.iot.us-east-
2.amazonaws.com",8883
[2019-09-23_18:47:30:265]OK
[2019-09-23_18:47:51:663]
[2019-09-23_18:47:51:663]+QMTOPEN: 3,0
[2019-09-23_18:47:52:412]AT+QMTCONN=3,"clientExample"
[2019-09-23_18:47:52:469]OK
[2019-09-23_18:47:55:161]
[2019-09-23_18:47:55:161]+QMTCONN: 3,0,0
[2019-09-23_18:47:55:871]AT+QMTSUB=3,1,"topic/example/tls",1
[2019-09-23_18:47:55:913]OK
[2019-09-23_18:47:58:777]
[2019-09-23_18:47:58:777]+QMTSUB: 3,1,0,1
[2019-09-23_18:48:02:002]AT+QMTPUB=3,0,0,0,"topic/example/tls","i am json :
"{"a":"1","b":"2"}""
[2019-09-23_18:48:02:059]OK
[2019-09-23_18:48:02:073]
[2019-09-23_18:48:02:073]+QMTPUB: 3,0,0
[2019-09-23_18:48:06:193]
[2019-09-23_18:48:06:193]+QMTRECV: 3,0,"topic/example/tls","i am json :
```

5 Appendix A References

Table 1: Related Documents and Links

SN	Document Name and Link	Remark
[1]	https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support	Azure mqtt-support document
[2]	http://mqttfx.jensd.de/index.php	MQTT.fx is a MQTT Client written in Java based on Eclipse Paho.
[3]	https://www.openssl.org/	OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library.
[4]	Quectel_BC66&BC66-NA_SSL_Appliation_Note	TLS/SSL AT commands and application note for BC66/BC66-NA
[6]	https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html	AWS MQTT-support document

Table 4: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
AWS	Amazon Web Services
CA	Certificate Authority
IoT	Internet of Things
IP	Internet Protocol
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrowband Internet of Things
NVRAM	Nonvolatile Random Access Memory
PDP	Packet Data Protocol

QoS	Quality of Service
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
URC	Unsolicited Result Code