

How to connect an MQTT client to publish and subscribe data

by David Schmeding

Prerequisites

This lesson assumes that you want to integrate the **Modbus/TCP** protocol with the Cybus Connectware. To understand the basic concepts of the Connectware, please check out the [Technical Overview](#) lesson. To follow along with the example, you will also need a running instance of the Connectware. If you don't have that, learn [How to install the Connectware](#). Although we focus on Modbus/TCP here, we will ultimately access all data via MQTT. So you should also be familiar with MQTT. If in doubt, head over to our [MQTT Basics](#) lesson.

Introduction

This article will teach you the integration of MQTT clients. In more detail, the following topics are covered:

- Creating client credentials and permissions
- Using the client registration feature
- Establishing connection with a client
- Publishing data
- Subscribing data

Selecting the tools

In this lesson we will utilize a couple of MQTT clients, each fulfilling a different purpose and therefore all having their right to exist.

Mosquitto

Mosquitto is an open source message broker which also comes with some handy client utilities. Once installed on your system it provides the `mosquitto_pub` and `mosquitto_sub` command line MQTT clients which you can use to perform testing or troubleshooting, carried out manually or scripted.

MQTT.fx

The client MQTT.fx is based on JavaFX and provides a graphical user interface to communicate with MQTT brokers. It offers convenient ways of configuring and establishing connections, creating subscriptions or publishing on topics. The program presents all data clearly and in well organized sections but also enables the user to select which data should appear on the monitor, should be hid or dumped in a file. Additionally it provides tools to decode the payload of messages, execute scripts or track the broker status.

Workbench

The Workbench is part of the Cybus Connectware. It is a flow-based, visual programming tool running a Node-RED instance to create data flows and data pre-processing on level of the Connectware. One way to access data on the Connectware utilizing the Workbench is via MQTT. The Workbench provides publish and subscribe nodes which serve as data sources or sinks in the modeled data flow and allows users to build applications for prototyping or debugging. But it is capable of a lot of more useful functions and nodes, for instance creating dashboards.

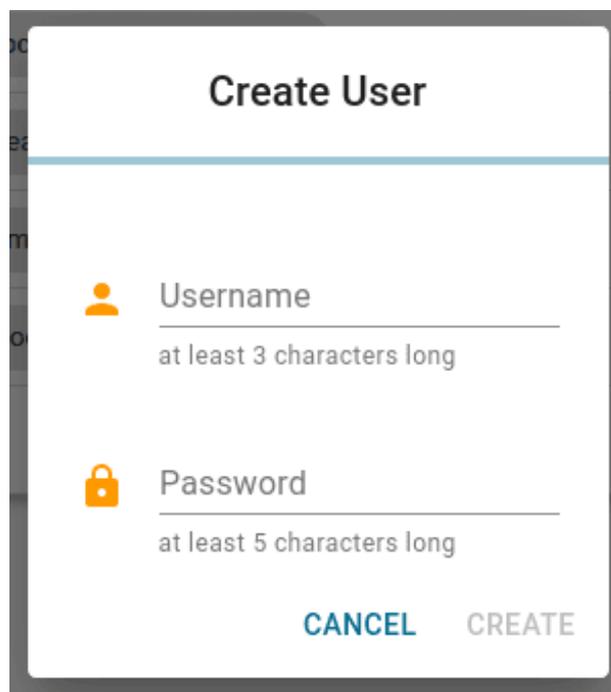
Configuring the Connectware

In order to connect a client to the Connectware it will need credentials for authorization. Those credentials must be created using the Admin UI of the Connectware. There are two ways of creating credentials which we will consider in the following.

Creating client credentials and permissions

Access the Admin UI of your Connectware and navigate to the section *User Management > Users*.

Click the (+) button in the upper right corner to add a user. We will create a separate user for every client we want to connect with.



The image shows a 'Create User' dialog box. It has a title bar with the text 'Create User'. Below the title bar, there are two input fields. The first is labeled 'Username' with a user icon and a note 'at least 3 characters long'. The second is labeled 'Password' with a lock icon and a note 'at least 5 characters long'. At the bottom of the dialog, there are two buttons: 'CANCEL' and 'CREATE'.

You are free in the choice of names and passwords. We created the following users for this example. The user for our Mosquitto client will be created in the following section making use of the second method.

User Management				admin	?	↗
Username	Roles	Grant Types	Identity Provider			
admin	connectware-admin	password token	local			
MQTT.fx		password token	local			
Workbench		password token	local			

Rows per page: 15 1-3 of 3 < >

Before a user can access an MQTT topic he needs to be granted the permissions for `read`, `write` or `readWrite` operations which correspond to subscribe, publish or subscribe and publish. After adding permissions click the Save-button in the lower right corner.

Add Permission

Resource
🔄 clients/#

⚙️ readWrite

CANCEL ADD

We grant all our clients access to the `clients` topic and specifying the wildcard `#` also every topic that is hierarchically under `clients`, for example the hypothetical topic `clients/status/active`.

Using the client registration feature

The second way of creating credentials for a user is the client registry process (also referred to as *self registration*). There are two variants of this process: The implicit form works with any MQTT client while the explicit form is for clients that use the REST API. In this lesson we will only look at the implicit variant. If you want to learn more on *self registration* and both of its forms, please consult the [Reference docs](#).

We will now step through the process of implicit client registration. At first we need to activate the self registration: In the Connectware Admin UI navigate to *User Management > Client Registry* and click the lock symbol in the upper right corner. This will temporarily unlock the self registration for clients. The next step is that the client tries to connect to the Connectware with a username that does not yet exist.

We utilize the `mosquitto_sub` client for this which is especially easy because we just need to use the command `mosquitto_sub -h localhost -p 1883 -u Mosquitto -P 123456 -t clients` assuming we are running the Connectware on our local machine and want to register the User `Mosquitto` with the password `123456` (you should of course choose a safe password!). The topic we chose using option `-t` is not relevant, this option is just required for issuing the command. We will explain this command in detail in the section about *Subscribing data*.

Shortly after we have issued this command and it exited with the message *"Connection error: Connection Refused: not authorized."* we can look at the Client Registry in the Admin UI and see the connection attempt of our Mosquitto client.

The screenshot shows the 'User Management' interface. At the top, there is a navigation bar with a hamburger menu, the text 'User Management', and the user 'admin' with a help icon and an external link icon. Below this is a table with the following columns: Username, Context, Timestamp, and Credential Type. There is one row in the table with the following data: Username: Mosquitto, Context: Auto registration via MQTT, client id: mosq-Dtzb86Mwrlpr0X9obE, Timestamp: 2020-07-22T09:42:14.971Z, and Credential Type: password. At the bottom right of the table, there is a pagination control showing 'Rows per page: 15' and '1-1 of 1'.

Username	Context	Timestamp	Credential Type
Mosquitto	Auto registration via MQTT, client id: mosq-Dtzb86Mwrlpr0X9obE	2020-07-22T09:42:14.971Z	password

Clicking on the list's entry will open the *Grant Access* dialogue. This summarizes the details of the request and the requested permissions (permissions can only be requested using the explicit method). To grant this client access and create a user click *Allow* in the lower right corner.

Grant Access

General Details

Username
Mosquitto

Context
Auto registration via MQTT, client id: mosq-Dtzb86MwrLpr0X9obE

Credential Type
password

Requested API Permissions

No API permissions requested

Requested Data Permissions

No Data permissions requested

CANCEL **ALLOW**

Now we take another look at *User Management > Users* and there we see the newly created user "Mosquitto". Since we are using the implicit method we now have to manually add permissions on the accessible topics like we did before.

Establishing connection with a client

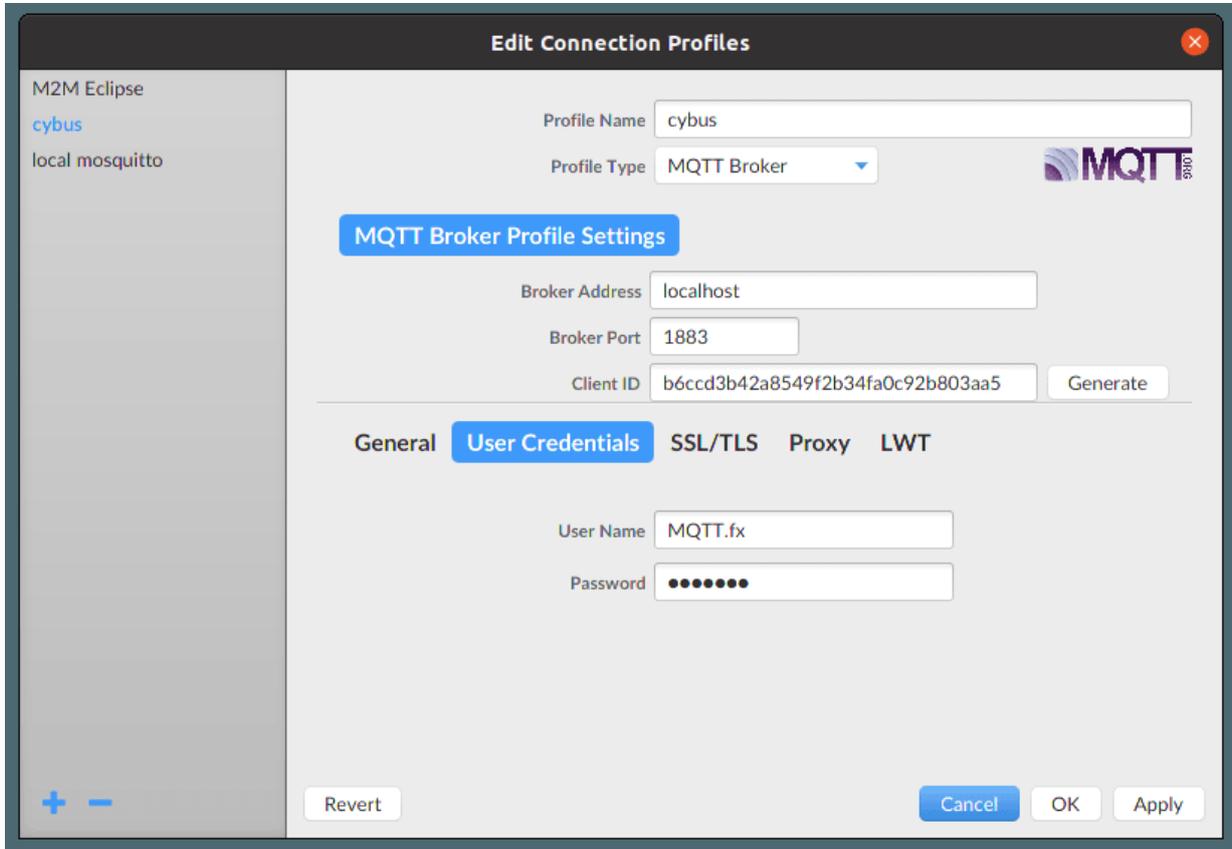
We are now ready to connect our clients to the Connectware via MQTT.

Mosquitto

We already made use of the client functions of Mosquitto by utilizing it for our self registration example. That also suggested that we do not explicitly need to establish a connection before subscribing or publishing, this process is included in the `mosquitto_sub` and `mosquitto_pub` commands and we supply the credentials while issuing.

MQTT.fx

Using MQTT.fx we need to configure the connection before connecting. After starting the program click the gear symbol next to the Connect-button. In the appearing window we can edit our connection profiles.

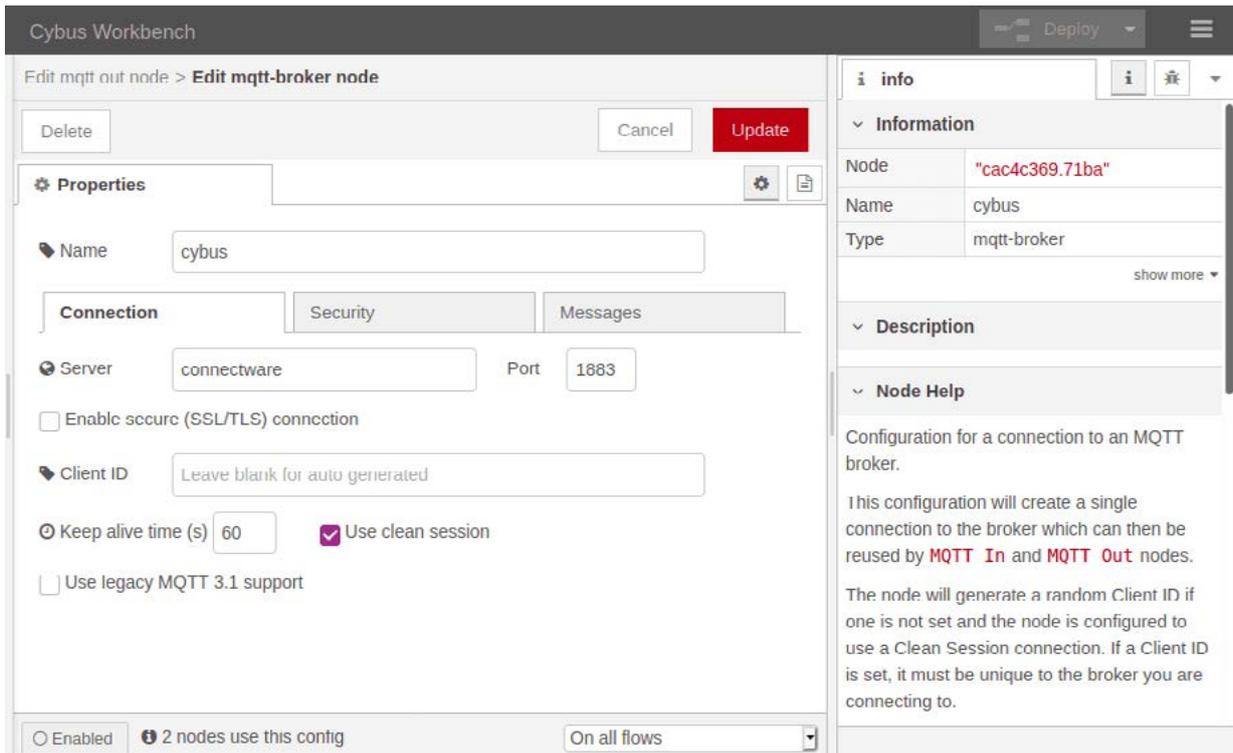


First we create a new profile by clicking the plus (+) in the lower left corner. We name the new profile "cybus" and define the broker address as "localhost" (exchange this with the address you are running your Connectware on). Then we generate a unique client ID by clicking the Generate-button and finally fill out the user credentials. All other settings can remain default for this example.

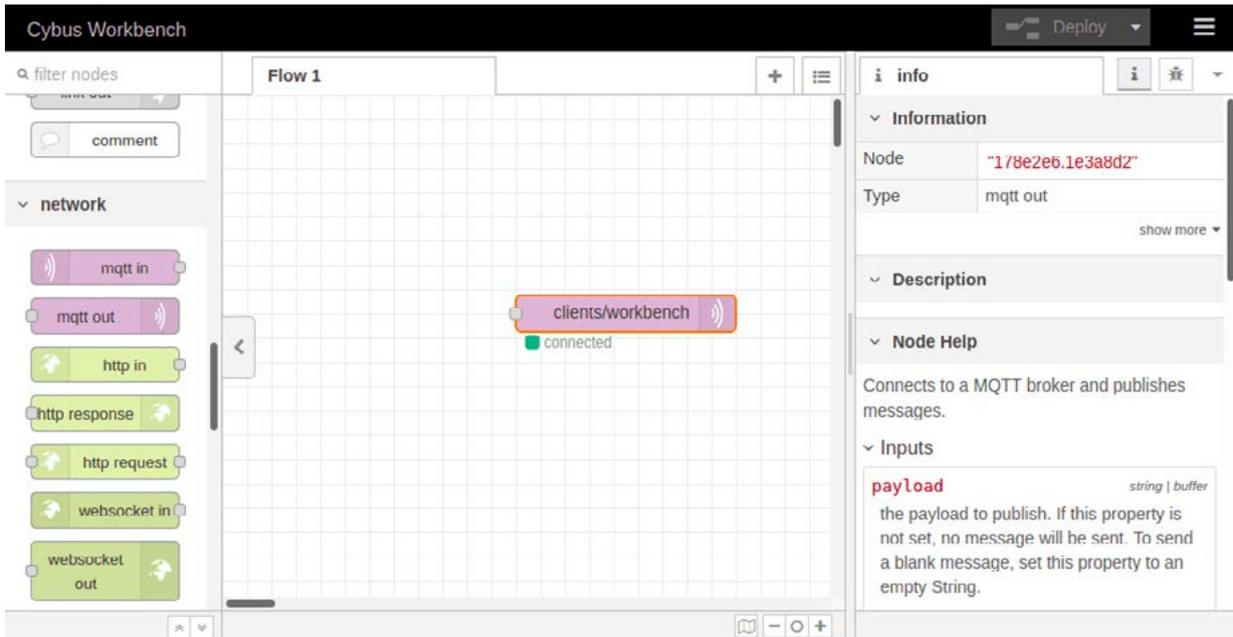
Workbench

The Workbench can be accessed through the Admin UI by clicking *Workbench* in the navigation bar. A new tab will open showing the most recent edited flow of our Workbench. If it is empty you can use this otherwise click the (+) button to the upper right of the actual shown flow. If you are new to the Workbench, we won't go into details about the functions and concepts in this lesson but it will demonstrate the most simple way of monitoring data with it.

Take a look at the left bar: There you see the inventory of available nodes. Scrolling down you will find the section *network* containing two MQTT nodes among others. The so-called *mqtt in* and *mqtt out* nodes represent the subscribe and publish operations. Drag the *mqtt out* node and drop it on your flow, double-clicking it will show you its properties. The drop-down-menu of the Server property will allow you to add a new mqtt broker by clicking the pencil symbol next to it.



We name this new connection "cybus" and define the server address as `connectware`. This is the way to address the local running Connectware, you won't be able to access it using `localhost`! Now switch to the Security tab and fill in username and password. Save your changes by clicking Add in the upper right corner. Make sure the just created configuration is active for the Server property. For the property Topic we choose the topic we selected for our clients: `clients/workbench`. The other properties will keep their default settings. Click Done in the upper right corner.



To apply the changes we just made click the button *Deploy* in the upper right corner. This will reset our flow and start it with the latest settings. You will now see that the MQTT node is displayed as "connected", assuming everything was configured correctly. We successfully established a connection between the Workbench and our Connectware.

Publishing data

Publishing data allows us to make it available. It does not necessarily mean that someone is actually receiving it but anyone with the permission to subscribe on the concerning topic could.

Mosquitto

Using Mosquitto publishing data can be achieved by a single command line utilizing `mosquitto_pub`. For this example we want to connect to the Connectware using the following options:

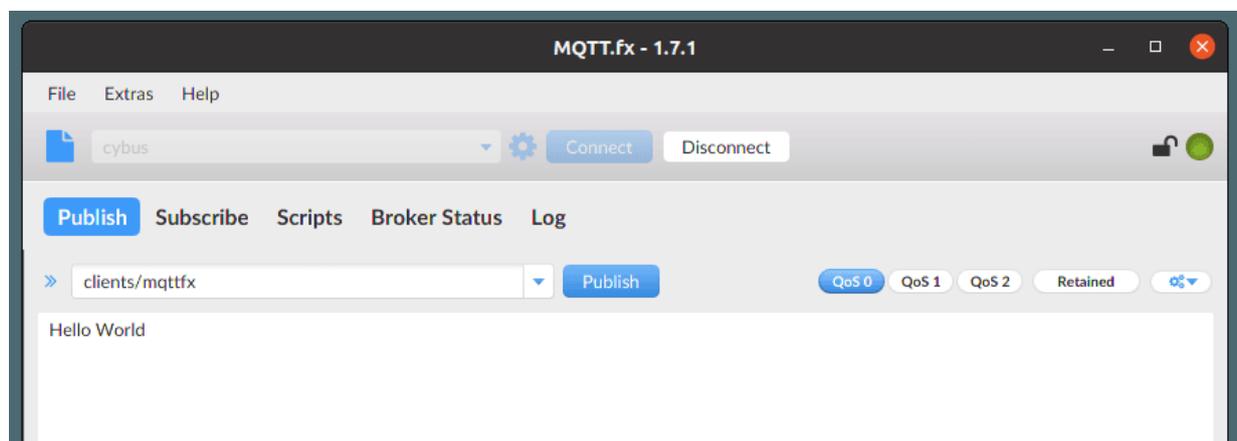
- Host to connect to: `-h localhost`
- Port to connect to: `-p 1883`
- Username: `-u Mosquitto`
- Password: `-P 123456`
- Topic to publish on: `-t clients/mosquitto`
- Message: `-m "Hello World"`

```
mosquitto_pub -h localhost -p 1883 -u Mosquitto -P 123456 -t clients/mosquitto -m "Hello World"
```

If successful the command will complete without any feedback message. We could not confirm yet if our message arrived where we expected it. But we will validate this later when coming to subscribing data.

MQTT.fx

Publishing with MQTT.fx is even simpler once the connection is configured and a connection is established (indicated by the gray/green circle in the upper right corner). If the indicator shows green, we are connected and ready to shoot some messages.



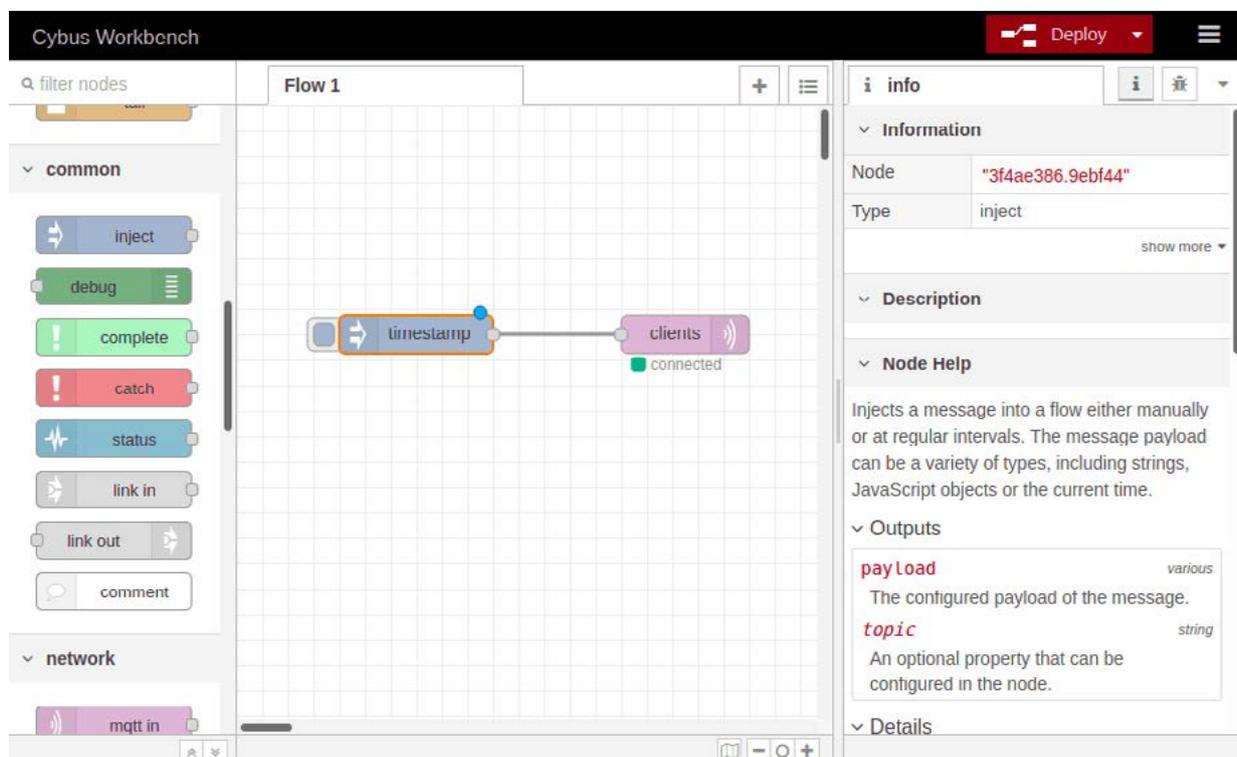
Switch to the *Publish* section. In the input line next to the *Publish*-button you define the topic you want to publish to. We will go for the topic `clients/mqttfx`. Then you click the big, blank box below and type a message, e.g. "Hello World". To publish this message click *Publish*.

Again we have no feedback if our message arrived but we will take care of this in the Subscribing data section.

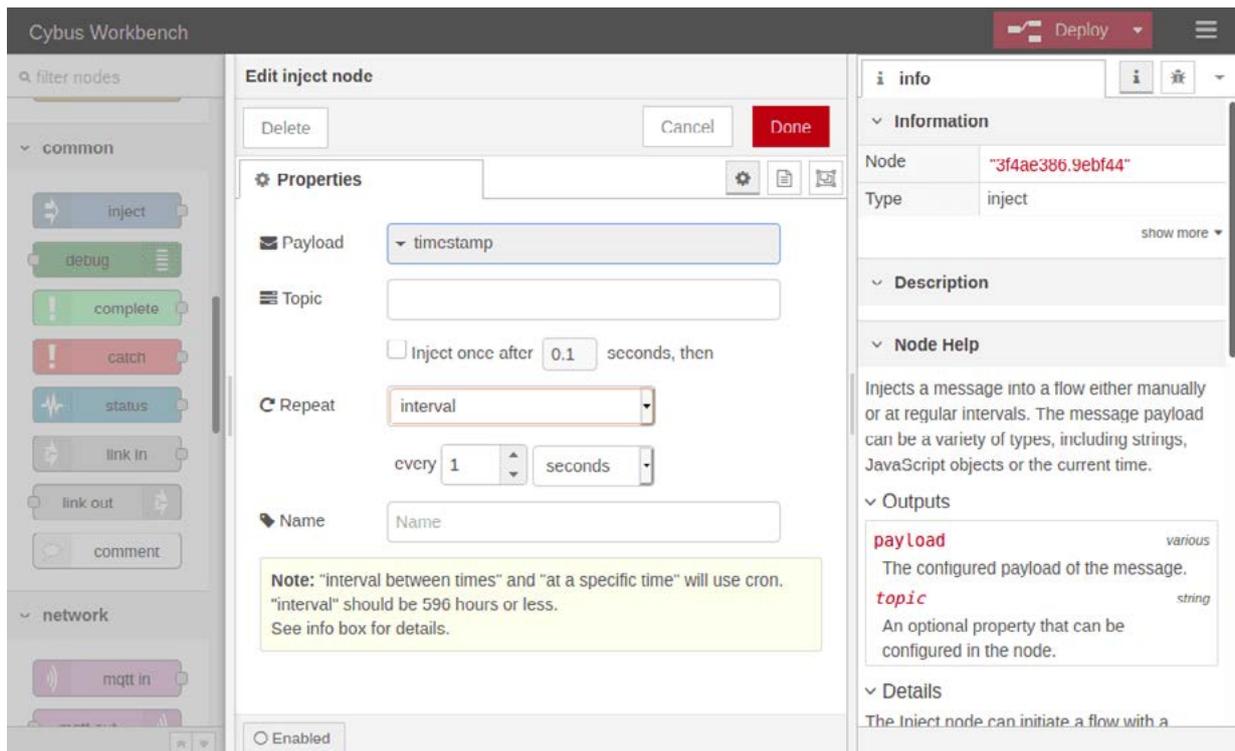
Configure the Workbench as data source

We already added a publishing node to our flow when we established a connection to the Connectware and configured it to publish on `clients/workbench`. Having this set the Workbench is ready to publish but the messages are still missing.

At this point we want to create a data source in our flow which periodically publishes data through the *mqtt out* node. We simply drag an *inject* node from the nodes bar into our flow and drop it left to the *mqtt out* node. Now we can draw a connection from the small gray socket of one to the other node.



Double-clicking the *inject* node, which is now labeled "timestamp", shows its properties. We can define the payload, but we will keep "timestamp", declare a topic to publish to, which we leave blank because we defined the topic in our MQTT node, and we can set if and how often repeated the message should be injected in our flow. If we do not define repeating a message will be injected only if we click the button on the left of the *inject* node in our flow.



We set repeating to an "interval" of every second and confirm the settings by clicking *Done* in the upper right corner. Now we have to *Deploy* again to apply our changes. We still have no confirmation that everything works out but we are confident enough to go on and finally subscribe to the data we are generating.

Subscribing data

Subscribing data means that the client will be provided with new data by the broker as soon as it is available. Still we do not have a direct connection to the publisher, the broker is managing the data flow.

Mosquitto

We already used `mosquitto_sub` to self register our client as a user. We could have also used `mosquitto_pub` for this but now we want to make use of the original purpose of `mosquitto_sub`: Subscribing to an MQTT broker. Not any broker but the broker of our Connectware.

We are using the following options:

- Host to connect to: `-h localhost`
- Port to connect to: `-p 1883`
- Username: `-u Mosquitto`
- Password: `-P 123456`
- Topic: `-t clients/workbench`

```
mosquitto_sub -h localhost -p 1883 -u Mosquitto -P 123456 -t clients/workbench
```

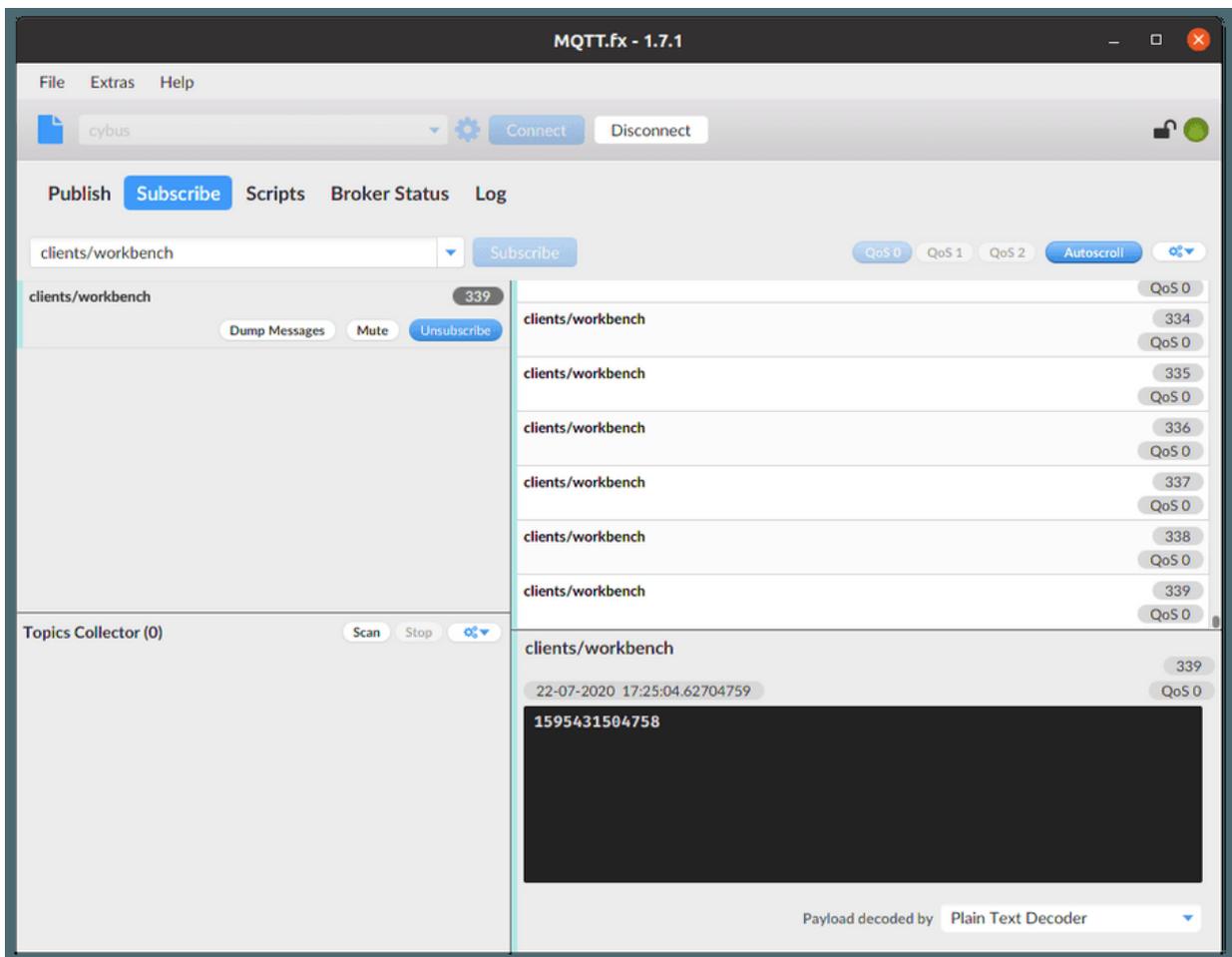
Again make sure that you have specified the address of your Connectware as host which does not necessarily have to be your local machine! We will subscribe to the topic `clients/workbench` where we are publishing the messages from our Workbench.

```
~$ mosquitto_sub -h localhost -p 1883 -u Mosquitto -P 123456 -t clients/workbench
1595429847062
1595429848062
1595429849064
1595429850067
1595429851069
1595429852070
1595429853070
1595429854070
1595429855072
1595429856073
```

And Tada! The mosquitto client now shows us every new message published on this topic – in this case the timestamps generated by our Workbench.

MQTT.fx

Already having the connection to our Connectware established it is a piece of cake to subscribe to a topic. Switch to the section *Subscribe*, type the topic to subscribe to in the input line (again `clients/workbench`), and click the *Subscribe*-button.

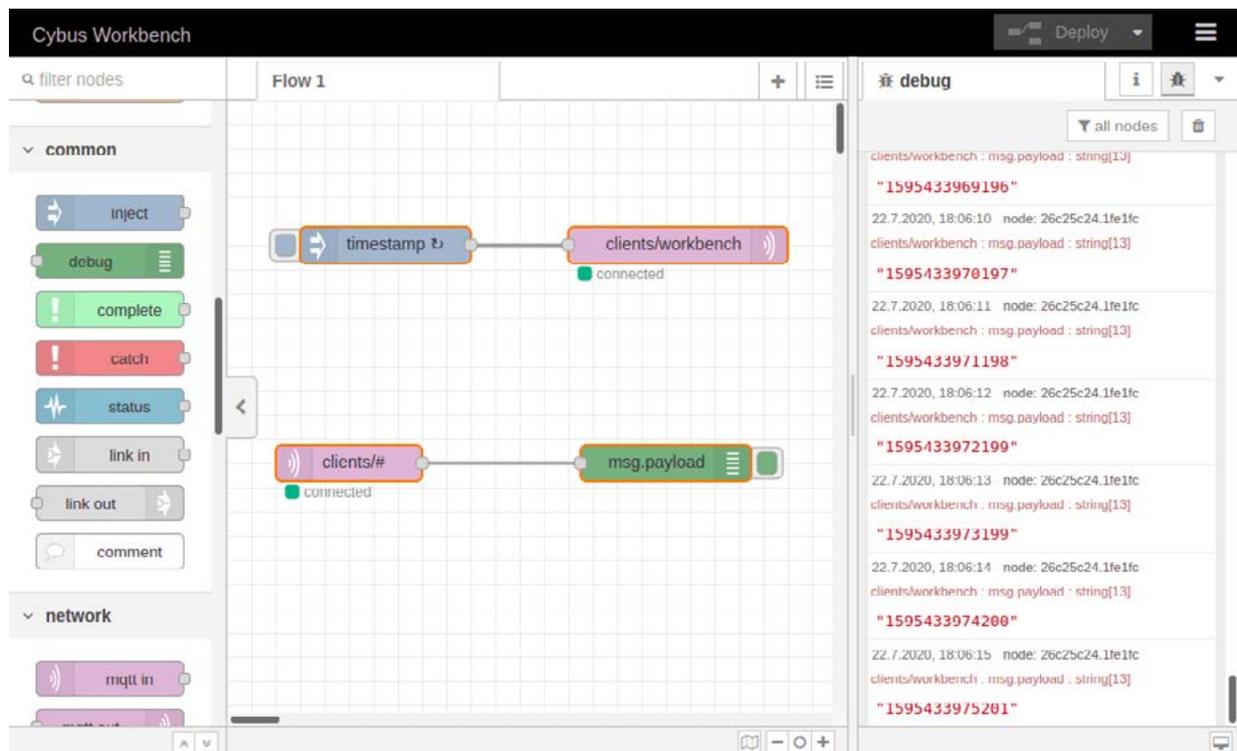


And there you see finely presented all the data flying in. You can also add more topics to be monitored simultaneously and manage which should be presented to you or be dumped in a file.

Workbench

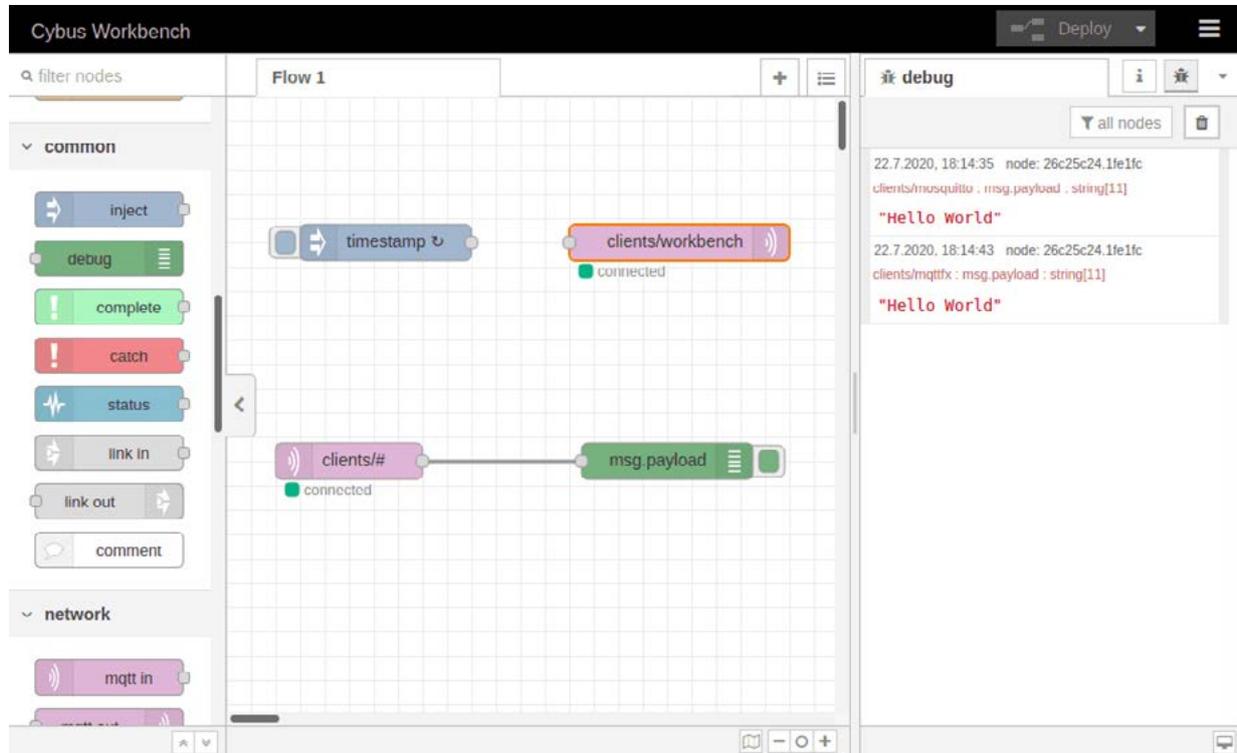
Eventually we will utilize the Workbench to monitor our data. But why just monitor it? The Workbench has a lot more to offer and exploring it will quickly give you a hint which possibilities lie ahead. But we will focus on that in another lesson.

At first we add an *mqtt in* node to the flow where we already created the data generator. Double-clicking it we choose the previously configured server "cybus" and the topic `clients/#`. The wildcard `#` defines, that we subscribe to every topic under `clients`. Be careful using wildcards since you could accidentally subscribe to a lot of topics which could possibly cause very much traffic and a message output throwing data from all the topics in a muddle. Confirm the node properties by clicking *Done* and add a debug node to the flow. Draw a connection from the *mqtt in* to the debug node and *Deploy* the flow.



Click the bug symbol in the upper right corner of the information bar on the right. We see there are already messages coming in: They are the timestamp messages created in the very same flow but taking a detour via the MQTT broker before plopping in our debug window. They are messing up the show so let us just cut the connection between the "timestamp" node and the publish node by selecting the connection line and pressing **Del** and *Deploy* the flow one more time.

Now that we have restored the quiet, let us finally validate if publishing with `mosquitto_pub` and MQTT.fx even works out. So issue the `mosquitto_pub` command again and also open MQTT.fx and publish a message like we did in the section before.



Et voilà! We can see two "Hello World" messages in our debug window and looking at the topics they were published on (written small in red right above the message text) we learn that one came from Mosquitto and the other from MQTT.fx.

Summary

This was quite a journey! We started in the realms of the Connectware, created credentials and permissions, learned about the magic of *self registration* and the trinity of MQTT clients. We went on and established connections between clients and brokers and sent messages to unobserved topics, struggling with the uncertainty of whether their destinations will ever be reached. But we saw light when we discovered the art of subscribing and realized that our troubles had not been in vain. In the end we received the relieving and liberating call of programmers: "Hello World"!

Where to go from here

The Connectware offers powerful features to build and deploy applications for *gathering, filtering, forwarding, monitoring, displaying, buffering*, and all kinds of *processing data*... why not build a dashboard for instance? For guides check out more of [Cybus Learn](#).

Cybus is a specialist for secure IIoT Edge software, headquartered in Germany. Cybus Connectware serves smart factories as a universal Edge and DevOps hub. Machine builders and providers of IIoT services use the Cybus Connectware as a software-based gateway. As early as 2017, Cybus published the first secure industrial connector for machine data according to today's DIN SPEC 27070 standard. Industry analyst Gartner named Cybus a worldwide "Cool Vendor". Today, the company counts medium-sized and large companies from numerous industrial sectors such as mechanical engineering, automotive and aviation among its customers.

Cybus GmbH · Osterstraße 124 · 20255 Hamburg · Germany · www.cybus.io · hello@cybus.io · (+49) 40 228 58 68 51