ThingsBoard (https://thingsboard.io/) is a widely used IoT portal that connects various remote devices to a centralized dashboard for monitoring, reporting, and notification of alarm conditions. Its many features and flexibility make it one of the most widely used device portals. It is available as an open-source community edition and a closed-source Professional edition (PE) with numerous additional features not found in the community edition. The Professional edition is available for installation on the customer's servers or as a cloud-hosted edition with a monthly or perpetual license. ThingsBoard also offers a hosted version, ThingsBoard Cloud, which alleviates the need to set up and manage an instance.

This document will describe the steps to connect ControlByWeb (CBW) devices to the ThingsBoard PE cloud-hosted edition using the data transport for MQTT. It will not provide detailed installation or setup of ThingsBoard – they provide many tutorials and videos that assist users in configuring and using their product. ThingsBoard has its own MQTT broker, so you do not need to provide this separately. The ThingsBoard documentation section, MQTT Device API Reference, is a good starting point for understanding how MQTT works with ThingsBoard. Features in this document require CBW device firmware version 3.13 or later.

CONNECTING TO THINGSBOARD CLOUD

If you don't already have a ThingsBoard account or instance you can connect to, you can sign up for their base-level cloud subscription, which has a free 30-day trial period. The base subscription covers 30 devices for \$ 10.00 USD per month. You sign up at https://thingsboard.cloud/signup. Once you have a validated account, you can view your cloud instances' overall dashboard.



ADDING A DEVICE TO THINGSBOARD

Devices in ThingsBoard require a unique identifier that is unique overall to the **thingsboard cloud**, not just your instance. Using the CBW serial number (such as 000CC8072C45), which is the Ethernet MAC address, will provide such a unique identifier.

From ThingsBoard, click **Entities**, **Devices**, **+** to add a new device, then click **Add** new device. Fill in a unique name, label, and description for this device. You can use the default profile for the Device profile. Device profiles are templates for groups of devices and provide a way to specify standard device alarms, rule chains, firmware files, communication parameters, and many other device settings. Then click **Next: Credentials**. Select **MQTT Basic** for the transport type. Enter the full MAC access without the dashes as the Client ID. This is your device's unique identifier. Next, you can optionally enter a username and password for the MQTT connection. Then click **Add** to complete adding the device to your system.

Add new device	? ×	Add new device	? ×
1 Device details	Credentials O Optional	Device details	Optional
Name* X412-072C45		Credentials type Access token X.509	MQTT Basic
Label Test 412		Client ID 000CC8072C45	Ū
Device profile* default	× /	User Name* Cbwuser	Ū
S gateway		Password	Ø []
Owner and groups Owner* rmontrose@controlbyweb.com Groups Development × Description Test X412 for development	×		
	Next: Credentials	Back	
	Cancel Add		Cancel Add

If you click the **Check connectivity** button, ThingsBoard will display a window showing how to use Mosquito to verify connectivity with ThingsBoard using your credentials. After we configure our CBW device, we will use it to test connectivity with ThingsBoard.

CONFIGURING DEVICE MQTT PUBLICATIONS

Next, we must configure the ControlByWeb device's MQTT interface to talk to ThingsBoard. Click on the **MQTT** tab and click on **Brokers**. Then, enter the MQTT device configuration information.

- The hostname and port are **mqtt.thingsboard.cloud**, port **1883**. To use an encrypted connection, you must generate a certificate in ThingsBoard and install it on the device (<u>see</u> <u>ThingsBoard documentation on MQTT security</u>). Secure connections use port **8883**. Leaving the hostname/IP blank or setting the port to 0 disables the broker.
- Enter the ClientID, Username, and Password you entered in the Device Credentials screen.
- ThingsBoard, especially the cloud version, cannot handle continuous streams of messages. Enter a **Throttle period** of 100ms between each message and adjust lower with testing.
- The unit will send an MQTT PING message every **Keep Alive** Interval unless there is other MQTT activity. The PING message will be sent if there is no activity during the minimum time. Note that this exchange takes 120 bytes, so you would want to increase this time to 5 minutes (300 seconds) or 10 minutes (600 seconds) for a cellular connection.
- Set the Topic Root to **v1/devices/me/telemetry**, the default topic for publishing readings (timeseries data). All data MQTT data sent to ThingsBoard is published using JSON payloads
- The Birth MQTT topic is sent when the device starts to register with the MQTT gateway. It is looking for a message published to v1/gateway/connect in the format {"device":"DeviceA"}, so we set the Birth Message to {"device":"\${clientID}"}.
- The Last Will topic is sent when the device disconnects. In practice, the device sends this before sending the Birth topic to ensure the gateway knows the device is connecting again. The Last Will topic is v1/gateway/disconnect with the message set to {"device":"\${clientID}"}.
- Publish heartbeat will send the specified packet at the specified interval. Depending on the heartbeat payload you specify, this will take about 180 bytes or more and shouldn't normally be used for cellular connections. The topic must be v1/devices/me/telemetry if enabled. Also, the default payload specified in 3.12 releases is not correctly formatted JSON and could cause the MQTT broker to disconnect. The default payload specified is

```
{"id":"${clientID}","upTime":"${upTime}","address":"${ip}:${port}}" and should be
{"id":"${clientID}","upTime":"${upTime}","address":"${ip}:${port}"}
```

• Save Changes.

Edit MQTT Broker		×	
Broker Name:	ThingsBoard		
Sparkplug B:	Yes No		
O Use Multiple Servers:	Yes No		
Hostname/IP:	ec2- us-east-2.compute.ama		
Port:	1883		
Client ID:	000CC8		
Username:	cbwuser		
Password:			
C Encrypted:	No 🗸		
Throttle Period:	100 Milliseconds		
A D (D)	Seconds		
Reconnection Delay:	30		
Keep Alive Interval:	30 Seconds		
O Topic Root:			
O Publish Heartbeat:	Yes No		
Clean Session:	Yes No		
Birth Topic:	v1/gatewav/connect		
	Prepend Topic Root		
Birth Message:	{"device":"\${clientID}"}		
		10	
Last Will Topic:	v1/gateway/disconnect		
	Prepend Topic Root		
O Last Will Message:	{"device":"\${clientID}"}		
		li	
	Save Changes	Cancel	

Next, we need to configure the **Publish / Subscribe** sections of the MQTT sensor device configuration.

Messages are sent as JSON packets to the **v1/devices/me/telemetry** MQTT topic, formatted in JSON. The message is formatted as:

{"keyword1":"\${token1}", {"keyword2":"\${token2}", {"keyword3":"\${token3}"}

The keywords are the names of the data that will appear as timeseries variable names in ThingsBoard. There is no standard for names in ThingsBoard, so you can pick any names that make sense to your application. It is recommended that they be kept short since the MQTT Payload can be a maximum of 500 bytes per published message. Suggested keywords are **rly1, din1, ain3, reg10, vin, and id.** The tokens are used to reference sensor values in the CBW device, and the actual value is replaced in the data published to ThingsBoard. The currently supported MQTT tokens are:

MAC Address	\${mac}	Name (Control Page Header)	\${name}	Vin	\${vin}
ClientID	\${clientID}	Firmware Revision	\${ver}	Register 1	\${register1}
Model	\${model}	Serial Number	\${ser}	Digital Input 1	\${digitalInput1}
IP Address	{qi}	Epoch Time Stamp (sec)	\${dateTime}	Analog Input 1	\${analogInput1}
HTTP Port	\${port}	Epoch Time Stamp (msec)	\${dateTimems}	Relay 1	\${relay1}
HTTPS Port	\${httpsPort}	Sequence Number (autoincrementing)	\${seq}		
RSSI (cellular)	\${rssi}	Up Time (sec)	\${upTime}		
Latitude Longitude	\${latitude} \${longitude}				

The tokens in the right column are specific to each device. They will differ based on the device's configuration and any expansion modules or additional sensors connected to it. The View MQTT Payload Tokens button will show a list tailored to each device.

By default, ThingsBoard will timestamp the data when the server receives it. If the MQTT connection is down or intermittent, MQTT messages are queued and sent when the connection is viable, but the time shown will be the received time, not the queued time. ThingsBoard supports the format where the timestamp is specified in the JSON payload using the **ts** keyword for the timestamp. The timeseries data in ThingsBoard will be timestamped using the Epoch millisecond timestamp from the device. In this case, the JSON payload format is:

{"ts":\${dateTimems},"values":
{"keyword1":"\${token1}",
{"keyword2":"\${token2}",
{"keyword3":"\${token3}"} }

Messages can be published either from the Control/Logic Tasks as a Scheduled, Conditional, or Automatic Reboot Task or based on a change of sensor device value in the Publish definition or as Log entries when the data is sent to the device's Log.

Edit MQTT Publication		×
Publication Name:	Pub Device Values	
Broker:	ThingsBoard Cloud	
Publish on Change:	Yes No	
Topic:	V1/devices/me/telemetry Prepend Topic Root	
🕄 IO:Payload:	{"ts". \${dateTimems},'values": {"id": "\${clientID};", "name":"\${nane}", "model":"\${model";" ain1":"\${nalogInput1}", "din2":"\${model";" ain3":"\${nalogInput3} ","ain4":"\${nalogInput3}", "th1": "\${relay1}", "th2": "\${relay2}", "rty3": "\${relay3}", "rty4": "\${relay4}","vin":"\${vin}":"\${seq}":"\${seq}","rssi"."\${ rssi}"}	
🕄 QOS: 🕑 Retain:	1 (At Least Once)	
	Save Changes Can	cel

For example, suppose you want to publish every time the Digital Input 1 changes state. In that case, you can select **Publish on Change** and then select that sensor device to trigger a publication. Only a single sensor device can be used as a trigger. Alternatively, you can specify the sensor device(s) in an MQTT Publication, and then in the **Control/Logic Tasks**, it will appear as an action that can be triggered. You can have changes in either of two different devices trigger a message publication, along with performing two other Tasks.

ControlByWeb devices often use the Flash Log to record activity. This is set up under the **Logging & Cloud** menu. The log is 512K of Flash memory that stores activity in a circular buffer. Logging can be specified to occur based on Tasks, changes on Devices, BASIC script, SNMP, Modbus, or HTTP/HTTPS XML/JSON API activity. When a log event occurs, the values for devices specified on the Logging page are captured and sent via Email, FTP, Cloud, or MQTT. Enable the MQTT section under SEND LOG FILE to send Log entries via MQTT. Note

• Send using scheduled task:	Yes No
Daily Send Time (HH:MM):	23 • = 00 •
C Email Log File:	Enable
O Publish Log File (MQTT):	C Enable
Broker:	ThingsBoard
Topic:	v1/devices/me/telemetry
Payload:	('ts':\$(logDateTimens), 'values'': ['seq':"\$(seq)', 'evt':"\$(logEventType)'', 'evtid':"\$(logEventID)', 'id':"\$(clientID)'', 'id':"\$(cl
	View MQTT Payload Tokens Test Log Publication

here that the payload can be up to 1000 characters in size. There are three additional variables you can use in your publication: *Log Epoch Time Stamp* (sec or ms) / \${logDateTime}, \${logDateTimes}, *Log Event Type or Source* \${logEventType} Log Entry ID \${logEventID}.

With MQTT, short messages are expected to be published with just the changed data. You could have a different MQTT Publication setup for each sensor or register on the device and have each configured to send when that sensor or register changes value. ThingsBoard has a screen widget called the Timeseries table that will show a continuous log of timeseries data as it is received with the timestamp. If you send each device in its own MQTT Publish message, the Timeseries Table will only show the value received; none of the existing values will be shown. If you want to see all the data for the device for each reading, then you would like to have all the device registers sent up simultaneously. I chose the latter for the example X412 (4 Analog inputs, 4 Relays) I used for this demonstration. The MQTT payload was set up as:

```
{"ts": ${dateTimems},"values": {"id": "${clientID}", "name":"${name}",
"model":"${model}", "ain1":"${analogInput1}",
"din2":"${digitalInput2}","ain3":"${analogInput3}", "ain4":"${analogInput4}",
"rly1": "${relay1}", "rly2": "${relay2}", "rly3": "${relay3}", "rly4":
"${relay4}","vin":"${vin}","seq":"${seq}","rssi":"${rssi}"} }
```

You may also want to send data periodically when the device starts. This will be a different MQTT publication that you can send based on a scheduled event once a day. Scheduled events are sent to each device bootup, so this can also be an initial message that the device is working. The Initial Settings message we defined was:

```
{"ts": ${dateTimems},"values": {"id": "${clientID}", "name":"${name}",
"model":"${model}","latitude": "${latitude}", "longitude": "${longitude}"} }
```

We set the Quality of Service (QOS) to 1 to ensure at least one transmission is sent for the publication. Since each publication has a timestamp, ThingsBoard will automatically remove duplicate timestamped messages. So, if the device sends the message multiple times before it receives an acknowledgment from the MQTT broker, it will not result in duplication in ThingsBoard.

TASKS/FUNCTIONS WED, 30 OCT 2024 18:37:44 CURRENTLY RUNNING NORMAL SCHEDULE							
SCHEDULED	SCHEDULED Add Scheduled Task +						
Name	Start Date/Time	Repeat	Actions	******	Next Occurrence	Run Mode	Edit
Publish MQTT Values 15 m	Thu, 24 Oct 2024 08:00:00	Every 15 Minutes	Trigger MQTT F Device Values	Publication Pub	Wed, 30 Oct 2024 18:45:00	Always	Edit X
Power On Message	Tue, 29 Oct 2024 08:00:00	Daily	Trigger MQTT F Initial Settings Log	Publication Pub	Thu, 31 Oct 2024 08:00:00	Always	Edit X
	CONDITIONAL Add Conditional Task +					Add Conditional Task +	
Name	Trigger			Actions			Edit
■ Analog Change If Analog Input 1 changes by 0.10 or Analog Input 3 changes by 1.0000 Trigger MQT		Trigger MQTT	Publication Pub Devic	e Values	Edit X		
Digital Chanrges If Digital Input 2 Changes		3	Trigger MQTT Publication Pub Device Values Edit		Edit X		
Relay Changes	If Relay 1 Changes or Ny Changes 12 Relay 2 Changes Trigger MQTT Publication Pub Device			e Values	Edit X		
Relay Changes	34 If Relay 3 Relay 4 Ch	Changes or hanges		Trigger MQTT Publication Pub Device Values			Edit X

The device's Control/Logic setup page is set to publish a message when any inputs or outputs change.

DEVICE MQTT SUBSCRIPTIONS

ThingsBoard communicates setting and control messages to devices using two methods: **Shared Attributes** and Remote Process Calls (**RPC**). Currently, ThingsBoard only supports Shared Attributes using MQTT for device control.

What is an Attribute? In ThingsBoard, a device (like our CBW controller) has timeseries data and attributes associated with the device. Attributes are values related to the device. There are three types of attributes: Client, Server, and Shared. As their names suggest, Client attributes are only visible to the client (device), and the server (ThingsBoard) cannot access the values. Server attributes are visible only in ThingsBoard – the device cannot see them. Shared attributes can be read and written by the device and the server to communicate between ThingsBoard and the device.

The device subscribes to Shared attribute changes using the **v1/devices/me/attributes** topic. ThingsBoard will publish a message to the device as a JSON packet in the format {"keyword":"value"}. You must first define the Shared Attributes for the device in ThingsBoard before they can be changed and sent to the device. The Shared Attributes name **must** match the device name in the MQTT Payload Token. For example, to control a relay, first create attributes named relay1, relay2, etc. These names <u>must</u> match those used to control devices in **state.json** or **state.xml**. Running **/state.json** will display the names. Since these will

be used to control the device, only relays and digital and analog outputs make sense to define as shared attributes used for control.

In ThingsBoard, go to the device and click the **Attributes** tab. Select **Shared Attributes** from the Entity attributes selector. Click the + to add a new attribute. Enter the attribute name exactly as shown in the MQTT Payload tokens, then define the attribute type. Relays and digital outputs will be defined as **Integer**, and analog outputs will be **Double**.



Repeat this for all the I/O devices you will want to control.

X D	(412-0 evice det	172C45 ails					?	×
<	Detail	s Attribut	es Latest teleme	etry Alarms	Events	Rel	ations	,
	Shared	d attributes	Entity attributes scope Shared attributes	•		+	G	۹
	La:	st update time	Key 🛧	Value				
	202	24-10-29 17:33:45	relay1	0				
	202	24-10-29 17:33:58	relay2	0				/
	202	24-10-29 17:34:17	relay3	0				/
	202	24-10-29 17:35:07	relay4	0				/

The next step is to define the MQTT subscription. Click Add Subscription on the CBW device under MQTT and then Publish / Subscribe. Select the MQTT Broker you have defined. The topic is v1/devices/me/attributes, and QOS is set to 1 to ensure you receive at least one message.

Edit MQTT Subscription		×
Subscription Name:	ThingsBoard Attribute Ch	
Broker:	ThingsBoard Cloud	
Торіс:	v1/devices/me/attributes	
0 QOS:	1 (At Least Once)	
	Save Changes	Cancel

When an attribute changes, like relay1, the device will receive the JSON message **{"relay1":1}**. The JSON is parsed and formatted to the message **relay1=1**, which is sent to the same parser used for parameters passed with **state.json** or **state.xml**. If you want to control the device using these older methods, you can do so using MQTT.

TESTING DEVICE TO DASHBOARD CONNECTION

Now, we want to verify connectivity between our device and ThingsBoard. From a ThingsBoard device, there is a **Latest Telemetry** tab. Click on that, and it will show the latest readings from the device. Trigger a change that will generate an MQTT Publish message. You should see the time series values change; the **Last update time** will reflect when the message was generated using the **ts** variable. It may help to set up a register or relay on the CBW device's Control page, which you can control and have set to trigger a message. You can look in the device's debug log at **/debug.html** and see the MQTT messages and any errors. If the messages are not received, verify the MQTT clientID, username and password, and the ThingsBoard MQTT address and port. Setting the Debug Console's Options setting to 20 will provide additional debugging detail. You can use **Mosquitto, as shown in the ThingsBoard documentation,** to verify the cloud server connection.

X412-072C45 Device details			0 ×
✔ Details Attributes	Latest telemetry	Alarms Events	Relations
Telemetry			+ Q
Last update time	Кеу 🛧	Value	
2024-10-29 17:34:17	ain1	2.49	Ť
2024-10-29 17:34:17	ain3	9.2061	Ť
2024-10-29 17:34:17	ain4	0.0	Ť
2024-10-29 17:34:17	din2	0	Ť
2024-10-29 17:34:17	id	000CC8072C45	Ť
2024-10-29 17:28:21	latitude	41.679588	Ť
2024-10-29 17:28:21	longitude	-111.873734	Ť
2024-10-29 17:34:17	model	X-412W-I	Ť
2024-10-29 17:34:17	name	X-412W	Ť.
2024-10-29 17:34:17	rly1	0	Î
2024-10-29 17:34:17	rly2	0	Ū
2024-10-29 17:34:17	rly3	0	Î
	Items per page: 30 🗸] 1 − 17 of 17 <	< > >I

Options: 20 Update 03/18/2025 15:22:53 MQTT_Ping(1) Sent 2 bytes of PIMGREQ packet to BrokerID=1. 2:0:2:6 03/18/2025 15:22:53 MandleIncomingAck() Received MQTT packet of type 0xD0. Len=0 Data= 2:0:2:6 03/18/2025 15:22:53 MonterioningAck() Received MQTT packet of type 0xD0. Len=0 Data= 2:0:2:6 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_STATE_START Attempt to send log batch to MQTT brokerID=1 battBackedMqttID=0 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 readSector=0 endSectorState 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Gocounter=0 readAddr=0 readSector=0 endSectorState 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 readSector=0 endSectorState 03/18/2025 15:23:00 mgttParsePublishLogPayload(1) STND_LOG_BATCH_GET_NEXT_LOG Getting thera "int: "\${digitalInput3}", "int: "\${digitalInput4}", "ny ris: "\${digitalInput4}", "ny ris: \${digitalInput4}", "ny ris: \${digitalInput4}", "ny ris: \${digitalInput4}", "int: "\${digitalInput4}", "int: "\${digitalInput4}", "int: "int: "in		
<pre>(3/18/2025 15:22:53 MQTT_Ping(1) Sent 2 bytes of PINGREQ packet to BrokerID=1. 2:0:2:6 (3/18/2025 15:22:53 MQTTreciveSingleIterationDataBuffered(1) brokerID=1 packetType=0x00 index=2 \$00\$00 2:0:2:6 (3/18/2025 15:22:53 handleIncomigAck() Received MQTT packet of type 0x00. Len=0 Data 2:0:2:6 (3/18/2025 15:22:3:00 sendClientLog(38) Attempt to send log entries over remote services: battBackedFtpCloudID=0 currentFtpCloudID=0. 2:0:2:6 (3/18/2025 15:22:3:00 sendClientLog(38) Attempt to send log entries over remote services: battBackedFtpCloudID=0 currentFtpCloudID=0. 2:0:2:6 (3/18/2025 15:23:3:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_STATE_START Attempt to send log batch to MQTT brokerID=1 battBackedMqttID=0 currentMqttID=0 logEntryLength=72 2:0:2:8 (3/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 endAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 LogBentryLength=72 2:0:2:8 (3/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 (3/18/2025 15:23:00 mqttParsePublishLogBayLoad(1) BrokerID=1 mqttClientPublishLogEntry spb=0 Len=937 payLoad=("ts":\${logDateTimems},"values": "\${digitalInput2}","din3":"\${digitalInput3}","din4":"\${digitalInput4}","rly 2:0:2:8 (3/18/2025 15:23:00 mqttParsePublishLogPayLoad(1) BrokerID=1 mared Len=15 mqtt_publish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time (),"evtid":"1","id":"000CC8062002","name":"X-410W,"model":"X-410CM- "","din1":"0","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": "3/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId= 2:0:2:8 (3/18/2025 15:23:00 MQTT_GetPacketId= 2:0:2:8 (3/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3 packetSize=645 remainingLength=642 headerSiz</pre>		
<pre>03/18/2025 15:22:53 MQTTreceiveSingleIterationDataBuffered(1) brokerID=1 packetType=0x00 index=2 \$D0\$00 2:0:2:6 03/18/2025 15:22:53 handleIncomingAck() Received MQTT packet of type 0x00. Len=0 Data 2:0:2:6 03/18/2025 15:23:00 sendClgatterationQ(38) Attempt to send log entries over remote services: battBackedFtpCloudID=0 currentFtpCloudID=0. 2:0:2:6 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_STATE_START Attempt to send log batch to MQTT brokerID=1 battBackedMqttID=0 currentMqttID=0 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 endAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=937 payload={"ts":\${logDateTimems},"values": {"seq":"\${seq":"s{seq}","evt":"\${logEventType}","evtid":"\${logEventTyp!","evtid":"\${logEventTyp!","evtid":"\${logEventTyp!","din1":"\${digitalInput3},"din4":\${logEventTyp!","int":"\${logEventTyp!","int":"\${logEventTyp!","int":"\${logEventTyp!","int":"\${int":"}{int":"\${int":"\${int":"\${int":"}{int":"\${int":"}{int":"}{int":"\${int":"}{int":"}{int":"\${int":"}{int":"}{int":"}{int":"}{int":"}{int":"}{int":"}{int":"}{int":"}{int":"}{int":"}{int":"</pre>	2:0:2:6	-
<pre>03/18/2025 15:22:53 handleIncomingXck() Received MQTT packet of type 0xD0. Len=0 bata= 2:0:2:6 03/18/2025 15:23:00 sendLightLog(38) Attempt to send log entries over remote services: batBackedFpCloudID=0 currentFtpCloudID=0. 2:0:2:6 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_STATE_START Attempt to send log batch to MQTT brokerID=1 battBackedMqttID=0 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounte=0 readAddr=0 endAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParserbublishLogPayload(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=987 payload=("ts";flogDateTimens},"values": {"seq":"\${seq}","evt":"\${logEventType}","evtid":"\${logEventID}","id":"\${clientID}","name":"\${name}","model":"\${model":"\${model":"\${model}","din1:"\${gigitalInput3}","din4":"\${digitalInput3}","din4":"\${digitalInput4}","rly 2:0:2:8 03/18/2025 15:23:00 mqttParserbublishLogPayload(1) BrokerID=1 Parsed len=615 mqttPublish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evtid":"1","id":"0%00CCB062002","name":"X-410W,"model":"X-410W- I","din1:"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","ru12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishLogEntry 2::0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishDigDerkerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2::0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishBacketID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2::0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTim==2020 sec @ M</pre>	/pe=0xD0 index=2 \$D0\$00 2:0:2:6	
<pre>03/18/2025 15:23:00 sendClientLog(38) Attempt to send log entries over remote services: battBackedFtpCloudID=0 currentFtpCloudID=0. 2:0:2:6 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_STATE_START Attempt to send log batch to MQTT brokerID=1 battBackedMqttID=0 currentMqttID=0 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 endAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mgttParsePublishLogPayload(1) BrokerID=1 mgttClientPublishLogEntry spb=0 len=987 payload={"ts":\${logDateTimems},"values": {"seq":"\${seq":"\${logEvenType}","evtid":"\${logEvenTDP","id":"\${cileIII]}","name":"\${name":"\${name":"\${model1":"\$(model1":"\${model1":"}{{m</pre>	Data= 2:0:2:6	
<pre>03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_STATE_START Attempt to send log batch to MQTT brokerID=1 battBackedMqttID=0 currentMqttID=0 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=937 payload=('ts:'\${logDateTimems}, "values": ("seq":"\${seq}","evt":"\${logEventType}","evtid":"\${logEventID}","id":"\${clientID}","name":"\${name}","mode1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod2":"\${mod2":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod2":"\${mod2":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod2":"\${mod2":"\${mod2":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"}{mod1":"\${mod1":"\${mod1":"\${mod1":"}{mod1":"}{mod1":"\${mod1":"\${mod1":"\${mod1":"\${mod1":"}</pre>	rvices: battBackedFtpCloudID=0 currentFtpCloudID=0. 2:0:2:6	
<pre>logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayLod(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=987 payLoad={"ts":\${logDateTimems},"values": {"seq":"\${seq}","evt":"\${logDateTimems},"values": {"seq":"\${seq}","evt":"\${logDateTimems},"idin4":"\${logtentTpp","idin4":"\${logtentTpp","idin4":"\${logtentTpp","stqii" "\${digitalInput2}","din3":"\${digitalInput3}","din4":"\${digitalInput4}","rly 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayLoad(1) BrokerID=1 Parsed len=615 mqtpublish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evti":"1","din1":"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly4":"0","rly4":"0","rus1":"2.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishIng brokerID=1 packetID=3, maxeRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishing brokerID=1 packetID=3, maxeRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishiNgthcopy(3) PacketID=3 packetID=3, maxeRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishiNgthcopy(3) PacketID=3 packetID=3, maxeRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishiNgthcopy(3) PacketID=3 packetID=3, maxeRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishiNgthcopy(3) PacketID=3 packetID=3, maxeRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03</pre>	send log batch to MQTT brokerID=1 battBackedMqttID=0 currentMqttID=/	3
<pre>03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG logCounter=0 readAddr=0 endAddr=0 readSector=0 endSectorStart=0 endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=987 payload={"ts":\${logDateTimems},"values": {"seq":"\${seq":"\${req":"\${logEventType}","evtid":"\${logEventDP","id":"\${clientD}","name":"\${name}","model":"\${model":"\${model":"\${model":"\${idigitalInput3}","din4":"\${digitalInput3}","din4":"\${digitalInput3}","rly 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 Parsed len=615 mqtt_publish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evtid":"1","id":"000CC8062002","name":"X-410KH","model":"X-410KH" ","din1":"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 AQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublish.30 Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 AddRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWithOutp() Alcong Vetors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 sendPublishWithOpy(3) PacketID=3 BrokerID=1 Sending Vetors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$0\$0\$0\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"000CC6062002?","name":</pre>	5 I I I I I I I I I I I I I I I I I I I	
<pre>endSectorEnd=4096 logEntryLength=72 2:0:2:8 03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayLoad(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=987 payLoad=("ts":\${logDateTimems},"values": {"seq":"\${seq}","evt":"\${logEventType}","evtid":"\${logEventID}","id":"\${clientID},"name":"\${name}","model":"\${model":"\${model","din1":"\${digitalInput3}","din4":"\${digitalInput3}","rly 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayLoad(1) BrokerID=1 Parsed Len=615 mqtt_publish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evtid":"1","id":"00CC8062002","name":"X-410W","model":"X-410CW- I","din1":"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 2:0:2:8 03/18/2025 15:23:00 kQITI_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 kQITI_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 adtLogLogLogLogLogLogLogLogLogLogLogLogLogL</pre>	er=0 readAddr=0 endAddr=0 readSector=0 endSectorStart=0	
<pre>03/18/2025 15:23:00 sendLogBatchToBroker(1) SEND_LOG_BATCH_GET_NEXT_LOG Getting the first log entry. readAddr=0 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayLoad(1) BrokerID=1 mqttClientPublishLogEntry spb=0 len=987 payLoad={"ts":\${logDateTimems},"values": {"seq":"\${seq}","evt":"\${logEventType}","evti":"\${logEventTD}","id":"\${logEventTD}","id":"\${logEventTD}","ame":"\${mame":"\${mame":"\${mame":"\${mame":"\${mame":"\${mame":"\${mame":"\${mame}}","model":"\${mame":"\${mame}}","model":"\${mame}","model":"\${model":"}{{model":"}{model":"\${model":"\${model":"\${model":"}{{mode</pre>		
<pre>03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 mqttLitentPublishLogEntry spb=0 len=987 payload=("ts":\${logDateTimems},"values": {"seq":"\${seq}","evt":"\${logEventType}","evtid":"\${logEventID}","id":"\${clientID}","name":"\${name}","model":"\${model":"\${model":"\${model}","din1":"\${digitalInput3}","din4":"\${logEventID}","id":"\${clientID}","name":"\${name}","model":"\${model":"\${model}","din1":"\${digitalInput3}","din4":"\${logEventID}","id":"\${clientID}","name":"\${name}","model":"\${model":"\${model":"\${model}","din1":"\${digitalInput3}","din4":"\${model}","din1":"\${digitalInput3}","din4":"\${model}","din1":"\${model}":"\${model</pre>	the first log entry. readAddr=0 2:0:2:8	
<pre>["sq1:"\${sq2}","evt":"\${logEventType}","evtid":"\${logEventTyp}","id":"\${logEventTyp}","name":"\${name}","model":"\${model":"\${model":"\${model":"\${digitalInput3}","din4":"\${logEventTyp}","ad":"\${logEventTyp}","name":"\${name}","model":"\${model":"\${model":"\${model":"\${digitalInput3}","din4":"\${logEventTyp}","ad":"ad":"ad":"ad":"ad":"ad":"ad":"ad</pre>	gEntry spb=0 len=987 payload={"ts":\${logDateTimems},"values":	
<pre>"\${digitalInput2}","din3":"\${digitalInput3}","din4":"\${digitalInput3}","rly 2:0:2:8 03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 Parsed len=615 mqtt_publish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evtid":"1","id":"00","din2":"1","din3":"0","din4":"0","rly1":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 calculatePublishPacketSize() PUBLISH packet remaining length=642 and packet size=645. 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishPacketSize() PUBLISH packetTo=3 a gocketSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 MQTT_Dublish(3) Publishing brokerID=1 packetID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWishWetTD=1 packetID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWishWetTD=1 Sending Vectors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 sendPublishWishWetTD=3 [0]=0:32 bytesToSend=645 2\$82\$05\$00\$171/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"00@CG8062002","name":"X-410W","model":"X-410W-","din1":"0","din3":"0","din3":"0","din3":"0","revt</pre>	}","name":"\${name}","model":"\${model}","din1":"\${digitalInput1}","din2	2":
<pre>03/18/2025 15:23:00 mqttParsePublishLogPayload(1) BrokerID=1 Parsed len=615 mqtt_publish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evtid":"1","id":"000CC8062002","name":"X-410W","model":"X-410CW- I","din1":"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) PublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_Publish(3) Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) ClientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"00CC8062002","name":"X-410W","nome":"X-410V","din1":"0","din3":"0","din3":"0","din3":"0","revt":"10.00","seq":"2","evt":"10.00","seq":"2","evt:"2","evt</pre>	ð:2:8	
<pre>()","evtid":"1","id":"000CC8062002","name":"X-410W","model1":"X-410W- I","din1":"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 2:0:2:8 03/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 calculatePublishPacketSize() PUBLISH packet remaining length=642 and packet size=645. 2:0:2:8 03/18/2025 15:23:00 MQTT_entilshPeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_PublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_Publish(3) Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 BrokerID=1 Sending Vectors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i di":"00CCB02002","name":"X-410W","model":"X-410W","din1":"0","din3":"0","din4":"0","r</pre>	_publish_buffer={"ts":1742332920000,"values":{"seq":"1","evt":"Time	
<pre>[1","din1":"0","din1":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0","rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt": 2:0:2:8 03/18/2025 15:23:00 QITI_GetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 calculatePublishPacketSize() PUBLISH packet remaining length=642 and packet size=645. 2:0:2:8 03/18/2025 15:23:00 MQTT_SerializePublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_Publish(3) Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 BrokerID=1 Sending Vectors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evt":"Time ()","evt":"1","i di":"00C68062002","name":"X-410W,"."model":"X-410W,"."","din1":"0","din3":"0","din3":"0","r</pre>		
<pre>2:0:2:8 2:0/18/2025 15:23:00 MQTT_GetPacketId(3) NextPacketId=4</pre>	'rly4":"0","vin":"12.3 V","rssi":"0","register1":"0.0","seq":"2","evt	14 I I
<pre>(03/18/2025 15:23:00 MQTT_cetPacketId(3) NextPacketId=4 2:0:2:8 03/18/2025 15:23:00 MQTT_serializePublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_serializePublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 03/18/2025 15:23:00 MQTT_Publish(3) Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3 maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1,"evt":"Time()","evtid":"1,"i d":"00C6805202","name":"X-410W," model":"X-410CW-T","din1":"0","din3":"0","din3":"0","r</pre>		
<pre>(03/18/2025 15:23:00 calculatePublishPacketSize() PUBLISH packet remaining Length=642 and packet size=645. 2:0:2:8 (03/18/2025 15:23:00 MQTT_evializePublishHeaderWithoutTopic() Adding QoS as QoS1 in PUBLISH flags. 2:0:2:8 (03/18/2025 15:23:00 MQTT_Publish(3) Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 (03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 (03/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 BrokerID=1 Sending Vectors ioVectorLength=4 2:0:2:8 (03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"000C6062002","name":"X-410W,"."model":"X-410W-T","din1":"0","din3":"0","din3":"0","r</pre>		
<pre>(93/18/2025 15:23:00 MQIT_SchlaftzePublishheaderWithoutiopic() Adding QoS as QoS1 in PUbLiSH flags. 2:012:8 (93/18/2025 15:23:00 MQTT_Publish(3) Publishing brokerID=1 packetId=3 packetSize=645 remainingLength=642 headerSize=5 QoS=1 from mqttClientPublishLogEntry 2:0:2:8 (93/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 (93/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 brokerID=1 Sending Vectors ioVectorLength=4 2:0:2:8 (93/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"000C68062002","name":"X-410W,"."model":"X-410W-T","din1":"0","din3":"0","din3":"0","r</pre>	=642 and packet size=645. 2:0:2:8	
<pre>D3/18/2025 15:23:00 MQI1_PUDIIsning brokerID=1 packetID=3 packetS12e=045 remainingLengtn=642 neaderS12e=5 QOS=1 from mqttClientPublishLogEntry 2:0:2:8 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 BrokerID=1 Sending Vectors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1,"evt:"Time ()","evtid":"1,"indel":"2,"idin1":"0"."din2":"1,"idin3":"0"."r</pre>	in PUBLISH flags. 2:0:2:8	
<pre>mqtclientPublishLogEntry 2:0:20 03/18/2025 15:23:00 addRecord(3) Added record at index 0. PacketID=3, maxRecordCount=100 updateTime=2020 sec @ MQTT_ReserveState() 2:0:2:8 03/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 BrokerID=1 Sending Vectors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1,"evt":"Time ()","evtid":"1,"i d":"000CC8062002","name":"X-410W," model":"X-410CW-T","din1":"0","din3":"0","din4":"0","r</pre>	045 remainingLength=642 headerSize=5 QoS=1 from	
<pre>03/18/2025 15:2:100 audkectord(5) Added record at index 0. PacketID=3, markectordcount=100 update11me=2020 set 0 mq11_keservestate() 2:0:2:0 03/18/2025 15:23:00 sendPublishWithCopy(3) PacketID=3 EnokerID=1 SendBy Vectors ioVectorLength=4 2:0:2:8 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"00C68062002","name":"X-410W," model":"X-410WL"."din1":"0"."din3":"0"."din4":"0"."r</pre>		
03/18/2025 15:25:00 Sendroulisimulincopy(5) Packet10=5 broker10=1 Sending Vectors TovectorLength=4 2:0:2:0 03/18/2025 15:23:00 transport_writev(3) clientId=3 [0]=0x32 bytesToSend=645 2\$82\$05\$00\$17v1/devices/me/telemetry\$00\$03{"ts":1742332920000,"values": {"seq":"1","evt":"Time ()","evtid":"1","i d":"000CC8062002"."name":"X-410W"."model":"X-410CW-T"."din1":"0"."din2":"1"."din3":"0"."din4":"0"."r	investoriestate() 2:0:2:8	
{"seq":"1","evt":"Time ()","evtid":"1","i {"seq":"1","evt":"Time ()","evtid":"1","i d":"000CC8062002","name":"X-410W","model":"X-410CW-T"."din1":"0"."din2":"1","din3":"0"."din4":"0"."r	10vectorLengtn=4 2:0:2:0	
[3=(1 ; ev. , Lame () ; ev. (1 = 1 ;	00500001/01/devices/me/ceremecryp000002 cs .1/42552520000, Values .	
	3"."0" "din4"."0" "r	
1/1"."0" "rlv2"."0" "rlv3"."0" "rlv4"."0" "vin"."12 3 V" "rcsi"."0" "registe 2:0:2:8	·0·2·8	
1/18/205 15:23:04 sendMessageVertor(1) BrokerTD=4 vertorsToReCent=645 Rute Sented. Rutes Remaining=645 2:0:2:8	t=0. Bytes Remaining=645 2:0:2:8	
03/18/2025 15:23:200 undateSecond(3) recordIndex=0 newState=MOTIPUbAckPendine(6) packetId=3 @ undateStatePublish() 2:0:2:8	packetId=3 @ updateStatePublish() 2:0:2:8	
03/18/2025 15:23:00 MOTT Publish(3) MOTTSuccess publishing BrokerID=1 packetId=3 throttleDelav=100 2:0:2:8	throttleDelay=100 2:0:2:8	
03/18/2025 15:23:00 mattllientPublishLogEntry(3): publish topic v1/devices/me/telemetry (23) BrokerID=1 PacketID=3 OoS=1 Retain=0 xResult =	lemetry (23) BrokerID=1 PacketID=3 QoS=1 Retain=0 xResult =	
MQTTSuccess(0)		
payload: {"ts":1742332920000,"values":{"seq":"1","evt":"Time ()","evtid":"1","id":"000CC8062002","name":"X-410W","model":"X-410CW-	:"1","id":"000CC8062002","name":"X-410W","model":"X-410CW-	
I","din1":"0","din2":"1","din3":"0","din4":"0","rly1":"0","rly2":"0","rly3":"0"," 2:0:2:8	2:0:2:8	
03/18/2025 15:23:00 waitForBrokerResponse(1) caller=mqttClientPublishLogEntry timeout=2000 ms maxRetries=200 2:0:2:8	neout=2000 ms maxRetries=200 2:0:2:8	
03/18/2025 15:23:00 MQTTreceiveSingleIteration() brokerID=1 packetType=0x40 index=4 @\$02\$00\$03 2:0:2:8	<=4 @\$02\$00\$03 2:0:2:8	
03/18/2025 15:23:00 handleIncomingAck() Received MQTT packet of type 0x40. Len=2 Data= \$00\$03 2:0:2:8	Data= \$00\$03 2:0:2:8	-
03/18/2025 15:23:00 deserializeSimpleAck(3) Received ACK for PacketID=3. 2:0:2:8	:8	
03/18/2025 15:23:00 handlePublishAcks(3) ACK packetID=3 deserialized with result: MQTTSuccess(0). 2:0:2:8	: MQTTSuccess(0). 2:0:2:8	-
03/18/2025 15:23:00 updateStateAck(3) recordIndex=0 packetId=3 currentState=6 newState=10 2:0:2:8	vState=10 2:0:2:8	

SETTING UP THE DASHBOARD

Under the Dashboard section of ThingsBoard, click the + button to add a new dashboard. Specify a name and optional description for the Dashboard. ThingsBoard allows different layouts and displays depending on whether the user connects using a desktop browser, tablet, or phone. Again, ThingsBoard documentation describes how to configure mobile-optimized dashboards—this tutorial will cover a typical desktop browser configuration. ThingsBoard also has a mobile application that can be used to connect to the Dashboard.

ThingsBoard	Dashboards >	All Current subscription Thin Status Trial	gsBoard Cloud Maker ends on the Nov 28, 2024	8	Ļ	:
🛧 Home	S All	Groups				
🖻 Plan and billing						
🛕 Alarms	Dashboards 🥰	Include customer entities		+	С	۹
Dashboards	Created time ↓	Title Customer name	e Groups			
III Solution templates						
🛦 Entities 🔨	2024-10-29 17:37:15	CBW Dashboard Test				:
🗔 Devices	2024-10-29 16:38:55	ThingsBoard IoT Gateways				:
E Assets						
Entity views						
🚹 Profiles 🛛 🗸						

Owners own dashboards, so if you have different user groups, you should first set up Customers and Users and assign them to the Dashboard.

Once you have created a Dashboard, click on it to see a blank pallet. One of the key concepts in ThingsBoard is **Entity Aliases**. An Entity refers to a Device or a group of Devices. Defining two Entity Aliases for your Dashboard is useful: **Device** and **Devices**. The **Devices** alias refers to a group of devices you want to appear on a single Dashboard. One common approach is to filter from all the devices based on the Profile and Device name. Since Alarms are set in the Profile, it is common for a customer or common devices to use the same **Device Profile**. Setting the Entity Aliases to the Profile, you will see all devices sharing the same

Profile. If you need to further filter the devices for the Dashboard, you can filter based on **Device's Name**, using a wild card expression "%file%". Customer, Asset, or other parameters can also filter a group of Devices. It's crucial to click **Resolve as multiple entities** to show multiple devices on the list.

Next, define an alias called **Device**. This is used to refer to a single device. Edit and set it so the *Entity is taken from dashboard state parameters*. This allows clicking on a Device and

then transitioning to a new Dashboard state, which only shows the details for that particular device. A Dashboard can have multiple states, and you can define Actions that will change the

	Dashboa >	> 📲 CBW Dashboard	Current subscription ThingsBoard Cloud Make Status Trial ends on the Nov 28, 2
	📚 home - 📔 📰		
	CBW Dashboa	rd Test	
	🛋 CBW Test		
W	Time Name UUID Typ	e AWOL Analog1 Analog2	Analog3 Analog4 DIN1 DIN2 Re
Î		No er	ntities found
Er	ntity aliases		×
	Alias name	Entity filter	Resolve as multiple entities
1.	Device	Entity taken from dashboard sta parameters	te 🍙 🖍 🗙
2.	Devices	Devices of type "default" and wit starting with '%410%'	h name 🛛 🖍 🗙
A	dd alias		Cancel Save

State. Each Dashboard state is a new, blank Dashboard that you can configure using ThingsBoard built-in widgets. To implement clicking on a device from a list of devices and then display the data for the single device, you need to:

- 1. Place an Entities table widget on your Dashboard.
- 2. Edit the Entities table. Set the data source to an Entity and enter the Entity alias **Devices**.
- 3. Add any data you want to display for all the devices on your table from the Timeseries data or Attributes (System, Shared, or Device).
- 4. On the Actions tab, add a new **Action.** It will perform on a **Row Click**, called Details for Controller; the Action will be to Navigate to a new dashboard state, and specify the state name "**DeviceView**." Click "**Set entity from widget**"

 Add a new Dashboard state called "DeviceView". In edit mode, select the state DeviceView. This will then display the Dashboard state DeviceView, a new Dashboard where you can add widgets. ThingsBoard allows a Dashboard to have many different dashboard states, and each state is a unique view of the device.

Edit action	×	Data Appearance Widget card Actions Layout \bigcirc Preview \times Decline \checkmark Apply
Action source* On row click	•	Datasource
Name* Details for Controller		Type Entity
lcon	•••	Entity elias* X Devices
Action Navigate to new dashboard state		
Deen right dashboard layout (mobile view)	X	Image: Second state Image: Second state<
Set entity from widget		# ✓ DIN1: din1 ▲ ✓ × # ✓ DIN2: din2 ▲ ✓ × # ✓ Relay1: rly1 ▲ ✓ ×
State entity parameter name By default		If we relay 2: rhy 2 If we relay 3: rhy 3 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy 4 If we relay 4: rhy
Dashboard state display option Normal	•	
	Cancel Save	Filter Create new

For this example, we will display the main Dashboard in the state Home and the Details in the state DeviceView.



Clicking on any of these Devices will then perform the Action to change the Dashboard state to DeviceView, and the Device entity will be set to the device we clicked on the row. We can customize a Dashboard for the single device to appear as:



To save new users time, we have provided an export of this dashboard; the rules chain a modified Control Widgets (see the addendum articles about that). These are in the file <u>CBW_ThingsBoard_Files.zip</u>. You can import these files using the same configuration described in this article.

ADDING WIDGETS TO THE DASHBOARD

ThingsBoard provides a variety of visual widgets to display data. Widgets can display single or multiple data items. ThingsBoard groups widgets in bundles containing multiple standard widgets, such as Charts or Tables. From the Dashboard, click on Edit and Add Widget to select a bundle and then a widget in the bundle.

Select widgets bund	le	Q 🛓 Import widget 🛛 Widg	ets bundles All widgets X
Charts sys 1	Cards by ()	Alarm widgets sys Total Total Alarms Treperbare Major Crewed Temperature Oficial Crewed Lore Handlify Warning Active	Tables type T • Type \$\u03c6 Mage WMA52 Drive \$\u03c6 Andrew • Type \$\u03c6 Mage WMA52 Drive \$\u03c6 29.54 • Type \$\u03c6 Mage WMA52 Drive \$\u03c6 29.24 • Type \$\u03c6 Mage WMA52 Drive \$\u03c6 29.24 • Type \$\u03c6 \$\u03c6 24.24 Drive \$\u03c6 20.24 • Type \$\u03c6 \$\u03c6 \$\u03c6 24.24 Drive \$\u03c6 24.24 • Did 13 \$\u03c6 \$\u03c6 \$\u03c6 24.24
Count widgets sys 1 Total 3 Device 296	Maps sys 1 Gogle (g) (i) (i) (i) (i) (i) (i) (i) (i) (i) (i	Analogue gauges sys 1	Buttons sys []
Control widgets sys () Round switch Mich control Switch control	Status indicators 5ys 2 being 100 x Biguid strength Diguid strength Diguid strength Diguid strength Diguid strength Diguid strength	SCADA symbols sys SCADA symbols SCADA symbols	Traditional SCADA fluid ys 1 system

Steps to adding a widget:

- Under Datasource, you usually select Entity alias and then Device to choose a single device unless you want to display data from multiple devices on the widget.
- If you have received data from the device, the timeseries data keys will be shown as a dropdown under Data key. Select at least one data item unless your control can display multiple data items, allowing for the specification of multiple data items.
- The widget's form is usually self-explanatory; you select or fill in the needed items. The Basic and Advanced tabs are in the widgets to specify additional parameters or customizations. You can enter CSS and HTML code to control the display, and some widgets allow entering JavaScript to manipulate the data values.
- Save the widget. It will appear at the bottom of the dashboard. Move and resize the widget as needed to position it on the dashboard.
- Save the dashboard.

CONTROL WIDGETS

Most of the widgets are used to display data. Some widgets allow control of output devices, such as the relays. These are in the Control Widget bundle. We can only use control widgets that enable the setting of an attribute value. Some of the older ThingsBoard widgets send an RPC message to the device, which requires additional customization of the Rule Chain. On the top right corner of each widget is a tooltip that lists if the widget can set an Attribute value or send an RPC message.

Select a Control Widget capable of setting attributes like the Power Button. Set the Target device as an Entity alias and select Device so that it will send the command to a single device. Then choose the timeseries data variable representing the output device's data to set the Initial state. This is also used to display the output device's current state. For the output control, select the Action to Set attribute and choose the name of the Shared attribute you defined earlier for each output device. Set the data type of the output device, which is usually Integer or Double.

ashboards	OBIT			
olution templates	🕚 Tubing F	Add widget: Power button		Basic Advanced ? X
Intities 🔨		Target device		Device Entity alias
Assets		Device		× 🖍
Entity views				
Gateways	Tank Leve	Behavior		
Device profiles	0.	Initial state 💿	Use time series 'rly1'	·
Power 'On'		×	Set 'relay1' attribute to: 1	/
Action	Set attr	ibute 👻	Set 'relay1' attribute to: 0	·)
Attribute scope	Shared	•	False	· ·
Attribute key*	relay1	×		
Value	Cor	nstant Function	Default	~
123 Integer 🗸	1	+	Relay1	
		Cancel Apply	24 🔶 px	- Q ()
lesources ^				
Widgets library		Cancel		Preview

The control allows you to enter a custom name and customize its look on the Dashboard. The Advanced tab provides additional customizations. You can also edit the widget and create your own custom bundle with special CSS, HTML, and JavaScript controls.

ADDENDUM: CERTIFICATE X.509 AUTHENTICATION

ThingsBoard allows you to specify an X.509 authentication certificate to secure your MQTT device connection. This is set up so that the device's certificate matches what is stored on the ThingsBoard MQTT broker to establish a secure, encrypted connection. The details for how to do this are listed on the ThingsBoard site at https://thingsboard.io/docs/pe/user-guide/certificates/ and https://thingsboard.io/docs/pe/user-guide/mqtt-over-ssl/. The particulars differ if you use your own ThingsBoard instance or ThingsBoard cloud. The pages provide detailed instructions that we won't repeat

here.

After you have followed the instructions, you will have:

- A server Certificate Authority or CA root certificate. This is prepared on the server and needs to be saved as a .pem file. It is used to validate the server certificate. This is entered using the MQTT Broker's Upload/View Client CA section.
- You can create a unique certificate for each device or a certificate chain that trusts a single root anchor certificate. This allows the single root anchor certificate to be updated or renewed without changing the certificates on every device. The ThingsBoard documentation provides instructions for both methods. After each approach, you will have a private key file key.pem and a public certificate file cert.pem. The cert.pem file is loaded to the Upload/View Client Certificate and key.pem is selected for the Upload/View Client Key in the MQTT Broker setup.
- In the ThingsBoard device setup, select X. 509 for Device Credentials and upload the generated cert.pem file.



• Set the MQTT Broker port to 8883.

Edit MQTT Broker	×			
Broker Name:	ThingsBoard			
Sparkplug B:	Yes No			
O Use Multiple Servers:	Yes No			
Hostname/IP:	ec2-18-117-242-216.us-east-2.compute.ama			
Port:	8883			
Client ID:	000CC8062002			
Username:	cbwuser			
Password:	*******			
C Encrypted:	Yes - Certificate Authentication ~			
Certificate:	Upload/View Client Certificate			
	Upload/View Client Key Upload/View Client CA			
A There will be started	Milliseconds			
	Seconds			
C Reconnection Delay:	30 Seconds			
Keep Alive Interval:	30 Seconas			
Topic Root:				
O Publish Heartbeat:	Yes No			
Clean Session:	Yes No			
Birth Topic:	v1/gateway/connect			
	Prepend Topic Root			
Birth Message:	{"device":"\${clientID}"}			
Last Will Topic:	v1/gateway/disconnect			
1 ast Will Mossago	{"device"."\${clientID}"}			
G Last Will Message:	ן שיאטים . אנטופוומטן ן			
	Save Changes Cancel			