MQTT Driver

Filename	MQTT.dll
Manufacturer	
Devices	Clients and Brokers compatible with specifications 3.1 and 3.1.1 of MQTT protocol
Protocol	MQTT over Ethernet TCP/IP and TLS/SSL
Version	1.0.30
Last Update	22/12/2021
Platform	Win32
Dependencies	IOKit v2.00 and the OpenSSL library
Superblock Readings	No
Level	31312

Introduction

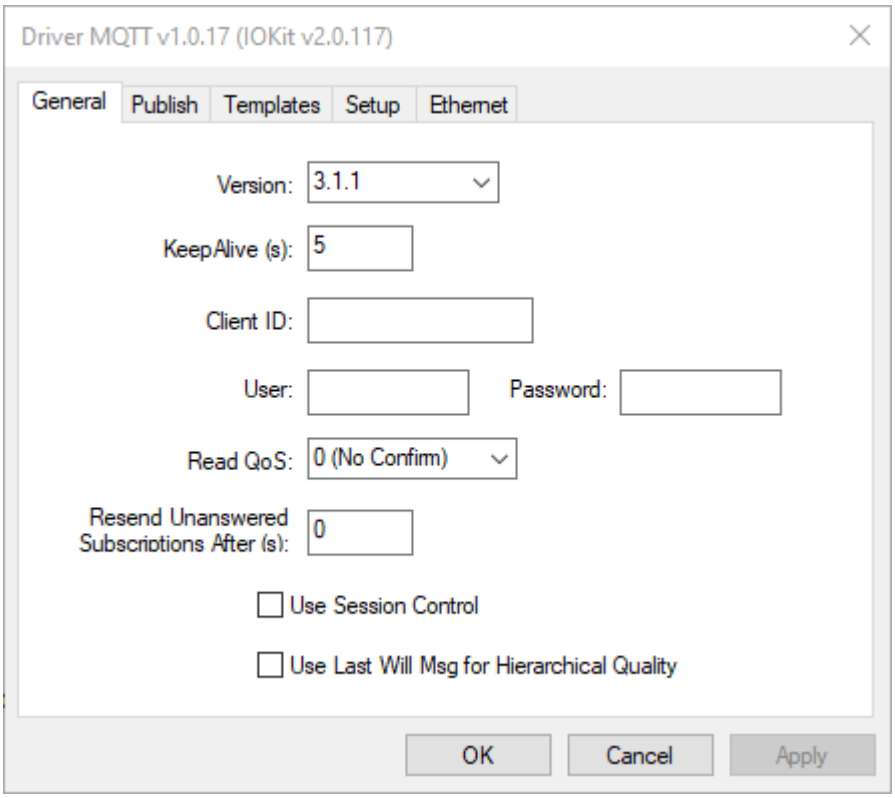
The MQTT Driver is a client that implements version 3.1.X of MQTT protocol by using an MQTT Broker, which is not provided.

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

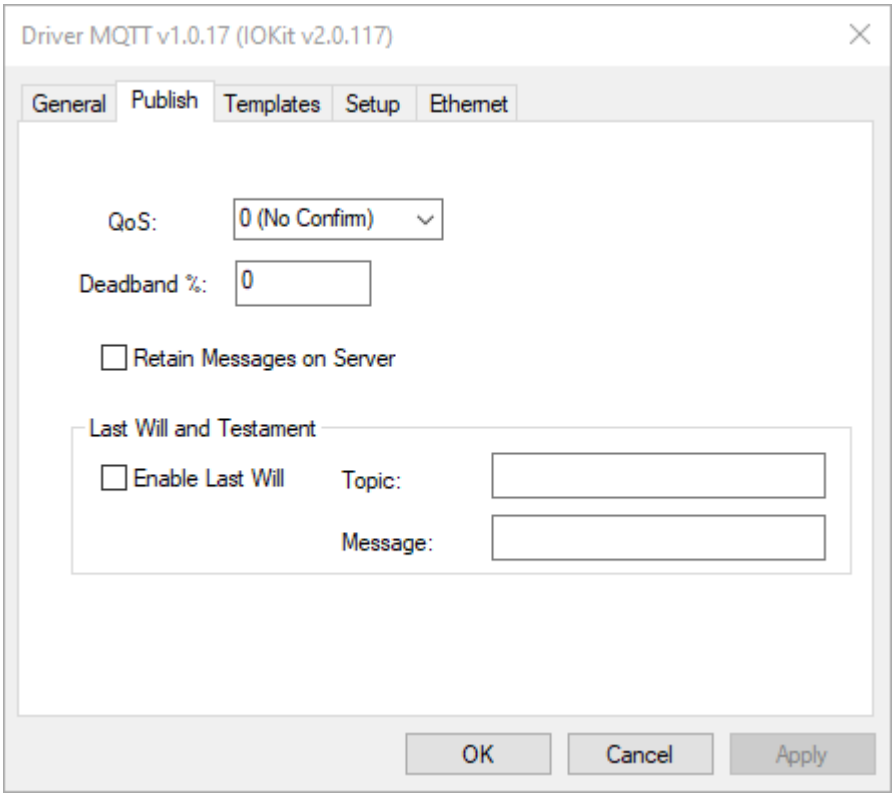
OPTION	DESCRIPTION
Version	Protocol version. Available options are **3.1** or **3.1.1**
KeepAlive (s)	Time, in seconds, that a client sends a **ping** message in MQTT protocol (**PingReq**) to check whether a connection with a Broker is still active. Default value of this option is 5 (five) seconds
Client ID	Name used by this client as an identifier to 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 along with the user name
Read QoS	Tells which QoS is used to request Tag subscriptions (QoS 0, 1, or 2). The Broker matches the requested QoS with the published QoS and then sends a minimum compatible data
Resend Unanswered Subscriptions After (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
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** and **2** messages sent by other clients can be recovered
Use Last Will Msg for Hierarchical Quality	When selecting this option, if this Driver receives the message configured in the **Message** option of the **Last**

OPTION	DESCRIPTION
	Will and Testament group on the **Publish** tab, then this message's topic is used to hierarchically define other Tags configured with a bad quality

Assuming that a Last Will message of a device is configured with an "Error" expression for the **Device527** item, when receiving that message all addressed Tags containing the expression "Device527" starting at the beginning of their address have their quality set as bad, that is, the value of the **Quality** property is set to 20, 24, or 28 (invalid). Examples of device addresses:

```
Device527/Temperature
Device527/AnalogInput01
Device527
```

For these Tags to return to a good quality (192), a topic must receive any value different from the message configured in the **Last Will and Testament** option.



Publish tab

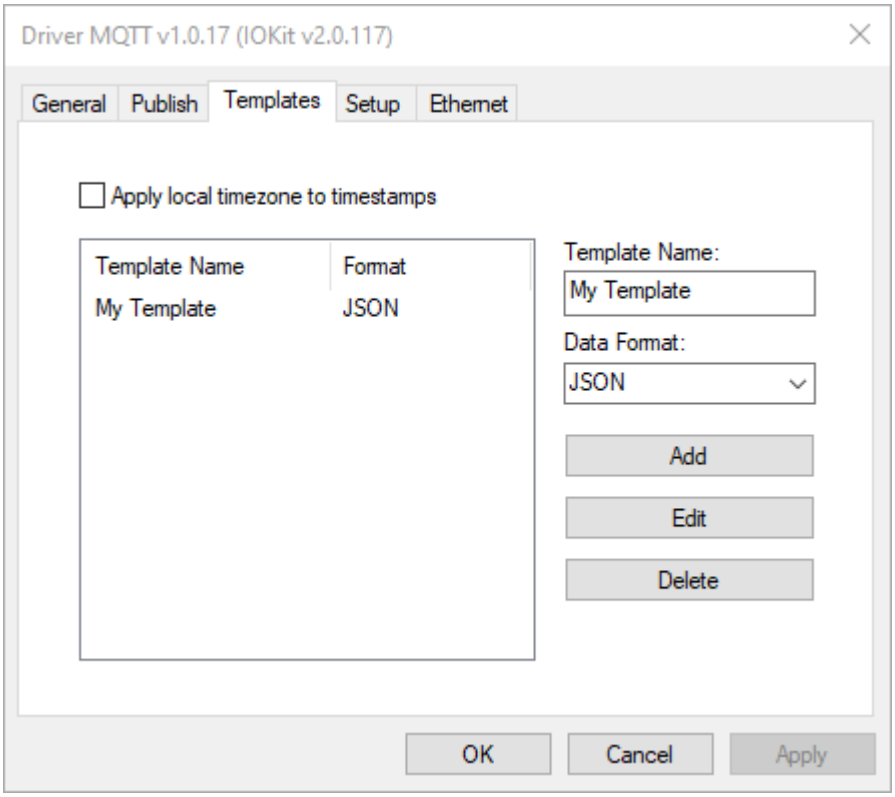
The available options on the **Publish** tab are described on the next table.

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 delivers), **QoS 1**: At least one or more delivers, or **QoS 2**: Exactly one deliver
Deadband %	Value for the relative dead band, applied when writing a numeric value to any Tag with no Template specified

OPTION	DESCRIPTION
Retain messages on Server	Selecting this option instructs a Broker to keep this client's messages 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 the Broker
Enable Last Will	When enabling this option, the server must keep in memory a topic and a message for the 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 disconnection (Last Will), such as "ELIPSE\goodbye"
Message	Message linked to a topic, such as "Goodbye"

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 Tags or Block Tags.



Templates tab

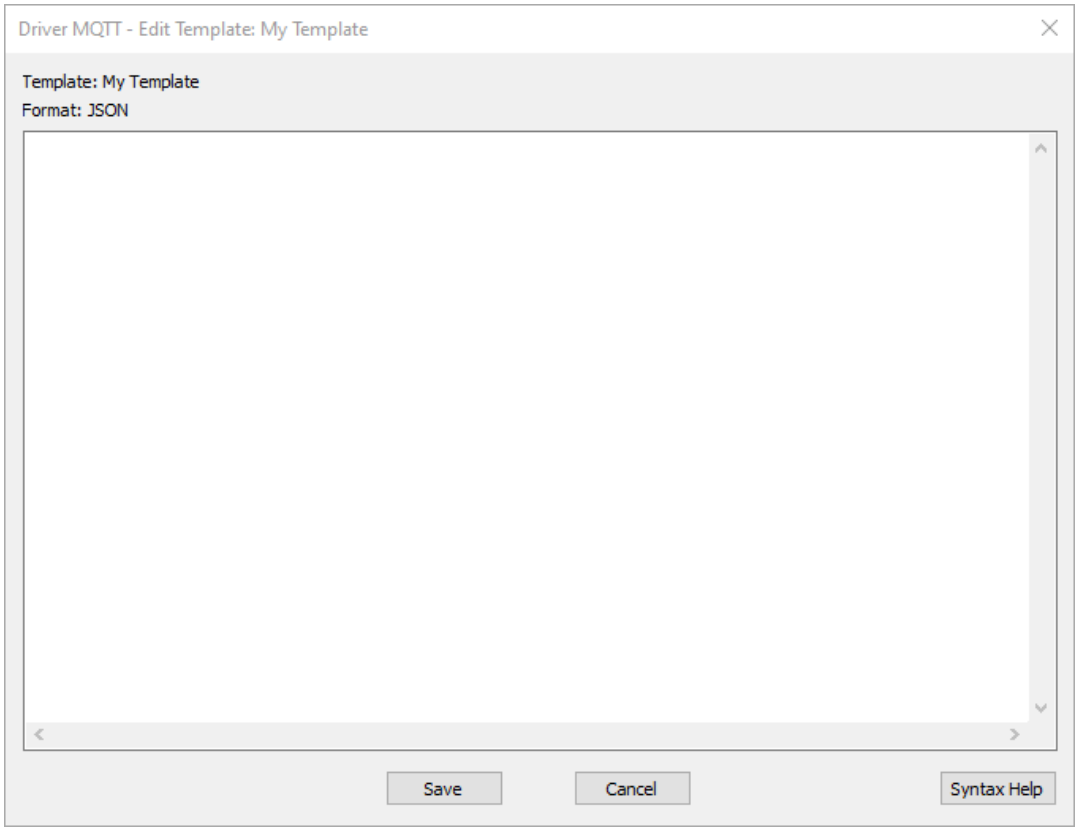
The available options on the **Templates** tab are described on the next table.

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 time zone used by the operating system. Otherwise, the received timestamps are displayed in UTC time zone

OPTION	DESCRIPTION
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 the 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**) along 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 **Ethernet** tab, users must configure an IP address and a TCP/IP port of an MQTT Broker. MQTT standard establishes TCP/IP port 1883 as a default port for direct unencrypted connections. TCP/IP port 8883 is defined as a default port for encrypted TLS/SSL connections without a certificate, and TCP/IP port 8884 is defined as a default port for TLS/SSL connections with a certificate.

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 Array, to indicate that the repeating members shall be processed independently, returning a read operation for each appearance. For more information, please see the section JSON Template .

Available options for the TS_TEXT keyword

OPTION	DESCRIPTION
%a	Short weekday
%A	Full weekday
%b	Short name of the month
%B	Full name of the month
%C	Century
%d	Day of the month, starting with 0 (zero)
%e	Day of the 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

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 two structures, 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. 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}
```

To extract the content of the previous message, a possible Template would be the following:

```
{"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
}
```

A possible Template for the previous example would be the following:

```
{
  "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 message and templates shall be JSON objects or arrays, starting with { or [.

Value/Object Repetition inside Arrays

Arrays can be used to receive a variable number of elements. Let´s take as an example the messages below:

Message 1:

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

Message 2:

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

We can see that inside the array, it is possible to find 1, 2, 3 or N objects containing temperature and humidity, and it might be interesting if a "Test" Block Tag, with 2 elements (Temperature and Humidity) can receive a reading at each object found.

To achieve this, you can declare the template with the keyword Repeat_E3VAL right after the '[' symbol that marks the array start. Example:

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

In this way, when we receive message 2, for example, we will receive a call at the block **OnRead** event for each object found. Example:

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

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
```

If more keywords E3VAL were declared at the template before the repetition point, those previous values will be repeated for each set. As an example, if we have declated the template as

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

we could have the block with 3 elements (Type, Temperature and Humidity), so we would receive the following values at each OnRead event:
"Test";11;12 '=> 1st OnRead
"Test";14;66 ' => 2nd OnRead
"Test";70;55 '=> 3rd 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. An example of a message in **CSV** format is the following:

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

An example of a Template for this format, to link to a Block Tag with four Block Elements, would be the following:

```
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 in values by the keywords. An example of a message in **XML** format is the following:

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

A possible Template for the previous message would be the following:

```
<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

Use the following syntax to configure Tags in **E3** or **Elipse Power**:

- **Device**: Not used
- **Item**: This field must use the following syntax

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

Or

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

Where:

- **<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. 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 its children. For example, for a value "station12#", the following values are returned:

```
"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. For example, for a value "station12/pump01/+", the following values are returned:

```
"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, in case this Tag must receive only one of the parameters
- **<Absolute Deadband>**: Absolute value of dead band, applied when writing numerical values
- **<Relative Deadband>**: Relative value of dead band, applied when writing numerical values

The next table contains examples of message addressing.

Examples of message 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 a 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 value **V1**. 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, then this new value is only published if the difference between those values 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, then this new value is only published if the difference between those values is equal to or greater than 5% of the current 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, then this new value is only published if the difference between those values is equal to or greater than 0.5 or equal to or greater than 5% of the current 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 in the **Publish** tab of the configuration window is bypassed.

To overcome an absolute dead band, the following condition must be met:

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

To overcome a relative dead band, the following condition must be met:

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

If a Tag has both absolute and relative dead bands, meeting any of the previous conditions triggers the publication.

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 in **JSON** format, according to the following syntax.

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

For example, let's suppose a device sends two types of messages. Message **1** contains values **A**, **B**, and **C** and Message **2** contains values **D**, **E**, **F**, and **B** again, in this order. If users want a Tag to receive **B** values coming from both 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
```

Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **MQTT** Driver.

Driver Configuration

I/O Interface configuration is performed on a Driver's configuration dialog box. To access the configuration of this dialog box in **E3** (version 1.0), follow these steps:

1. Right-click a Driver object (IODriver).
2. Select the **Properties** item on the contextual menu.
3. Select the **Driver** tab.
4. Click **Other parameters**.

In **E3** version 2.0 or later, click **Configure driver** on a Driver's toolbar. In **Elipse SCADA**, follow these steps:

1. Open the Organizer.
2. Select a Driver on Organizer's tree.
3. Click **Extras** on the **Driver** tab.

Currently, an I/O Interface allows opening only one connection for each Driver. This means that, if users want to access two serial ports, they must add two Drivers to an application and then configure each Driver for each serial port.

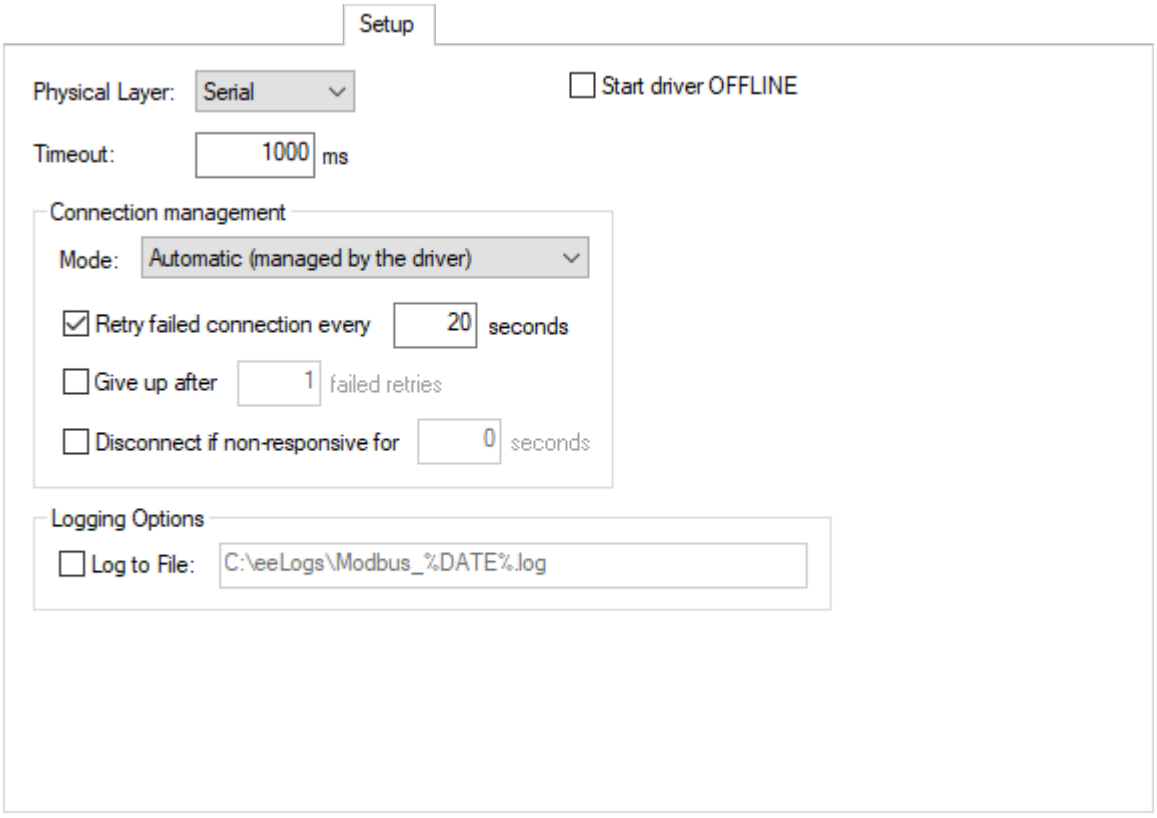
Configuration Dialog Box

The I/O Interfaces dialog box allows configuring the I/O connection used by a Driver. This dialog box contains the **Setup**, **Serial**, **Ethernet**, **Modem**, and **RAS** tabs, described on the next topics. If a Driver does not implement a specific I/O connection, its corresponding tab is not available for configuration. Some Drivers may contain additional tabs, specific for each Driver, on the configuration dialog box.

Setup Tab

The **Setup** tab contains Driver's general configurations. This tab is divided into the following groups:

- **General configurations**: Configurations of Driver's physical layer, time-out, and initialization mode
- **Connection management**: Configurations on how the I/O Interface keeps a connection and which recovery policy is used on failure
- **Logging options**: Controls the generation of log files



Setup tab

General options on the Setup tab

OPTION	DESCRIPTION
Physical Layer	Select the physical layer on a list. Available options are **Serial**, **Ethernet**, **Modem**, and **RAS**. The selected interface must be configured on its specific tab
Timeout	Configure a time-out, in milliseconds, for the physical layer. This is the amount of time an I/O interface waits to receive any byte from reception's buffer
Start driver OFFLINE	Select this option so that the Driver starts in **Offline** mode or stopped. This means that the I/O interface is not created until this Driver is configured to **Online** mode by using a Tag in an application. This mode enables a dynamic configuration of an I/O interface at run time

Options on the Connection management group

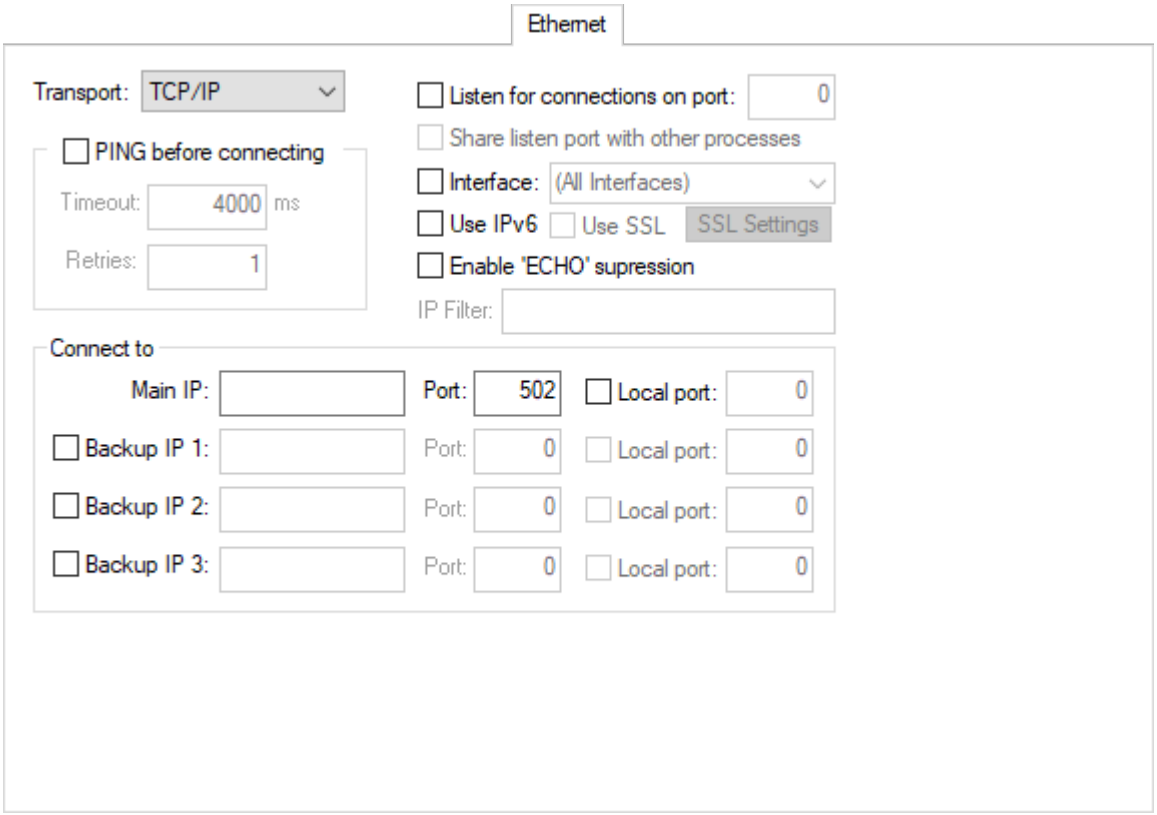
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection. Please check topic **Driver Statuses** for more details
Retry failed connection every ... seconds	Select this option to enable a Driver's connection retry in a certain interval, in seconds. If the **Give up after failed retries** option is not selected, the Driver keeps retrying until a connection is performed, or until the application is stopped
Give up after ... failed retries	Enable this option to define a maximum number of connection retries. When the specified number of consecutive connection retries is reached, the Driver goes to the **Offline** mode, assuming that a hardware problem was detected. If a Driver establishes a successful connection, the number of unsuccessful retries is cleared. If this new connection is lost, then the retry counter starts at zero
Disconnect if non-responsive for ... seconds	Enable this option to force a Driver to disconnect if no byte was received by the I/O interface during the specified time-out, in seconds. This time-out must be greater than the time-out configured in the **Timeout** option

Options on the Logging Options group

OPTION	DESCRIPTION
Log to File	Enable this option and configure the name of a file to write a log. Log files can be large, so use this option for short periods of time, only for testing and debugging purposes. If the **%PROCESS%** macro is used in the log file name, it is replaced by the ID of the current process. This option is particularly useful when using several instances of the same Driver in **E3**, thus allowing each instance to generate a separate log file. For example, when configuring this option as **c:\e3logs\drivers\sim_%PROCESS%.log**, a file named **c:\e3logs\drivers\sim_00000FDA.log** is generated for process **0FDAh**. Users can also use the **%DATE%** macro in the file name. In this case a log file is generated every day (in the format **aaaa_mm_dd**). For example, when configuring this option as **c:\e3logs\drivers\sim_%DATE%.log**, a file named **c:\e3logs\drivers\sim_2005_12_31.log** is generated in 12/31/2005 and a file named **c:\e3logs\drivers\sim_2006_01_01.log** is generated in 01/01/2006

Ethernet Tab

Use this tab to configure parameters of an **Ethernet** Interface. These parameters, except port configurations, must also be configured for use in the **RAS**.



Ethernet tab

Available options on Ethernet tab

OPTION	DESCRIPTION
Transport	Select **TCP/IP** for a TCP socket (stream). Select **UDP/IP** to use a UDP socket (connectionless datagram)
Listen for connections on port	Use this option to wait for new connections in a specific IP port, common in Slave Drivers. If this option remains unselected, the Driver connects to the address and port specified in the **Connect to** option
Share listen port with other processes	Select this option to share the listen port with other Drivers and processes
Interface	Select the local network interface, identified by its IP address, that is used by the Driver to establish and receive connections, or select the **(All Interfaces)** item to use any local network interface
Use IPv6	Check this option to force the Driver to use IPv6 addresses on all Ethernet connections. If this option is unchecked the Driver will work with IPv4 addresses
Enable 'ECHO' suppression	Enable this option to remove the echo from received data. An echo is a copy of sent data, which can be returned before a reply message
IP Filter	List of restricted or allowed IP addresses from where a Driver accepts connections (Firewall). Please check the

OPTION	DESCRIPTION
	IO.Ethernet.IPFilter property for more details
PING before connecting	Enable this option to execute a **ping** command, that is, check whether a device can be reached on a network, for a device before trying a socket connection. This is a quick way of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from the **ping** command. Users must use the **ping** command to check the normal reply time, configuring this option for a value above that average. Usually this value can be configured between 1000 and 4000 milliseconds, that is, between one and four seconds • **Retries**: Number of retries of a **ping** command, not counting the first attempt. If all attempts fail, then the socket connection is aborted

Available options on the Connect to group

OPTION	DESCRIPTION
Main IP	Type the IP address of the remote device. Users can use an IP address separated by dots, as well as a URL. In case of a URL, the 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 IP port of the remote device, between 0 (zero) and 65535
Local port	Select this option to use a fixed local IP port when connecting to a remote device
Backup IP 1, 2, and 3	Indicate here the IP address, the IP port, and the fixed local IP port of up to 3 (three) backup addresses of a remote device

General Configurations

This section contains information about the configuration of general **I/O Tags** and **Properties** of I/O Interfaces.

I/O Tags

General I/O Interfaces Tags (N2/B2 = 0)

The Tags described next are provided for all supported I/O Interfaces.

IO.IOKitEvent

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1 (minus one)
B2 Parameter	0 (zero)
B3 Parameter	0 (zero)
B4 Parameter	1 (one)
Size Property	4 (four)
ParamItem Property	IO.IOKitEvent

This Block returns Driver events generated by several sources in I/O Interfaces. The **TimeStamp** property of this Block represents the moment this event occurred. The Block Elements are the following:

- **Element 0**: Type of event. Possible values are **0**: Information, **1**: Warning, or **2**: Error
- **Element 1**: Source of event. Possible values are **0**: Driver (specific of a Driver), **-1**: IOKit (generic events of I/O Interfaces), **-2**: **Serial** Interface, **-3**: **Modem** Interface, **-4**: **Ethernet** Interface, or **-5**: **RAS** Interface
- **Element 2**: Error number, specific for each source of event
- **Element 3**: Event message, a **String** specific for each event

NOTE

A Driver keeps a maximum number of 100 events internally. If additional events are reported, older events are discarded.

IO.PhysicalLayerStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	2 (two)
String Configuration	IO.PhysicalLayerStatus

This Tag indicates the status of a physical layer. Its possible values are the following:

- **0**: Physical layer stopped, that is, the Driver is in **Offline** mode, the physical layer failed when initializing, or exceeded the maximum number of reconnection attempts
- **1**: Physical layer started but not connected, that is, the Driver is in **Online** mode but the physical layer is not connected. If the **Connection management** option is configured as **Automatic**, the physical layer can be connecting,

disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured as **Manual**, then the physical layer remains in this status until forced to connect

- **2**: Physical layer connected, that is, the physical layer is ready for use. This **DOES NOT** mean the device is connected, only the access layer is working

IO.SetConfigurationParameters

Type of Tag	Block Tag
Type of Access	Read-Only
B1 Parameter	-1 (minus one)
B2 Parameter	0 (zero)
B3 Parameter	0 (zero)
B4 Parameter	3 (three)
Size Property	2 (two)
ParamItem Property	IO.SetConfigurationParameters

Use this Tag to change any property of a Driver's configuration dialog box at run time.

This Tag works only while a Driver is in **Offline** mode. To start a Driver in **Offline** mode, select the **Start driver OFFLINE** option on the Driver's configuration dialog box. Users can write to a PLC Tag or to a Block Tag containing the parameters to change. Writing individual Block Elements is not supported, the whole Block must be written at once.

In **Elipse SCADA**, users must use a Block Tag. Every parameter to configure uses two Block Elements. For example, if users want to configure three parameters, then the size of the Block must be 6 (3 × 2). The first Element is the property's name, as a **String**, and the second Element is the property's value, according to the next example.

```
// 'Block' must be a Block Tag with automatic reading,
// scan reading, and automatic writing disabled.
// Configure all parameters
Block.element001 = "IO.Type" // Parameter 1
Block.element002 = "Serial"
Block.element003 = "IO.Serial.Port" // Parameter 2
Block.element004 = 1
Block.element005 = "IO.Serial.BaudRate" // Parameter 3
Block.element006 = 19200
// Writes the whole Block
Block.Write()
```

When using **E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use Driver's **Write** method to send all parameters to the Driver, without creating a Tag, according to the next example.

```
Dim arr(6)
' Configure all array elements
arr(1) = "IO.Type"
arr(2) = "Serial"
arr(3) = "IO.Serial.Port"
arr(4) = 1
arr(5) = "IO.Serial.BaudRate"
arr(6) = 19200
' There are two methods to send parameters
' Method 1: Using an I/O Tag
tag.WriteEx arr
' Method 2: Without using a Tag
Driver.Write -1, 0, 0, 3, arr
```

A variation of the previous example uses a bidimensional array.

```
Dim arr(10)
' Configure all array elements. Notice the array was resized
' to 10 elements. Empty array elements are ignored by a Driver
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
Driver.Write -1, 0, 0, 3, arr
```

A Driver does not validate parameter names or passed values, therefore be careful when writing parameters and values. The **Write** method fails if the configuration array is incorrectly created. Users can check Driver's log or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of the error.

```
Dim arr(10), strError
arr(1) = Array("IO.Type", "Serial")
arr(2) = Array("IO.Serial.Port", 1)
arr(3) = Array("IO.Serial.BaudRate", 19200)
If Not Driver.WriteEx -1, 0, 0, 3, arr, , , strError Then
  MsgBox "Failed configuring Driver parameters: " + strError
End If
```

IO.WorkOnline

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	4 (four)
String Configuration	IO.WorkOnline

This Tag informs the current status of a Driver and allows starting or stopping the physical layer. Possible values are the following:

- **0 - Driver Offline**: Physical layer is closed or stopped. This mode allows a dynamic configuration of a Driver's parameters using the **IO.SetConfigurationParameters** Tag
- **1 - Driver Online**: Physical layer is open or executing. While in **Online** mode, the physical layer can be connected or disconnected and its current status can be checked on the **IO.PhysicalLayerStatus** Tag

In the next example, using **E3**, a Driver is configured to **Offline** mode, its COM port is changed, and then configured to **Online** mode again.

```
'Configure to Offline mode
Driver.Write -1, 0, 0, 4, 0
'Change port to COM2
Driver.Write -1, 0, 0, 3, Array("IO.Serial.Port", 2)
'Configure to Online mode
Driver.Write -1, 0, 0, 4, 1
```

The **Write** method can fail when configuring a Driver to **Online** mode, that is, writing the value 1 (one). In this case, the Driver remains in **Offline** mode. The cause of failure can be:

- Type of physical layer incorrectly configured, probably an invalid value was configured in the **IO.Type** property

- Driver may have run out of memory
- Physical layer probably did not create its working thread. Search the log file for a message "Failed to create physical layer thread!"
- Physical layer could not start. The cause of failure depends on the type of physical layer. It can be an invalid serial port number, a failure when starting Windows Sockets, or a failure when starting TAPI (modem), among others. This cause is recorded on the log file

IMPORTANT

Even if the configuration of a Driver to **Online** mode is successful, this does not necessarily mean the physical layer is ready to use, that is, ready to execute input and output operations with an external device. The **IO.PhysicalLayerStatus** Tag must be checked to ensure the physical layer is connected and ready for communication.

Properties

These are general properties of all supported I/O Interfaces.

IO.ConnectionMode

9 Controls the management mode of a Connection. Possible values are **0**: Automatic mode, where a Driver manages the connection or **1**: Manual mode, where an application manages the connection.

IO.GiveUpEnable

◩ When configured to True, defines a maximum number of reconnection attempts. If all reconnection attempts fail, the Driver enters the **Offline** mode. When configured to False, the Driver tries until a reconnection is successful.

IO.GiveUpTries

9 Number of reconnection attempts before this one is aborted. For example, if the value of this property is equal to 1 (one), a Driver tries only one reconnection when the reconnection is lost. If this one fails, a Driver enters the **Offline** mode.

IO.InactivityEnable

◩ Configure to True to enable and to False to disable inactivity detection. The physical layer is disconnected if inactive for a certain period of time. The physical layer is considered inactive only if it is capable of sending data but not capable of receiving it back.

IO.InactivityPeriodSec

9 Number of seconds to check for inactivity. If the physical layer is inactive for this period of time, it is disconnected.

IO.RecoverEnable

◩ Configure to True to enable a Driver to recover lost connections and to False to leave a Driver in **Offline** mode when a connection is lost.

IO.RecoverPeriodSec

9 Delay time between two connection attempts, in seconds.

NOTE

The first reconnection is executed immediately after a connection is lost.

IO.StartOffline

Configure to True to start a Driver in **Offline** mode and to False to start a Driver in **Online** mode.

NOTE

It is pointless to change this property at run time, as it can only be changed when a Driver is already in **Offline** mode. To configure a Driver in **Online** mode at run time, write the value 1 (one) to the **IO.WorkOnline** Tag.

IO.TimeoutMs

Defines a time-out for the physical layer, in milliseconds. One second is equal to 1000 milliseconds.

IO.Type

Defines the type of physical interface used by a Driver. Possible values are the following:

- **N or None**: Does not use a physical interface, that is, a Driver must provide a customized interface
- **S or Serial**: Uses a local serial port (COM*n*)
- **M or Modem**: Uses a local modem, internal or external, accessed via TAPI (*Telephony Application Programming Interface*)
- **E or Ethernet**: Uses a TCP/IP or UDP/IP socket
- **R or RAS**: Uses a **RAS** (*Remote Access Server*) Interface. A Driver connects to a RAS device using the **Ethernet** Interface and then sends an **AT** (*dial*) command

Statistical Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of I/O Interfaces statistics.

I/O Tags

Tags of I/O Interface statistics (N2/B2 = 0)

The Tags described next display statistics for all I/O Interfaces.

IO.Stats.Partial.BytesRecv

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1101
Configuration by String	IO.Stats.Partial.BytesRecv

This Tag returns the number of bytes received in the current connection.

IO.Stats.Partial.BytesSent

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1100
Configuration by String	IO.Stats.Partial.BytesSent

This Tag returns the number of bytes sent through the current connection.

IO.Stats.Partial.TimeConnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1102
Configuration by String	IO.Stats.Partial.TimeConnectedSeconds

This Tag returns the number of seconds a Driver is connected in the current connection or 0 (zero) if a Driver is disconnected.

IO.Stats.Partial.TimeDisconnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1103
Configuration by String	IO.Stats.Partial.TimeDisconnectedSeconds

This Tag returns the number of seconds a Driver is disconnected since the last connection ended or 0 (zero) if a Driver is connected.

IO.Stats.Total.BytesRecv

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1001
Configuration by String	IO.Stats.Total.BytesRecv

This Tag returns the number of bytes received since a Driver was loaded.

IO.Stats.Total.BytesSent

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1000
Configuration by String	IO.Stats.Total.BytesSent

This Tag returns the number of bytes sent since a Driver was loaded.

IO.Stats.Total.ConnectionCount

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1004
Configuration by String	IO.Stats.Total.ConnectionCount

This Tag returns the number of connections a Driver already established, successfully, since it was loaded.

IO.Stats.Total.TimeConnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1002
Configuration by String	IO.Stats.Total.TimeConnectedSeconds

This Tag returns the number of seconds a Driver remained connected since it was loaded.

IO.Stats.Total.TimeDisconnectedSeconds

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	1003
Configuration by String	IO.Stats.Total.TimeDisconnectedSeconds

This Tag returns the number of seconds a Driver remained disconnected since it was loaded.

Properties

Currently, there are no properties defined specifically to display I/O Interface statistics at run time.

Ethernet Interface Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of an **Ethernet** Interface.

I/O Tags

Tags of an Ethernet Interface (N2/B2 = 4)

The Tags described next allow controlling and identifying an **Ethernet** Interface at run time and they are also valid when the **RAS** Interface is selected.

IMPORTANT
These Tags are available **ONLY** while a Driver is in **Online** mode.

IO.Ethernet.IPSelect

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	4 (four)
N4 Parameter	0 (zero)
String Configuration	IO.Ethernet.IPSelect

Indicates the active IP address. Possible values are **0**: The main IP address is selected, **1**: The first alternative or backup IP address is selected, **2**: The second alternative or backup IP address is selected, or **3**: The third alternative or backup IP address is selected.

If the **Ethernet** or **RAS** Interface is connected, this Tag indicates which one of the four configured IP addresses is in use. If the Interface is disconnected, this Tag indicates which IP address is used first on the next connection attempt.

During the connection process, if the active IP address is not available, the I/O Interface tries to connect using the next alternative IP address. If the connection with the alternative IP address works, it is configured as the active IP address (automatic switchover).

To force a manual switchover, write values from 0 (zero) to 3 (three) to this Tag. This forces a reconnection with the specified IP address (**0**: Main IP address or **1, 2, 3**: Alternative IP address) if the Driver is currently connected. If the Driver is disconnected, this Tag configures the active IP address for the next connection attempt.

IO.Ethernet.IPSwitch

Type of Tag	I/O Tag
Type of Access	Write-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	4 (four)
N4 Parameter	1 (one)
String Configuration	IO.Ethernet.IPSwitch

Any value written to this Tag forces a manual switchover. If the main IP address is active, then the Driver tries to connect to each one of the alternative or backup IP addresses and back to the main IP address until a connection is established.

If the Driver is disconnected, this Tag configures the active IP address for the next connection attempt.

Properties

These properties control the configuration of an **Ethernet** Interface.

NOTE

The **Ethernet** Interface is also used by the **RAS** Interface.

IO.Ethernet.AcceptConnection

◩ Configure to False if the Driver must not accept external connections, that is the Driver behaves as a master, or configure to True to enable the reception of connections, that is, the Driver behaves as a slave.

IO.Ethernet.BackupEnable[2,3]

◩ Configure to True to enable an alternative or backup IP address. If the reconnection attempt with the main IP address fails, the Driver tries to use an alternative IP address. Configure to False to disable its usage.

IO.Ethernet.BackupLocalPort[2,3]

9 Local port number to be used when connecting to an alternative IP address of a remote device. Used only if **IO.Ethernet.BackupLocalPortEnable** is equal to True.

IO.Ethernet.BackupLocalPortEnable[2,3]

◩ Configure to True to force the use of a specific local port when connecting to an alternative or backup IP address of a remote device or configure to False to use any available local port.

IO.Ethernet.BackupIP[2,3]

A Alternative or backup IP address of a remote device. Users can use a numerical address, as well as a device's host name, such as "192.168.0.7" or "SERVER2".

IO.Ethernet.BackupPort[2,3]

9 Port number of an alternative or backup IP address of a remote device, used with the **IO.Ethernet.BackupIP** property.

IO.Ethernet.IPFilter

A List with a comma-separated IPv4 or IPv6 addresses, which defines from which addresses a Driver accepts or blocks connections. Users can use asterisks, such as "192.168.*.*", or intervals, such as "192.168.0.41-50", in any part of IP addresses. To block an IP address or a range of IP addresses, use the tilde ("~") character at the beginning of the address. Examples:

- **192.168.0.24**: Accepts only connections from IPv4 address 192.168.0.24
- **192.168.0.41-50**: Accepts connections from IPv4 addresses from 192.168.0.41 to 192.168.0.50
- **192.168.0.***: Accepts connections from IPv4 addresses from 192.168.0.0 to 192.168.0.255
- **fe80:3bf:877::*:* (expands to fe80:03bf:0877:0000:0000:0000:*:*)**: Accepts connections from IPv6 addresses from fe80:03bf:0877:0000:0000:0000:0000:0000 to fe80:03bf:0877:0000:0000:0000:ffff:ffff
- **192.168.0.10, 192.168.0.15, 192.168.0.20**: Accepts connections from IPv4 addresses 192.168.0.10, 192.168.0.15, and 192.168.0.20
- **~192.168.0.95, 192.168.0.***: Accepts connections from IPv4 addresses from 192.168.0.0 to 192.168.0.255, except the IPv4 address 192.168.0.95

When a Driver receives a connection attempt, the list of filters is scanned sequentially from left to right, searching for a specific authorization or block for the IP address where the connection comes from. If no element on the list corresponds to the IP address, the authorization or block are dictated by the last element of that list:

- If the last element on the list is an authorization, such as "192.168.0.24", then all IP addresses not found on the list are blocked
- If the last element on the list is a block, such as "~192.168.0.24", then all IP addresses not found on the list are authorized

If an IP address appears on more than one filter on the list, the leftmost filter has precedence. For example, in case of "~192.168.0.95, 192.168.0.*", the IP address 192.168.0.95 fits both rules, but the rule that wins is the leftmost one, "~192.168.0.95", and therefore this IP address is blocked.

When **IOKit** blocks a connection, it logs a message "Blocked incoming socket connection from {IP}!".

In case of UDP connections in broadcast listen mode, where a Driver can receive packets from different IP addresses, blocks or permissions are performed at each packet received. If a packet is received from a blocked IP address, it logs a message "Blocked incoming packet from {IP} (discarding {N} bytes)!".

IO.Ethernet.ListenIP

A IP address of the local network interface that a Driver uses to establish and receive connections. Leave this property empty to use any local network interface.

IO.Ethernet.ListenPort

9 Number of the IP port used by a Driver to listen to connections.

IO.Ethernet.MainIP

A IP address of a remote device. Users can use a numerical address, as well as a device's host name, such as "192.168.0.7" or "SERVER2".

IO.Ethernet.MainLocalPort

9 Local port number to use when connecting to the main IP address of a remote device. This value is only used if the **IO.Ethernet.MainLocalPortEnable** property is equal to True.

IO.Ethernet.MainLocalPortEnable

◩ Configure to True to force the use of a specific local port when connecting to the main IP address of a remote device or configure to False to use any available local port.

IO.Ethernet.MainPort

9 Number of the IP port of a remote device, used with the **IO.Ethernet.MainIP** property.

IO.Ethernet.PingEnable

◩ Configure to True to enable sending a **ping** command to the IP address of a remote device, before trying to connect to the socket. This socket's connection time-out cannot be controlled, therefore sending a **ping** command before connecting is a fast way to detect if the connection is going to fail. Configure to False to disable a **ping** command.

IO.Ethernet.PingTimeoutMs

9 Delay time to wait for a response from a **ping** command, in milliseconds.

IO.Ethernet.PingTries

9 Maximum number of attempts of a **ping** command. Minimum value is 1 (one), including the first **ping** command.

IO.Ethernet.ShareListenPort

◩ Configure to True to share a listening port with other Drivers and processes or False to open a listening port in exclusive mode. To successfully share a listening port, all Drivers and processes that use that port must open it in shared mode. When a listening port is shared, each incoming connection is distributed to one of the processes listening. This way, if a Slave Driver only supports one connection at a time, users can use several instances of this Driver listening on the same port, therefore simulating a Driver with support for multiple connections.

IO.Ethernet.SupressEcho

◩ Configure to True to eliminate echoes in communication. An echo is the unwanted reception of an exact copy of all data packets a Driver sends to a device.

IO.Ethernet.Transport

A Defines a transport protocol. Possible values are **T or TCP**: Uses the TCP/IP protocol or **U or UDP**: Uses the UDP/IP protocol.

IO.Ethernet.UseIPv6

◩ Configure to True to use IPv6 addresses on all Ethernet connections or configure to False to use IPv4 addresses (default).

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.30	12/22/2021	M. Salvador	• Templates and messages now can start with an array (Case 31200) • Added read-only support for object repetition inside arrays (case 31239) • Message publishing now works with more than one value at the template. Block writes 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

VERSION	DATE	AUTHOR	COMMENTS
			Broker (*Case 30683*).
1.0.18	03/01/2021	G. Beal	• Added a new individual dead band configuration for 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)