MQTT Driver

File Name	MQTT.dll
Manufacturer	Message Queuing Telemetry Transport
Devices	Clients and Brokers compatible with specifications 3.1, 3.1.1, and 5 of MQTT protocol
Protocol	MQTT over Ethernet TCP/IP and TLS/SSL
Version	1.0.42
Last Update	03/24/2026
Platform	Win32
Dependencies	IOKit version 2.0 and OpenSSL library
Superblock Readings	No
Level	31312

Introduction

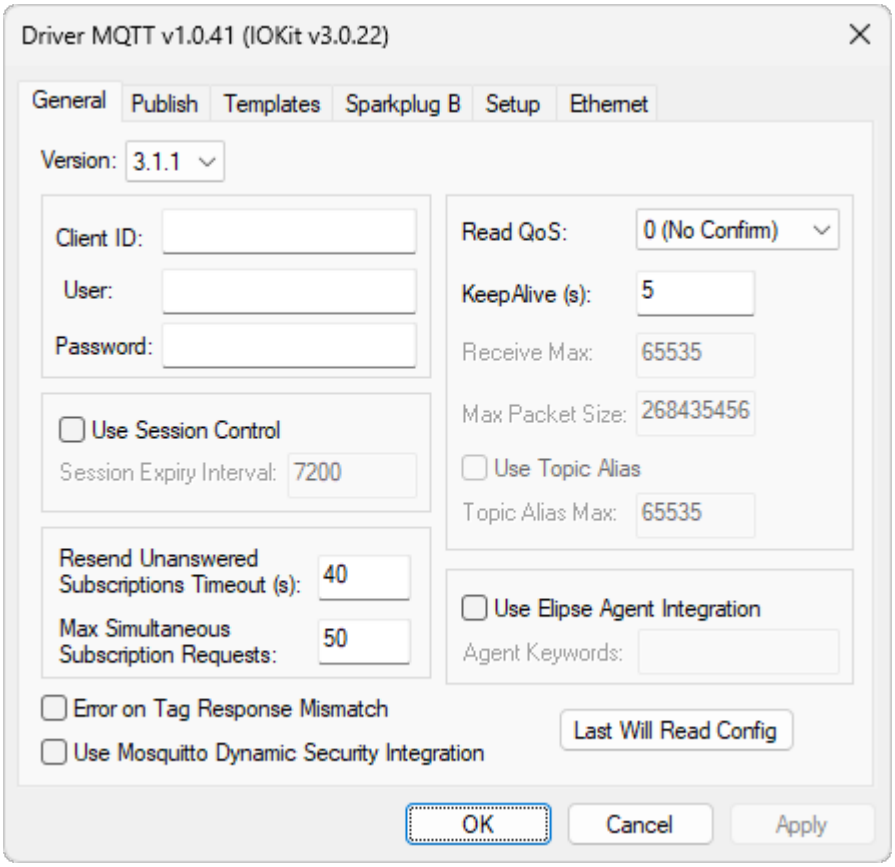
The MQTT Driver is a **Client** that implements versions **3.1**, **3.1.1**, and **5** of the standard MQTT protocol, or **Vanilla**, with additional support for **Sparkplug B** protocol, using an external MQTT Broker, not provided.

This Driver can receive and extract values from received messages, as well as send messages that are processed by other **Clients**.

Connections among **Clients** are not performed directly, only using a Broker.

Driver's Configuration Parameters

This Driver's **[P]** configuration parameters are not used. All settings are performed on the configuration window, shown on the next figure.



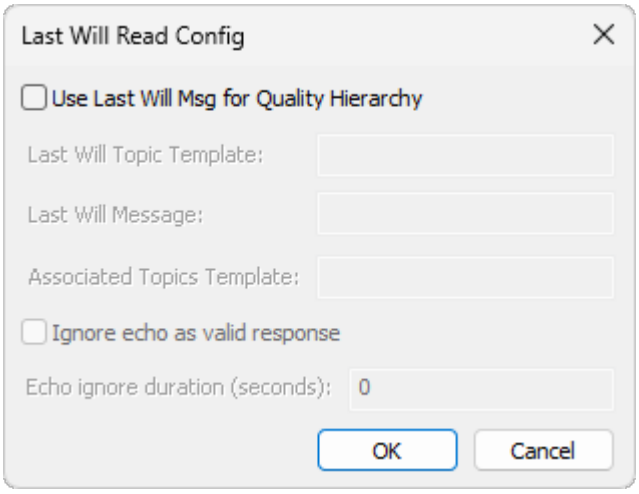
General tab

The available options on the **General** tab are described on the next table.

Available options on the General tab

OPTION	DESCRIPTION
Version	Protocol version. Available options are **3.1**, **3.1.1**, or **5**
Client ID	Name used by this **Client** as an identifier for all other **Clients**. There must not be another **Client** connected to the same Broker with that name
User	Name of a user used in connection messages (**ConnAck**), if needed and allowed by a Broker
Password	Password used together with the user name
Read QoS	Indicates the QoS (*Quality of Service*) code used to request Tag subscriptions. Possible values are **0**: No Confirm, **1**: Confirm, or **2**: Handshake. A Broker compares the requested QoS code with the QoS code originally published, then sending at least a compatible data
KeepAlive (s)	Time, in seconds, that a **Client** sends a **ping** message in MQTT protocol (**PingReq**) to check whether a connection with a Broker is still active. The default value of this option is 5 (five) seconds
Receive Max	Available only in version **5** of MQTT protocol, defines how many QoS 1 (one) or 2 (two) messages can be received from a Broker simultaneously before being acknowledged
Max Packet Size	Available only in version **5** of MQTT protocol, defines the maximum size, in bytes, for MQTT packets

OPTION	DESCRIPTION
Use Topic Alias	Available only in version **5** of MQTT protocol. When selecting this option, a numeric identifier is used internally to replace a long topic, thus reducing the size of packets
Topic Alias Max	Available when the **Use Topic Alias** option is selected. Limits how many topics can be replaced by numeric identifiers during communication
Use Session Control	Informs whether this Driver must keep the status of the active session, that is, if a disconnection occurs and then a connection occurs, the QoS 1 (one) and 2 (two) messages sent by other **Clients** can be recovered
Session Expiry Interval	Available only in version **5** of MQTT protocol, defines the time, in seconds, that a session remains stored on a Broker after disconnection
Resend Unanswered Subscriptions Timeout (s)	Informs the number of seconds to resend a subscription of an item if it did not receive the first value during the informed period
Max Simultaneous Subscription Requests	Maximum number of item subscription requests to be sent at each message
Error on Tags Response Mismatch	Informs whether this Driver must return an error for Tags whose message does not match the configured Template
Use Mosquitto Dynamic Security Integration	Enables integration of control or configuration of Mosquitto permissions, by using pre-defined MQTT messages, triggered by Tags
Use Elipse Agent Integration	Enables integration with an **Elipse Software** Agent, a diagnostics tool for computers with Windows
Last Will Read Config	Open the Last Will Read Config window, shown on the next figure, to configure the Last Will message received by this Driver



Last Will Read Config window

The available options on the Last Will Read Config window are described on the next table.

Available options on the Last Will Read Config window

OPTION	DESCRIPTION
Use Last Will Msg for Quality Hierarchy	When selecting this option, all other settings on this page are enabled for editing, except the **Echo ignore duration (seconds)** option. If this Driver receives the message configured in the **Last Will Message** option on a topic that matches the one defined in the **Last Will Topic Template** option, all topics that match the value configured in the **Associated Topics Template** option are marked with a bad quality 20, 24, or 28 (invalid)
Last Will Topic Template	MQTT topic Template for the Last Will message
Last Will Message	Last Will message, such as "Goodbye"
Associated Topic Template	Template of MQTT topics associated to the Last Will message, that is, the topics that are marked with a bad quality
Ignore echo as valid response	When selecting this option, the **Echo ignore duration (seconds)** option is enabled for editing
Echo ignore duration (seconds)	This option defines a period, in seconds, during which a message received on a topic is considered an echo sent by a Broker when this Driver publishes to that topic. Thus, this message is disregarded as a valid communication, and the topics corresponding to the value of the **Associated Topics Template** option are not marked with a good quality (192)

MQTT topic Templates obey the next specifications:

- They can include fixed fields with predefined names
- The + (plus sign) character is used as a wildcard to represent any topic at a specific level
- Variable fields used for checking or differentiating between Last Will messages
- The # (number sign) character is used as a wildcard for multiple levels, used exclusively at the end of a topic

These specifications comply with the MQTT standard, except for variable fields, which follow the **<VARIABLE_NAME>** format.

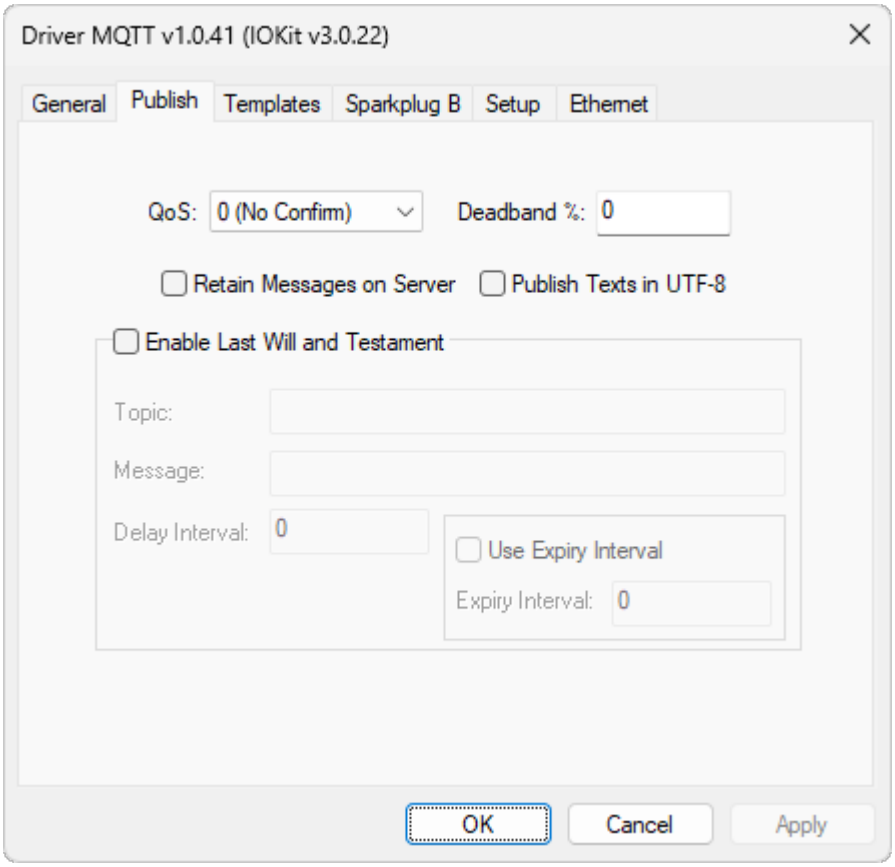
For example, consider a topic structure following the pattern *city, city code, device, device identifier* and, later, the corresponding parameters, such as "city/003/device/147/...".

To configure the **Last Will Topic Template** option, users can use the format "city/<CITY_ID>/device/<DEVICE_ID>/lastwill". For the **Associated Topic Template** option, on the other hand, users can use the format "city/<CITY_ID>/device/<DEVICE_ID>/#".

Thus, when a Last Will message is received, all Tags following the format with the same **<CITY_ID>** and **<DEVICE_ID>** from the received Last Will message are marked with a bad quality. If there are other fields in the topic that vary and are not fixed **Strings**, such as *city* or *device*, nor repeated in the associated topics, such as **<CITY_ID>** and **<DEVICE_ID>**, the + (plus sign) character can be used. For example, if the associated topics follow the pattern "<APPLICATION_NAME>/<CITY_ID>/<DEVICE_ID>", the **Associated Topic Template** option can be configured as "+/<CITY_ID>/<DEVICE_ID>/#".

This demonstrates the system's flexibility, allowing to configure how the Last Will message affects Tags and enabling a distinction between different Last Will messages. It is essential that the Templates are followed and that all variable fields, defined in the format **<VARIABLE_NAME>** in the **Last Will Topic Template** option, are found with the same values in the **Associated Topic Template** option.

For these Tags to return to a good quality, the value 192, the corresponding topic must receive a message with a value different from the one configured in the **Last Will Message** option. However, if the Tag's *Item* parameter starts with the ! (exclamation point) character, that Tag does not have its quality marked as bad upon receiving the Last Will message, nor returns to a good quality upon receiving a new value.



Publish tab

The available options on the **Publish** tab are described on the next table.

Available options on the Publish tab

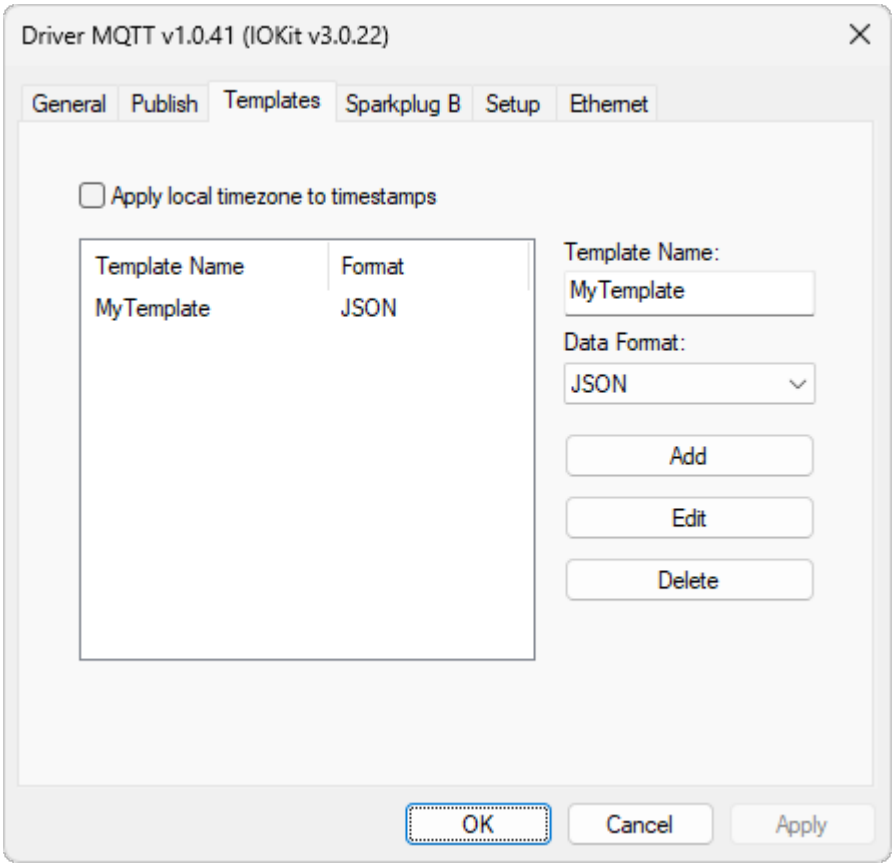
OPTION	DESCRIPTION
QoS	Indicates the service level of messages published by this Driver. The available options are **QoS 0**: There is almost always a deliver (there may have zero or more deliveries), **QoS 1**: At least one or more deliveries, or **QoS 2**: Exactly one deliver
Deadband %	Value for the relative dead band applied when writing a numeric value to any Tag without Templates
Retain messages on Server	Selecting this option instructs a Broker to keep this **Client**'s messages together with their QoS, so that **Clients** subscribing in the future can receive accumulated messages. When a new **Client** subscribes, the last message of each topic, if available, must be sent to that **Client** by a Broker

OPTION	DESCRIPTION
Publish Texts in UTF-8	Informs if **Text**-type Tags must have their values converted to the **UTF-8** format before writing them. This avoids a wrong interpretation of messages with special characters by other **Clients**
Enable Last Will	When enabling this option, a server must keep in memory a topic and a message for a **Client**, which are sent to all other **Clients** when there is an unordered disconnection, that is, without sending a *Disconnect* message
Topic	Topic or address sent during a disconnection (*Last Will*), such as "ELIPSE\goodbye"
Message	Message linked to a topic, such as "Goodbye"
Delay Interval	Available only in version **5** of MQTT protocol, defines the time, in seconds, that a Broker waits after an unexpected disconnection before publishing a *Last Will* message
Use Expiry Interval	Available only in version **5** of MQTT protocol, enables the **Expiry Interval** option
Expiry Interval	Available only in version **5** of MQTT protocol, defines the time, in seconds, after which the *Last Will* message is discarded by a Broker if it has not been delivered yet

NOTE

To define a different value of **QoS**, **Retain**, and **Expiry Interval** for each Tag, users can configure a Tag with parameters *N1* equal to 1 (one), *N2* equal to **QoS** (zero, one, or two), *N3* equal to **Retain** (zero or one), and *N4* equal to **Expiry Interval**, in seconds, if the version of MQTT protocol is equal to **5** (five).

On the **Templates** tab, users can define different patterns for data on MQTT messages, whose elements can be automatically extracted or filled by this Driver using Links with Tags or Block Tags.



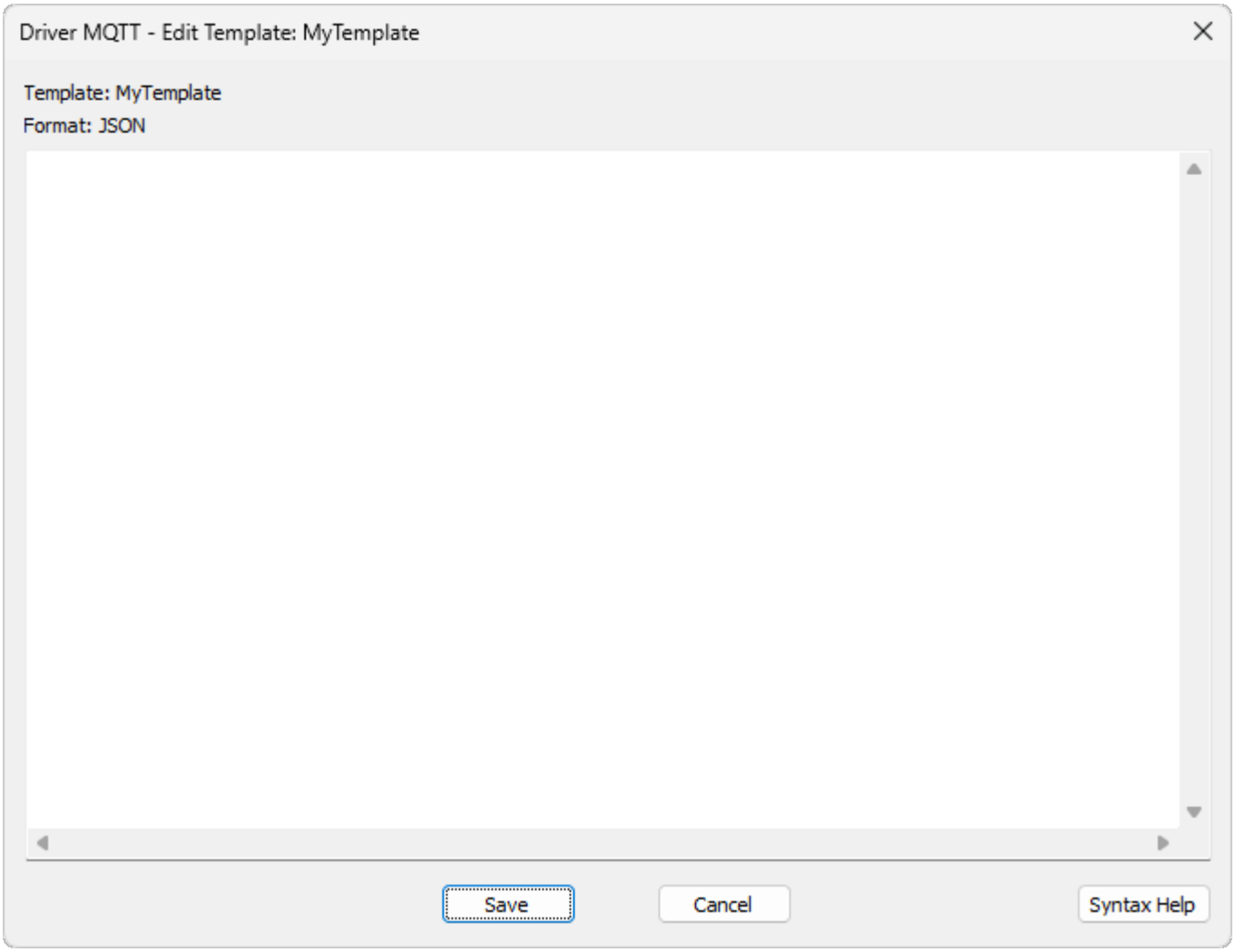
Templates tab

The available options on the **Templates** tab are described on the next table.

Available options on the Templates tab

OPTION	DESCRIPTION
Apply local timezone to timestamps	When this option is selected, the received timestamps are converted to the timezone used by the operating system. Otherwise, the received timestamps are displayed in **UTC** timezone
Template Name	Informs the name of a Template. Templates with the same name are not allowed
Data Format	Declares the format of a Template's message. The available options are **JSON**, **CSV**, or **XML**. The **JSON** format is preferred for use with MQTT, because it allows defining any type of data structure with less bytes
Add	Adds a Template to the list of Templates
Edit	Opens a window for editing the selected Template
Delete	Deletes the selected Template from the list of Templates

Click **Edit** to open a window for editing the selected Template. The name and format of a Template are displayed at the top. The Template itself must be inserted in the edit box.



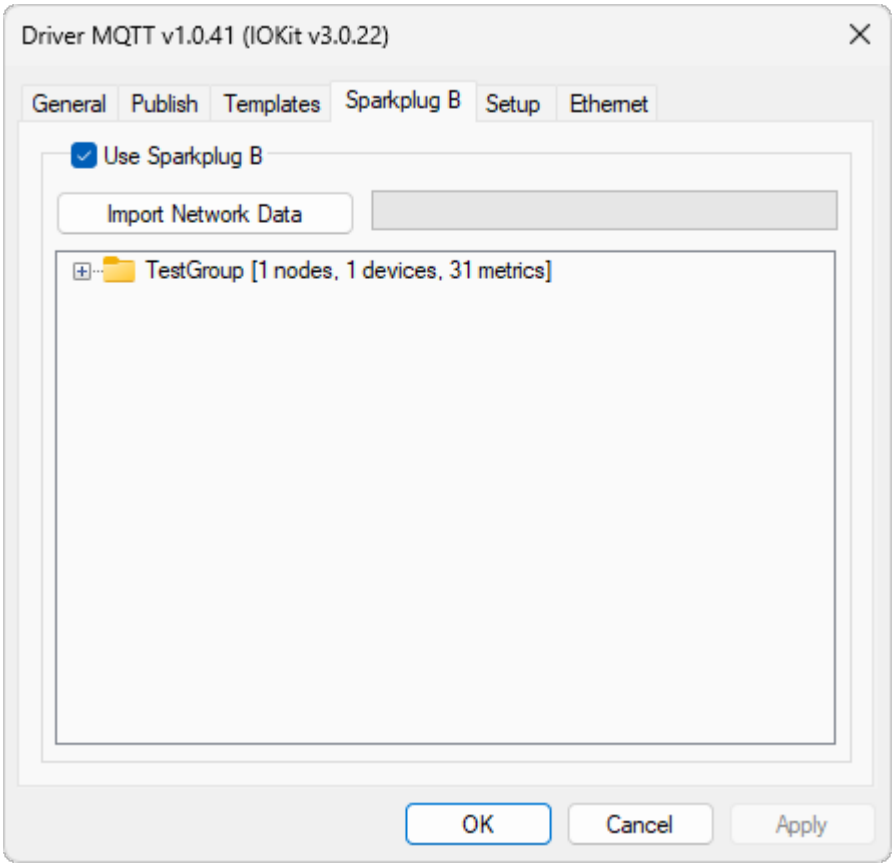
Window for editing a Template

The available options on the window for editing a Template are described on the next table.

Available options on the window for editing a Template

OPTION	DESCRIPTION
Save	Confirms all changes and closes this window
Cancel	Discards all changes and closes this window
Syntax	Informs the syntax of a message, in the selected format (**JSON**, **CSV**, or **XML**) together with the keywords used during the process of replacing them by timestamps, qualities, or values. For more information, please check topic **Template Syntax**

On the **Sparkplug B** tab, users can enable this communication format via MQTT and import groups, nodes, devices, and metrics retained in a Broker.



Sparkplug B tab

The available options on the **Sparkplug B** tab are described on the next table.

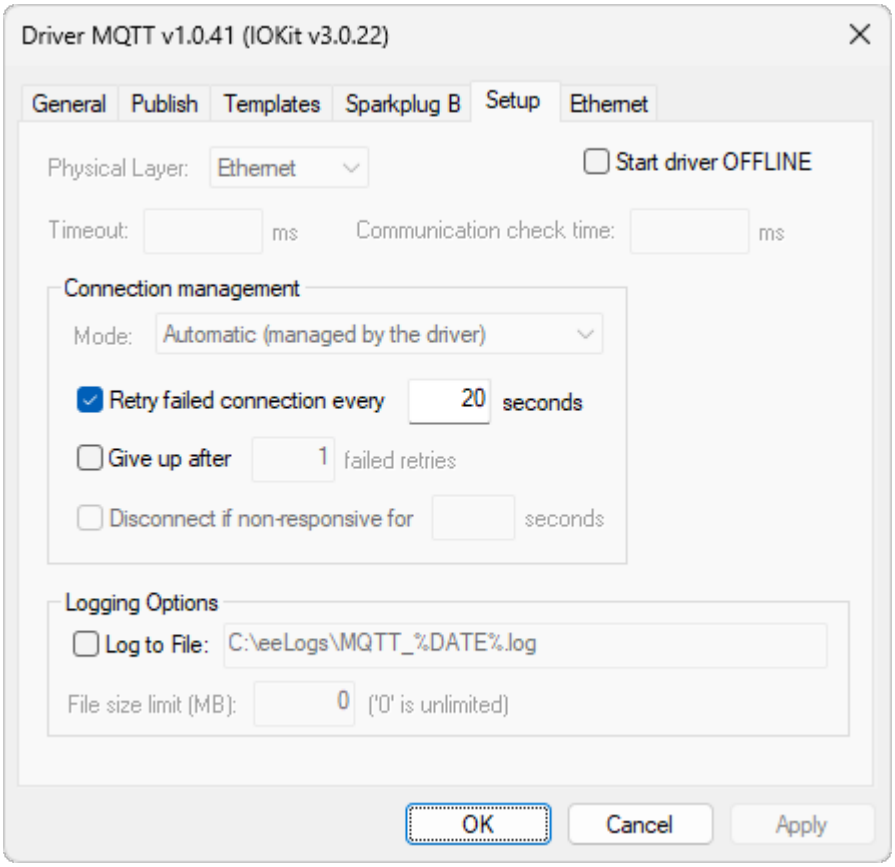
Available options on the Sparkplug B tab

OPTION	DESCRIPTION
Use Sparkplug B	Select this option for this Driver to stop supporting **Vanilla** standard and start acting as a Primary Application, following a distinct configuration of Tags
Import Network Data	Discovers and imports groups, nodes, devices, and metrics retained in a Broker, building a tree for visualization

NOTE

For this import process to work correctly, the connection settings must be filled on the **General** and **Ethernet** tabs. The import process automatically creates Tags on the **Tag Browser** window.

The **Setup** tab contains general settings for this Driver.



Setup tab

The available options for this Driver on the **Setup** tab are described on the next table.

NOTE

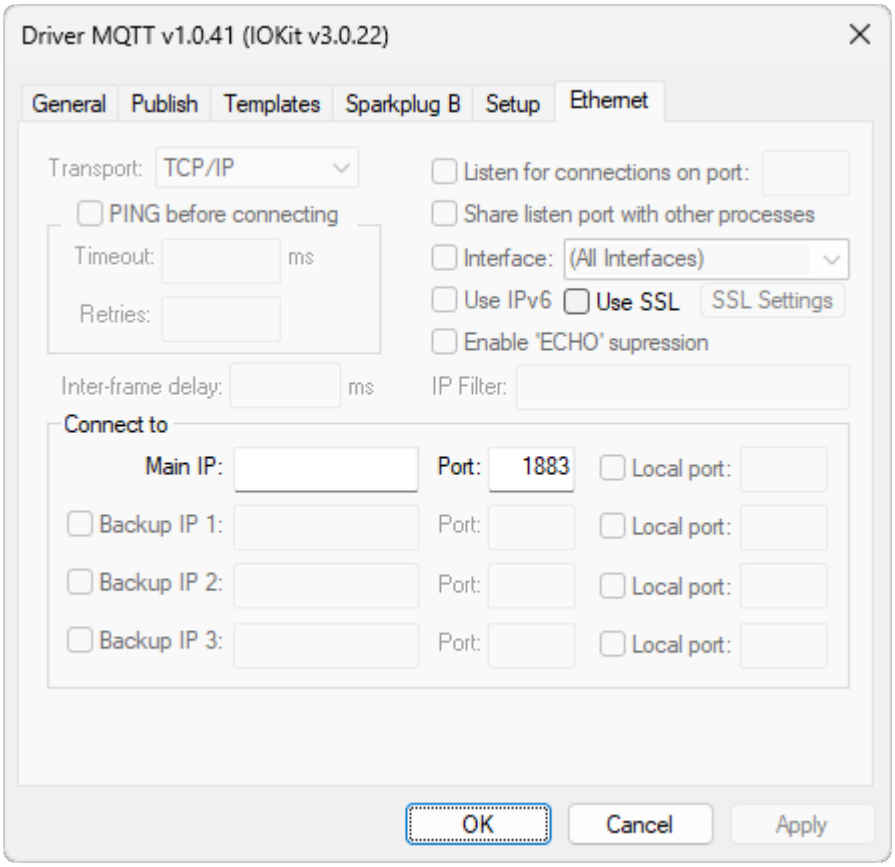
The options of the **Setup** tab that are not described on the next table are deactivated and have no effect on the configuration of this Driver.

Available options on the Setup tab

OPTION	DESCRIPTION
Start driver OFFLINE	Select this option for this Driver to start in **Offline** mode or stopped. This means that the **Ethernet** interface is not created until users configure this Driver in **Online** mode by using a Tag in an application. This mode enables a dynamic configuration of the **Ethernet** interface at run time
Retry failed connection every *n* seconds	Select this option to enable a connection retry of this Driver in an *n* interval, in seconds. If the **Give up after *n* failed retries** option is not selected, this Driver keeps retrying until the connection is established, or until an application is stopped
Give up after *n* failed retries	Enable this option to define a maximum number of connection retries. When the specified number of consecutive connection retries is reached, this Driver goes to the **Offline** mode, assuming that a hardware problem was detected. If this Driver establishes a successful connection, the number of retries is zeroed. If this new connection is lost, then the counter of retries starts at zero

OPTION	DESCRIPTION
Log to file	Enable this option and configure the name of the file in which logs are written. Log files can be large, therefore use this option for short periods of time, only for testing and debugging purposes. In case of using the **%PROCESS%** macro in the name of a log file, it is replaced by the identifier of the current process. This option is particularly useful when using several instances of the same Driver in **Elipse E3**, in **Elipse Power**, or in **Elipse Water**, thus allowing each instance to generate a separate log file. For example, when configuring this option with the value "c:\e3logs\drivers\mqtt_%PROCESS%.log", it generates a file c:\e3logs\drivers\mqtt_00000FDA.log for the **0FDAh** process. Users can also use the **%DATE%** macro in a file name. In this case a log file per day is generated, in the format **aaaa_mm_dd**. For example, when configuring this option with the value "c:\e3logs\drivers\mqtt_%DATE%.log", it generate a file c:\e3logs\drivers\mqtt_2005_12_31.log on 12/31/2025 and a file c:\e3logs\drivers\mqtt_2026_01_01.log on 01/01/2006. Analogously, the **%DATE_HOUR%** macro generates a log file per hour, in the format **aaaa_mm_dd_hh**
File size limit (MB)	Configure the size limit of a log file, in megabytes. A value equal to 0 (zero) means that there is no size limit for a log file

The **Ethernet** tab allows configuring parameters of the **Ethernet** interface.



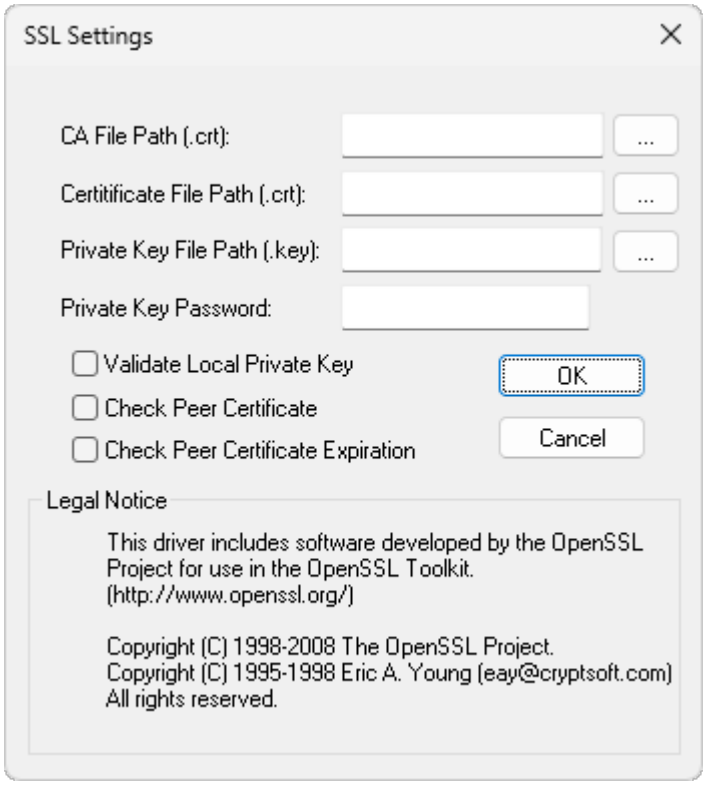
Ethernet tab

The available options for this Driver on the **Ethernet** tab are described on the next table.

NOTE
The options of the **Ethernet** tab that are not described on the next table are deactivated and have no effect on the configuration of this Driver.

Available options on the Ethernet tab

OPTION	DESCRIPTION
Use SSL	When selecting this option, the SSL (*Secure Sockets Layer*) protocol is used for a secure connection with a Broker. The **SSL Settings** is enabled to configure the certificates
SSL Settings	Opens the SSL Settings window to configure the certificates, shown on the next figure
Main IP	Type the IP address of a remote device. Users can use either an IP address separated by dots or a URL. For a URL, this Driver uses the available DNS service to map that URL to an IP address, such as "192.168.0.13" or "Server1"
Port	Type the TCP/IP port of a remote device, between 0 (zero) and 65535



SSL Settings window

The available options on the SSL Settings window are described on the next table.

Available options on the SSL Settings window

OPTION	DESCRIPTION
CA File Path (.crt)	Name and path of a file for a Certificate of Authority (CA). Click ... to select a file
Certificate File Path (.crt)	Name and path of a file for a **Client**'s SSL Certificate. Click ... to select a file
Private Key File Path (.key)	Name and path of a file with private keys for a **Client**'s SSL Certificate. Click ... to select a file
Private Key Password	Password of a private key, if applicable
Validate Local Private Key	Enables or disables validating the file defined in the **Private Key File Path (.key)** option
Check Peer Certificate	Enables or disables checking a **Client**'s SSL Certificate
Check Peer Certificate Expiration	Enables or disables checking the expiration date of a **Client**'s SSL Certificate

The I/O Tags available for this Driver are described next.

IO.CommunicationStatus

Read-only I/O Tag that informs the communication status of this Driver. Indicates how communication works relative to receiving valid data inside a time period set during the configuration. Possible values are **0 - Inactive communication**: This Driver did not receive valid data or stopped receiving data after *n* milliseconds, as configured on the **Setup** tab, or **1 - Active communication**: This Driver is receiving valid data.

IO.PhysicalLayerStatus

Read-Only I/O Tag that indicates the status of the physical layer. Possible values are **0**: The physical layer is stopped, that is, this Driver is in **Offline** mode, the physical layer failed during initialization, or exceeded the maximum number of reconnection retries, **1**: The physical layer was started but not connected, that is, this Driver is in **Online** mode, but the physical layer is not connected, or **2**: The physical layer is connected, that is, the physical layer is ready for use. This **DOES NOT** mean that a device is connected, only that the access layer is working.

IO.SetConfigurationParameters

Read-Only Block Tag that allows changing any configuration property of this Driver at run time. This Tag only works while this Driver is in **Offline** mode. To do so, select the **Start driver OFFLINE** option on the **Setup** tab. Writing individual Block Elements is not supported, a whole Block must be written at once.

IO.WorkOnline

Reading and writing I/O Tag that informs the current status of this Driver and allows starting or stopping the physical layer. Possible values are **0 - Driver Offline**: The physical layer is closed or stopped. This mode allows a dynamic configuration of this Driver's parameters using the **IO.SetConfigurationParameters** Tag or **1 - Driver Online**: The physical layer is open or executing. While it is in **Online** mode, the physical layer can be connected or disconnected and the current status can be checked in the **IO.PhysicalLayerStatus** Tag.

Template Syntax

To extract the content of a message, users must declare a Template, which allows informing that message's format and which parts must be transformed into data. Templates can have a **JSON**, **CSV**, or **XML** type. Each Template must use keywords, which must replace the values to extract. The available keywords are described on the next table.

Available keywords for Templates

KEYWORD	DESCRIPTION
TS_TEXT(format)	Timestamp, as a text, used as a Block Tag's or I/O Tag's timestamp. The meaning of each field is described on table **Available options for the TS_TEXT keyword**
TS_UNIX	Timestamp, in seconds, since 1970 (**UNIX** format). This value can be a number or a text, and it is used as a Block Tag's or I/O Tag's timestamp, such as 1504198675 or "1504198675"
TS_DAYS1900	Timestamp, as a number, with the number of days since Jan,1st 1900, also known as Gregorian calendar, used as Block Tag's or I/O Tag's timestamp, such as 43595.37373843. This keyword is only available for Templates in **JSON** or **XML** format
TS_UNIX_TZ_SECONDS+ **TS_UNIX_TZ_SECONDS-**	Timezone, in seconds, informed separately from the **TS_UNIX** keyword. User must inform a + (plus) or - (minus) symbol to indicate whether seconds must be added or subtracted, respectively, from the **TS_UNIX** keyword to represent a message's timestamp. This keyword is only available for Templates in **JSON** format
QL_OPC	Quality, in OPC DA standard, using a byte. This value is used directly as a quality value in a Block Tag or I/O Tag, without transformations. Please check table **OPC DA standard** for more information
QL_BOOL	Quality, in Boolean standard. If a value is greater than 0 (zero) or a "true" or "TRUE" expression, quality is good. If a value is equal to 0 (zero) or a "false" or "FALSE" expression, quality is bad
V1, V2, ... V50	Declares one or more values, which can be extracted individually for I/O Tags, by specifying a third addressing parameter (*param*). If all values of a message are mapped to a single Block Tag, then users can use an **E3VAL** keyword for any value to extract, indistinctly
E3VAL	Specifies a value that is extracted in the sequence it occurs for a Block Tag, each value to its own Block Element. If there is only one **E3VAL** value in the message, the Template can also be used with an I/O Tag
DUMMY	A variable field, but whose value must not be sent to I/O Tags
Repeat_E3VAL	Keyword that can only be used inside a **JSON**-type array, to indicate that the repeating members must be read independently by this Driver, returning a reading operation for each set. For more information, please check topic **JSON Template**

Available options for the TS_TEXT keyword

OPTION	DESCRIPTION
%a	Short weekday
%A	Full weekday
%b	Short name of a month
%B	Full name of a month
%C	Century
%d	Day of a month, starting with 0 (zero)
%e	Day of a month, starting with a space
%f	Milliseconds, from 0 (zero) to 999
%h	Hour, in 12-hour format
%H	Hour, in 24-hour format
%m	Month
%M	Minute
%p	AM or PM
%S	Seconds
%y	Year with two digits
%Y	Year with four digits
%Z	Name of a timezone, an international code that transforms time to GMT
%+	GMT offset in the format +-HH:MM

The next code contains examples of timestamps formatted using the **TS_TEXT** keyword.

```
"2014-07-11T15:26:37Z" -> "TS_TEXT(%y-%m-%dT%H:%M:%SZ)"
"Mon Jul 10 11:04:47 BRT 2017" -> "TS_TEXT(%a %b %d %H:%M:%S %Z %Y)"
"2018-05-02T10:29:28.622-02:00" -> "TS_TEXT(%Y-%m-%dT%H:%M:%S.%f%+)"
```

OPC DA standard

BIT	7	6	5	4	3	2	1	0
Description	Q	Q	S	S	S	S	L	L

Available options for the OPC DA standard

OPTION	DESCRIPTION
QQ	Two quality bits
SSSS	Four sub-status bits
LL	Two limit bits

OPTION	DESCRIPTION
QQ	**0**: BAD, **1**: UNCERTAIN, or **3**: GOOD
SSSS	**0**: BAD_NONSPECIFIC, **1**: BAD_CONFIGERROR, **2**: BAD_NOTCONNECTED, **3**: BAD_DEVICEFAILURE, or **4**: BAD_SENSORFAILURE
SSSS	**0**: UNCERT_NONSPECIFIC, **1**: UNCERT_LASTUSABLEVALUE, or **4**: UNCERT_SENSORNOTACCURATE
SSSS	**0**: GOOD_NONSPECIFIC, **1**: GOOD_LOCALOVERRIDE, or **6**: GOOD_NONSPECIFICLOCALTIMESTAMP
LL	**0**: FREE, **1**: LOW, **2**: HIGH, or **3**: CONST

JSON Template

JSON (*JavaScript Object Notation*) is a format for data exchange that is easily understood by humans, and also simple so that programs can perform information processing and extraction. The **JSON** format is build on top of a collection of pairs containing name and value and an ordered list of these pairs.

An **Object** is an unordered list of pairs of names and values. An object is delimited by braces, each name is defined between quotation marks followed by a colon and its respective value, and each pair is separated by commas.

An **Array** is an ordered list of values. An array is delimited by brackets and its values are separated by commas. Values can be numbers or texts, which must be represented between quotation marks. The next code contains an example of a message in **JSON** format.

```
{"s":1, "t":"2014-07-11T15:26:37Z", "q":192,"c":1, "x":-1.234,"y":0.234, "z":-0.234}
```

The next code contains a possible Template to extract the content of the previous message.

```
{"s":"DUMMY", "t":"TS_TEXT(%y-%m-%dT%H:%M:%SZ)","q":"QL_OPC", "c":"DUMMY", "x":"E3VAL","y":"E3VAL",
"z":"E3VAL"}
```

In this case, users must create a Block Tag with three Block Elements, and each Block Element receives one of the **x**, **y**, and **z** values, respectively.

The next code contains an example of a message in **JSON** format.

```
{
  "n_channels":2,
  "timestamp":1504198675,
  "hash":"1842E0F97392F08BDF996961A8333832AB06D113",
  "battery":5.13,
  "gmt":-3,
  "tag_channels":["Analog1","Analog2"],
  "value_channels":[28.100,27.200],
  "tag_units":["°C","°C"],
  "alarm_low":[0,0],
  "alarm_high":[0,0],
  "buzzer_state":0
}
```

The next code contains a possible Template for the previous example.

```
{
  "n_channels":"DUMMY",
  "timestamp":"TS_UNIX",
  "hash":"DUMMY",
  "battery":"V1",
  "gmt":"V2",
  "tag_channels":["V3","V4"],
  "value_channels":["V5","V6"],
  "tag_units":["V7","V8"],
  "alarm_low":["V9","V10"],
  "alarm_high":["V11","V12"],
  "buzzer_state":"V13"
}
```

NOTE

Both messages and Templates must be objects or arrays, therefore they must start with a brace ({) or bracket ([) character, respectively.

Repetition of Values

Arrays can be used to receive a variable number of elements. The next codes show examples of messages.

Message 1 (one):

```
{
  "MessageType": "Test",
  "TagData": [
    {
      "Temperature": 11,
      "Humidity": 50
    },
   {
      "Temperature": 14,
      "Humidity": 80
    }
  ]
}
```

Message 2 (two):

```
{
  "MessageType": "Test",
  "TagData": [
    {
      "Temperature": 11,
      "Humidity": 12
    },
   {
      "Temperature": 14,
      "Humidity": 66
    }
   {
      "Temperature": 70,
      "Humidity": 55
    }
  ]
}
```

Inside an array, users can have 1 (one), 2 (two), 3 (three), or *n* sets of temperature and humidity in a single message, and it might be interesting if a "Test" Block Tag with two Elements, **Temperature** and **Humidity**, could receive a reading at each set found.

To do so, declare the Template with the **Repeat_E3VAL** keyword right after the bracket character that marks the array start, as in the next example.

```
{"MessageType":"DUMMY","TagData":[Repeat_E3VAL{"Temperature":"E3VAL","Humidity":"E3VAL"}]}
```

This way, when receiving message 2 (two), for example, there is a call to Block Tag's **OnRead** event for each set, according to the next example.

```
Sub [Sensor-001_OnRead]()
  Application.Trace Item("Temperature").Value
  Application.Trace Item("Humidity").Value
End Sub
```

Example of a log:

```
910         2021-08-05 10:06:15.826              0x3FF8  0x3210  APPTRACE          11              14
911         2021-08-05 10:06:15.826              0x3FF8  0x3210  APPTRACE          12              14
912         2021-08-05 10:06:15.826              0x3FF8  0x3210  APPTRACE          14              14
913         2021-08-05 10:06:15.826              0x3FF8  0x3210  APPTRACE          66              14
914         2021-08-05 10:06:15.826              0x3FF8  0x3210  APPTRACE          70              14
915         2021-08-05 10:06:15.826              0x3FF8  0x3210  APPTRACE          55              14
```

When declaring more **E3VAL** values before the repetition point, these values are repeated for each set. As an example, for a Template declared according to the next code.

```
{"MessageType":"E3VAL","TagData":[Repeat_E3VAL{"Temperature":"E3VAL","Humidity":"E3VAL"}]}
```

Users can have a Block Tag with 3 (three) Elements, type, temperature, and humidity, and the following values would be received at each **OnRead** event.

```
"Test";11;12 ' => First OnRead
"Test";14;66 ' => Second OnRead
"Test";70;55 ' => Third OnRead
```

CSV Template

A **CSV Template** allows processing values separated by semicolons. Values, timestamps, and qualities to extract from a message must be replaced by keywords. The next code contains an example of a message in **CSV** format.

```
10/18/2016 22:30:45; 22BAC23OOP10; "High Level";98,3;m
```

The next code contains an example of a Template for this format, to link to a Block Tag with four Block Elements.

```
TS_TEXT(%m/%d/%y %H:%M:%S);E3VAL;E3VAL;E3VAL;E3VAL
```

XML Template

An **XML Template** allows processing values in an MQTT message sent in **XML** format. Users must declare a message's format and replace fields into nodes to be transformed into values by the keywords. The next code contains an example of a message in **XML** format.

```
<updates>
  <update>
    <id>TEMP23A</id>
    <value>34.5</value>
    <quality>true</quality>
    <timestamp>10/18/2016 22:30:45</timestamp>
  </update>
</updates>
```

The next code contains a possible Template for the previous message.

```
<updates>
  <update>
    <id>V1</id>
    <value>V2</value>
    <quality>QL_BOOL</quality>
    <timestamp>TS_TEXT(%m/%d/%y %H:%M:%S)</timestamp>
  </update>
</updates>
```

Tag Configuration

This section contains the syntax for the configuration of Tags in **Elipse E3**, in **Elipse Power**, or in **Elipse Water** when using **Vanilla MQTT** or **MQTT Sparkplug B** standard.

Vanilla MQTT Tags

- **Item**: This field must use the next syntax

```
<Topic>[;<Template>[;<Param>]]
```

Or

```
<Topic>[(Deadband:<Absolute Deadband>)][(Deadband:<Relative Deadband>%)]
```

The ! (exclamation point) character can be used before the **<Topic>** to indicate an exception to the **Associated Topic Template** option.

In which:

- **<Topic>**: Address of an item or topic on a client's database. According to MQTT standard, each item on a client's database can correspond to a free data set, transmitted in **Text** format. Topics can be organized in a tree, and in this case users must use a slash mark to separate levels, according to the next example

```
"station12/pump01/pressure1"
"station12/pump01/pressure2"
```

A client can also use a number sign (#) and a plus sign (+) to address topics. In the first case, it must be added to a response's request all items containing in its name the current item or the children of the current item. For example, the value "station12#" returns the next values.

```
"station12/pump01"
"station12/pump01/pressure1"
"station12/pump01/pressure2"
"station12/pump02"
"station12/pump02/pressure1"
"station12/pump02/pressure2"
```

In the second case, it must be added to a response's request all items on the same level of the current item. For example, the value "station12/pump01/+" returns the next values.

```
"station12/pump01/pressure1"
"station12/pump01/pressure2"
```

- **<Template>**: Informs the name of a Template used as a model to interpret data from a message, with the purpose of mapping fields in a message for Tags (timestamp, quality, and values). This Template must be registered on the **Templates** tab
- **<Param>**: Informs the name of one of the Template's parameters, if this Tag must receive only one of the parameters
- **<Absolute Dead Band>**: Dead band as an absolute value, applied when writing numerical values

- **<Relative Dead Band>**: Dead band as a relative value, applied when writing numerical values

The next table contains examples of addressing messages.

Examples of addressing

ITEM	DESCRIPTION
"station12/pump01"	In this case the message read or written for the item is a **Text**-type Tag (**String**) without any processing
"station12/pump01;pumpdata"	Applies a **pumpdata** Template to the content of **station12/pump01**. If that Template describes more than one value, users must use a Block Tag in which each Block Element is a value from that Template
"station12/pump01;pumpdata;V1"	Same case as the previous one, but retrieving only the first value of the Template and linking it to a Tag, which does not need to be a Block Tag. In this case the Template must describe the **V1** value. For more information, please check the **Template** tab on topic **Driver's Configuration Parameters**
"station12/pump01(Deadband:0.5)"	If the current value of this topic is a number and a new number value is written to this Tag, then this new value is only published if the difference between the current value and the new value is equal to or greater than 0.5
"station12/pump01(Deadband:5%)"	If the current value of this topic is a number and a new number value is written to this Tag, then this new value is only published if the difference between the current value and the new value is equal to or greater than 5% of the old value
"station12/pump01(Deadband:0.5)(Deadband:5%)"	If the current value of this topic is a number and a new number value is written to this Tag, then this new value is only published if the difference between the current value and the new value is equal to or greater than 0.5 or equal to or greater than 5% of the old value

NOTE

When a dead band is configured directly in a Tag, this Tag must not be associated to a Template. In this case, the dead band configured on the **Publish** tab of this Driver's configuration window is ignored for this Tag.

To overcome an absolute dead band, the next condition must be met.

```
|(Current Value) - (New Value)| ≥ (Absolute Deadband)
```

To overcome a relative dead band, the next condition must be met.

```
|(Current Value) - (New Value)| ≥ |(Current Value)| * (Relative Deadband) / 100%
```

If a Tag is configured simultaneously with both absolute and relative dead bands, meeting any of the previous conditions triggers the publication of the value.

If an information appears in more than one message and a Tag must receive values from all those messages, users can specify an array of addresses, according to the next syntax.

```
["Topic1;Template1;Param1","Topic2;Template2;Param2","Topic3;Template3;Param3"]
```

For example, suppose that a device sends a message **1** that contains values **A**, **B**, and **C** and a message **2** that contains values **D**, **E**, **F**, and **B** again, in this order. If users want a Tag to receive **B** values coming from any of the messages, configure the array in the next format.

```
["device023/msg1;template1;V2","device023/msg2;template2;V4"]
```

In this case, the second value configured in **template1**, received in topic **device023/msg1**, and the fourth value configured in **template2**, received in topic **device023/msg2**, are sent to this Tag.

NOTE

Users can repeat topics, such as two or more Tags using the same topic, but one of the other parameters, *template* or *param*, must be different.

Examples of Tags:

```
Tag1: "device023/msg1" - OK
Tag2: "device023/msg1;T1" - OK
Tag3: "device023/msg1;T1;V2" - OK
Tag4: "["device023/msg1;T1;V2","device023/msg1;T1;V3"] - Not accepted. The first element repeats
Tag3
```

Publishing Values

When this Driver needs to publish messages to other clients, perform a Tag writing. If a Tag does not have a linked Template, then the informed value is sent to the topic. In this case, if it is a number value, this value is converted to text. If it is already a text, it is sent unchanged.

If a Tag contains a linked Template, then the declared variables inside that Template are filled with the values from the Tag and then sent. Notice that, when using a simple Tag, the Template must have only one variable declared. If more than one variable is needed, use a Block Tag, in which each Block Element corresponds to a variable inside the Template, in the same order.

To execute a writing to a Block Tag, the application must write the whole Block Tag, by using the **Write** method in a script. To do so, first configure the value of each one of the Elements and then write the whole Block Tag. The **AllowRead** and **AllowWrite** properties must be configured with the False value. The next code contains an example of writing a Template with 3 (three) variables.

```
Set Block1 = Application.GetObject("DriverMQTT").Item("Block1")
Block1.Item("Elemento1").Value = 123
Block1.Item("Elemento2").Value = "Abnormal Situation"
Block1.Item("Elemento3").Value = 555
Block1.Write 1 ' synchronous writing, 2 = asynchronous writing
```

MQTT Sparkplug B Tags

It is recommended to use the Tags available on the **Tag Browser** window automatically assembled during the import process. However, if any metric was not discovered or needs to be configured manually, the Tags follow the next pattern:

- **Item**: This field must follow the syntax indicated next

```
<Group>/<Node>;<Metric>
```

Or

```
<Group>/<Node>/<Device>;<Metric>
```

In which:

- **<Group>**: Identifier for a **Sparkplug** group (*Group ID*), which logically groups nodes in an MQTT Broker
- **<Node>**: Identifier of a **Sparkplug** node (*Node ID*), which represents a device or gateway that publishes or subscribes data in a group
- **<Device>**: Identifier of a **Sparkplug** device (*Device ID*), optionally used to represent a specific device connected to a node
- **<Metric>**: Name of a **Sparkplug** metric, or property, that is read or written, as defined in protocol's payload

NOTE

Tag values are updated by using callbacks when received by the **NDATA** or **DDATA** topics from **Sparkplug B** standard. When writing to a Tag, the content is analyzed, or parsed, converted to the format expected by the recipient, and then sent via **NCMD** (*Node Command*) or **DCMD** (*Device Command*) command.

Eclipse Distribution License Version 1.0

This project includes Paho MQTT C Client DLLs, distributed under the Eclipse Distribution License v1.0.

Driver's Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.42	03/24/2026	A. Fetzner	• Fixed a failure in SSL/TLS connection when using a

VERSION	DATE	AUTHOR	COMMENTS
			password-protected private key (*Case 39619*).
1.0.41	11/11/2025	A. Fetzner	• Implemented support for the **Sparkplug B** standard, acting as *Primary Application* (*Case 30445*).
1.0.40	09/03/2025	A. Fetzner M. Ludwig	• Fixed a high usage of CPU that occurred when the connection to a Broker failed or was interrupted (*Case 38571*). • Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37998*).
1.0.39	05/14/2025	A. Fetzner	• Updated the properties window of this Driver and adjustments to the reconnection settings with a Broker (*Case 37814*). • Fixed a GPF (*General Protection Failure*) that occurred when a Broker initiated a disconnection (*Case 37794*). • Added new settings to *Last Will* messages received by this Driver, allowing them to be customized individually through dedicated Templates (*Case 37678*).
1.0.38	04/02/2025	A. Fetzner	• Fixed a memory leak during data publishing (*Case 37217*). • Implemented support for version **5** of MQTT protocol. OpenSSL protocol was updated to version **3** and also fixed a connection error with HiveMQ Brokers (*Case 32094*).
1.0.37	01/20/2025	M. Salvador	• Created an option on the **Publish** tab to define whether texts (**Strings**) are sent in **UTF-8** format. The same option is used during reception to process **Strings** received from **UTF-8** format to **ASCII** format, thus keeping consistency between sending and receiving (*Case 34202*).

VERSION	DATE	AUTHOR	COMMENTS
			• Created an option to return error to Tags when a message does not match the configured Template (*Case 34301*). • Now, when writing to a Tag, this Driver parses based only on the Template referenced in the same Tag, not on other Templates that may be referenced for the same topic (*Case 34797*). • Created an option **Return Error for Tags With Response Mismatch** for when a received message is not in the same order defined by the Template linked to a Tag (*Case 35658*). • Created an option using the *N* parameters to inform the value of the **QoS** and **Retain** properties per Tag (*Case 35659*). • Fixed a general failure when using the SSL protocol while receiving a large message (*Case 37042*).
1.0.30	12/22/2021	M. Salvador	• Templates and messages can now start as arrays (*Case 31200*). • Added read-only support for repeating objects inside arrays (*Case 31239*). • Message publishing now works with more than one Template value and block writings are allowed (*Case 31242*).
1.0.22	05/13/2021	G. Beal	• Fixed the processing of **TS_UNIX**, **TS_UNIX_TZ_SECONDS+** and **TS_UNIX_TZ_SECONDS-** keywords for Templates in **JSON** format (*Case 30878*).
1.0.20	04/01/2021	G. Beal	• Fixed an issue that delayed the initial connection to a Broker (*Case 30683*).
1.0.18	03/01/2021	G. Beal	• Added a new individual dead band configuration for

VERSION	DATE	AUTHOR	COMMENTS
			each Tag (*Case 30517*).
1.0.15	06/16/2020	M. Salvador	• Added a relative dead band (**Deadband %**) configuration, applied to all numerical Tags (*Case 28886*).
1.0.14	05/04/2020	G. Beal	• Implemented a new window for editing Templates (*Case 28650*). • Modified the comparison between messages and Templates in **JSON** format so that the order of name and value pairs within an object does not affect their match (*Case 28678*).
1.0.8	12/05/2019	M. Ludwig	• Added a syntactic interpretation in **JSON** format only with attribute names that are identical with Template. Non-existing attributes are ignored (*Case 26316*).
1.0.7	02/01/2019	M. Salvador	• Fixed problems with performance and time-out checks. • Updated the timestamp when an item is not linked to a Template. • Added support for several references to a single item with different Templates.
1.0.1	11/07/2018	M. Salvador	• Initial version of this Driver.

Headquarters
Rua Mostardeiro, 322/Cj. 902, 1001 e 1002
90510-002 — Porto Alegre — RS
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

Branch in Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist.
807 — Kaohsiung City — Taiwan
Phone: (+886 7) 323-8468
Fax: (+886 7) 323-9656
E-mail: evan@elipse.com.br

Check our website for information about a representative in your country.

www.elipse.com.br

kb.elipse.com.br
forum.elipse.com.br
www.youtube.com/elipsesoftware
elipse@elipse.com.br

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)