www.AKCP.com

SP+ and WTG MQTT Manual

Introduction

MQTT is a new feature on SP+ and WTG units. It allows you to receive sensor values and sensor status changes via MQTT from the device, up to 4 MQTT brokers.
The following “What is MQTT?” introduction information is based on the HiveMQ FAQ documentation.

What is MQTT?

MQTT is a standardized protocol for messaging and data exchange (developed by OASIS, ISO/IEC 20922:2016). The protocol uses a **publish/subscribe architecture**.
The technology provides a scalable and cost-effective way to connect devices together with minimal protocol overhead. It is able to deliver data over the Internet in near real-time and with guarantees of delivery.
MQTT was originally designed to connect sensor nodes over communication networks that are unreliable or high-latency, or both – it is lightweight, which enables low-cost device communication.
MQTT uses TCP. Due to ordering requirements MQTT over UDP is not possible.

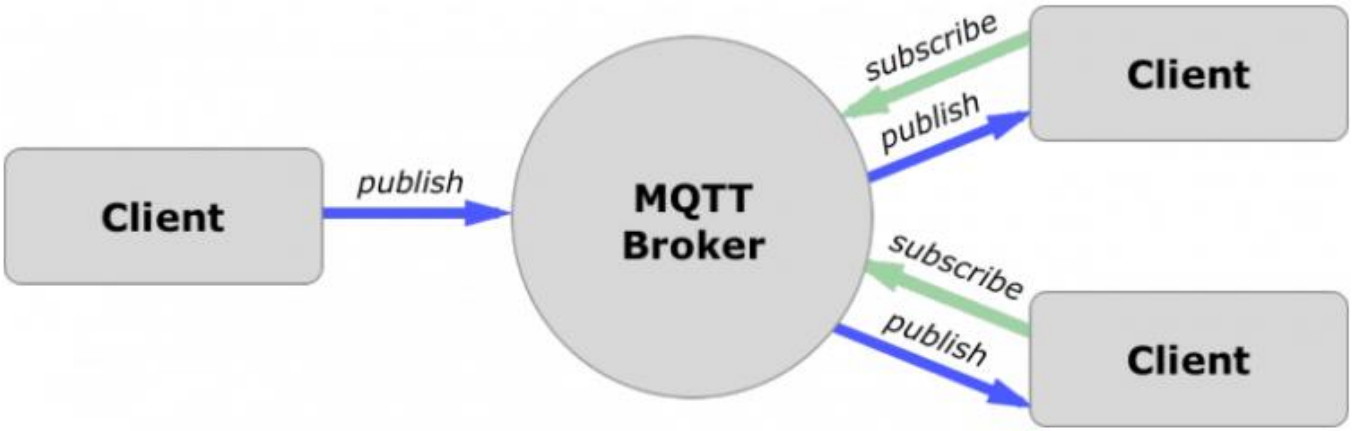
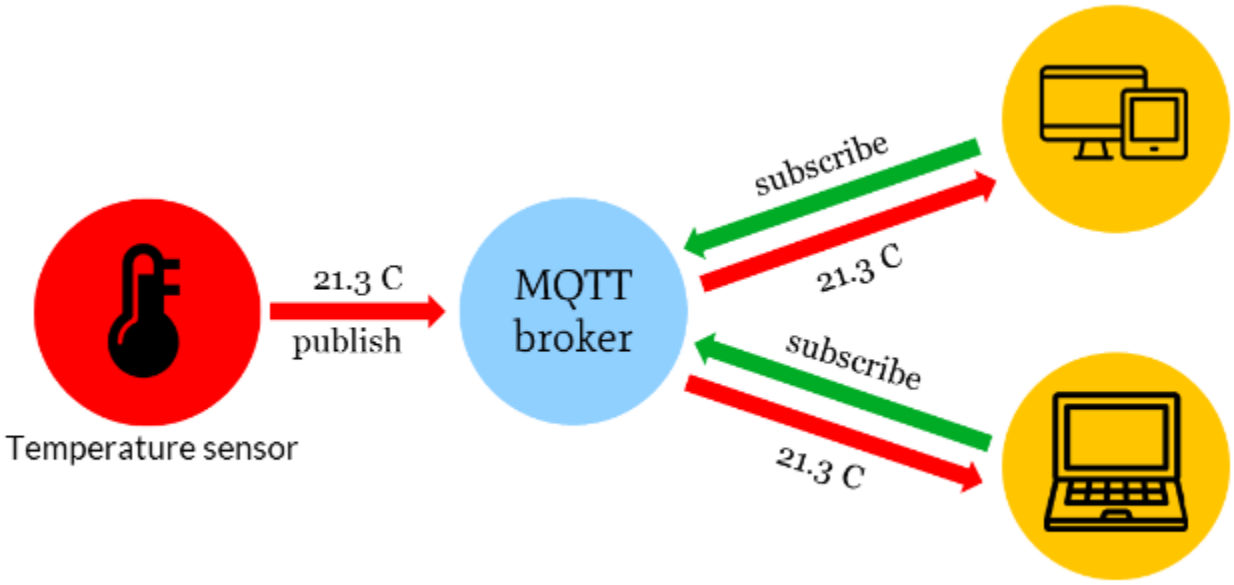


Image source 1sheeld.com

MQTT follows a Publish/Subscribe paradigm. The sender (Publisher) and receiver (Subscribers) of messages communicate via so-called **topics** and are decoupled from each other. The connection between them is handled by the MQTT **broker**. The task of the broker is to receive and filter all incoming messages, determine who is interested in each message, and distribute them correctly to the subscribers.
A **client** (any device that operates an MQTT library and connects to an MQTT broker over a network) doesn’t have to pull the information it needs, the broker pushes the information to the client whenever something new is available.

For example, monitoring a temperature sensor via MQTT works like the following:



Schematic data flow from sensor (machine) to devise (machine)

Image source 1sheeld.com

YouTube videos:

HiveMQ Introduction to MQTT
https://www.youtube.com/watch?v=z4r4hIZcp40

MQTT Protocol – How it works
https://www.youtube.com/watch?v=4QWISyd7SOo

MQTT Topics

Communication in MQTT is based on the **topic** principle. An MQTT topic is a UTF-8 string that the broker uses to filter messages for each connected client. To receive messages, the client must **subscribe** to the topic. A topic can have one or more topic **levels**. Each topic level is separated by a slash (Topic Level Separator).
Each topic must contain at least 1 character and that the topic string permits empty spaces. Topics are case-sensitive.

Image source hivemq.com

Topics support **wildcard characters**. When a client subscribes to a topic, it can subscribe to the exact topic of a published message or it can use wildcards to subscribe to multiple topics simultaneously. A wildcard can only be used to subscribe to topics, not to publish a message. There are two different kinds of wildcards: single-level and multi-level.

Single Level wildcard - replaces one topic level: +

Image source hivemq.com

Multi Level wildcard - covers many topic levels: #

Image source hivemq.com

MQTT Cluster

In an MQTT cluster, multiple brokers are handling the same topics.
If a broker receives new data from a sensor/device, it will distribute the new data between all other MQTT brokers for redundancy (using inter-node communication channels).
In the event of a broker failure or communications problem (called a netsplit event), the data would be still available from other cluster members.
If the failed cluster member becomes online again, the sensor data it has missed during the offline period would be replicated over from the other nodes.

From the subscribed clients point of view, they would get the same data and same topics from any of the brokers participating in the cluster.
From the publisher device/sensor point of view, they can publish their data to any of the brokers participating in the cluster, knowing that their data will be replicated automatically to other brokers.

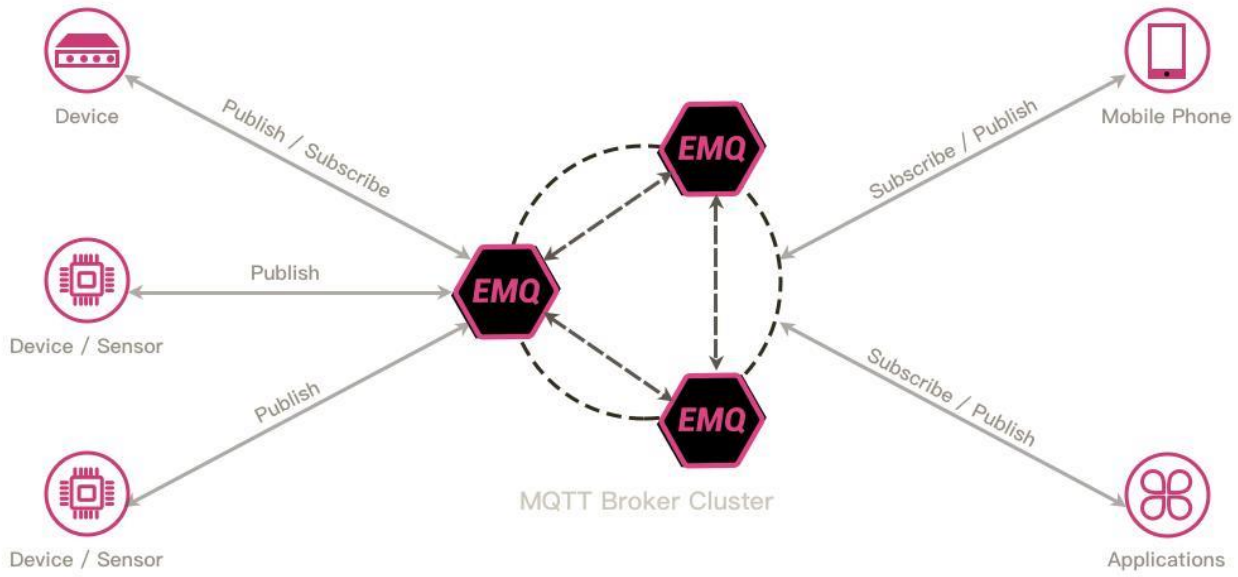


Image source programmersought.com

SP+ and WTG MQTT firmware features

MQTT can be enabled for SP+ and WTG units in the **Settings** menu (see below for configuration).
The current implementation supports MQTT v3.1.1 without encryption as described here:
https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf
We are using JSON strings as MQTT Data type to pass sensor information.

The following features are supported:

- sending sensor status change events on appropriate events from ALL sensors (wired, virtual, etc).
- periodically (interval is set on WebUI) send (update) all sensor values via MQTT
- up to 4 MQTT servers can be defined in the settings, the device will randomly select a server to send the MQTT values

The SP+ and WTG units can **queue up to 1020 MQTT messages**, if the connection is lost to the MQTT server. These messages contain status- and value changes of all online sensors.
The messages in the queue are kept even if the device is powered off, and will be re-sent to the MQTT server once the connection has been reestablished again.

Important note: in the current implementation, all MQTT messages are being stored on the internal flash of the device. If you have a lot of sensors and the MQTT sending interval is low, the device's internal flash will wear out quickly. Therefore, for production usage, you must set the MQTT broadcast interval to at least 15 minutes to avoid damaging the unit.

MQTT transaction speed

The device's communication with an MQTT broker takes some time, due to the nature of composing and sending out MQTT packets. The device sends the MQTT messages one by one, and if during the transaction between the device and the broker is interrupted for some reason, the communication will be restarted from scratch.

Normally on an Ethernet connection, using un-secure MQTT the communication is fast, about 2-4 messages per second.
The encrypted MQTTS (on H7 units) is always slower than an unencrypted MQTT due to encryption overhead and handling of the security certificates, about 1 message per 2 seconds using an Ethernet connection.

If you're using MQTT over a GSM modem connection, the communication speed can become much slower, and it also depends on the quality of your mobile network provider.

MQTT topics

SP+ topics:

`spp/<DEVICE_MAC_ADDRESS>/sensor/status_change/<SENSOR_COMPOUND_ID>`
`spp/<DEVICE_MAC_ADDRESS>/sensor/value_change/<SENSOR_COMPOUND_ID>`

These topics are used for ALL sensor types, including wired and wireless sensors on SP+ and WTG units.

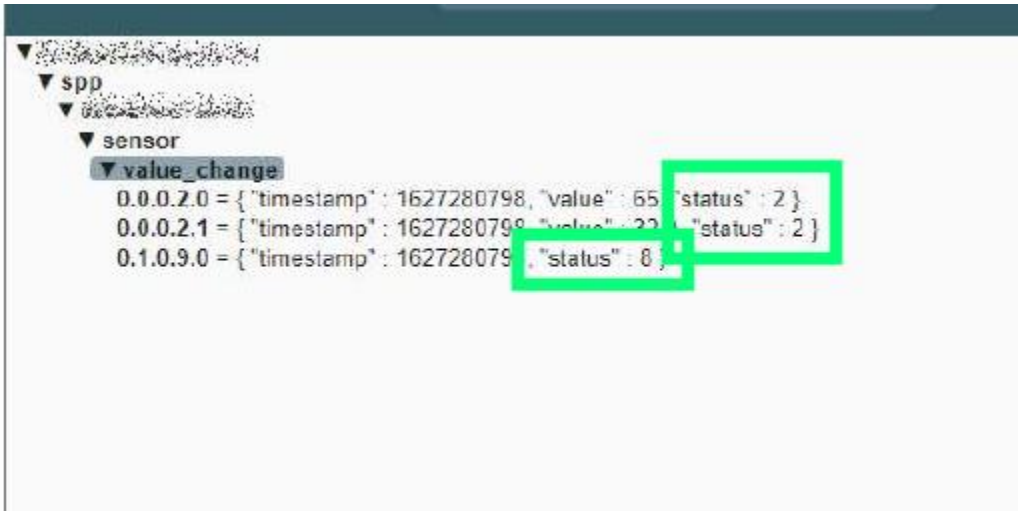
WTS topic:

`wts/DEVICE_ID/graph/PORT.SUBPORT`

This topic is only available on WTG units and is only for the wireless sensors.
While wireless sensor values will be pushed to both wts/ and spp/ prefixes, if somebody requires only the wireless sensors processing, it is better to subscribe to the WTS topic.

Sensor status codes

The sensor statuses are sent in the MQTT packets, represented by numbers.
Below you can find the values of each sensor status with their respective numbers.



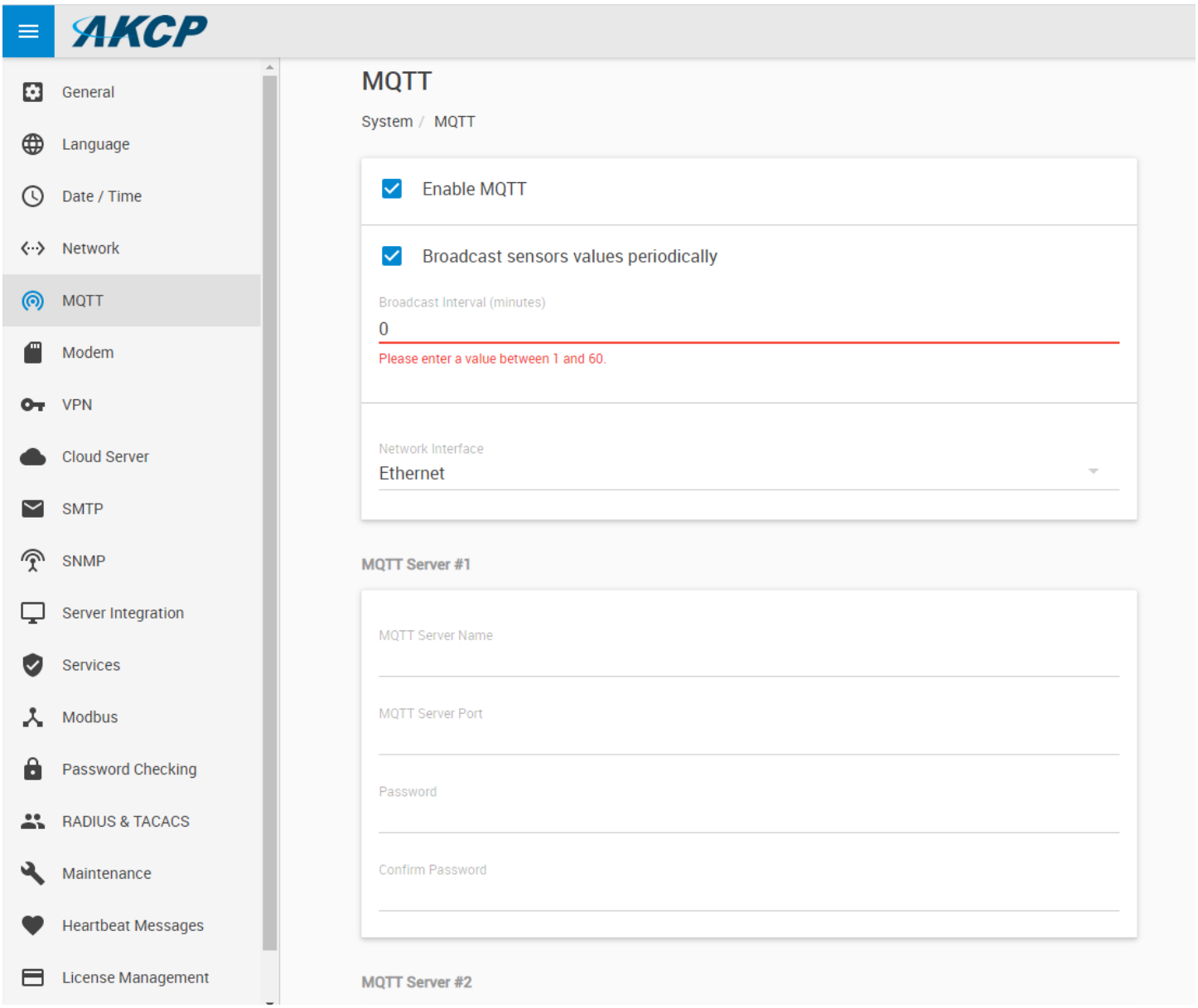
NOSTATUS = 1 : Sensor is in an unknown or undefined state
SENSORNORMAL = 2 : Sensor is in normal state
HIGHWARNING = 3 : High warning state for analog sensor types
HIGHCRITICAL = 4 : High critical state for analog sensor types
LOWWARNING = 5 : Low warning state for analog sensor types
LOWCRITICAL = 6 : Low critical state for analog sensor types
SENSORERROR = 7 : Sensor error state

Additional statuses (not all applies to WTS sensors):

SWITCH_LOW_OUT = 8 : Low state for output (switch) sensor types
SWITCH_HIGH_OUT = 9 : High state for output (switch) sensor types
NO_VOLT_PRESENT = 10 : No voltage detected (only when SP+ polls the SwitchRelayOutputVoltStatus OID of switch output sensor from SP devices)
VOLT_PRESENT = 11 : Voltage detected (only when SP+ polls the SwitchRelayOutputVoltStatus OID of switch output sensor from SP devices)
12 : Reserved value for future use, currently unused
STATUS_ACKED = 13 : Sensor status is acknowledged
STATUS_OFFLINE = 14 : Sensor status is offline
UNREACHABLE = 15 : Sensor is unreachable

SP+ and WTG MQTT configuration

Open **Settings** page -> **MQTT**



1. Click **Enable MQTT**
2. Click **Broadcast sensor values periodically**
3. Set the broadcast interval between 1-60 minutes (set below 15 minutes ONLY for testing)
4. Set the network interface which MQTT will broadcast on; for wired network choose Ethernet. GSM (Modem) and Wi-Fi (on WTG units) are also supported.

Note: if the "Broadcast sensor values periodically" option is not turned on, the unit will only send MQTT packages when a sensor status or value reading change occurs.

Important note: in the current implementation, all MQTT messages are being stored on the internal flash of the device. If you have a lot of sensors and the MQTT sending interval is low, the device's internal flash will wear out quickly. Therefore, for production usage, you must set the MQTT broadcast interval to at least 15 minutes to avoid damaging the unit.

MQTT server parameters:

Up to 4 servers can be configured. If you have an MQTT cluster, you can define all cluster nodes here and all nodes in the cluster will receive the messages.

Depending on the MQTT server's configuration, you may need to specify username/password.
This is a server setting; WTG/SP+ can connect without authentication (leave these fields blank).
If your server requires authentication to accept data from the WTG/SP+, by default the device's MAC address is used as the Username.

For testing MQTT, toggle the flag "send values periodically" and set 1 minute for broadcasting interval. You should see new values appearing in MQTT.
After testing, set this value to 15 minutes. DO NOT USE 1 minute value in a production setup!

MQTTS support (H7 units only):

On the newer H7 platform the secured, encrypted MQTTS is also supported in addition to the existing unencrypted MQTT.

MQTT communication methods supported for MQTT on H7:

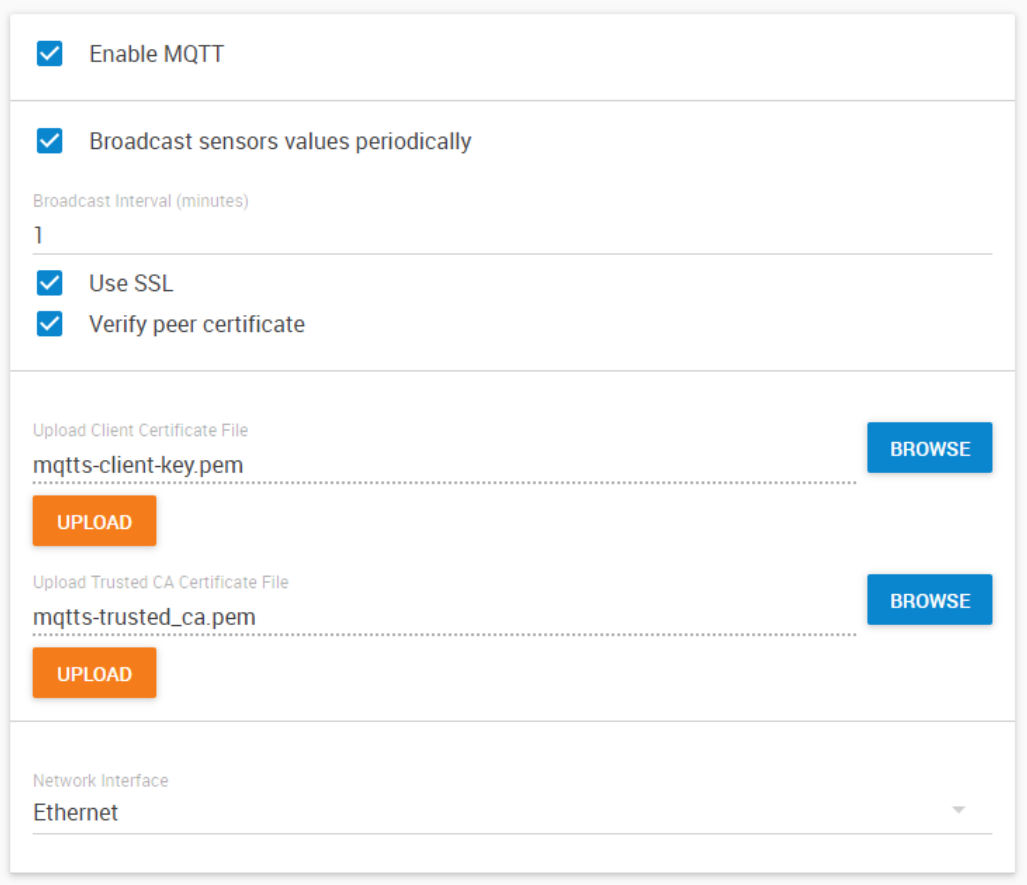
- unencrypted, unauthenticated – same as on F7 platform
- unencrypted, authenticated (username and password) – same as on F7 platform
- encrypted, unauthenticated
- encrypted, using client certificate
- encrypted, authenticated (username and password + SSL certificate)

For using the SSL (encrypted) method, a client .PEM x509 certificate has to be uploaded.
It has to be in the same format as the HTTPS certificate: contain the client's private key and certificate combined, without password.

The maximum size of the uploaded .PEM file must not exceed 8 Kilobytes.

The client certificate doesn't support multiple client and server certificates combined into one .PEM file for certificate chain validation.
The server's certificate has to be uploaded separately, if required (see below).

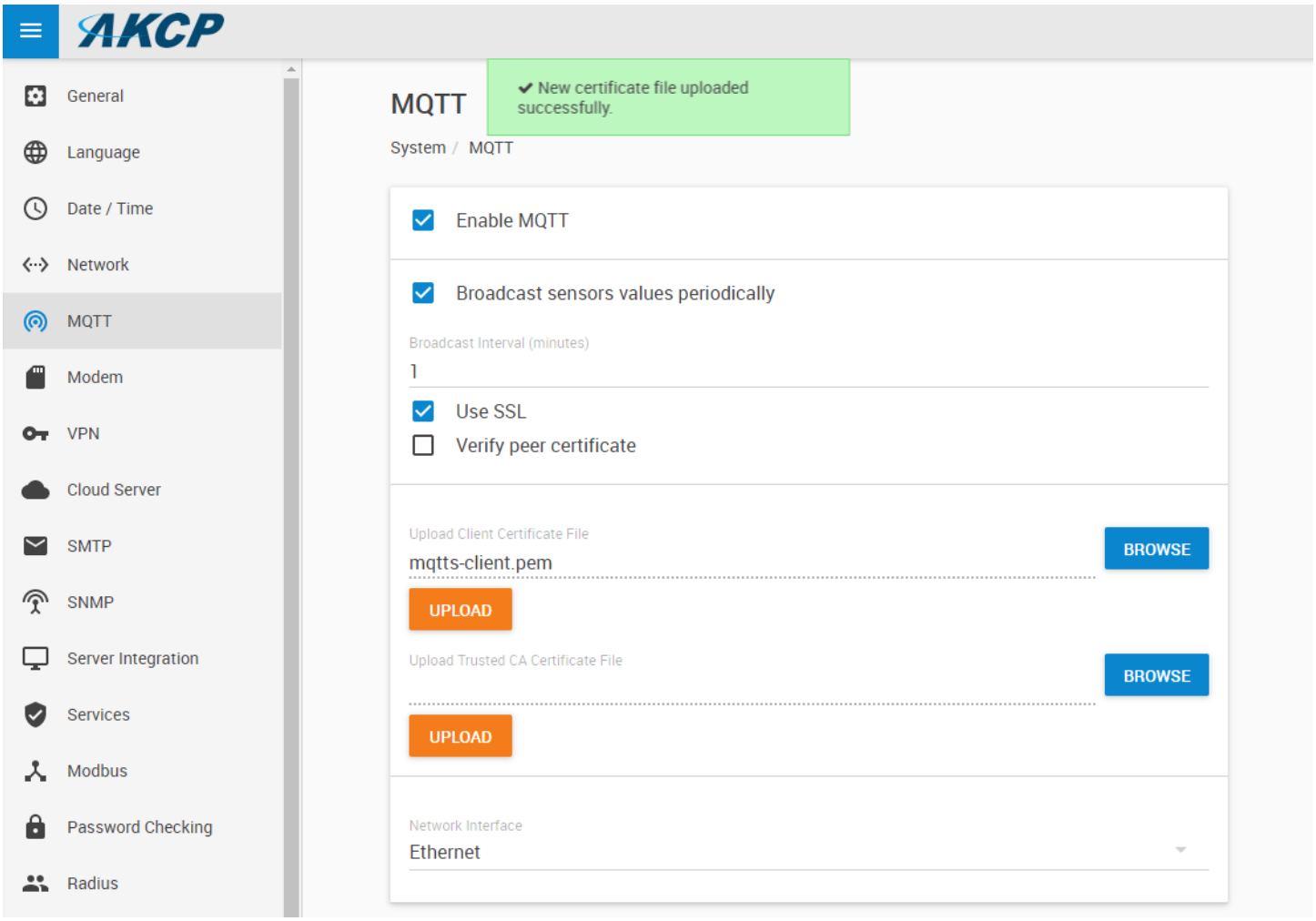
If the option "verify peer certificate" is turned on, the trusted CA certificate has to be uploaded separately, and it has to be valid (not expired).
If this option is turned off, then the server CA certificate is not validated (it could be expired).



On this example picture, the MQTTS is configured with the encrypted SSL certificate method, the certificate validation is turned on, and certificates are being uploaded for MQTTS.
You will need to browse and upload each .PEM certificate separately. Once uploaded and the settings are saved, the certificate file names will not be displayed, but they will be in use.

If the .PEM file upload fails and you get an error message, there is a problem with the certificate's format. Please refer to the "*Adding Security Certificates to AKCP products*" manual for making a correct .PEM file.

Note: the maximum size of the uploaded .PEM file must not exceed 8 Kilobytes.



Important: when you upload the .PEM certificate files, after selecting the file with "Browse" you also need to click on the "**Upload**" button for each uploaded client and/or server certificate file.
If you forget to click "Upload" and just click on "Save" at the bottom of the page, it will not upload the certificate files, which will in turn fail the MQTTS feature.

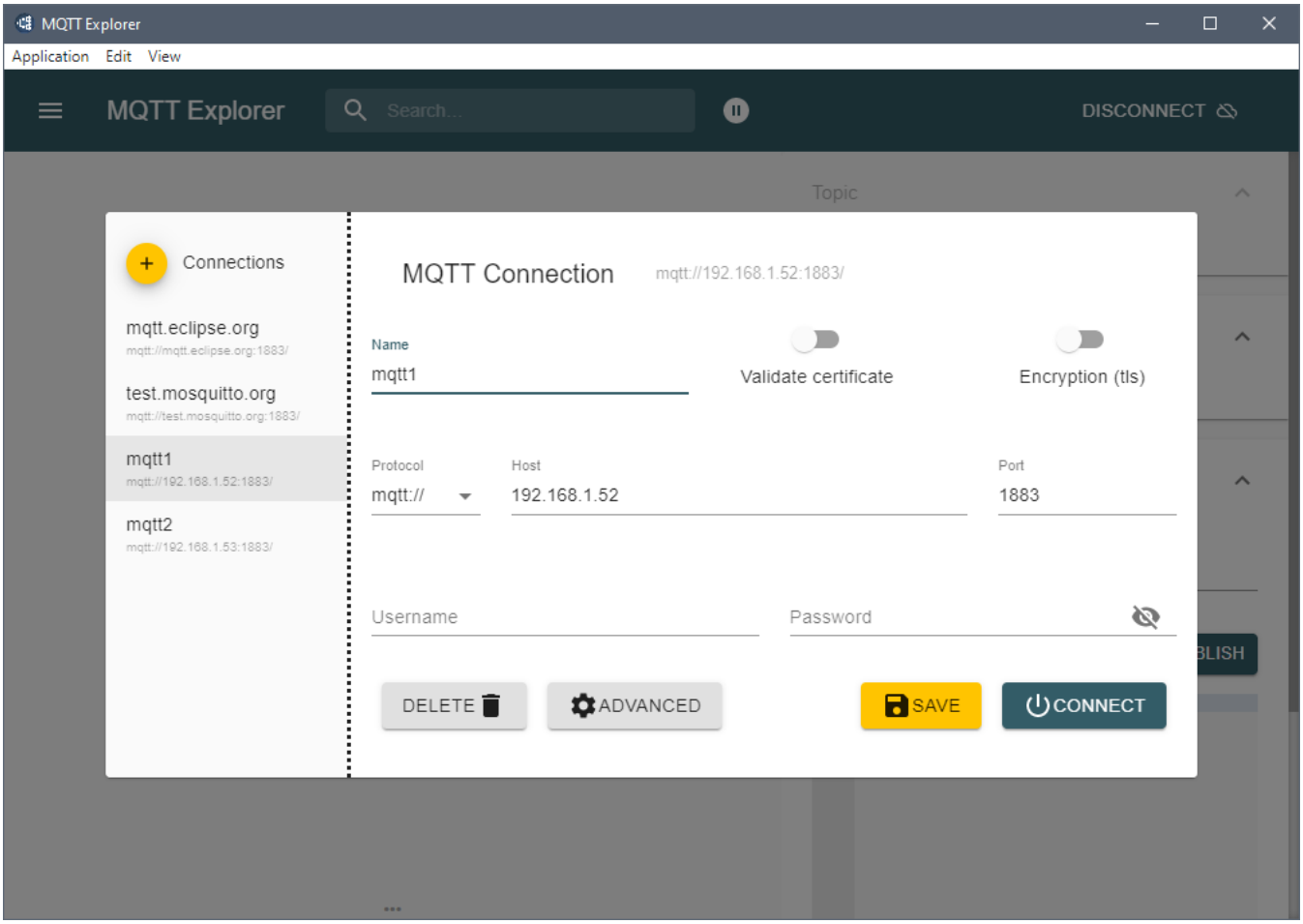
If the .PEM file format is correct and the upload is successful, you will see a green popup message indicating that the file has been uploaded successfully.
If you don't see this popup, then your certificate file is not yet uploaded.

Monitoring MQTT

To monitor MQTT, we recommend using the free MQTT Explorer application:

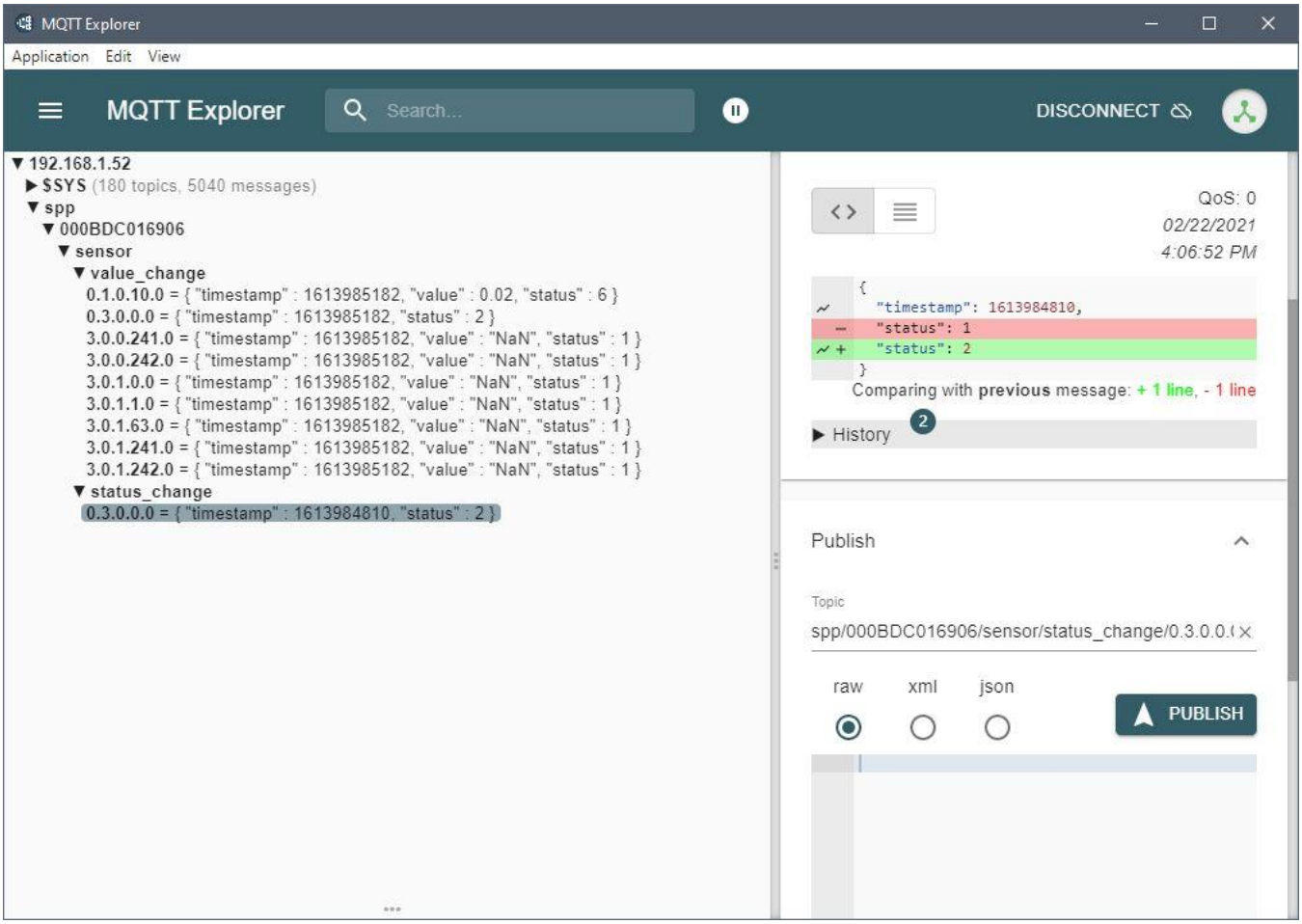
MQTT Explorer (versatile Windows MQTT client)
http://mqtt-explorer.com/

Configure MQTT Explorer to connect to an MQTT server:



Note: you would need an MQTT broker already set up and running, before you can connect to it. Depending on the server's configuration, you may need to specify username/password.

MQTT values will appear from your SP+ or WTG unit:

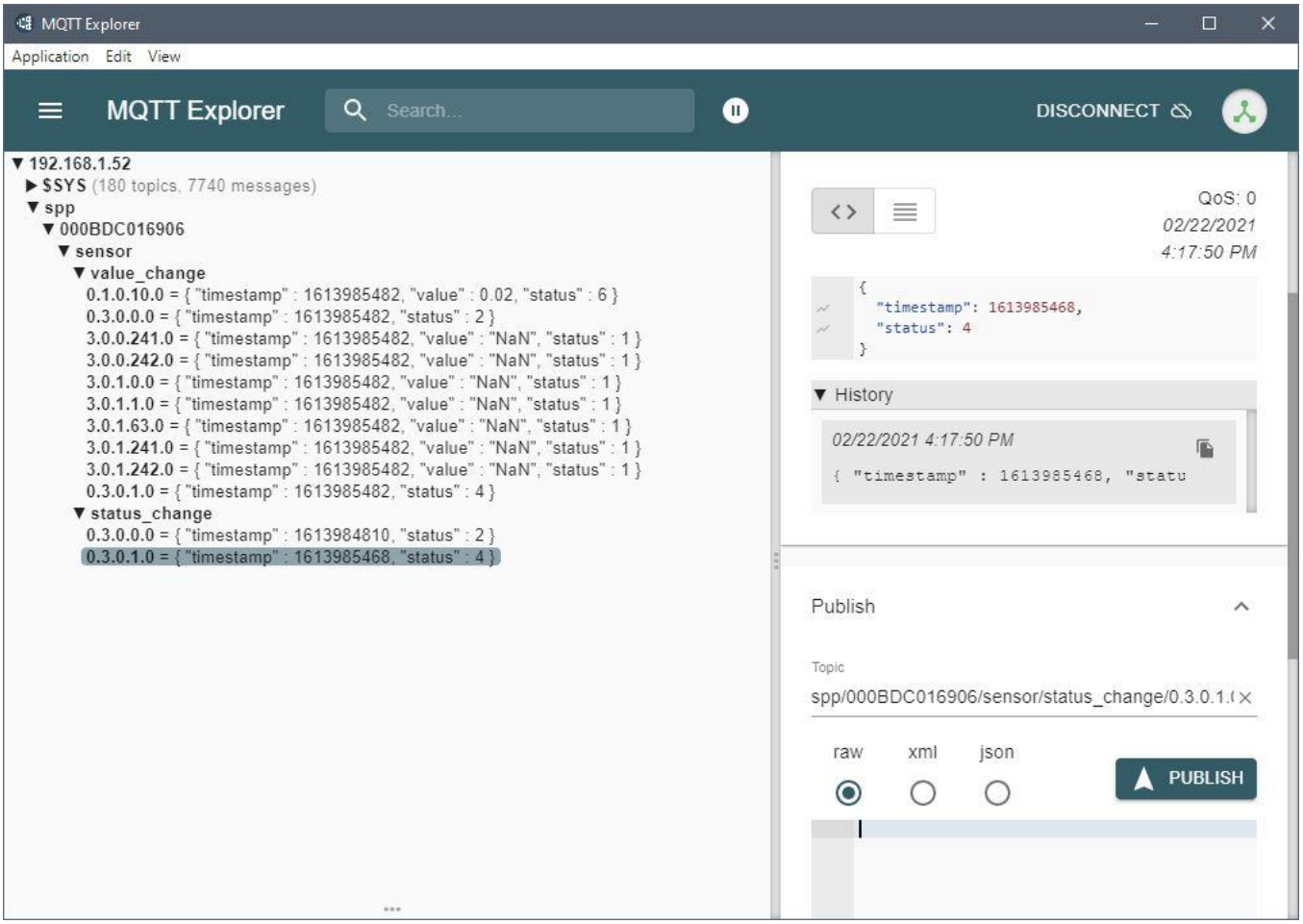


As noted earlier, the MQTT topic will have the following format:

spp/${DeviceID}/sensor/status_change/${CompoundID}
spp/${DeviceID}/sensor/value_change/${CompoundID}

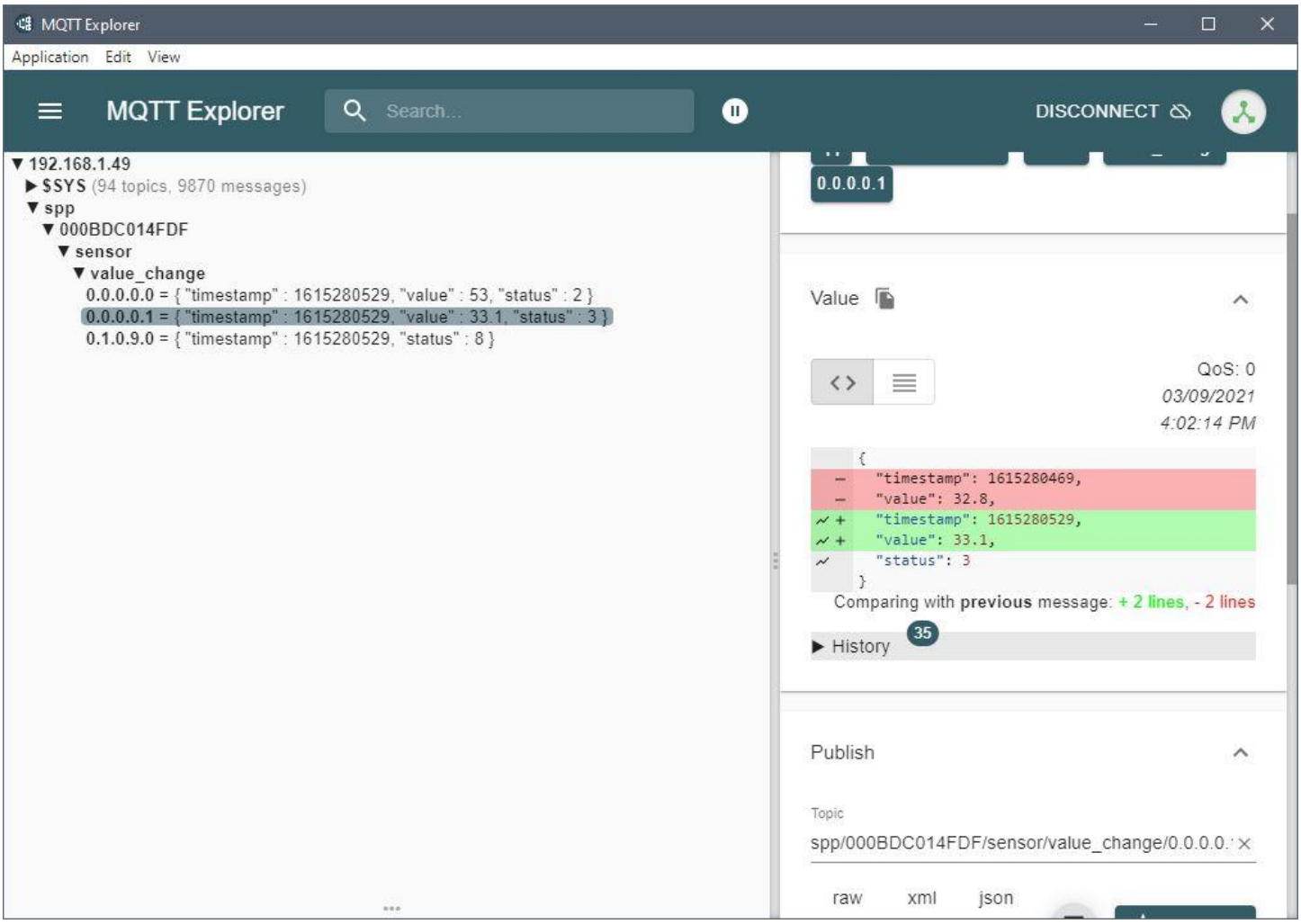
The *Device ID* is the MAC address of the unit.

When a new sensor becomes online on the device, a new topic will appear with a new Sensor ID:



Example

This screenshot below shows an SPX+ unit with a Temperature/Humidity sensor plugged in Port 1.
The **Device ID** is **000BDC014FDF** which is the MAC address of the unit.
The **Temperature sensor** has the **Sensor ID 0.0.0.0.1**



Example MQTT Explorer configuration with online broker

In the following example we will use the public MQTT test server from HiveMQ.
We will use an SPX+ with a Temperature & Humidity sensor connected.

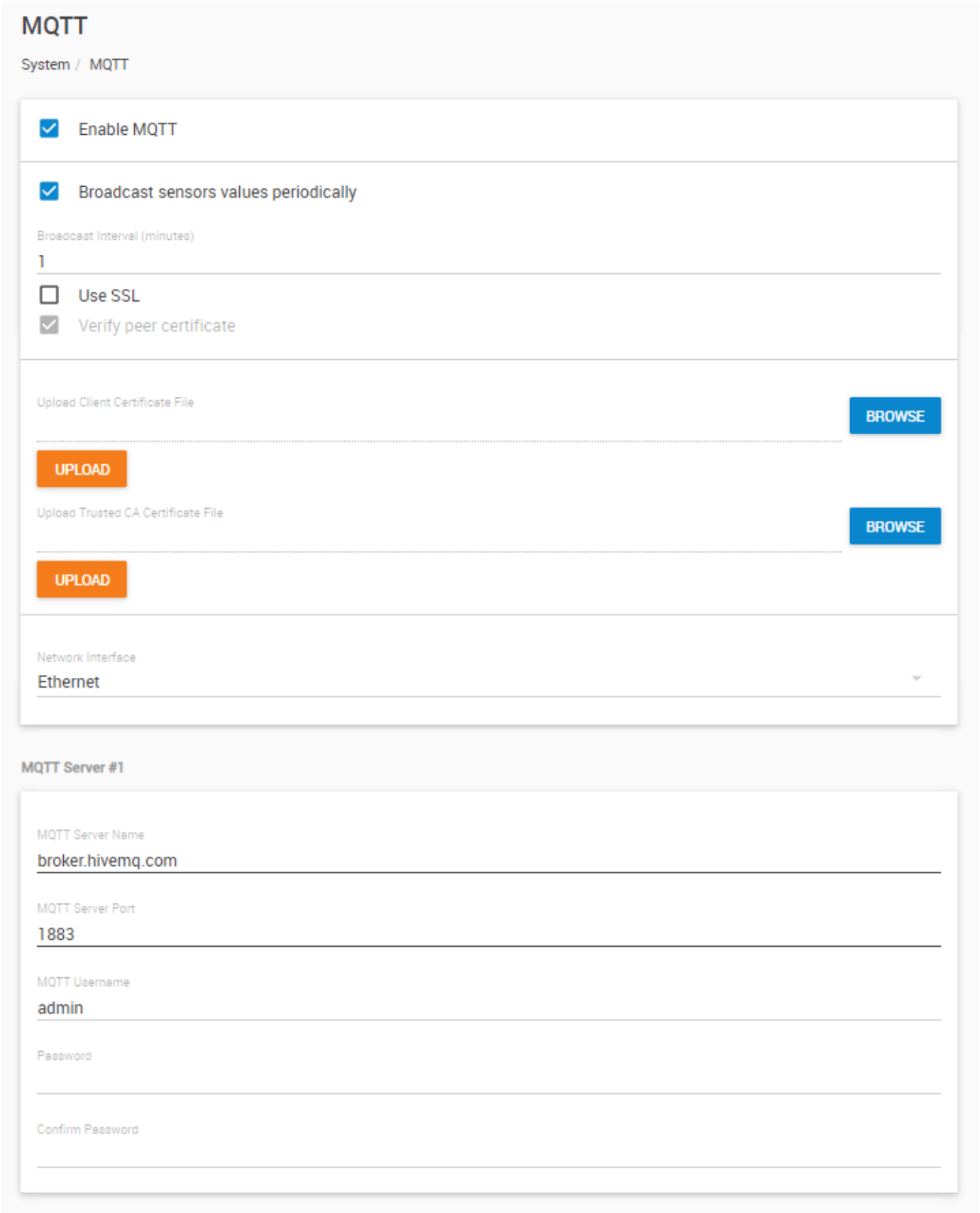
On the webpage, the MQTT connection settings are shown as follows:

MQTT connection settings
Host: broker.hivemq.com
TCP Port: 1883

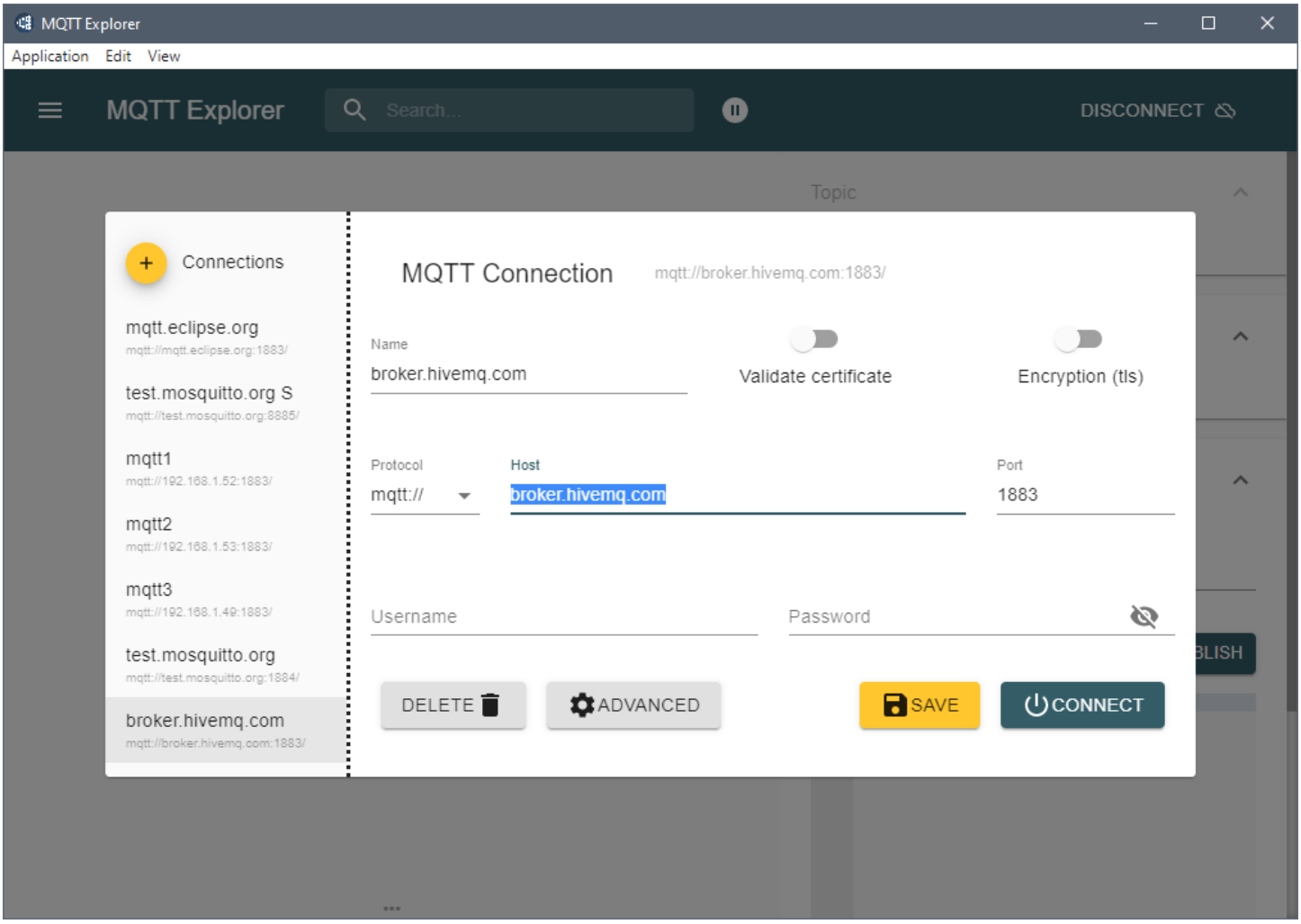
First, configure the MQTT server settings on the SPX+ side.

1. Click "**Enable MQTT**".
2. Click "**Broadcast sensor values periodically**" and set it to **1 minute interval**.
3. The HiveMQ test broker doesn't use SSL or authentication, therefore **don't enable SSL** and **do not specify a password** (leave it blank). You may use any value for the username, it will not be used (however, the SPX+ WebUI form requires a username parameter to save the settings – for this reason we use the value "admin").
4. Fill out the connection details for MQTT Server #1: **broker.hivemq.com** and default port **1883**.
5. Scroll down and click "**Save**".

See the screenshot on the following page.


MQTT
System / MQTT
Enable MQTT
Broadcast sensors values periodically
Broadcast Interval (minutes)
1
Use SSL
Verify peer certificate
Upload Client Certificate File
BROWSE
UPLOAD
Upload Trusted CA Certificate File
BROWSE
UPLOAD
Network Interface
Ethernet
MQTT Server #1
MQTT Server Name
broker.hivemq.com
MQTT Server Port
1883
MQTT Username
admin
Password
Confirm Password

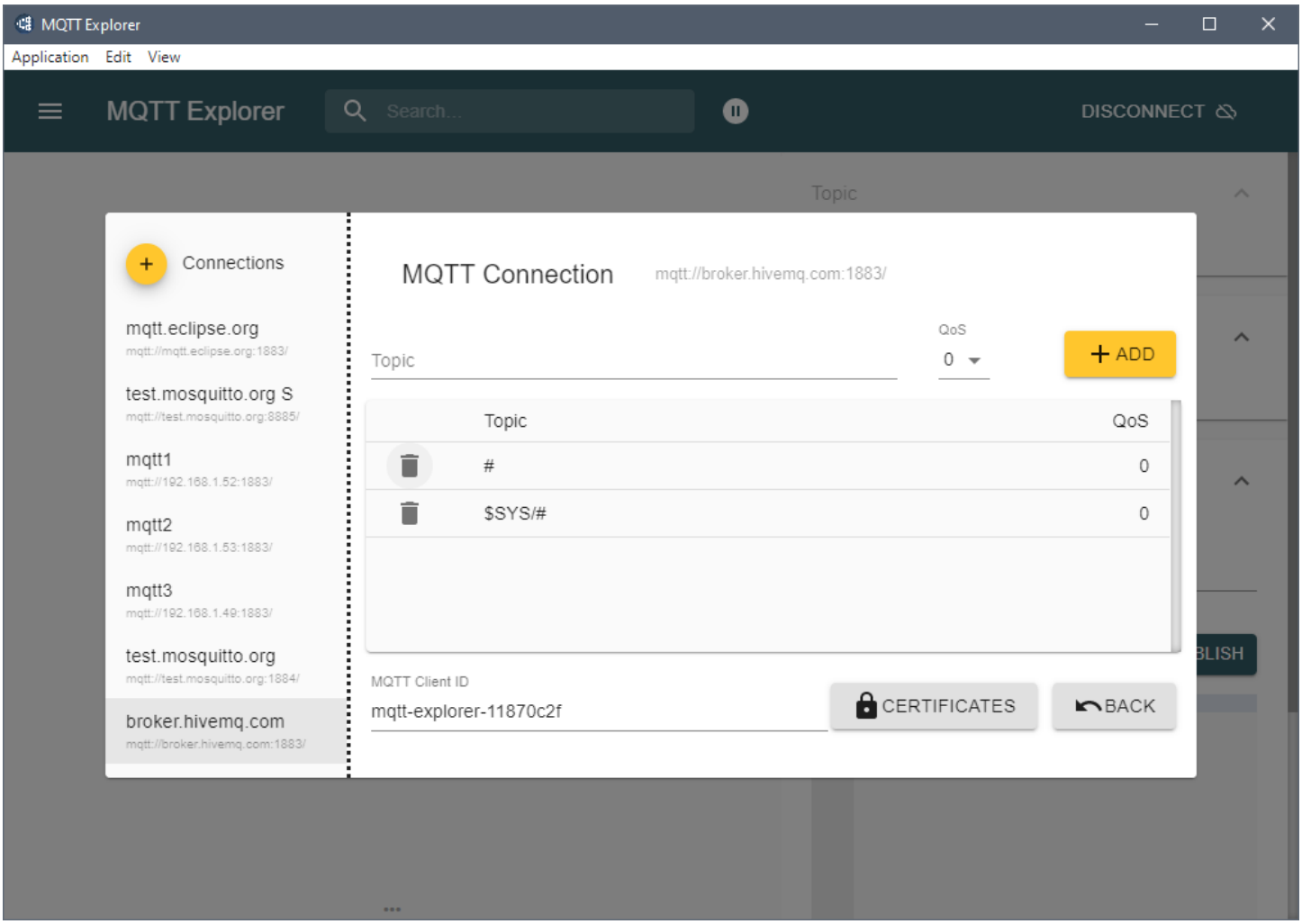
Next, start MQTT Explorer and create a new connection (click on the yellow + sign).



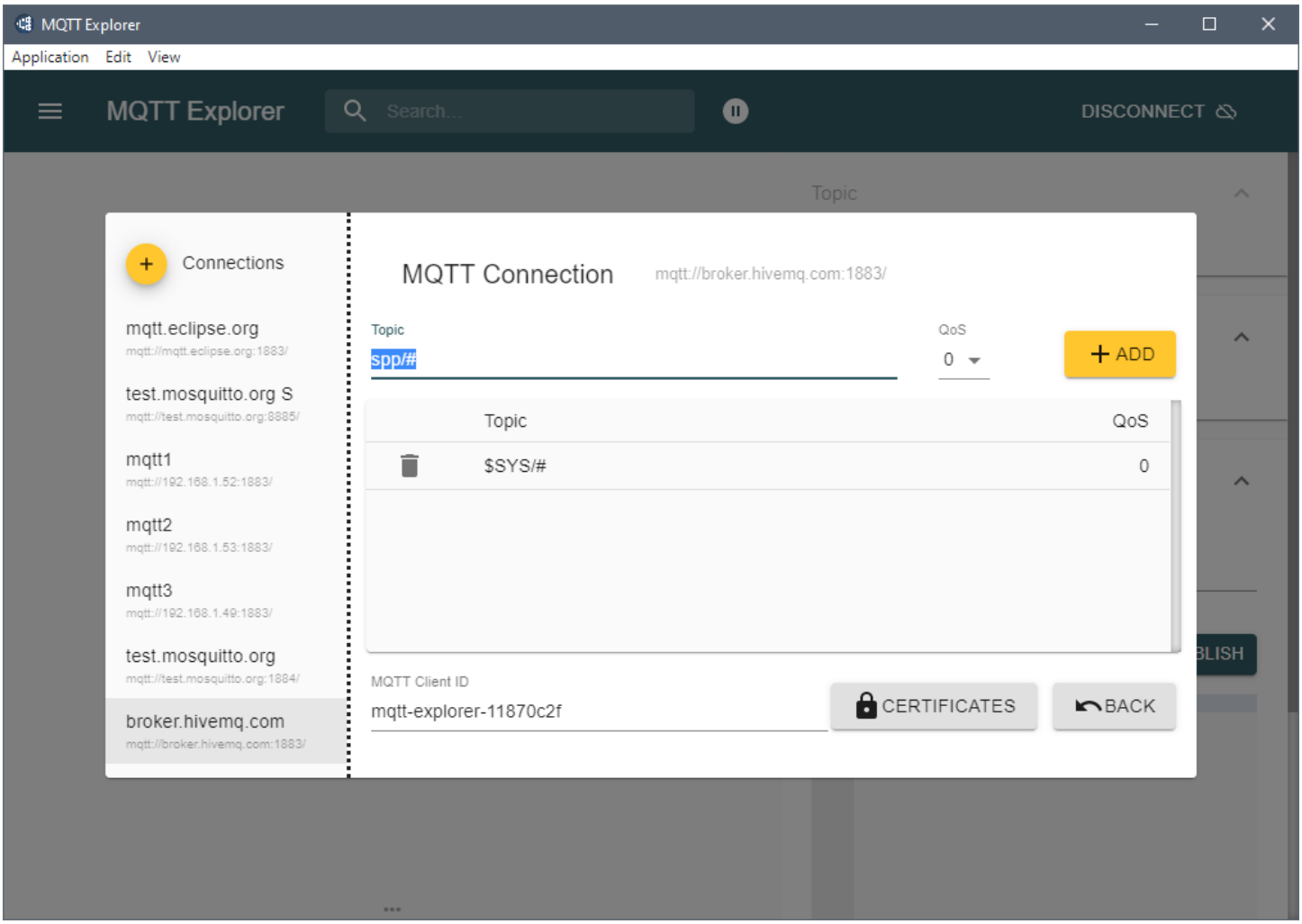
Specify the same connection settings as for the SPX+:

1. **Connection name**: it can be set to any value, here we use broker.hivemq.com
2. Do not enable “Validate certificate” and “Encryption (tls)” settings, leave them turned off.
3. Fill out the **hostname to connect to**: broker.hivemq.com
4. Do not specify any username or password to connect.

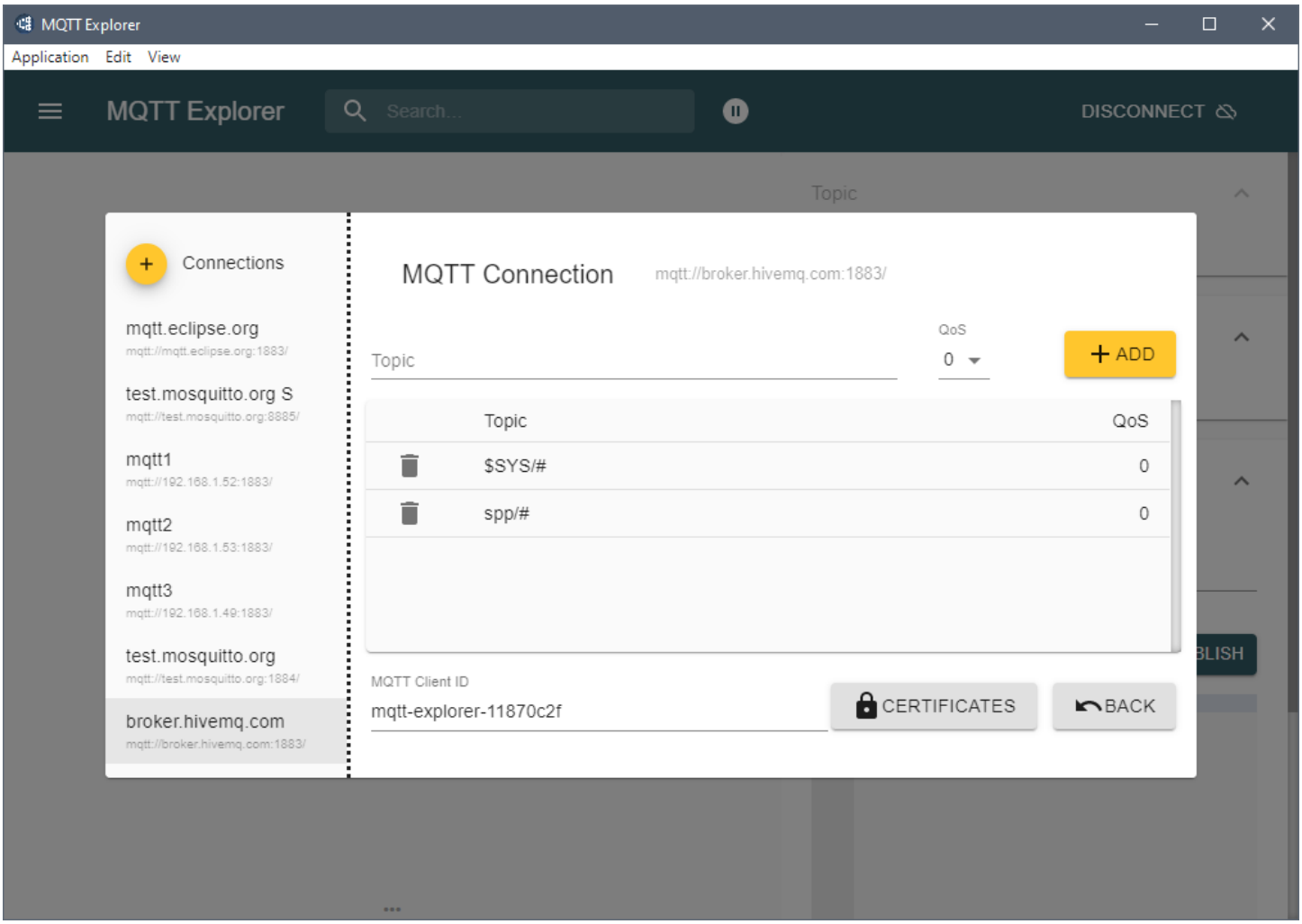
Next, click on “**Advanced**” button to set the MQTT topic, since we don’t want to subscribe to all topics that are available on this public broker.



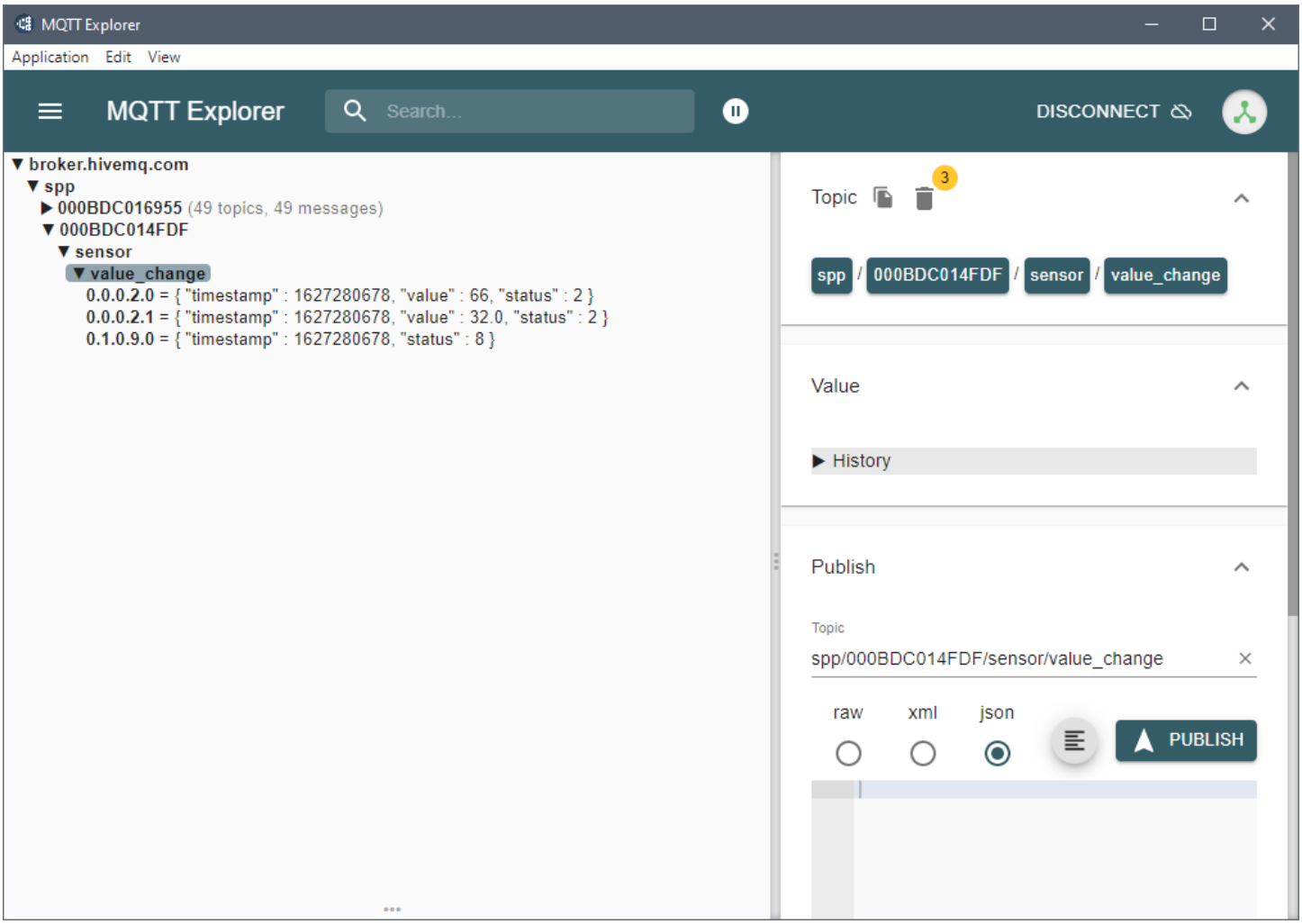
The default topic configuration is # which subscribes to all available topics on the server.
We don’t want to get values from other devices, therefore remove this setting with the delete icon.



Next, specify the topic that we are interested in.
This will be the **spp/#** topic and subtree, which will subscribe to any AKCP devices publishing to this MQTT server.
Type in spp/# and click "**Add**".

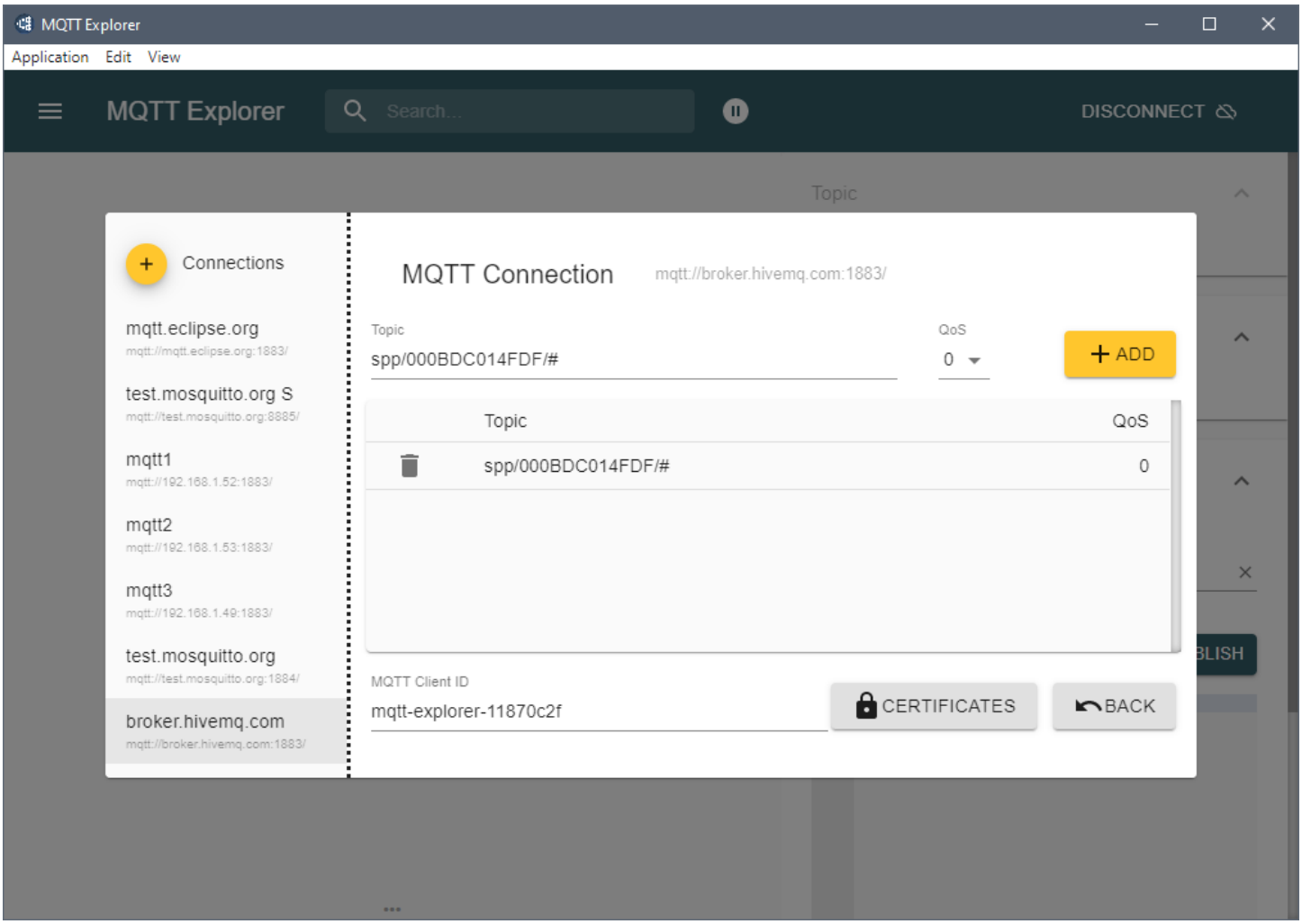


The topic subscription setting will be saved, and you can click “**Back**” button and connect to the MQTT server.



If all settings are correct, the connected SPX+ hostname and all its sensors will be displayed after 1-2 minutes.
In our example that is the 000BDC014FDF host, but there is already another AKCP device visible in this topic.

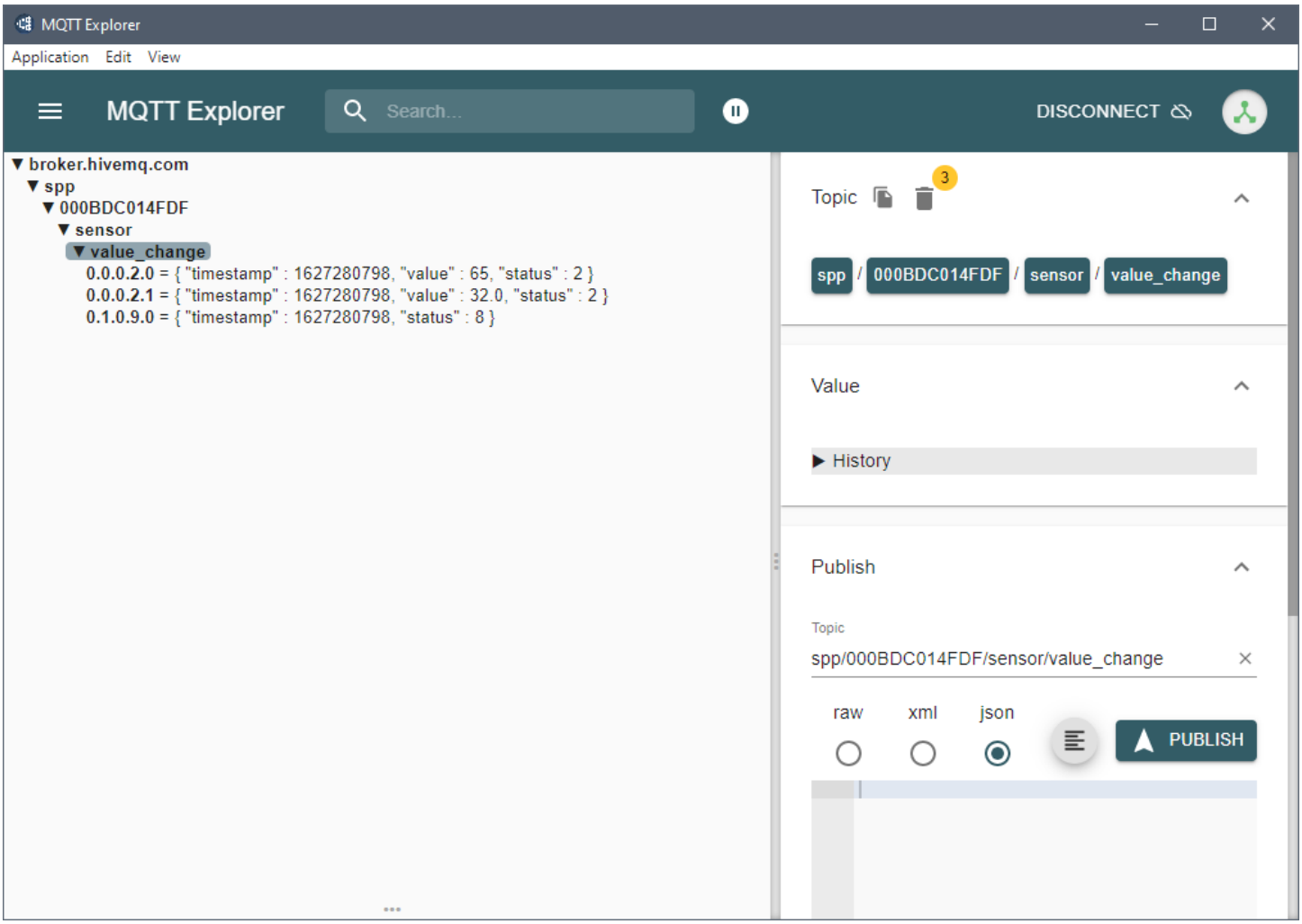
If you only want to subscribe to and display the values for a single host, disconnect the server and go back to the “Advanced” configuration to specify a different topic setting.



On this screenshot we reconfigured the topic to only display sensor values and statuses from our example host 000BDC014FDF and ignore all others.

The topic setting is spp/ 000BDC014FDF/#

Please review the first part of the manual about using MQTT wildcards.



As shown on this example screenshot, with this topic setting, only our SPX+ device and its sensors are subscribed to.

As noted earlier, remember that the supported MQTT topic settings on WTG and SP+ units have the following format:

spp/${DeviceID}/sensor/status_change/${CompoundID}
spp/${DeviceID}/sensor/value_change/${CompoundID}

You can narrow down the topic subscription, based on your needs.

Example MQTTS configuration with online broker

In the following example we will use the public MQTT test server from Mosquitto (test.mosquitto.org). We will use an SPX+ with a Temperature & Humidity sensor connected.

On the webpage, the MQTTS connection settings are shown as follows:

MQTTS connection settings (encrypted, authenticated)
Host: test.mosquitto.org
TCP Port: 8885

MQTTS connection requires certificates.
The Mosquitto test server's certificate **mosquitto.org.crt** can be downloaded from the webpage, or use this link:
http://test.mosquitto.org/ssl/mosquitto.org.crt

To generate the SPX+ unit's client certificate that is accepted by this test server, follow the steps below.

Note: You can directly run the Openssl commands below in a Linux terminal or on Windows if the Openssl package is installed. You cannot run Openssl on the SPX+ or WTG units; the request must be generated on their behalf.

Generate a CSR using the openssl utility
Generate a private key:

```
openssl genrsa -out spxclient.key
```

Generate the CSR:

```
openssl req -out spxclient.csr -key spxclient.key -new
```

Note that you must specify custom values for the following parameters when generating the request, or it will not be accepted by the test.mosquitto.org server:

1. Country Name (2 letter code)
2. State or Province Name (full name)
3. Organization Name (eg, company)
4. Common Name (e.g. server FQDN or YOUR name)

Copy-paste the contents of the `spxclient.csr` file to the webpage at http://test.mosquitto.org/ssl/
If there's any issues with your request (ex. missing common name field) the webpage will tell you.
If the request is accepted, the certificate file **client.crt** will be downloaded automatically.

At this stage you should have 3 files ready: **spxclient.key, client.crt** and **mosquitto.org.crt**
Additional steps are necessary to create the .PEM certificate files that can be uploaded on the SPX+.

Preparing the test .PEM certificate files

Open the file **spxclient.key** with Notepad++, go to the end of the file and copy-paste the contents of **client.crt** just below it.
Now you should have a file with the private key and certificate combined.
Save this file as **spxtest.pem**

Rename the file **mosquitto.org.crt** to **mosquitto.org.pem**

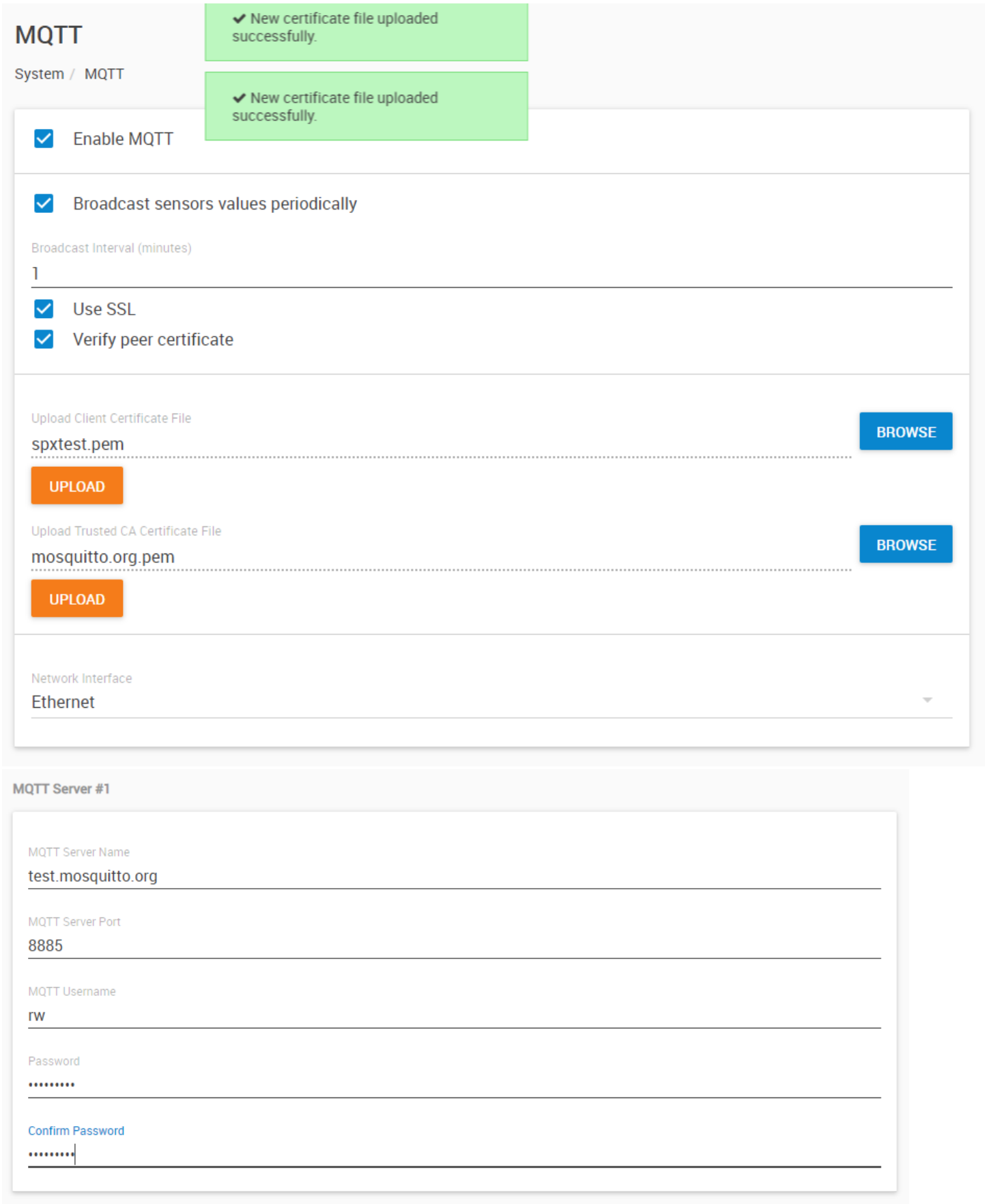
Now you should have 2 files which can be uploaded to SPX+:
spxtest.pem (client certificate)
mosquitto.org.pem (server certificate)

Set up MQTTS on SPX+

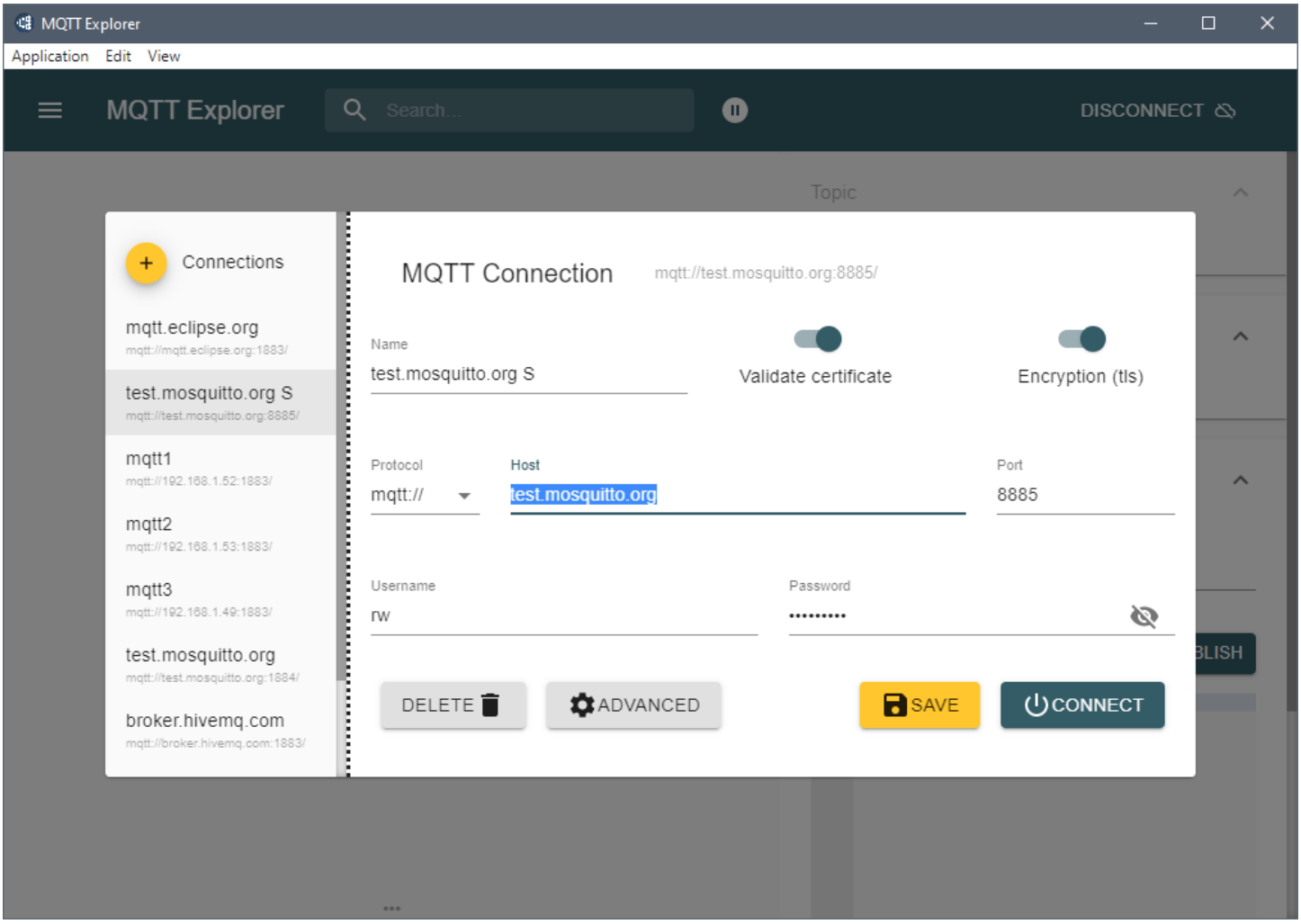
Configure the MQTTS settings on the SPX+ side as follows.

6. Click "**Enable MQTT**".
7. Click "**Broadcast sensor values periodically**" and set it to **1 minute interval**.
8. Click "**Use SSL**" to enable MQTTS
9. Click "**Verify peer certificate**" to enable the certificate verification – this is optional
10. **Browse to** and then **press Upload button** for the file **spxtest.pem** client certificate
11. **Browse to** and then **press Upload button** for the file **mosquitto.org.pem** server certificate
12. You should see 2 green popup messages saying that the certificate files have been uploaded successfully. If you don't see these, there is a problem with the .PEM certificate files, try creating them again.
13. Fill out the connection details for MQTT Server #1: **test.mosquitto.org** and port **8885**.
14. Specify MQTT username and password as **rw** and **readwrite** respectively
15. Scroll down and click "**Save**".

See the screenshot on the following page.


MQTT
System / MQTT
✔ New certificate file uploaded successfully.
✔ New certificate file uploaded successfully.
Enable MQTT
Broadcast sensors values periodically
Broadcast Interval (minutes)
1
Use SSL
Verify peer certificate
Upload Client Certificate File
spxtest.pem
BROWSE
UPLOAD
Upload Trusted CA Certificate File
mosquitto.org.pem
BROWSE
UPLOAD
Network Interface
Ethernet
MQTT Server #1
MQTT Server Name
test.mosquitto.org
MQTT Server Port
8885
MQTT Username
rw
Password
Confirm Password

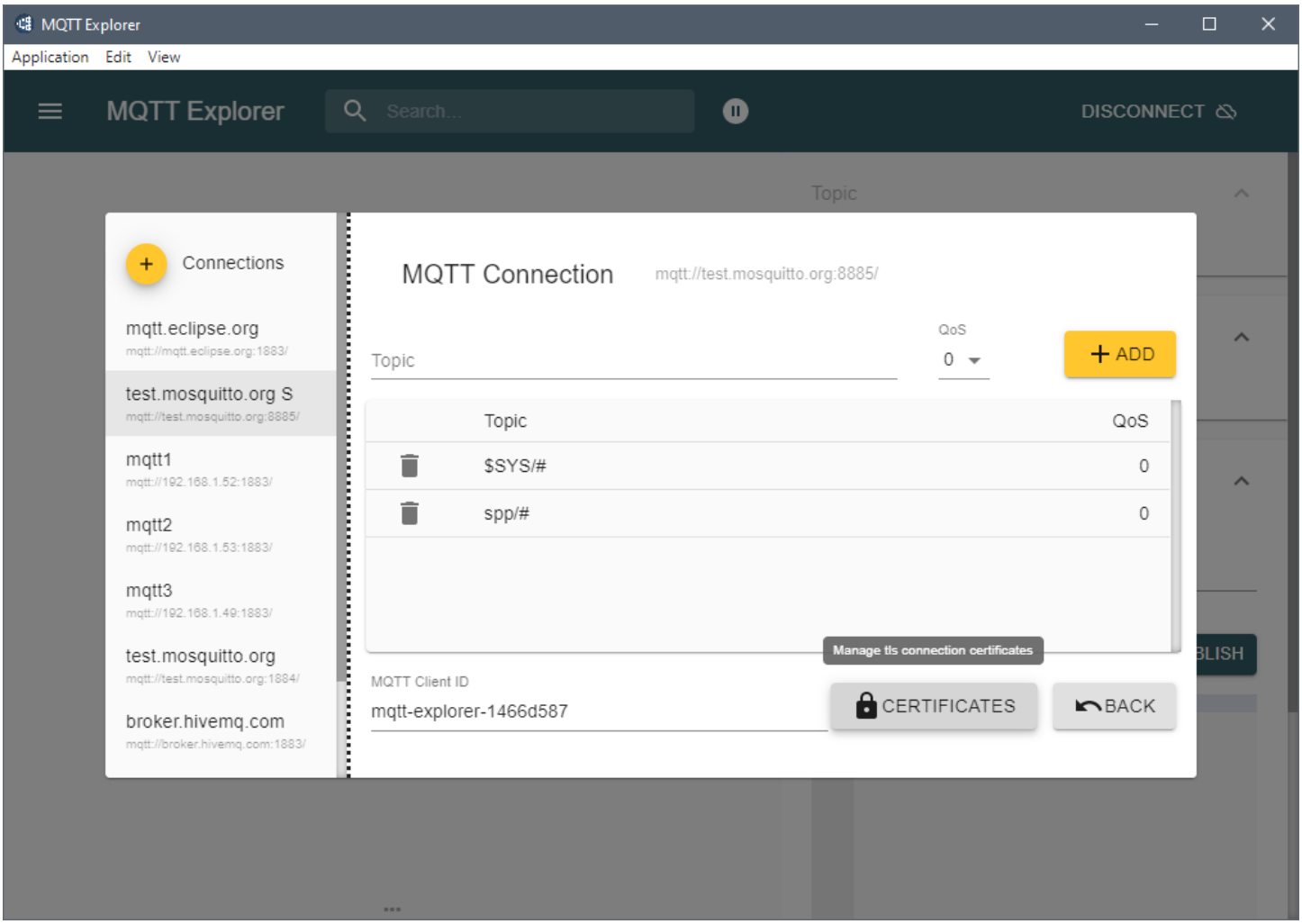
Next, start MQTT Explorer and create a new connection (click on the yellow + sign).



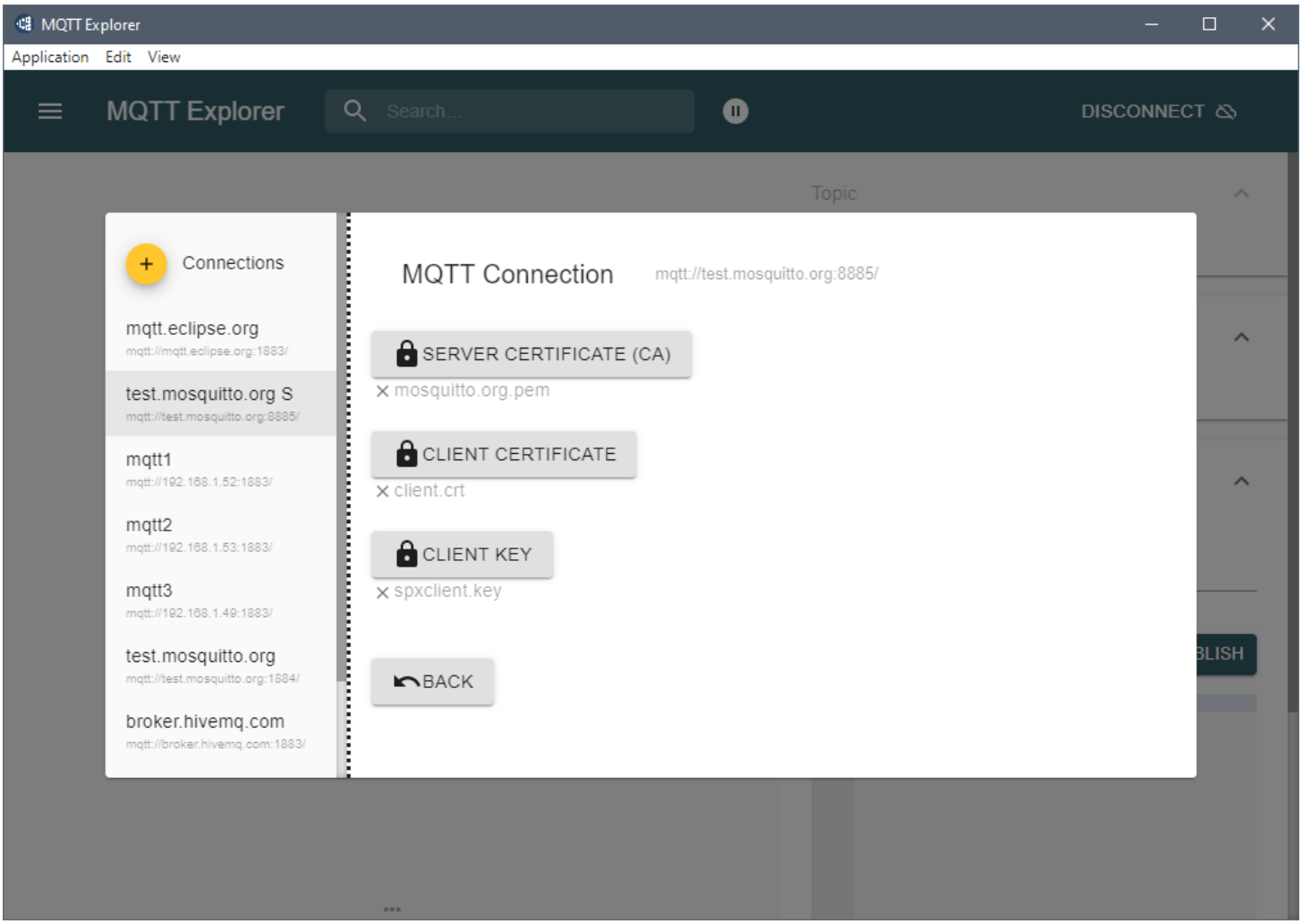
Specify the same connection settings as for the SPX+:

5. **Connection name**: it can be set to any value, here we use test.mosquitto.org S
6. Enable "**Validate certificate**" and "**Encryption (tls)**" settings
7. Fill out the **hostname to connect to**: test.mosquitto.org
8. Specify the username and password to connect: rw / readwrite

Next, click on "**Advanced**" button to set the certificate files for the connection.



Click on the **Certificates** button first.

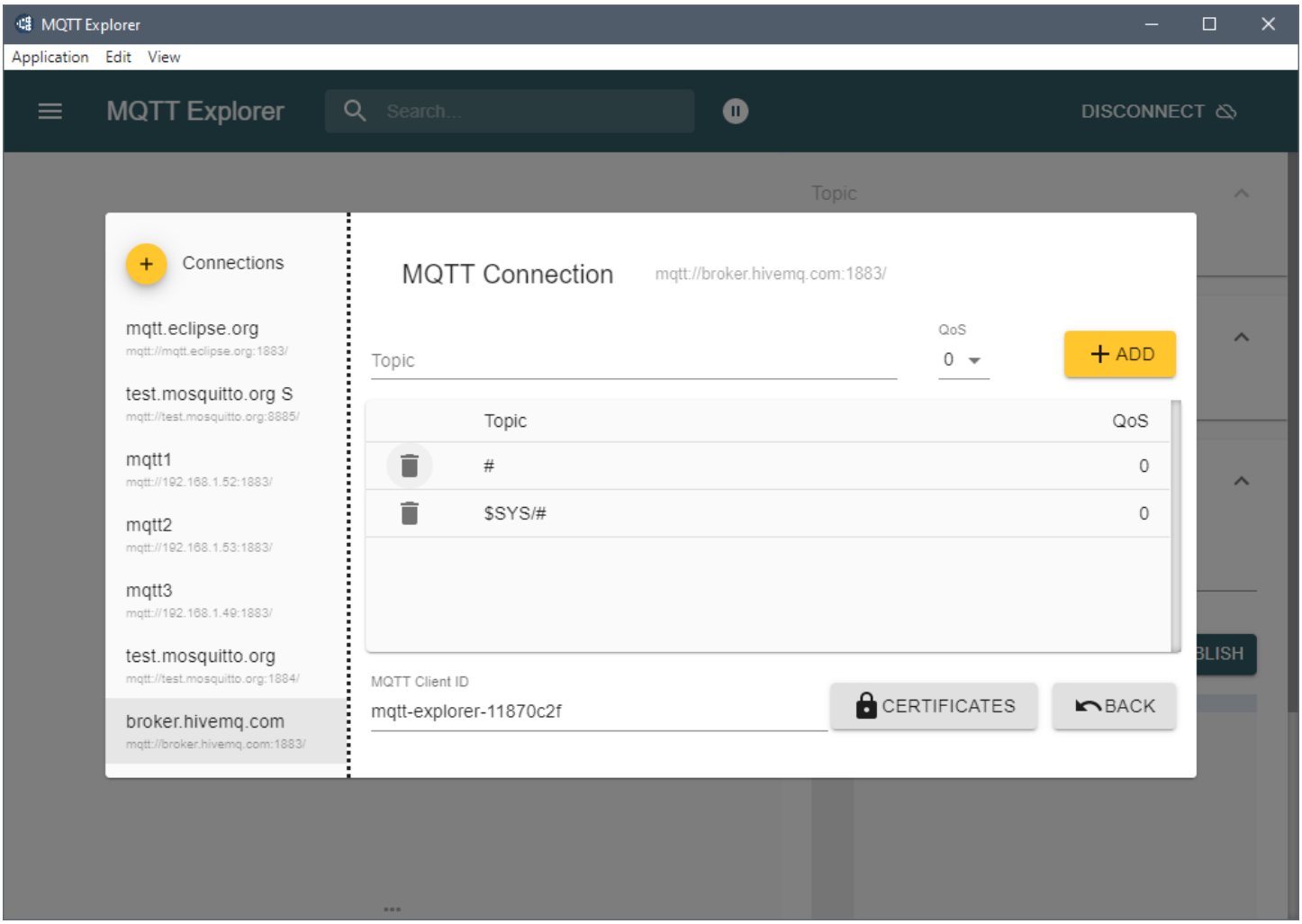


Choose the certificate files from your PC:

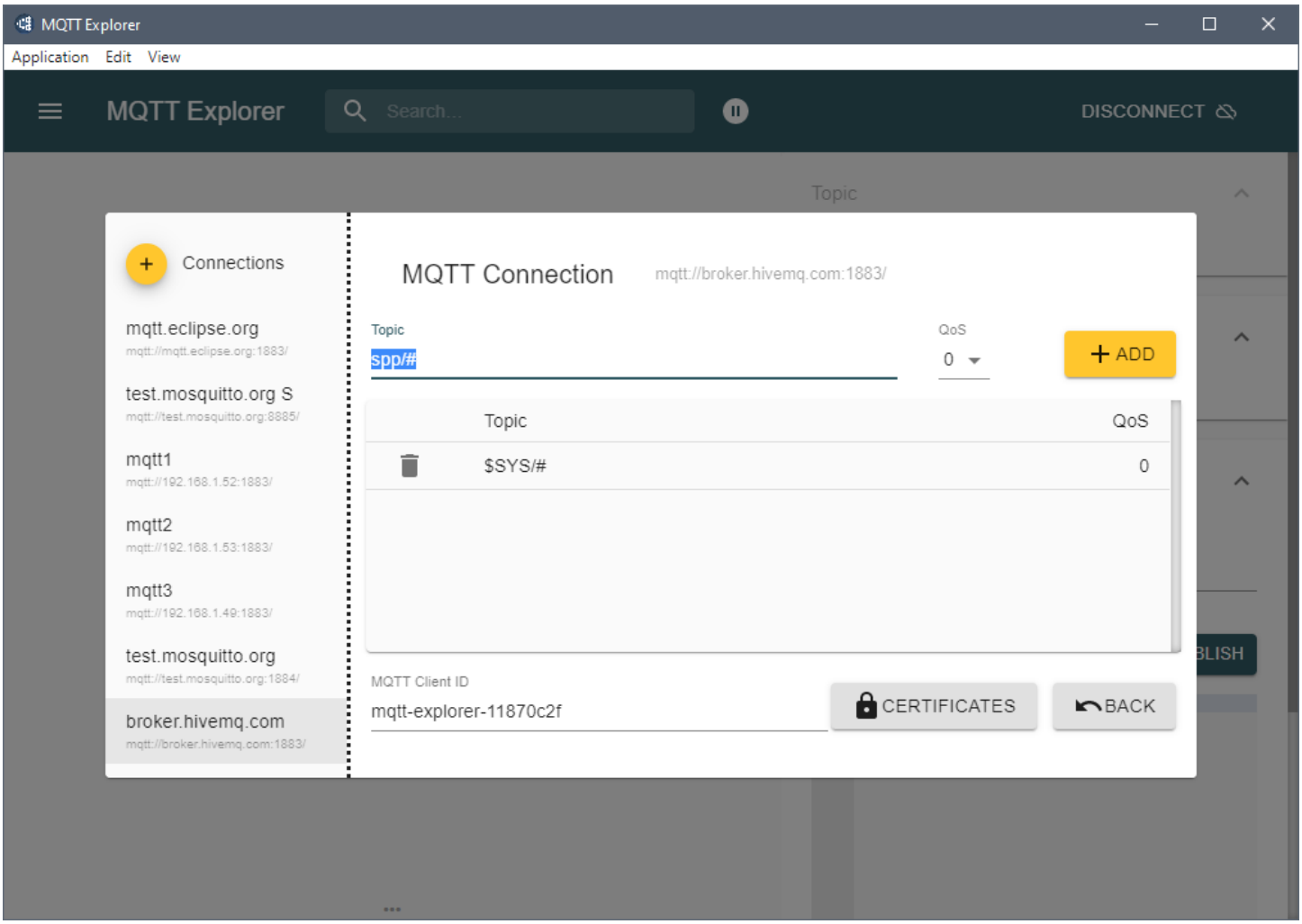
Server certificate (CA): **mosquitto.org.pem**
Client certificate: **client.crt**
Client key: **spxclient.key**

Then click on the **Back** button.

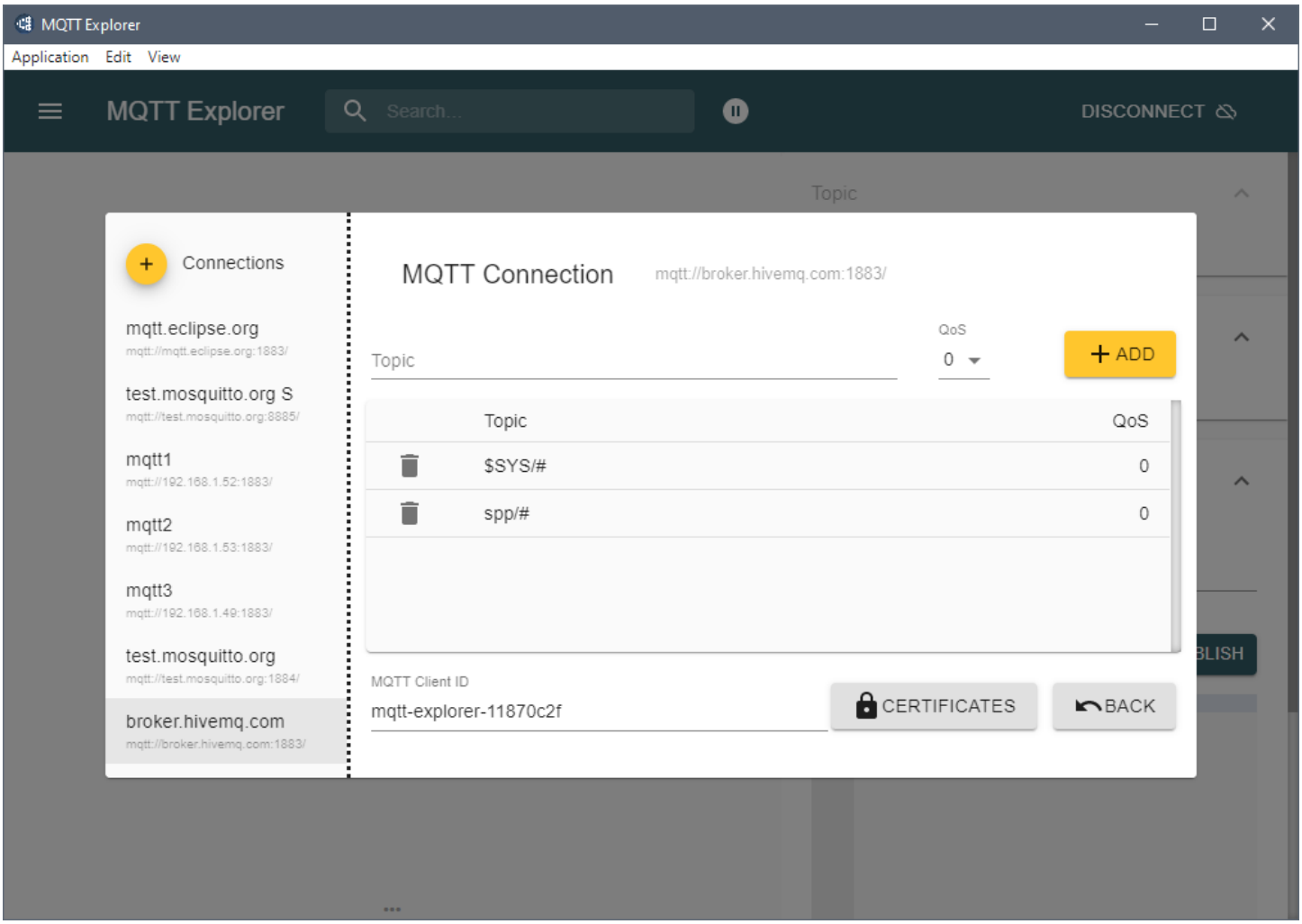
We will configure the MQTT topic subscription, since we don't want to subscribe to all topics that are available on this public broker.



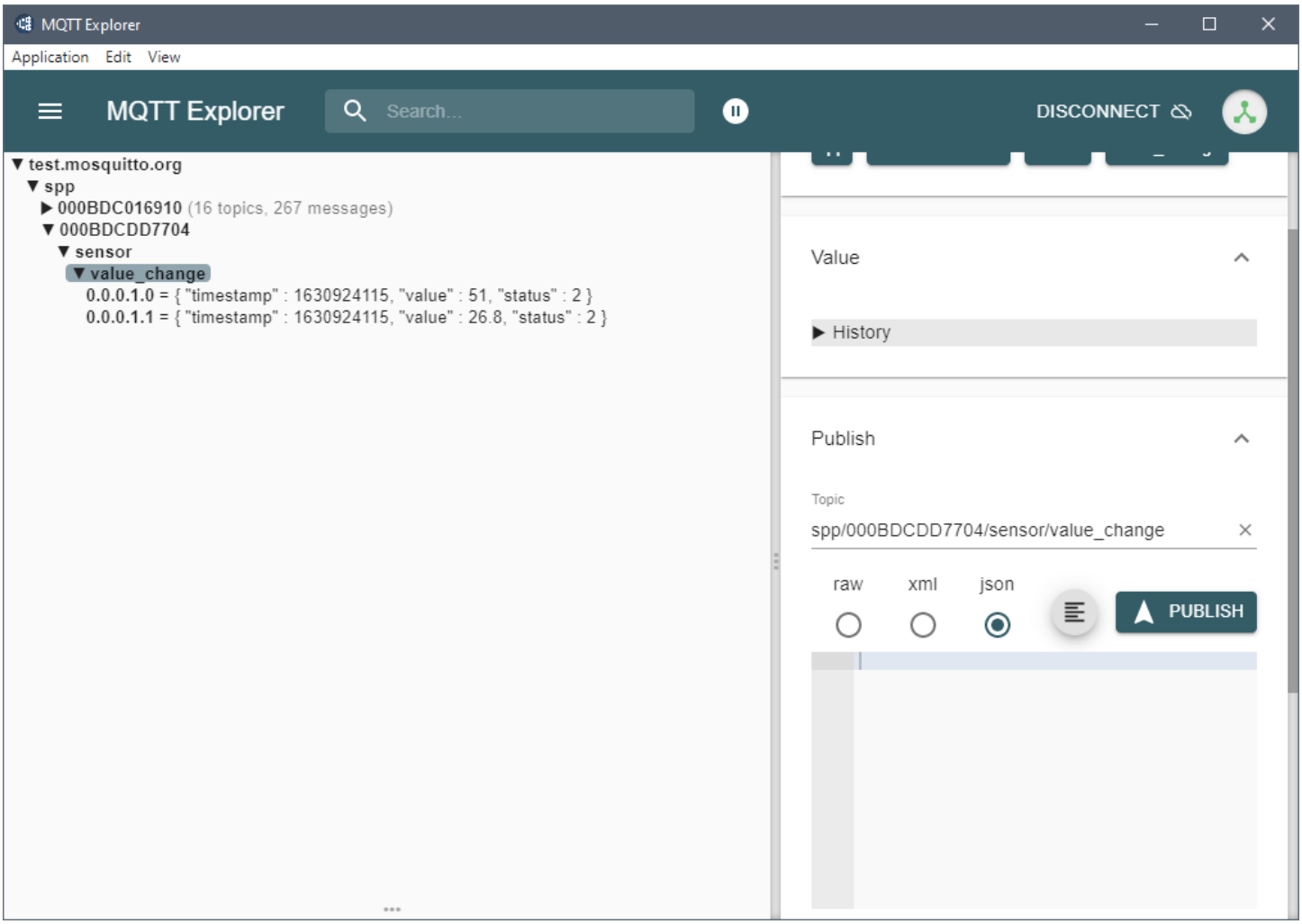
The default topic configuration is # which subscribes to all available topics on the server.
We don’t want to get values from other devices, therefore remove this setting with the delete icon.



Next, specify the topic that we are interested in.
This will be the **spp/#** topic and subtree, which will subscribe to any AKCP devices publishing to this MQTT server.
Type in spp/# and click "**Add**".

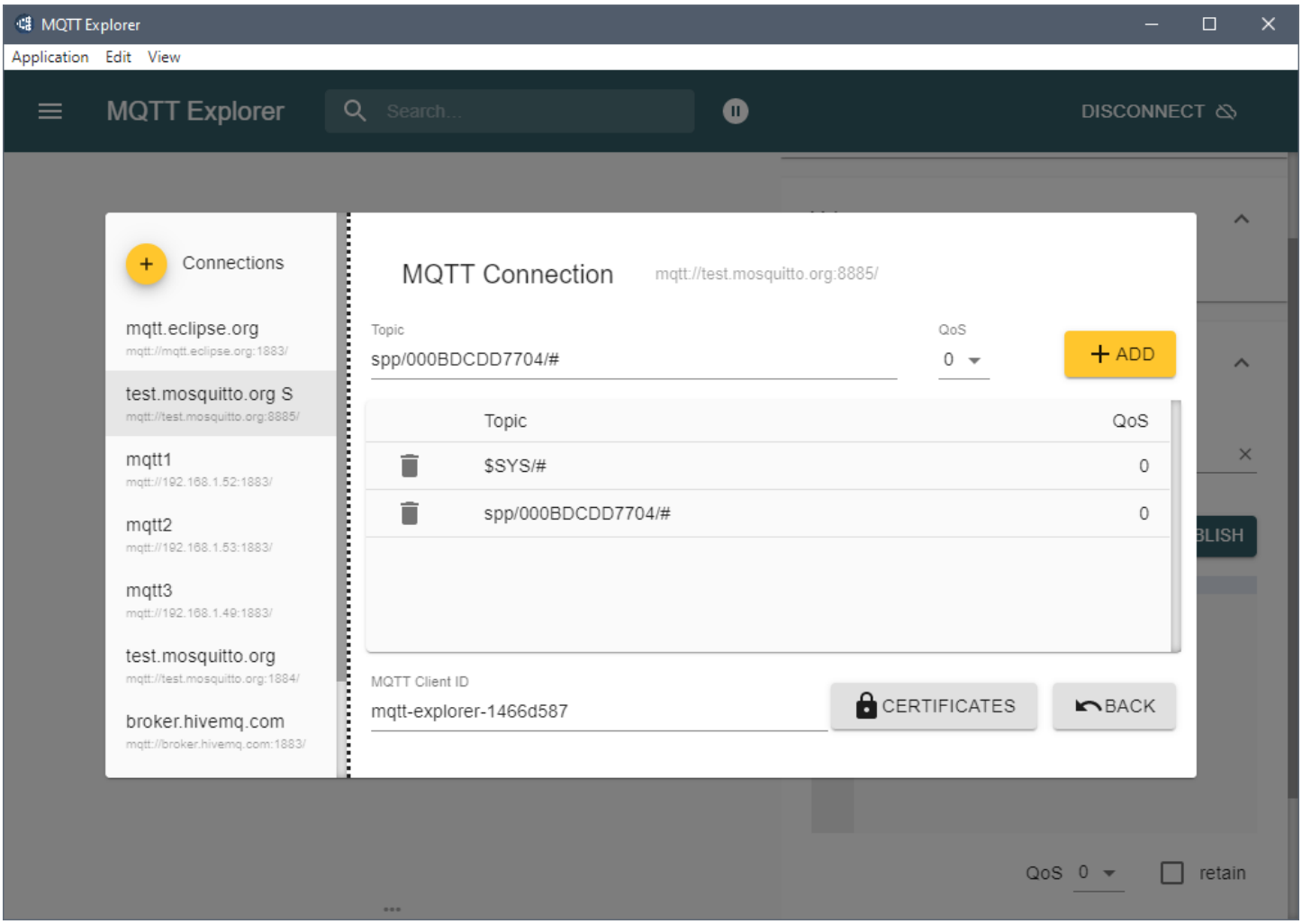


The topic subscription setting will be saved, and you can click “**Back**” button and connect to the MQTT server.



If all settings are correct, the connected SPX+ hostname and all its sensors will be displayed after 1-2 minutes.
In our example that is the 000BDCDD7704 host, but there is already another AKCP device visible in this topic.

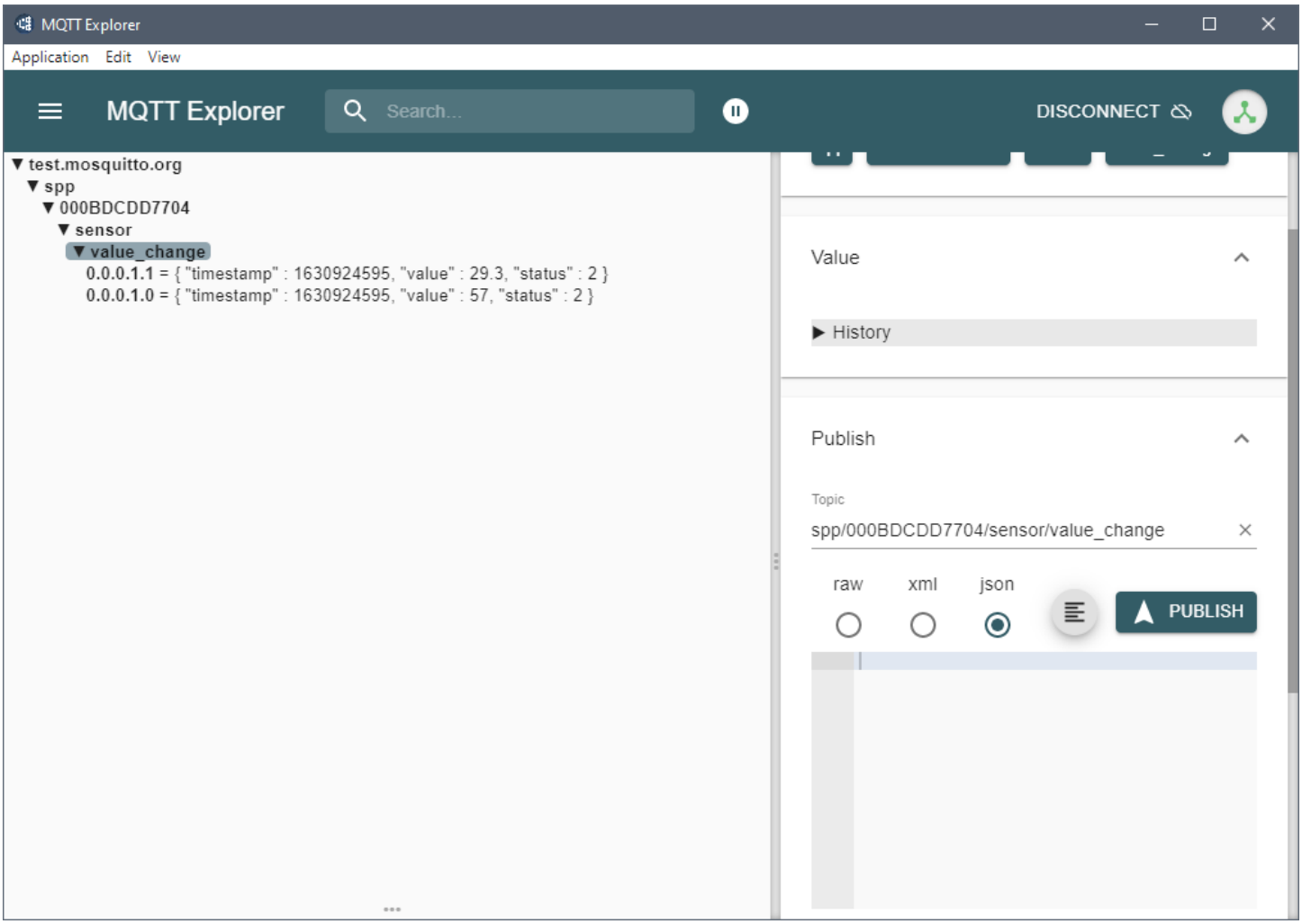
If you only want to subscribe to and display the values for a single host, disconnect the server and go back to the “Advanced” configuration to specify a different topic setting.



On this screenshot we reconfigured the topic to only display sensor values and statuses from our example host 000BDCDD7704 and ignore all others.

The topic setting is spp/000BDCDD7704/#

Please review the first part of the manual about using MQTT wildcards.



As shown on this example screenshot, with this topic setting, only our SPX+ device and its sensors are subscribed to.

As noted earlier, remember that the supported MQTT topic settings on WTG and SP+ units have the following format:

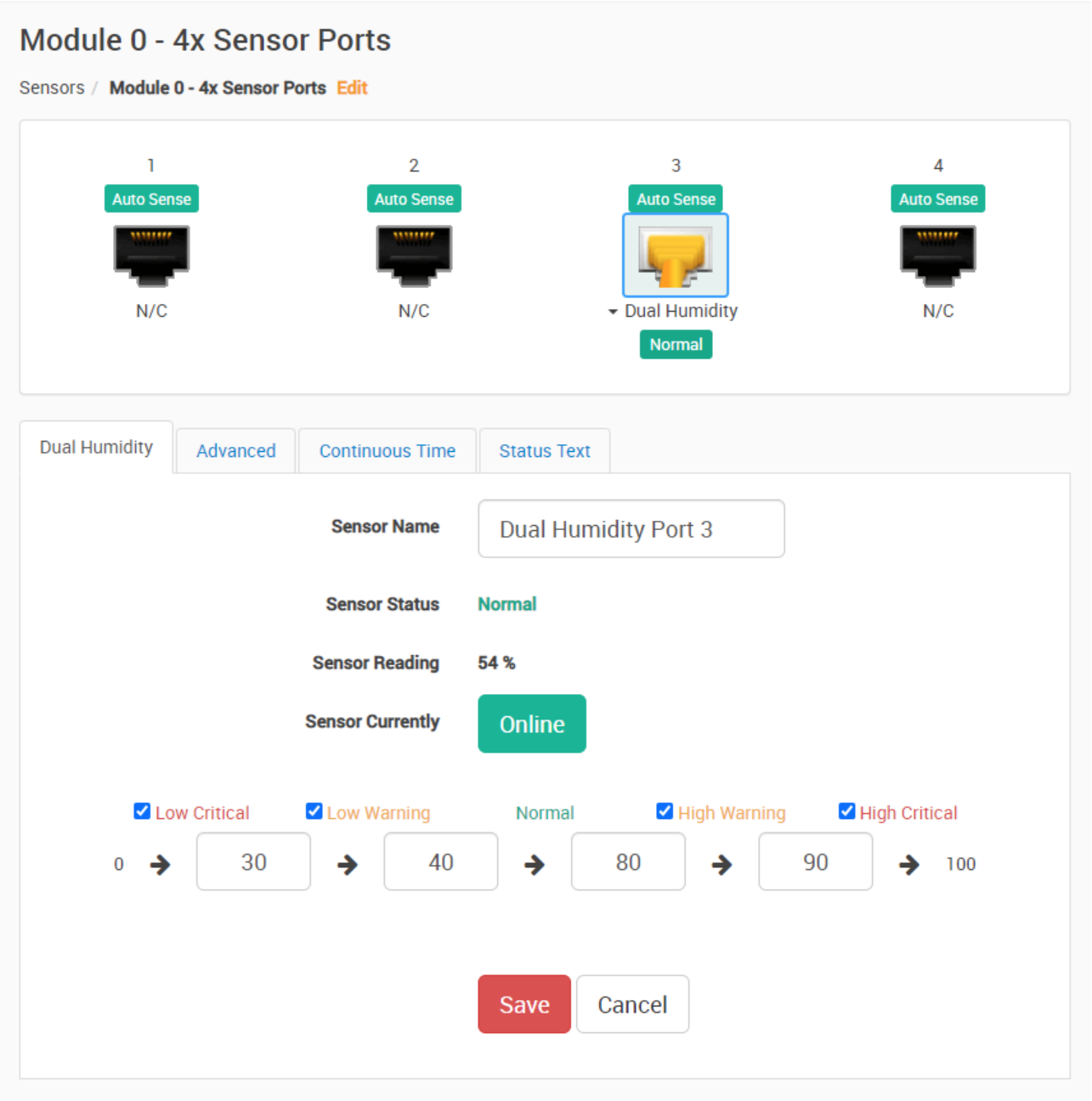
spp/${DeviceID}/sensor/status_change/${CompoundID}
spp/${DeviceID}/sensor/value_change/${CompoundID}

You can narrow down the topic subscription, based on your needs.

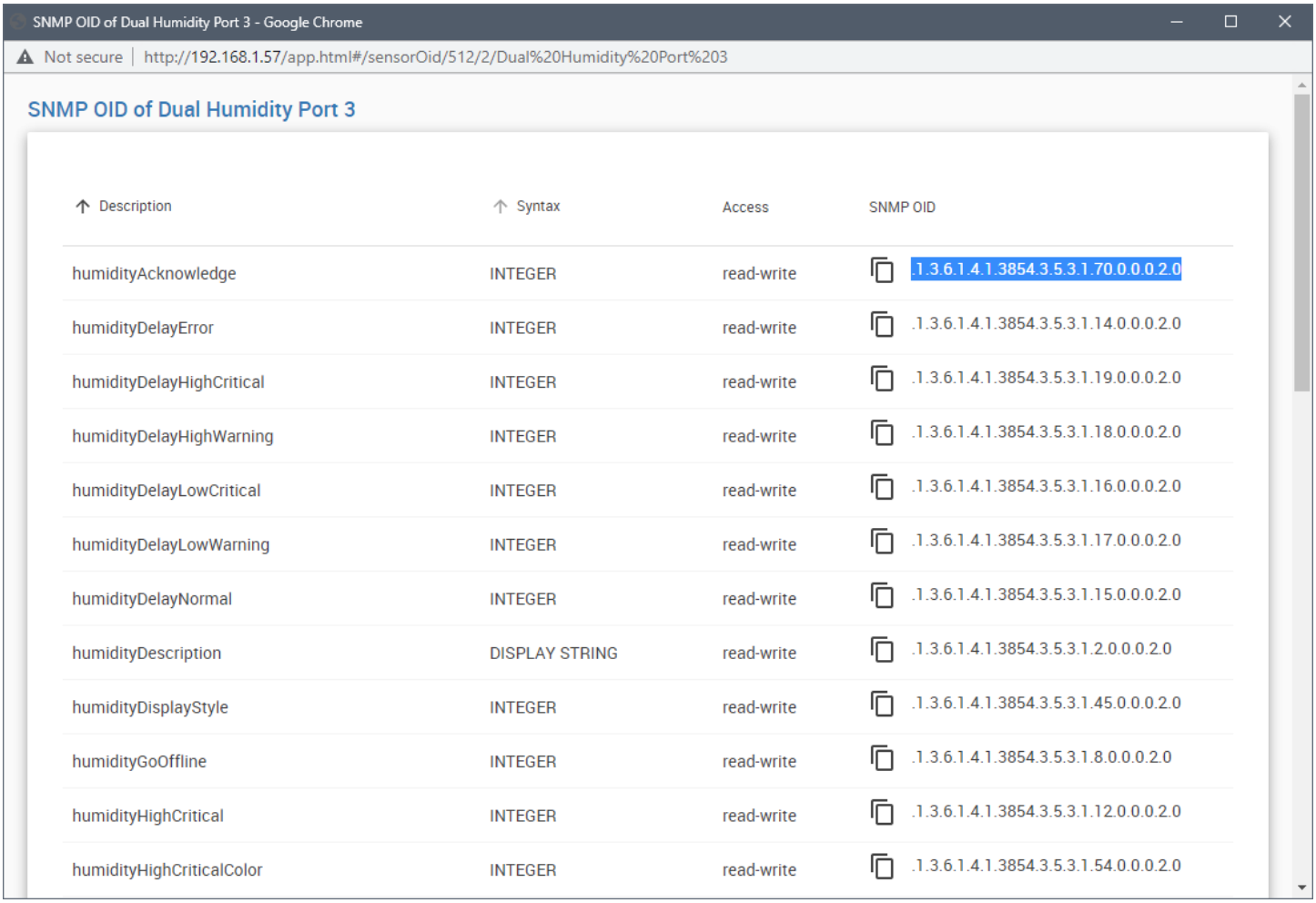
How to find sensor compound ID

The compound ID could be found from any SNMP OID of the given sensor.

For example: I have temp/humidity sensor connected to port 3 on SPX.
The active sensor is the Humidity sensor.

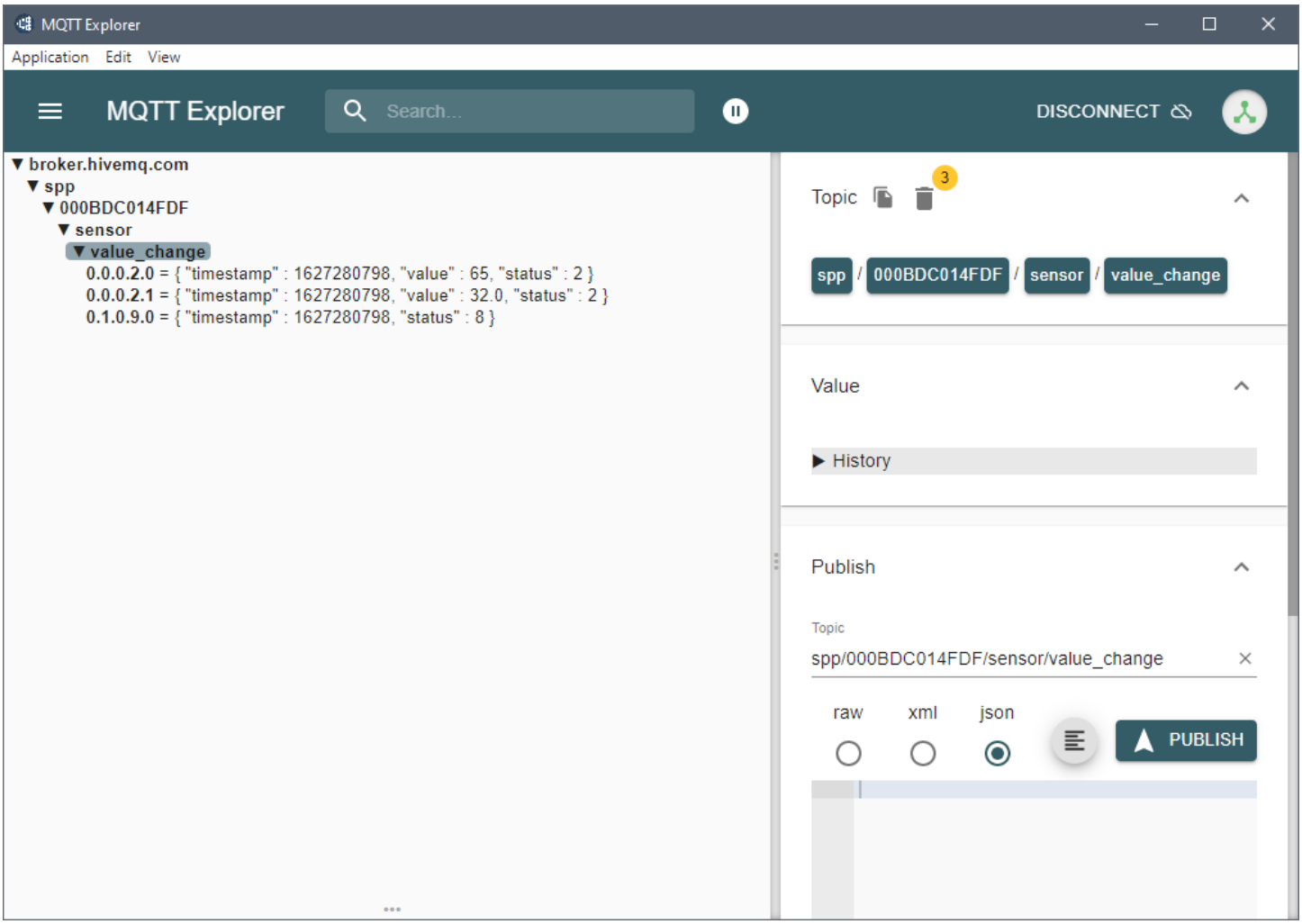


Then open “Get SNMP OID” window and choose any OID for the humidity sensor:



The last 5 digits of the OID is the compound ID, for example this:
.1.3.6.1.4.1.3854.3.5.3.1.70.**0.0.0.2.0**

Then associate the same compound ID when viewing MQTT data.
On the screenshot below, you can see “0.0.0.2.0” compound ID belonging to the humidity sensor:


MQTT Explorer
Application Edit View
MQTT Explorer
Search...
DISCONNECT
broker.hivemq.com
spp
000BDC014FDF
sensor
value_change
0.0.0.2.0 = { "timestamp" : 1627280798, "value" : 65, "status" : 2 }
0.0.0.2.1 = { "timestamp" : 1627280798, "value" : 32.0, "status" : 2 }
0.1.0.9.0 = { "timestamp" : 1627280798, "status" : 8 }
Topic
3
spp / 000BDC014FDF / sensor / value_change
Value
History
Publish
Topic
spp/000BDC014FDF/sensor/value_change
raw
xml
json
PUBLISH

Detailed explanation of compound ID structure

Compound ID consists of:

Expansion port . Board position . PCard position . Sensor port . Sensor sub-port

0: Main board	0: Main board	PCard	Starting	Starting
1: Exp chain	1: Internal board	I2C address	from 0	from 0
2: Exp I2C BEB	3: Virtual board			
3: WTSes	4: *Software board			
	6: Modbus board			

* = not used

Examples of Compound IDs:

Dry Contact on Main RJ45 Sensor port 1:	0.0.0.0.0
Temperature sensor on Main RJ45 Sensor port 3 subport 2:	0.0.0.2.1
Virtual sensor port 6:	0.3.0.5.0
Temperature sensor on port 3 of Sensor4:	0.0.1.2.0
Relay port 2 of Relay PCard, module 1.2 on BEB:	2.1.2.1.0
Temperature sensor on first WT-TH on WTG:	3.0.0.0.0

Please contact support@akcp.com if you have any further technical questions or problems.

Thanks for Choosing AKCP!