Elipse RESTAPIClient Driver

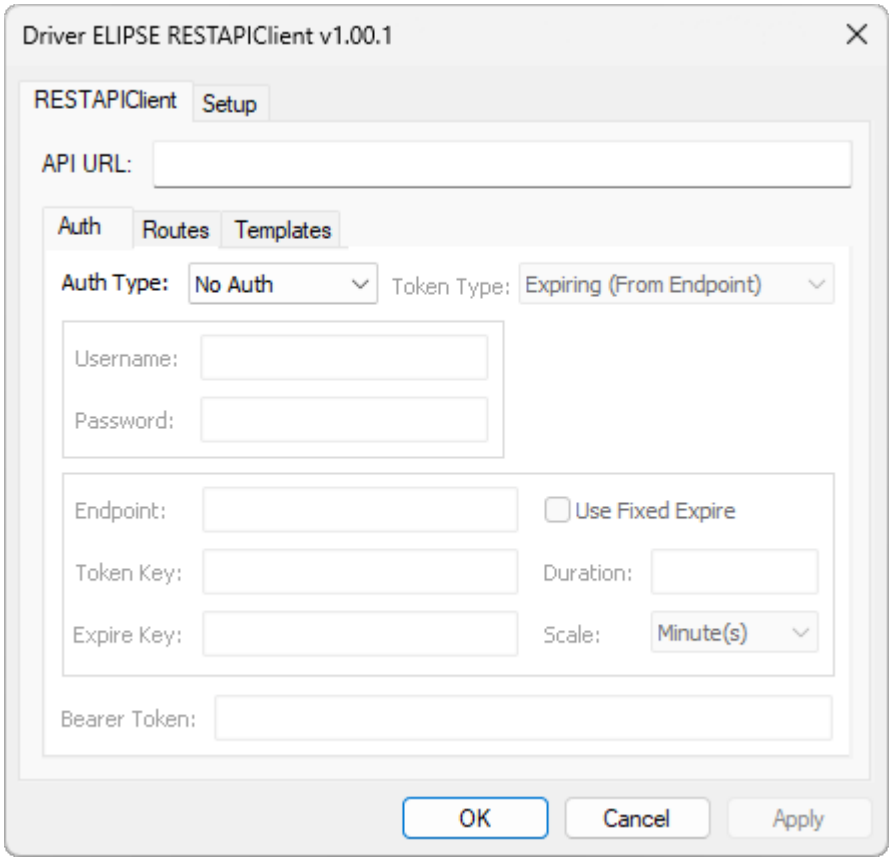
File Name	RESTAPIClient.dll
Manufacturer	Elipse Software Ltda
Devices	RESTful APIs
Protocol	HTTP or HTTPS
Version	1.0.3
Last Update	12/22/2025
Platform	Win32
Dependencies	IOKit version 3.0
Superblock Readings	No
Level	0

Introduction

This Driver implements the HTTP or HTTPS protocol, allowing applications developed by **Elipse Software** to communicate with RESTful APIs, using standard HTTP methods and supporting the **JSON** (*JavaScript Object Notation*) data format.

Driver's Configuration Parameters

The **[P]** configuration parameters of this Driver are not used. All configurations are performed on the configuration window, shown on the next figure.

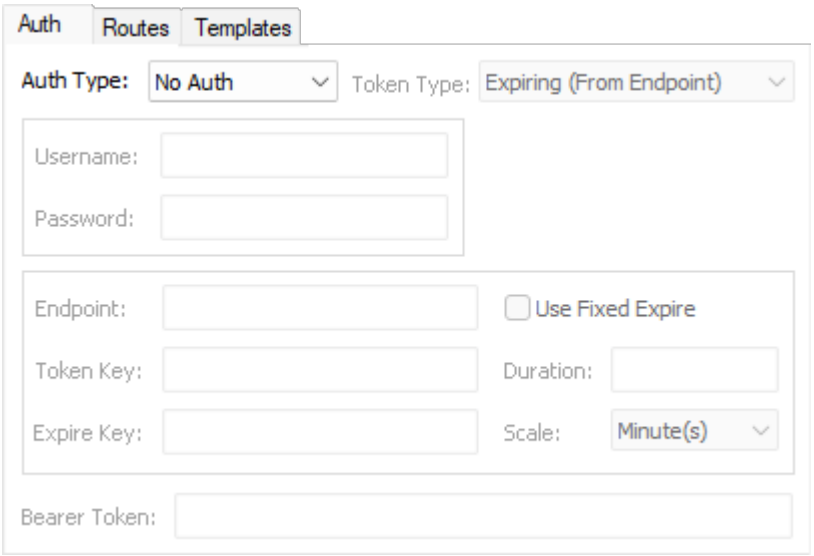


RESTAPIClient tab

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

Available options on the RESTAPIClient tab

OPTION	DESCRIPTION
API URL	Path of an API (*Application Programming Interface*), formed by a protocol, **HTTP** or **HTTPS**, by a domain, such as "https://api.example.com", and, when needed, by a TCP/IP port, such as "https://api.example.com:8080", defined before the specific routes that identify the resources of an API
Auth	Select this tab to configure the authentication of an API. For more information, please check table **Available options on the Auth tab**
Routes	Select this tab to configure the specific routes of an API. For more information, please check table **Available options on the Routes tab**
Templates	Select this tab to configure the templates in **JSON** format for requests and responses from an API. For more information, please check table **Available options on the Templates tab**



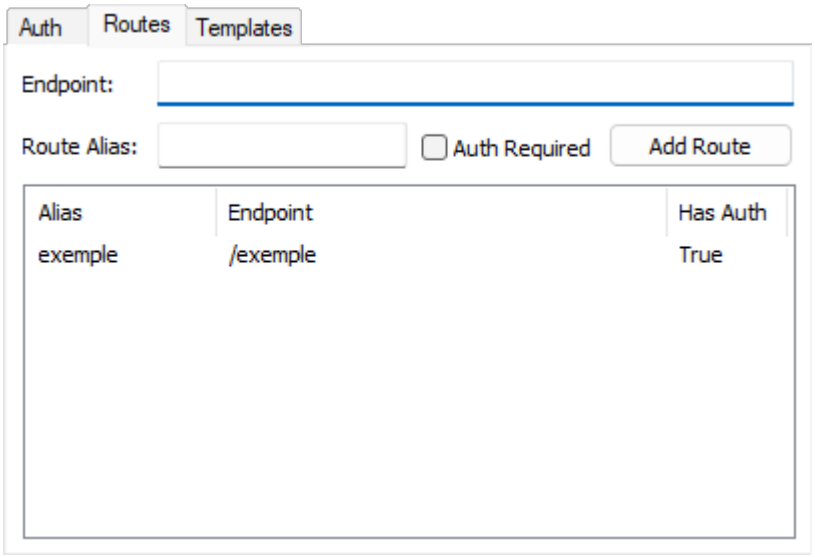
Auth tab

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

Available options on the Auth tab

OPTION	DESCRIPTION
Auth Type	Defines the type of authentication used by this Driver. The available options are **No Auth**, **Basic Auth**, or **Bearer Token**
Token Type	Defines the type of Bearer Token. This option is only enabled when the **Auth Type** option is configured as **Bearer Token**. The available options are **Expiring (From Endpoint)**, **Non-Expiring (From Endpoint)**, or **Non-Expiring (Provided)**

OPTION	DESCRIPTION
Username	Defines the name of a user. Available only for authentications of type **Basic Auth** or **Bearer Token** with types **Expiring (From Endpoint)** or **Non-Expiring (From Endpoint)**
Password	Defines the password of a user. Available only for authentications of type **Basic Auth** or **Bearer Token** with types **Expiring (From Endpoint)** or **Non-Expiring (From Endpoint)**
Endpoint	Defines the endpoint used to retrieve a Bearer Token, such as "/auth/token". Available only for the authentication of type **Bearer Token** with types **Expiring (From Endpoint)** or **Non-Expiring (From Endpoint)**
Token Key	Defines a key in the endpoint's response **JSON**-type file that contains a Bearer Token, such as "access_token". Available only for the authentication of type **Bearer Token** with types **Expiring (From Endpoint)** or **Non-Expiring (From Endpoint)**
Expire Key	Defines a key in the endpoint's response **JSON**-type file that contains the duration of a token, in seconds, such as "expires_in". Available only for the authentication of type **Bearer Token** with type **Expiring (From Endpoint)**
Use Fixed Expire	Defines a fixed time of expiration of a manually-provided token, such as 3600 seconds. Selecting this option deactivates the **Expire Key** option. Available only for the authentication of type **Bearer Token** with type **Expiring (From Endpoint)**
Duration	Defines the validation time of a token, which must be a number value. Available only when the **Use Fixed Expire** option is selected
Scale	Defines a time unit used for the Duration option. The available options are **Second(s)**, **Minute(s)**, **Hour(s)**, or **Day(s)**. Available only when the **Use Fixed Expire** option is selected
Bearer Token	Defines a fixed token used in authentication. Available only for the authentication of type **Bearer Token** with type **Non-Expiring (Provided)**



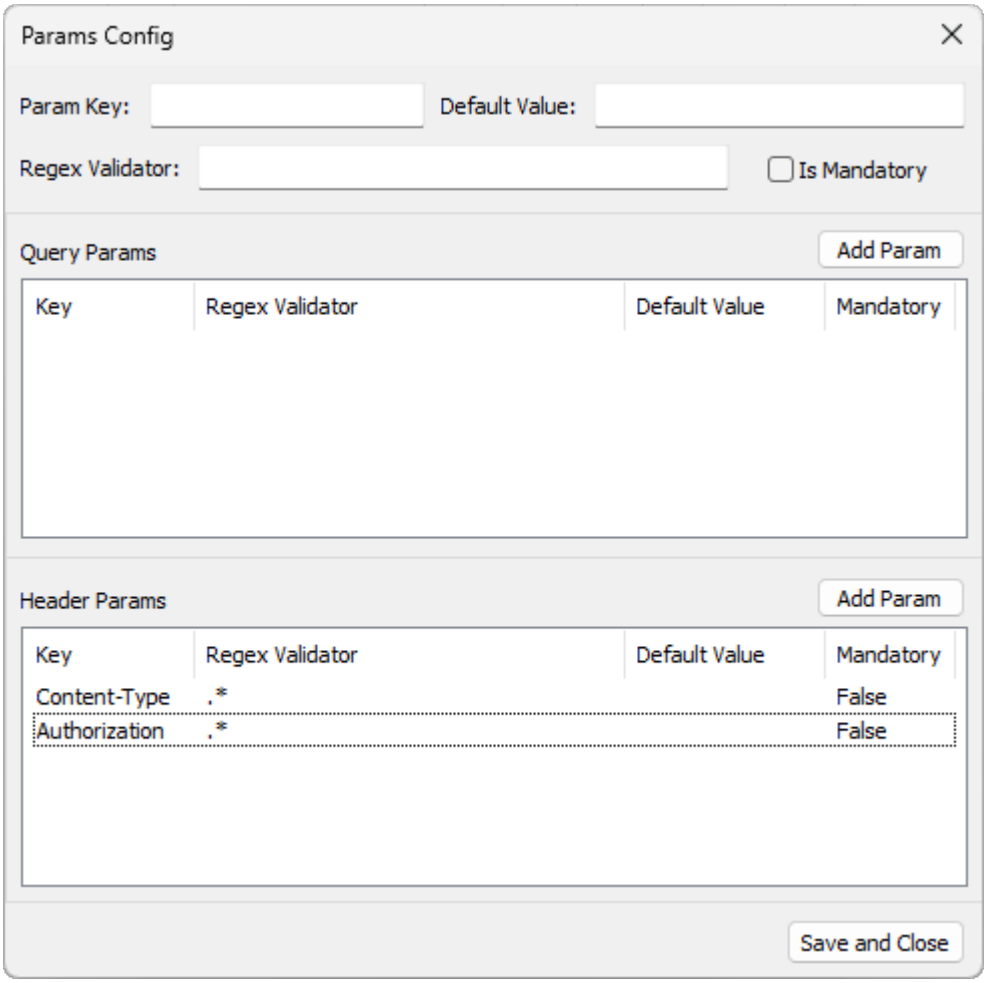
Routes tab

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

Available options on the Routes tab

OPTION	DESCRIPTION
Endpoint	Defines the endpoints used in requests. Each endpoint must start with a slash character followed by a path, such as "/api/users". Users can also define dynamic segments in an endpoint, which are then replaced by values configured via Tags in a request. These segments must be indicated by colons, such as "/api/company/:companyId/user/:userId"
Route Alias	Defines an alias for a route. This alias is used later to link Tags from an application to a specific route. The value of this option is case-sensitive
Auth Required	This option indicates that a specific route uses the authentication method configured in the **Auth Type** option of the **Auth** tab
Add Route	Adds a route to the table of configured routes

Right-clicking a route displays the **Edit** option, which opens the Params Config window, and the **Delete** option, which allows removing the selected route. Double-clicking a route also opens the Params Config window, shown on the next figure, which allows configuring specific parameters of that route.



Params Config window

The available options on the Params Config window are described on the next table.

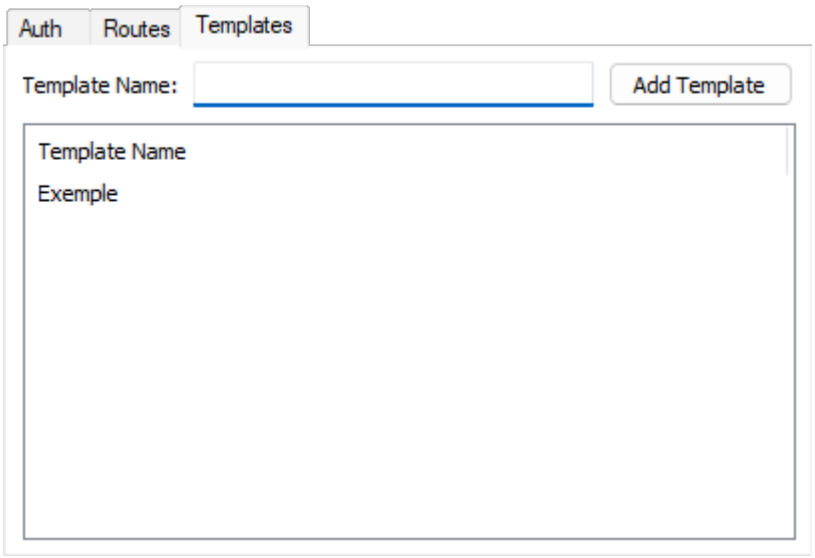
Available options on the Params Config window

OPTION	DESCRIPTION
Param Key	Defines the name of a parameter used in a request
Default Value	Defines a default value for a parameter. If no value is provided, this option is considered as empty
Regex Validator	Allows defining a regular expression to validate a value assigned to a parameter, if needed
Is Mandatory	Selecting this option prevents sending a request if a parameter is not configured or if it is configured with an empty value
Add Param	Adds a new parameter to the Query Params or Header Params tables. Each one of these tables has its corresponding **Add Param** option, which allows inserting a query parameter or a header parameter, respectively
Save and Close	Saves the configured data and closes this window

Right-clicking a parameter on any of these tables displays the **Delete** option, which allows removing the selected parameter.

The **Content-Type** and **Authorization** parameters are automatically inserted as **header**-type parameters on the Header Params table. These parameters cannot be removed nor configured later using Tags.

In case of using a type of authentication different from the available options on the **Auth** tab, such as using an API key, a route can be configured to not use authentication, that is, deselect the **Auth Required** option, and in this case the header parameters can be configured, defining the respective values of the **Default Value** column both manually and at run time using Tags.



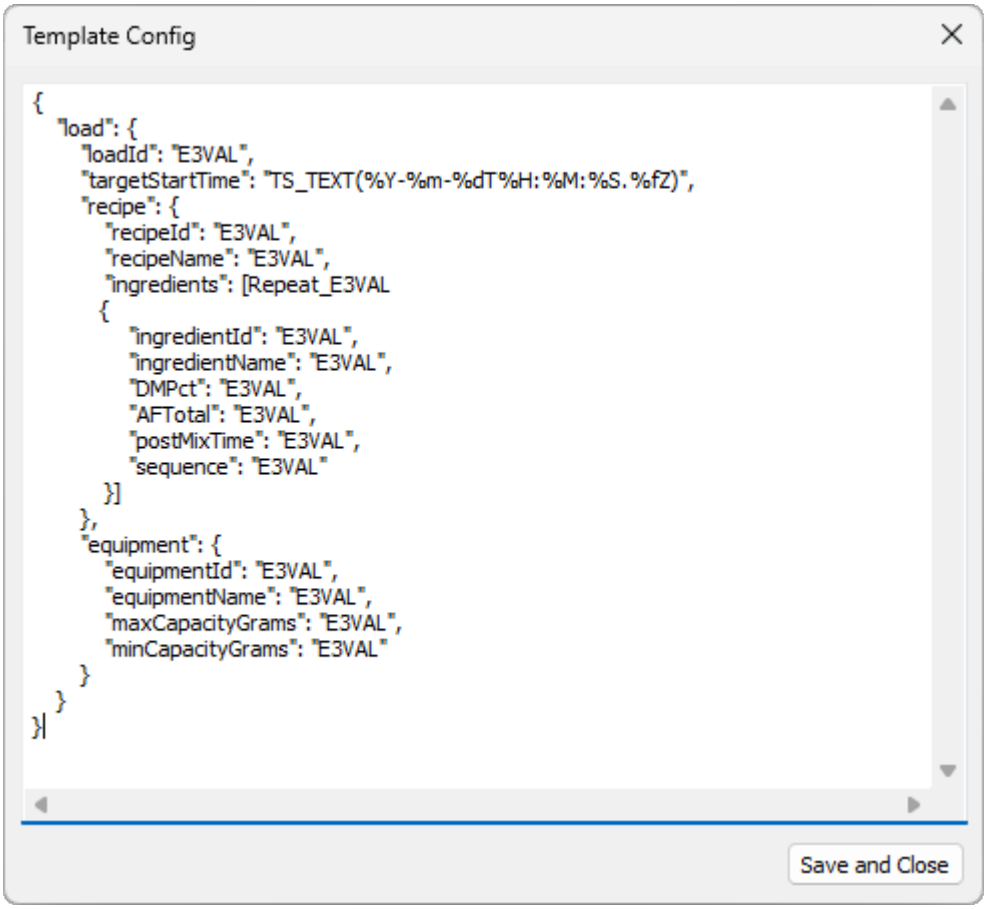
Templates tab

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

Available options on the Templates tab

OPTION	DESCRIPTION
Template Name	Defines the name of a template, in **JSON** format, that is used for requests and responses from an API
Add Template	Adds a template to the table of configured templates

Right-clicking a template displays the **Edit** option, which opens the Template Config window, and the **Delete** option, which allows removing the selected template. Double-clicking a template also opens the Template Config window, shown on the next figure, which allows configuring a template in **JSON** format, according to the standard syntax of this format described in the next topic.



Template Config window

The available option on the Template Config window is **Save and Close**, which saves the configured data and then closes this window.

Syntax of a JSON Template

To extract or assemble the content of a message, users must declare a template, which allows specifying the format of this message and which parts must be transformed into data. Each template must use keywords placed in the position of the values to extract. The available keywords are described on the next table.

Available keywords for templates

KEYWORD	DESCRIPTION
TS_TEXT(format)	Timestamp, in **Text** format, used as the timestamp for a Block Tag or I/O Tag. The meaning of each field is described on table **Available options for the TS_TEXT keyword**
TS_UNIX	Timestamp, in seconds since 1970, or **UNIX** format. This value can be a number or a text, and it is used as the timestamp for a Block Tag or I/O Tag, that is, 1504198675 or "1504198675"
QL_OPC	Quality, in OPC DA standard, using 1 (one) byte. This value is used directly as the quality value in a Block Tag or I/O Tag, without transformation. For more information, please check table **OPC DA standard**
QL_BOOL	Quality in Boolean standard. If the value is greater than 0 (zero) or the "true" or "TRUE" expression, quality is good.

KEYWORD	DESCRIPTION
	If the value is 0 (zero) or the "false" or "FALSE" expression, quality is bad
E3VAL	Specifies a value extracted in the sequence that it occurs for a Block Tag, each value in a Block Element. If there is only one **E3VAL** value in a 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	A keyword that can only be used inside an array in **JSON** format to indicate that repeated elements inside that array must be processed independently by this Driver, returning a reading for each set. A template can have this keyword only once

Available options for the TS_TEXT keyword

OPTION	DESCRIPTION
%a	Abbreviated weekday, in English
%A	Full weekday, in English
%b	Abbreviated month, in English
%B	Full month, in English
%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	Timezone, an international code that converts time to a **GMT** (*Greenwich Mean Time*) format
%+	Offset of GMT time in the format **±HH:MM**

The next code contains examples of timestamps formatted by 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
QQ	Possible values are **0**: BAD, **1**: UNCERTAIN, or **3**: GOOD
SSSS	Possible values are **0**: BAD_NONSPECIFIC, **1**: BAD_CONFIGERROR, **2**: BAD_NOTCONNECTED, **3**: BAD_DEVICEFAILURE, or **4**: BAD_SENSORFAILURE
SSSS	Possible values are **0**: UNCERT_NONSPECIFIC, **1**: UNCERT_LASTUSABLEVALUE, or **4**: UNCERT_SENSORNOTACCURATE
SSSS	Possible values are **0**: GOOD_NONSPECIFIC, **1**: GOOD_LOCALOVERRIDE, or **6**: GOOD_NONSPECIFICLOCALTIMESTAMP
LL	Possible values are **0**: FREE, **1**: LOW, **2**: HIGH, or **3**: CONST

Examples of Configuring Templates in JSON Format

A file in **JSON** format containing an array with multiple data sets must include the **Repeat_E3VAL** keyword in the template right after the opening bracket of that array, according to the next example.

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

In this case, when receiving a message with 2 (two) data sets, for example, the **OnRead** event of a Block Tag is triggered for each set, as shown in the next code.

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

If other **E3VAL**-type values are declared before or after a **Repeat_E3VAL**-type value, these are replicated for each set, with the **Repeat_E3VAL**-type value always as the last in the receiving Block Tag, according to the next example.

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

A Block Tag with the Block Elements **Type**, **Description**, **Temperature**, and **Humidity** can generate, for each OnRead event, values according to the next code.

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

A Case Without Reading Repetition Per Set

If users want to trigger an **OnRead** event for each set, a template can be declared without a **Repeat_E3VAL**-type value, using only one **E3VAL**-type keyword to capture the array, according to the next code.

```
{
  "MessageType": "DUMMY",
  "TagData": "E3VAL",
  "MessageDesc": "DUMMY"
}
```

In this case, the received array is expanded based on the number of elements, resulting in a single reading on the **OnRead** event, in the format of the next example.

```
array(11, 12); array(14, 66); array(70, 55) => Single OnRead event
```

If a template contains more than one **E3VAL**-type keyword, that is, non-**DUMMY**, the array is not expanded, according to the next example.

```
{
  "MessageType": "E3VAL",
  "TagData": "E3VAL",
  "MessageDesc": "E3VAL"
}
```

This reading is performed on a single **OnRead** event, in the format of the next example.

```
"Test"; array(array(11, 12), array(14, 66), array(70, 55)); "Description"=> Single OnRead event
```

Tag Reference

This section contains information about the configuration of this Driver's PLC and Block Tags.

Block Tag for Receiving

Read-Only

Use this Block Tag for reading data received from a request to an API, configuring its parameters according to the next table.

Item	*<Route Alias>;<Template Name>*
B1	0 (zero)
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

The size of this Block Tag depends on the linked template and corresponds to the number of elements in the file in **JSON** format whose keyword is different from **DUMMY**, except when the only existing keyword different from **DUMMY** is an array. In this case, that array is expanded and this Block Tag has the same size of the received array.

NOTE

In case of a Block Tag for Receiving whose linked template contains the **Repeat_E3VAL** keyword, the data set related to this keyword appears at the last positions of this Block Tag.

Block Tag for Sending

Write-Only

Use this Block Tag for writing data related to a request to an API, configuring its parameters according to the next table.

Item	*<Route Alias>;<Template Name>*
B1	0 (zero)
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

The size of this Block Tag is the sum of the number of parameters required in a request, that is, header, route, query, and body.

The first Block Elements correspond to the header parameters added on the **Params Config** window, except for the **Content-Type** and **Authorization** parameters, which are ignored.

Next, there are route parameters, defined on the **Routes** tab and linked to an endpoint, such as "/api/company/:companyId/user/:userId", in which the parameters are **:companyId** and **:userId**.

Next, the query parameters are defined, also configured on the **Params Config** window.

The remaining Elements of this Block Tag are designed to the definition of body data of a request, that is, values defined by the template linked to this Block Tag, if it exists.

The number of Block Elements designed to the body of a request corresponds to the number of items in the linked template and all Elements must be filled. However, if the special **Repeat_E3VAL** array is present, this array is considered as a single Block Element, unlike what happens in the **Block Tag for Receiving**, in which it is expanded. In addition, this array remains in the original position of the template and it is not moved to the end of this Block Tag.

To write to this array, users must inform in an application an array of arrays, in which each element represents a data set to insert. Each set is interpreted as an array and values are linked by position with the fields defined in the template.

Example of a template.

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

Example of writing the corresponding element in this Block Tag.

```
array(array(11, 12), array(14, 66), array(70, 55))
```

Example of the final result, that is, the generated code in **JSON** format.

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

NOTE

Writing invalid data to the parameters, either because they do not match the configured **regular expressions** or because they have an incompatible data type, fails this writing operation.

PLC Tag

Write-Only

Use a PLC Tag to send a request to an API by configuring its parameters according to the next table.

Item	*<Route Alias>;<Template Name>*
N1	1 (one)
N2	From 0 (zero) to 3 (three)
N3	0 (zero)
N4	0 (zero)

Any writing operation to this Tag executes a request using the route configured in the *Item* parameter and, when needed, sends the body of a message based on the linked template.

The *N2* parameter maps the HTTP request method. Possible values are **0**: **GET** method, **1**: **POST** method, **2**: **PUT** method, or **3**: **DELETE** method.

NOTE

If not all mandatory elements for a request are configured in the **Block Tag for Sending**, this writing operation fails. Likewise, if a request fails for any other reason, this writing operation is not completed successfully.

Documentation of I/O Interfaces

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

Configuration of a Driver

I/O Interface configuration is performed on a Driver's configuration dialog box. To access the configuration of this dialog box in **Elipse E3** in 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 **Elipse 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 one of these Drivers for each serial port.

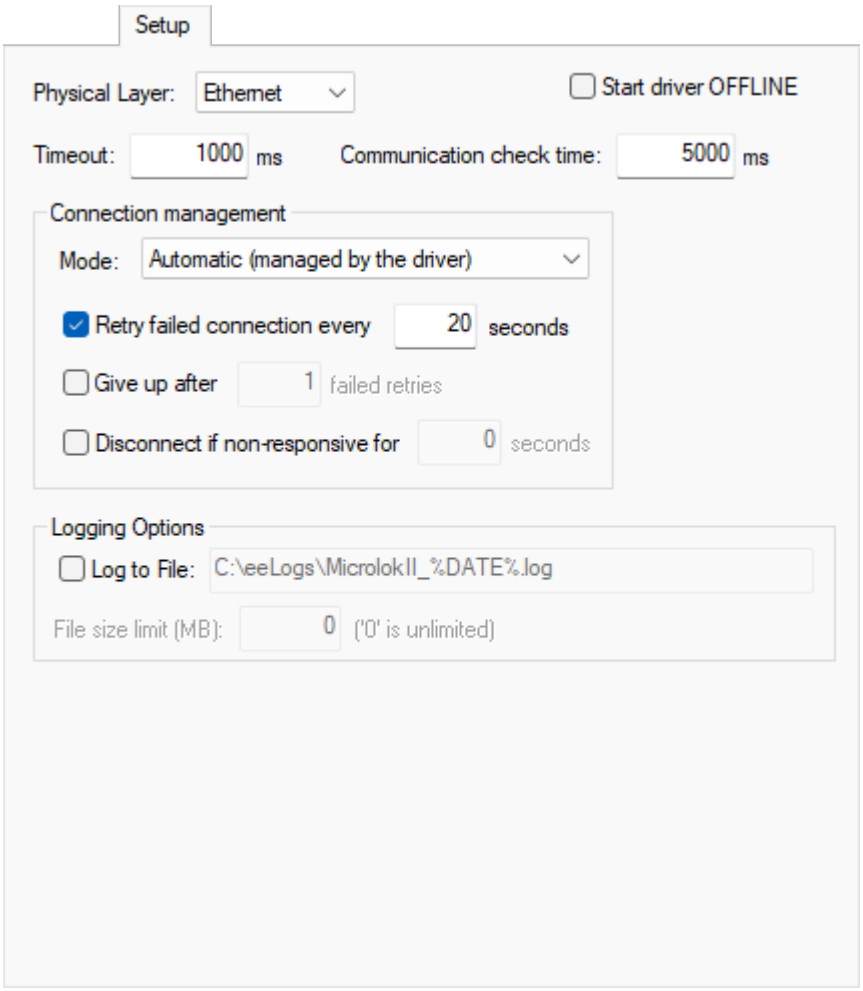
Configuration Dialog Box

The dialog box of I/O Interfaces 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 that Driver, on the configuration dialog box.

Setup Tab

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

- **General configurations**: Configurations of a 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 the reception's buffer
Communication check time	Set the time, in milliseconds, to define the interval at which communication is considered to be in an inactive state. As long as an I/O Driver receives valid data, its communication state is considered active. However, if during operation an I/O Driver does not receive valid data inside this period of time, the state is considered inactive. The communication state is shown in the **IO.CommunicationStatus** Tag
Start driver OFFLINE	Select this option so that a 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
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, this 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, a 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 identifier of the current process. This option is particularly useful when using several instances of the same Driver in **Elipse E3**, thus allowing each instance to generate a separate log file. For example, when configuring this option with value "c:\e3logs\drivers\sim_%PROCESS%.log", it generates a file named c:\e3logs\drivers\sim_00000FDA.log 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 with value "c:\e3logs\drivers\sim_%DATE%.log", it generates a file named c:\e3logs\drivers\sim_2005_12_31.log in 12/31/2005 and a file named c:\e3logs\drivers\sim_2006_01_01.log in 01/01/2006. Similarly, the **%DATE_HOUR%** macro generates one log file per hour, in the format **aaaa_mm_dd_hh**
File size limit (MB)	Configure the log file size limit, in megabytes. A value equal to 0 (zero) means that there is no size limit for the log file

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.CommunicationStatus

Type of Tag	I/O Tag
Type of Access	Reading
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	0 (zero)
N4 Parameter	6 (six)
String Configuration	IO.CommunicationStatus

This Tag informs the communication status of a Driver. It indicates how communication works relative to receiving valid data within a time period arbitrated in the configuration. For more information, please check topic **Setup Tab**. Possible values are **0 - Inactive communication**: The Driver did not receive valid data or stopped receiving data after *n* milliseconds, as configured in the properties window, or **1 - Active communication:** The Driver is receiving valid data.

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 an 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**: Message of an event, 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. Possible values are the following:

- **0**: Physical layer stopped, that is, a 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, a Driver is in **Online** mode but the physical layer is not connected. If the **Connection management** option is configured with the value **Automatic**, the physical layer can be connecting, disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured with the vlaue **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 a device is connected, only that 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 that 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 3 (three) parameters, then the size of the Block must be 6 (six, 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 writings 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 **Elipse E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use the **Write** method of a Driver to send the parameters directly to that 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 the log of a Driver or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of an 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 using the **IO.PhysicalLayerStatus** Tag

In the next example, using **Elipse 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 may fail when configuring a Driver to **Online** mode, that is, writing the value 1 (one). In this case, this 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
- This 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 this 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, in which a Driver manages the connection or **1**: Manual mode, in which an application manages the connection.

IO.GiveUpEnable

◩ When configured to True, defines a maximum number of reconnection attempts. If all reconnection attempts fail, a Driver enters the **Offline** mode. When configured to False, a 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 connection is lost. If this one fails, this 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 then 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

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.

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.3	12/22/2025	A. Fetzner	• Fixed problems when parsing data in **JSON** format and in communication using the HTTPS protocol (*Case 39116*).
1.0.1	10/28/2025	A. Fetzner	• 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)