CLA-VAL **MQTT Configuration**

D22 - D35

User Manual

Table of Contents

1 INTRODUCTION

The purpose of this document is to provide a clear and comprehensive explanation of the MQTT protocol, detailing its functionality, configuration, and the steps necessary to connect our devices (D22, D35, CV-Log-35, MD35) to an MQTT broker. It is primarily aimed at engineers and technicians who want to implement a communication architecture based on MQTT. By the end of the document, readers will be able to understand the protocol, configure their devices properly, ensure effective and secure communication.

2 MQTT PROTOCOL

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish/subscribe messaging protocol designed for low-bandwidth and high-latency. MQTT is widely used in Internet of Things (IoT) applications, where sensors, devices, and machines need to communicate efficiently with servers or other devices. Its minimal resource requirements make it ideal for constrained environments.

2.1 BASIC PRINCIPLES OF THE MQTT

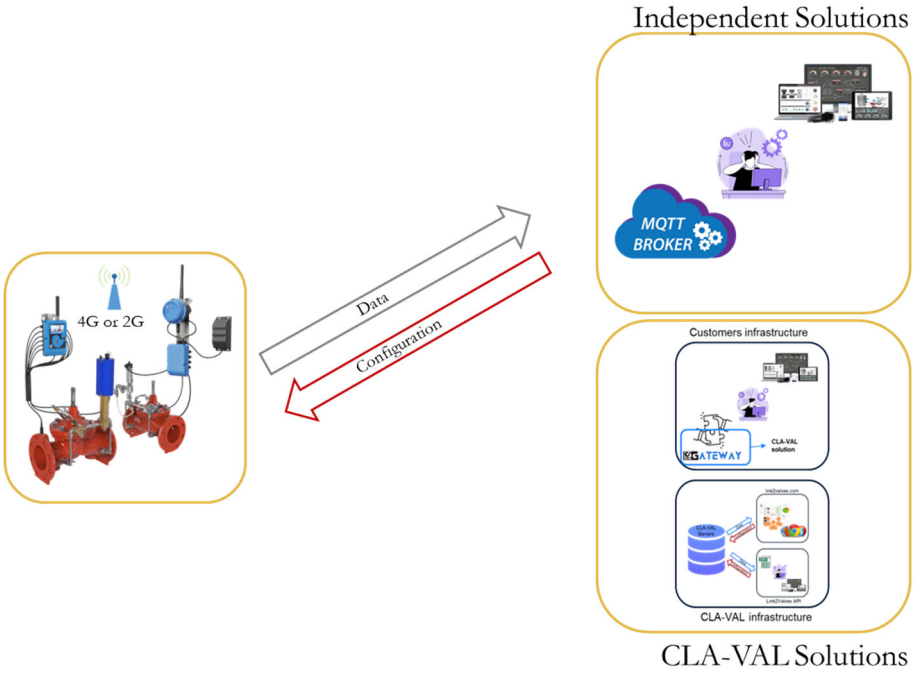
MQTT operates on a publish/subscribe model, which differs from traditional client-server architectures like HTTP.

In MQTT:

- **Publisher:** The device (D22 or D35) that sends (publishes) messages to specific "topics". For a D22 device there is a "data" topic, a "config" topic and "info" topic. For a (CV-Log-35, D35, MD35) there are two topics "data" and "config".
- **Subscriber:** A SCADA application that receives the data and the configs from topics it subscribes to.
- **Broker:** The intermediary that manages the communication between publishers and subscribers. It ensures that messages are delivered to the appropriate subscribers based on the topics.

To use MQTT on the CLA-VAL devices, users must configure devices to enable communication through this protocol (this procedure will be explained in the next chapter of this document). Additionally, they will need to configure the MQTT broker and handle the data processing and integration into their database or SCADA system.

By using MQTT, users can manage communication within their own network without needing to go through CLA-VAL infrastructure, but this requires development and setup on their side for data processing. MQTT communication is based on a publish/subscribe model, where the CLA-VAL D22 and D35 devices must be connected to a mobile network (4G or 2G) and capable of communication. These devices send data and their configurations to the broker, and from there, the data is retrieved from the client's SCADA system.



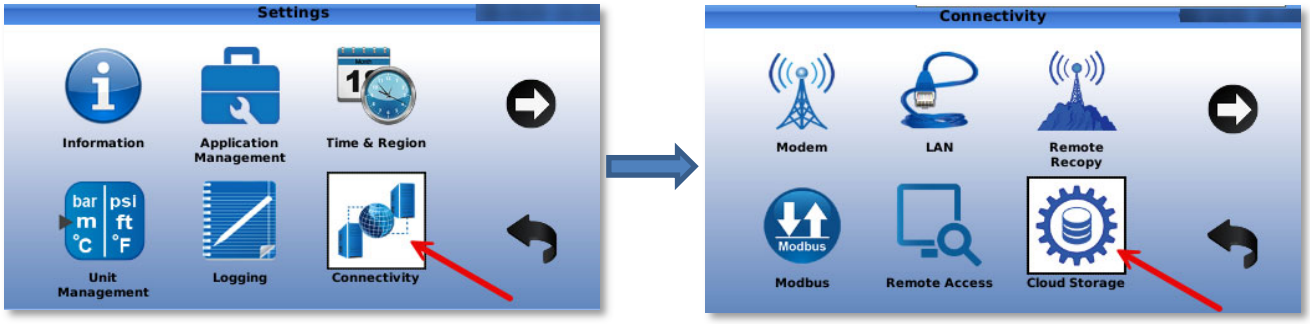
It is also possible to remotely modify certain parameters of the devices, such as control curves, variables, or the data logging interval and transfer interval, through the broker, allowing flexible adjustments.

3 DEVICES CONFIGURATION FOR MQTT

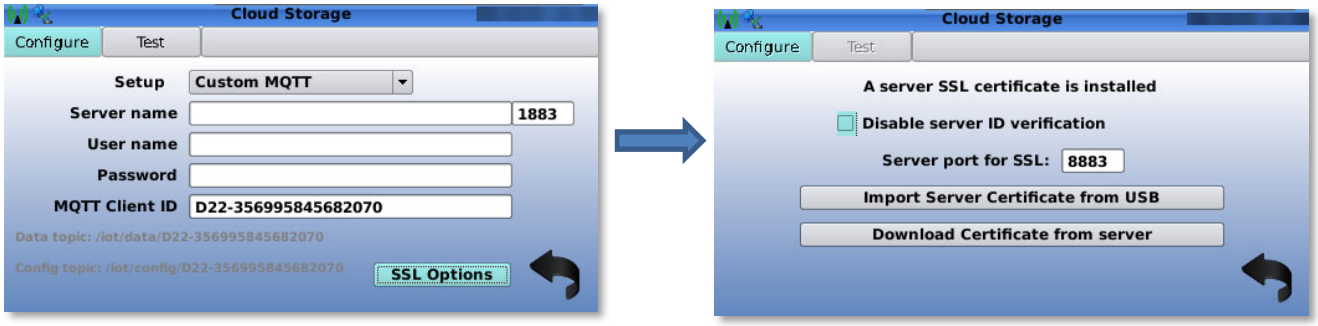
3.1 D22 CONFIGURATION FOR MQTT

To use the MQTT protocol, the D22 device needs to be running the R-engine version 2.8.7 or later and it must be connected to either a 4G or 2G network.

To configure the MQTT, perform a long click on the down/settings button of the D22 to access the settings menu. Once in the settings, click on "**Connectivity**," and then, on the connectivity page, select "**Cloud Storage**".



1. Inside the "**Cloud Storage**" choose the option "**Custom MQTT**" and configure the following settings:



2. Insert the "**Server's name or host**" and the port number (by default it is set to 1883).
3. Insert "**Username**" and "**Password**".
4. "**MQTT Client ID**" by default is set to D22-IMEI (it can be defined by the user, this can change the format for the Data topic, Config topic and Info topic).
5. In the "**SSL Options**" menu, it is possible to determine the server port for SSL (by default it is set to 8883), in this menu it is possible to download the certificate or import it from USB.

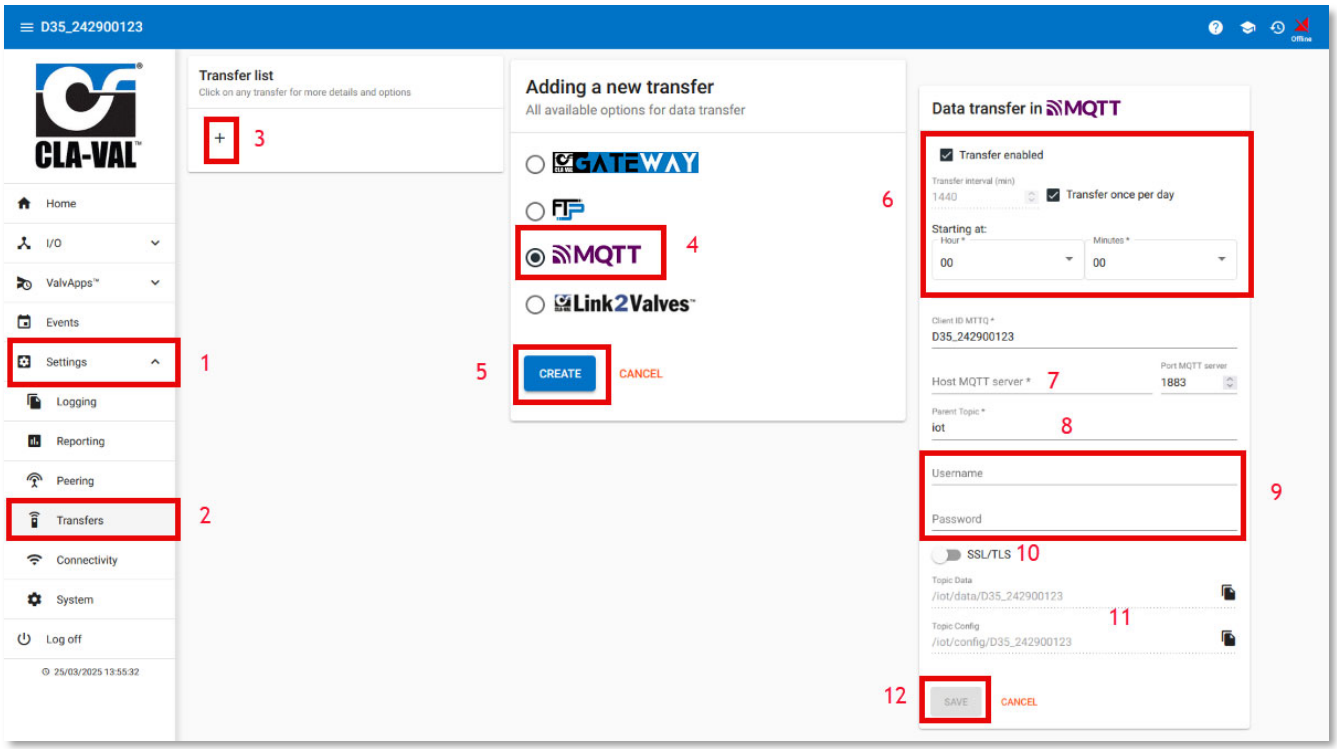
By completing these steps, the D22 is enabled to publish its data and configurations to the broker and can also receive remote configurations from the broker.

3.2 D35 & CV-LOG-35 CONFIGURATION FOR MQTT

To use the MQTT protocol, the D35 or CV-LOG-35 device needs to be running the engine version 2.3.2 or higher and it must be connected to either a 4G or 2G network.

To configure MQTT, connect the D35 or CV-LOG-35 to the PC via local Wi-Fi.

1. Open the web interface and navigate to "**Settings**".
2. Then select "**Transfers**".
3. Click the plus "**+**" button to add a new communication option.
4. Select "**MQTT**" as the communication method.
5. Click "**CREATE**" to complete the addition of the method.



To complete the configuration, follow the instructions below:

6. Enable the transfer, set the transfer interval and the starting hour of the transfers.
7. Insert the "**Server's name or host**" and the port number (by default it is set as 1883).
8. Insert the "**Parent Topic**" (by default it is set as iot).
9. Inset the "**Username**" and "**Password**".
10. The SSL/TLS button enables to choose between two different security standards (by default the SSL port is set as 8883).
11. "**Client ID MQTT**" by default is set to D35-SerialNumber (it can be defined by the user, this can change the format for the Data topic and Config topic).
12. By clicking on the "**Save**" button, the MQTT configuration is confirmed and stored, enabling the D35 to transmit its data and configurations to the broker, as well as receive updated configurations from the broker.

4 DATA FORMAT AND REMOTE CONFIGURATION VIA MQTT

Via MQTT, it is possible to modify the control curves, actions, adjust variables, as well as change both the data logging frequency and the data transmission frequency. Below, you can find an example of code for each of these modifications.

To use the commands below, the user must send them to the "**config**" topic.

4.1 MODIFYING CONTROL CURVES VIA MQTT

Here's an example of how to modify a control curve. To do this, you need to insert the number of the control curve you want to modify in "**CurveId**". Further down, in the "X" and "Y" fields, you can modify the existing points or add new ones using the same syntax.

```
{
   "profileChanges":[
      {
         "curveId":3,
         "profile":[

            {
               "X":4,
               "Y":2.5
            },
            {
               "X":8,
               "Y":2.5
            },
            {
               "X":12,
               "Y":6
            },
            {
              "X":15,
              "Y":12
            },
            {
              "X":18,
              "Y":14
            },
            {
              "X":20,
              "Y":14.9
            }
         ]
      }
   ]
}
```

4.2 MODIFYING VARIABLES VIA MQTT

The example below shows how to modify variables. To modify a variable, you need to insert its name and the desired value.

```
{
   "settingChanges":[
      {
         "name":"P_SP",
         "value":12.0
            },
      {
         "name":"Pos_SP",
         "value":75.0
            },
      {
         "name":"Q_SP",
         "value":2.0
      }
         ]
      }
```

4.3 MODIFYING LOGGING & TRANSFER INTERVAL

The example below shows how to modify the logging interval and/or the data transfer interval. The recording interval is expressed in seconds, while the transfer interval is expressed in minutes. The text remains the same; the only thing you need to modify are the values.

```
{
 "logging":{
            "interval":60,
            "transfer": 1440
               }}
```

4.4 MODIFYING ACTIONS VIA MQTT

The example below shows how to modify the parameters of a configured action. For each parameter, the modification must be made within the specified ranges. Please check the image below.

```
{
    "actions": [
        {
            "id": 1,
            "enabled": true,
            "name": "Alarm1",
            "output": {
                "port": "SO1",
                "value": 85.0,
                "send_data": false,
                "activation_delay": 10,
                "deactivation_delay": 3,
                "minimum_activation_duration": 30,
                "priority": 0,
                "activation_oneshot": false,
                "deactivation_action": "restore",
                "deactivation_value": null
            },
            "operations": [
                {
                    "condition": {
                        "id": 1,
                        "type": "value",
                        "port": "AI1",
                        "reference":"AI3",
                        "comparison": ">",
                        "hysteresis": 1.0
                    },
                    "operator": "and"
                },
                {
                    "condition": {
                        "id": 2,
                        "type": "value",
                        "port": "AI2",
                        "constant": 31,
                        "comparison": "<",
                        "hysteresis": 2.0
                    },
                    "operator": null
                }
            ]
        }
    ]
}
```