GE Ethernet SRTP Driver

Filename	GEETH.dll
Manufacturer	FANUC Corporation
Devices	90-30, 90-70, and Versamax
Protocol	GE SRTP
Version	2.0.9
Last Update	09/03/2025
Platform	Win32
Dependencies	IOKit version 2.0 or later
Superblock Readings	Yes
Level	0

Introduction

The GE Ethernet SRTP Driver Driver communicates with GE 90-30, 90-70, and Versamax devices using the GE SRTP protocol via Ethernet.

Device Preparation

This section contains information about configuring a connection from a device to this Driver. This device must be configured to support connections in **Ethernet TCP/IP** mode.

Connecting a Computer to a PLC

The devices used to test this Driver are the following:

- BASE 10-SLOT (IC693CHS391D)
- SERIES 90-30 POWER SUPPLY (IC693PWR321R)
- CPU 351 (IC693CPU351-DF)
- ETHERNET CONTROLLER (IC693CMM321-BA)
- INPUT SIMULATOR MODULE (IC693ACC300D)
- OUTPUT 12/24 0.5A 32PT POS (IC693MDL753D)
- STAT. MGR to PC (IC693CBL316A) cable
- TRANSCEIVER

The steps for assembling these devices are described next.

1. Plug the **POWER SUPPLY**, **CPU 351**, **ETHERNET CONTROLLER**, **INPUT**, and **OUTPUT** into the rack (**BASE 10-SLOT**) as explained on document *Series 90-30 Programmable Controller Installation Manual*.
2. Connect one side of the **STAT** cable on port 1 (one) of the **CPU** and the other side on the computer's serial port. This step is only necessary if the IP address of the PLC is not configured yet.

3. Connect one end of the **TRANSCEIVER** to the corporate Ethernet and the other end to the **ETHERNET CONTROLLER** (side connection).

4. Plug the **POWER SUPPLY** feed cable to an outlet. Its **PWR** frontal led must be turned on.

5. Use **LM90** software to program the PLC and to modify its IP address according to the network used. For more information about the IP address to use, please check the *Appendix D - Assigning IP Addresses of Host Communications Toolkit for C/C++ Application User's Manual (GFK - 0870B)*. If **LM90** returns a time-out error, there are errors in the installation of the device. Please check the serial port used by **LM90** to communicate.

Detecting Whether a PLC has a Valid IP Address

To detect whether a PLC has a valid IP address, users must use Windows **ping** command, according to the next example.

```
C:\WINDOWS>ping 128.0.0.3
Pinging 128.0.0.3 with 32 bytes of data:
Reply from 128.0.0.3: bytes=32 time<10ms TTL=32
Reply from 128.0.0.3: bytes=32 time<10ms TTL=32
Reply from 128.0.0.3: bytes=32 time<10ms TTL=32
Reply from 128.0.0.3: bytes=32 time<10ms TTL=32
```

If this **ping** command receives a reply, this configuration is correct. Otherwise, the reply of this **ping** command must look like the one on the next example.

```
C:\WINDOWS>ping 234.2.3.18
Pinging 234.2.3.18 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
```

In this case, return to **LM90** and configure an IP address compatible with the network.

Driver Configuration

This Driver does not use **[P]** configuration parameters. All configurations of **IOKit** library, and the ones specific to this Driver, must be performed on this Driver's configuration dialog box or on the **Extra Settings** option in **Elipse SCADA**.

In this properties dialog box, the **GEETH** tab contains specific settings for this Driver. The other tabs refer to the communication settings of **Elipse Software**'s **IOKit** library.

For more information about the configuration of **IOKit** library, please check topic **Documentation of I/O Interfaces**.

Configuring Properties

This topic contains information about the properties available on the **GEETH** tab, including the value of offline property **Strings** that can be programmed by users when starting an application in **Offline** mode.

In **Elipse E3**, **Elipse Power**, or **Elipse Water** applications, the value of these settings can also be defined at run time. To do this, start this Driver in **Offline** mode by selecting the **Start driver OFFLINE** option on the **Setup** tab of the properties dialog box.

The configuration options for this Driver are listed on the next table.

Configuration options for GE Ethernet SRTP Driver

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
Setup	Physical Layer	IO.Type	Text	By default, use the **Ethernet** option

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
	Timeout	IO.TimeoutMs	Number	A time limit, in milliseconds, to receive data from a device's response. For example, the value 1000 defines a 1 (one) second limit
Ethernet	Transport	IO.Ethernet.Transport	Text	By default, use the **TCP** option
	Main IP	IO.Ethernet.MainIP	Text	IP address of a device, in the format **[0-255].[0-255].[0-255].[0-255]**
	Port	IO.Ethernet.MainPort	Number	Value of the TCP/IP port of the Ethernet connection
GEETH	Source ID (Master)	GEETH.SourceID1	Number	First value, from left to right, referring to the identification of a message's source
		GEETH.SourceID2	Number	Second value, from left to right, referring to the identification of a message's source
		GEETH.SourceID3	Number	Third value, from left to right, referring to the identification of a message's source
		GEETH.SourceID4	Number	Fourth value, from left to right, referring to the identification of a message's source
	Destination ID (Slave)	GEETH.DestinationID1	Number	First value, from left to right, referring to the identification of a message's destination
		GEETH.DestinationID2	Number	Second value, from left to right, referring to the identification of a message's destination
		GEETH.DestinationID3	Number	Third value, from left to right, referring to the identification of a message's destination
		GEETH.DestinationID4	Number	Fourth value, from left to right, referring to the identification of a message's destination

All offline properties must be configured via PLC Tags in **String** format, by using the *N1* parameter equal to -1 (minus one), the *N2* parameter equal to 0 (zero), the *N3* parameter equal to 0 (zero), and the *N4* parameter equal to 3 (three). For more details and examples, please check topic **Documentation of I/O Interfaces**.

Tag Reference

All Tags of this Driver are configured with numerical values by using their **N** and **B** parameters.

[N] and [B] Parameters of Block or PLC Tags

Use a PLC or Block Tag to read or write data to the memory of a PLC.

N1/B1	Not used
N2/B2	Data type. For more information, please check topic **Data Types**
N3/B3	Address of a point, usually starting at 1 (one)
N4/B4	Maximum number of characters, in case of a **String** data type, that is, the *N1* parameter equal to 27 or 28

NOTE

As the structure of GE SRTP protocol allocates a maximum of 200 bytes for a data area, please check topic **Maximum Number of Elements per Block Tag** for more information about the maximum number of Elements a Block Tag supports, according to the type of data to process.

Reading a PLC Status Word

Function 999

Used to check the last value of a **PLC Status Word** returned by a device, via script of an **OnRead** event of a Tag Block reading, according to the parameters described on the next table.

B1	Not used
B2	999
B3	Not used
B4	Not used

Each **OnRead** event returns a Block with the structure of Elements on the next table to interpret the last received **PLC Status Word**.

Available Elements on PLC Status Word Tag

ELEMENT	DESCRIPTION	VALUE
1	**Oversweep** flag. Meaningful only when constant sweep mode is active	Possible values are **0**: There is no oversweep condition or **1**: Constant sweep value exceeded
2	Constant sweep mode	Possible values are **0**: Constant sweep mode deactivated or **1**: Constant sweep mode activated
3	PLC **Fault Entry** since last reading	Possible values are **0**: PLC's fault table unchanged since last reading or **1**: PLC's fault table was changed since last reading by this device
4	PLC **I/O Fault Entry** since last reading	Possible values are **0**: PLC's I/O fault table unchanged since last reading or **1**: PLC's I/O fault table was changed since last reading by this device
5	PLC **Fault Entry** available	Possible values are **0**: PLC's fault table is empty or **1**: One or more fault entries in PLC's fault table
6	PLC **I/O Fault Entry** available	Possible values are **0**: PLC's I/O fault table is empty or **1**: One or more fault entries on PLC's I/O fault table
7	Programmer attachment flag	Possible values are **0**: No programmer's attachment found or **1**: Programmer's attachment found
8	Enables or disables front panel switch configuration	Possible values are **0**: Outputs enabled or **1**: Outputs disabled
9	Runs or stops front panel switch configuration	Possible values are **0**: Stop or **1**: Run
10	Protected OEM	Possible values are **0**: No OEM protection or **1**: OEM protection in effect
11	Not used	0 (zero)
12	Not used	0 (zero)
13	PLC status	Possible values are **0**: Run I/O enabled, **1**: Run I/O disabled, **2**: Stop I/O disabled, **3**: CPU stopped by failure, **4**: CPU stopped, **5**: CPU suspended, or **6**: Stop I/O enabled

Collecting Event History

Function 200

Used to collect several history records from a SER PLC, via script of an **OnRead** event of a Block Tag reading, according to the parameters described on the next table.

B1	Not used
B2	200
B3	Not used
B4	Not used

Each **OnRead** event returns a Block with the following structure of Elements, if there is any available event:

- **Element 0**: Circuit number
- **Element 1**: Transition status. Possible values are **1**: Normal or **0**: Alarm

NOTE

The **TimeStamp** property of this Block Tag returns the **time of the day** of a data collecting.

Reading a SER PLC Status

Use a PLC Tag to check the current status of a SER PLC, accessing the **%Q2016** memory address, according to the parameters described on the next table.

B1	Not used
B2	2 (two)
B3	2016
B4	Not used

Possible reading values for this PLC Tag are **0**: Offline or **1**: Online.

Reading a SER PLC Event History Status

Use a PLC Tag to check for events on a SER PLC event history by accessing the **%Q2011** memory position, according to the parameters described on the next table.

B1	Not used
B2	2 (two)
B3	2011
B4	Not used

Possible values for this PLC Tag are **0**: Without events or **1**: With events.

Collecting Horner SER300/CDC300 Events

Function 300

Used to collect events from a Horner SER300/CDC300 device by reading a Block Tag, according to the parameters described on the next table.

B1	Not used
B2	300
B3	Initial record of an event block generated by a CDC300 device
B4	Not used

The event is returned as a Block with the following structure of Elements:

- **Element 0**: New Event Input Data (from zero to 15 bits)
- **Element 1**: New Event Input Data (from 16 to 31 bits)
- **Element 2**: New Event Input Data (from 32 to 47 bits)
- **Element 3**: New Event Input Data (from 48 to 63 bits)
- **Element 4**: New Event Input Data (from 64 to 79 bits)
- **Element 5**: New Event Input Data (from 80 to 95 bits)
- **Element 6**: New Event Input Data (from 96 to 111 bits)
- **Element 7**: New Event Input Data (from 112 to 127 bits)
- **Element 8**: New Event Input Data (from 128 to 143 bits)
- **Element 9**: New Event Input Data (from 144 to 159 bits)
- **Element 10**: New Event Input Data (from 160 to 175 bits)
- **Element 11**: New Event Input Data (from 176 to 191 bits)
- **Element 12**: New Event Input Data (from 192 to 207 bits)
- **Element 13**: New Event Input Data (from 208 to 223 bits)
- **Element 14**: New Event Input Data (from 224 to 239 bits)
- **Element 15**: New Event Input Data (from 240 to 255 bits)
- **Element 16**: Current Time (**Date and Time** format)
- **Element 17**: Current Input Data (from zero to 15 bits)
- **Element 18**: Current Input Data (from 16 to 31 bits)
- **Element 19**: Current Input Data (from 32 to 47 bits)
- **Element 20**: Current Input Data (from 48 to 63 bits)

- **Element 21**: Current Input Data (from 64 to 79 bits)
- **Element 22**: Current Input Data (from 80 to 95 bits)
- **Element 23**: Current Input Data (from 96 to 111 bits)
- **Element 24**: Current Input Data (from 112 to 127 bits)
- **Element 25**: Current Input Data (from 128 to 143 bits)
- **Element 26**: Current Input Data (from 144 to 159 bits)
- **Element 27**: Current Input Data (from 160 to 175 bits)
- **Element 28**: Current Input Data (from 176 to 191 bits)
- **Element 29**: Current Input Data (from 192 to 207 bits)
- **Element 30**: Current Input Data (from 208 to 223 bits)
- **Element 31**: Current Input Data (from 224 to 239 bits)
- **Element 32**: Current Input Data (from 240 to 255 bits)
- **Element 33**: SER300 Module Command Register
- **Element 34**: SER300 Module Response Register
- **Element 35**: SER300 Module Status Register
- **Element 36**: SER300 Module Firmware Version Number
- **Element 37**: New Output Data (from zero to 15 bits)
- **Element 38**: New Output Data (from 16 to 31 bits)
- **Element 39**: New Output Data (from 32 to 47 bits)
- **Element 40**: New Output Data (from 48 to 63 bits)
- **Element 41**: Input Mask Word Index
- **Element 42**: Input Mask

NOTE

The **TimeStamp** property of this Block Tag returns the **New Event Time Stamp** of a data collecting.

Collecting PAC System RX3i Events

Function 400

Used to collect several records from the event history of an RX3i device, via **OnRead** script event of a Block Tag, according to the parameters described on the next table.

B1	Not used
B2	400
B3	Not used
B4	Equal to 1 (one) for automatic cleaning of the event history of an RX3i device (optional)

Each **OnRead** event returns a Block Tag with the following structure of Elements, if there is any event available:

- **Element 0**: Event status
- **Element 1**: Channel number
- **Element 2**: Event counter of a Channel
- **Element 3**: Event counter of all Channels
- **Element 4**: Current status

NOTE

The **TimeStamp** property of this Block Tag returns the **time of the day** of a data collecting.

Reading a PAC System RX3i Event History Status

Use a PLC Tag to check for the presence of events in the event history of an RX3i device, by accessing the **%W00001** memory position, according to the parameters described on the next table.

B1	Not used
B2	40
B3	1 (one)
B4	Not used

Possible values for this PLC Tag are **0**: Without events or **1**: With events.

Cleaning a PAC System RX3i Event History

Use a writing to a PLC Tag to release the whole content of the event history of an RX3i device, by writing 0 (zero) to the **%W00001** memory position, according to the parameters described on the next table.

B1	Not used
B2	40
B3	1 (one)
B4	Not used

The writing value for this PLC Tag is 0 (zero, releases events).

NOTE

The *B2* parameter equal to 400 can perform an automatic cleaning of an event history, if configured together with the *B4* parameter equal to 1 (one).

Data Types

Available data types

TAG TYPE	N2 OR B2	DESCRIPTION	OPERATION
PLC or Block	0	Discrete Inputs (**%I**), **Bit**	Reading and Writing
PLC or Block	1	Discrete Inputs (**%I**), **Byte**	Reading and Writing
PLC or Block	2	Discrete Outputs (**%Q**), **Bit**	Reading and Writing
PLC or Block	3	Discrete Outputs (**%Q**), **Byte**	Reading and Writing
PLC or Block	4	Discrete Temporaries (**%T**), **Bit**	Reading and Writing
PLC or Block	5	Discrete Temporaries (**%T**), **Byte**	Reading and Writing
PLC or Block	6	Discrete Internals (**%M**), **Bit**	Reading and Writing
PLC or Block	7	Discrete Internals (**%M**), **Byte**	Reading and Writing
PLC or Block	8	Special Contacts A (**%SA**), **Bit**	Read-Only
PLC or Block	9	Special Contacts A (**%SA**), **Byte**	Read-Only
PLC or Block	10	Special Contacts B (**%SB**), **Bit**	Read-Only
PLC or Block	11	Special Contacts B (**%SB**), **Byte**	Read-Only
PLC or Block	12	Special Contacts C (**%SC**), **Bit**	Read-Only
PLC or Block	13	Special Contacts C (**%SC**), **Byte**	Read-Only
PLC or Block	14	System Fault (**%S**), **Bit**	Reading and Writing
PLC or Block	15	System Fault (**%S**), **Byte**	Reading and Writing
PLC or Block	16	Genius Global Data (**%G**), **Bit**	Reading and Writing

TAG TYPE	N2 OR B2	DESCRIPTION	OPERATION
PLC or Block	17	Genius Global Data (**%G**), **Byte**	Reading and Writing
PLC or Block	18	Analog Input table (**%AI**), unsigned 16-bit **Word**	Reading and Writing
PLC or Block	19	Analog Input table (**%AI**), signed 16-bit **Word**	Reading and Writing
PLC or Block	20	Analog Output table (**%AQ**), unsigned 16-bit **Word**	Reading and Writing
PLC or Block	21	Analog Output table (**%AQ**), signed 16-bit **Word**	Reading and Writing
PLC or Block	22	Register table (**%R**), unsigned 16-bit **Word**	Reading and Writing
PLC or Block	23	Register table (**%R**), signed 16-bit **Word**	Reading and Writing
PLC or Block	24	Register table (**%R**), signed 32-bit **DWord**	Reading and Writing
PLC or Block	25	Register table (**%R**), unsigned 32-bit **DWord**	Reading and Writing
PLC or Block	26	Register table (**%R**), 32-bit IEEE floating point **Float**	Reading and Writing
PLC or Block	27	Register table (**%R**), **String**	Reading and Writing
PLC or Block	28	Register table (**%R**), reverse **String**	Reading and Writing
PLC or Block	29	Register table (**%R**), 64-bit IEEE floating point **Double**	Reading and Writing
PLC or Block	30	Local Memory (**%L**), unsigned 16-bit **Word**	Reading and Writing
PLC or Block	31	Local Memory (**%L**), signed 16-bit **Word**	Reading and Writing
PLC or Block	32	Local Memory (**%L**), signed 32-bit **DWord**	Reading and Writing
PLC or Block	33	Local Memory (**%L**), unsigned 32-bit **DWord**	Reading and Writing
PLC or Block	34	Local Memory (**%L**), 32-bit IEEE floating point **Float**	Reading and Writing
PLC or Block	35	Register table (**%L**), 64-bit IEEE floating point **Double**	Reading and Writing
PLC or Block	40	Word Memory (**%W**), unsigned 16-bit **Word**	Reading and Writing
PLC or Block	41	Word Memory (**%W**), signed 16-bit **Word**	Reading and Writing
PLC or Block	42	Word Memory (**%W**), signed 32-bit **DWord**	Reading and Writing

TAG TYPE	N2 OR B2	DESCRIPTION	OPERATION
PLC or Block	43	Word Memory (**%W**), unsigned 32-bit **DWord**	Reading and Writing
PLC or Block	44	Word Memory (**%W**), 32-bit IEEE floating point **Float**	Reading and Writing
PLC or Block	45	Register table (**%W**), 64-bit IEEE floating point **Double**	Reading and Writing
PLC or Block	50	UDT Word - unsigned	Reading and Writing
PLC or Block	51	UDT Short Int - signed	Reading and Writing
PLC or Block	52	UDT I32 - signed	Reading and Writing
PLC or Block	53	UDT U32 - unsigned	Reading and Writing
PLC or Block	54	UDT R32 Float	Reading and Writing
PLC or Block	55	UDT R64 Double	Reading and Writing
PLC or Block	56	UDT Boolean (byte)	Reading and Writing
PLC or Block	57	UDT String	Reading and Writing
PLC	100	PLC clock	Reading and Writing
PLC	500	Read UDT	Read-Only

Maximum Number of Elements per Block Tag

Maximum number of Elements per Block Tag

DATA TYPE	LIMIT
Bit	Number of Elements less or equal to 200
Byte	Number of Elements less or equal to 200
Word	Number of Elements less or equal to 100
DWord or Float	Number of Elements less or equal to 50
Double	Number of Elements less or equal to 25
Text	Number of Elements multiplied by the *B4* parameter is less or equal to 200

User Data Types

This Driver allows reading Tags with addressing by **Strings**, relative to user-defined objects or types, known as **UDT** (*User Data Types*).

A **UDT**-type Tag is addressed using the *Item* parameter, informing the path of an object, and also using the *N2* parameter, in which users define a data type. For more information, please check topic **Data Types**.

To read UDT Tags, users must create a PLC Tag with the *N2* parameter equal to 500 and with the **AdviseType** property equal to **AlwaysInAdvise**. At each scan of this Tag, all PLC and Block Tags with a **UDT** data type are updated and read in groups of 40 for each request.

If users inform in the *Item* parameter an object that represents an array of basic types, they can create a Block Tag and each Block Element corresponds to an element of that array.

Addressing Examples

- To read the **%SB80** bit in **Bit** mode:
  - **N2**: 10 (equal to **%SB** in **Bit** mode)
  - **N3**: 80 (reads the 80th bit)
  - **N4**: 0 (zero)

- To read the **%R594** register in **Word** mode:
  - **N2**: 20 (equal to **%R** always in **Word** mode)
  - **N3**: 594 (register number to read)
  - **N4**: 0 (zero)

- To read a 100 bytes in **Text** mode, starting from address **%R594**:
  - **N2**: 27 (equal to **%R** in **Text** mode)
  - **N3**: 594 (starts in **%R594**)
  - **N4**: 100 (number of characters to read)

At protocol level, this Tag performs a reading of 50 registers (**Words**), starting from **%R594** up to **%R644**, to receive all 100 bytes that compose the text (characters in **ASCII** format).

- To read from **%G10** up to **%G50** variables, inclusive, in **Byte** mode, from a PLC with an IP address equal to 4.0.0.4 using the gef_cfg.ini file as an example:
  - **B2**: 17 (equal to **%G** in **Byte** mode)
  - **B3**: 10 (starts at **%G10**)
  - **B4**: 0 (zero)

- To read 60 bits from a PLC with an IP address equal to 3.0.0.1 in **Bit** mode, starting from address **%SA5**:
  - **B2**: 8 (equal to **%SA** in **Bit** mode)
  - **B3**: 5 (starts at **%SA5**)
  - **B4**: 0 (zero)

Documentation of I/O Interfaces

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

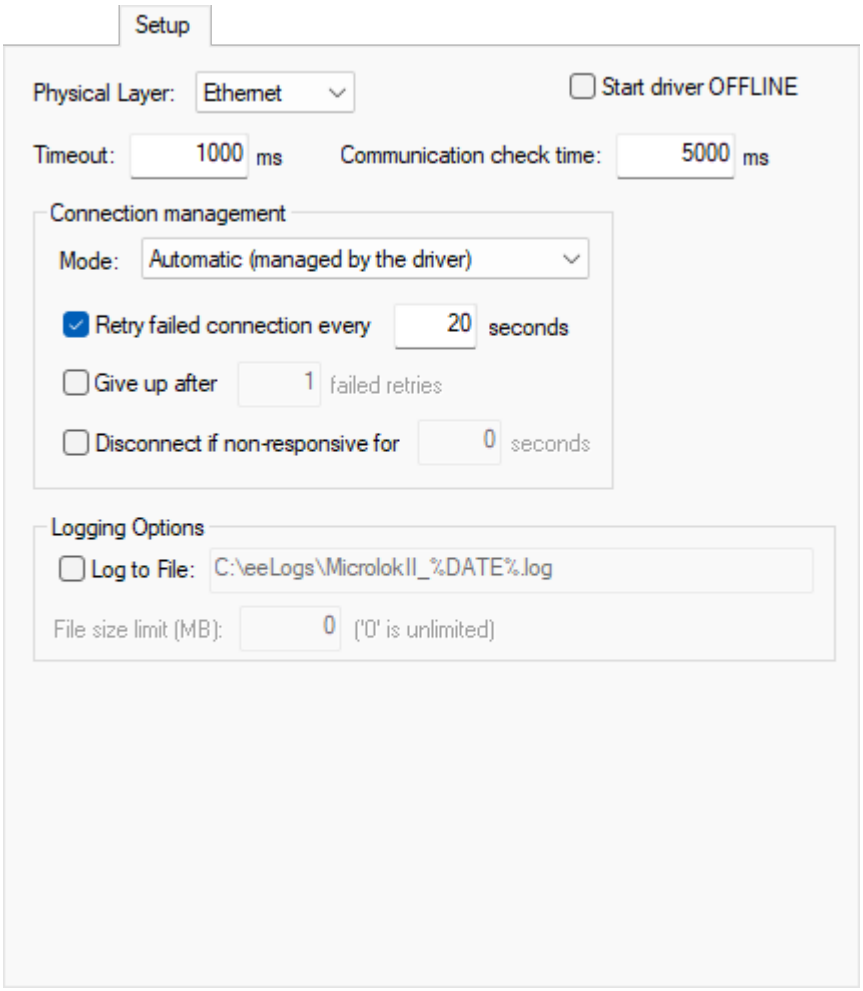
Configuration Dialog Box

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

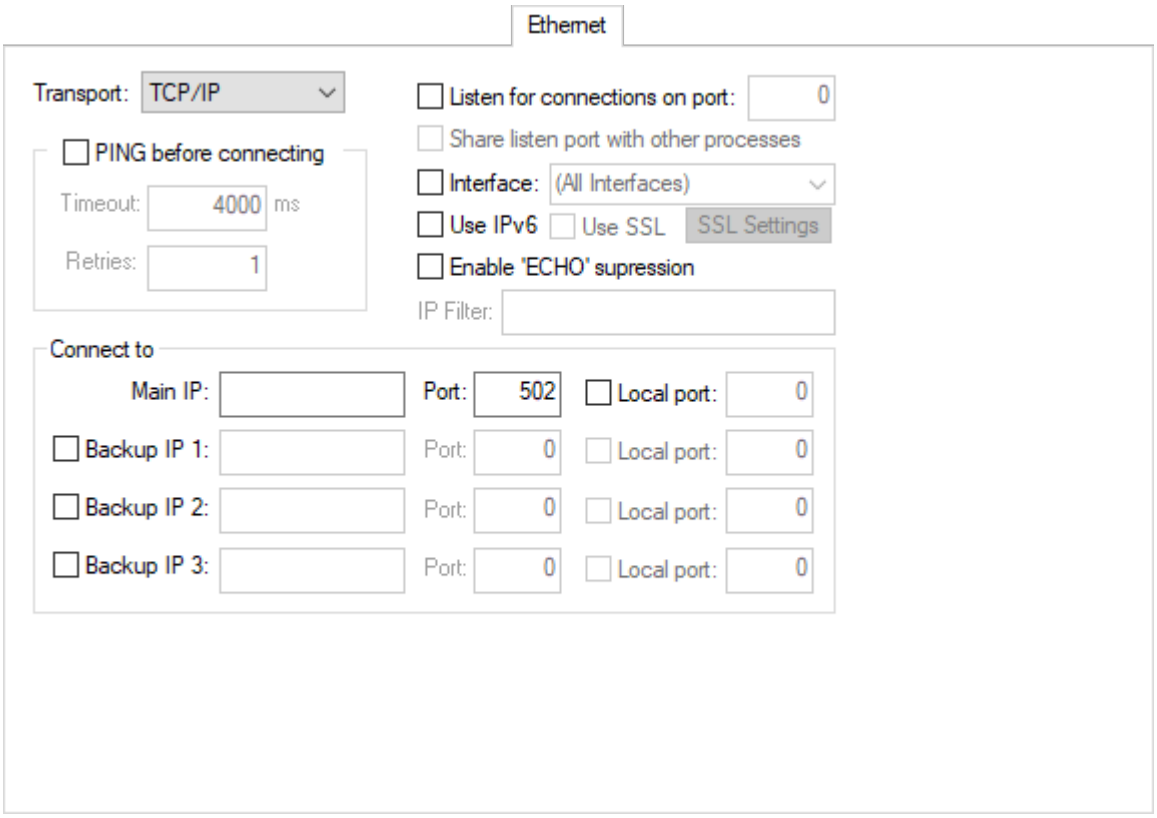
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every ... 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

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** Interface.



Ethernet tab

Available options on the Ethernet tab

OPTION	DESCRIPTION
Transport	Select the value **TCP/IP** for a TCP socket (*stream*) or select the value **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, a 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 listening port with other Drivers and processes
Interface	Select the local network interface, identified by its IP address, that a Driver uses to establish and receive connections, or select the value **(All Interfaces)** to allow connection in any network interface
Use IPv6	Select this option to force a Driver to use addresses in **IPv6** format on all Ethernet connections. Leave this option deselected to use the **IPv4** format
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 **IO.Ethernet.IPFilter** property for more information
PING before connecting	Enable this option to execute a **ping** command, that is, to check whether a device can be reached on a network, for a device before trying a socket connection. This is a quick way

OPTION	DESCRIPTION
	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 a **ping** command. Users must use a **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 1 (one) and 4 (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 a remote device. Users can use an IP address separated by dots, as well as a URL. In case of a URL, a 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 a 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 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.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.

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 attempt to connect.

During the connection process, if the active IP address is not available, the I/O Interface tries to connect using the other 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 address or **1, 2, 3**: Alternative address) if a Driver is currently connected. If a Driver is disconnected, this Tag configures the active IP address for the next attempt to connect.

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 first alternative or backup IP address is activated, and so on for all alternative IP addresses and returning to the main address until a connection is established.

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

IO.Ethernet.SocketState

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	4 (four)
N4 Parameter	2 (two)
String Configuration	IO.Ethernet.SocketState

The Value property of this Tag corresponds to socket states as a map of bits:

- **Bit 0**: 0 (zero, not listening) or 1 (one, listening)
- **Bit 1**: 0 (zero, disconnected) or 1 (one, connected)

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 a Driver must not accept external connections, that is, if a Driver behaves as a master, or configure to True to enable the reception of connections, that is, if a 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, a Driver tries to use an alternative IP address. Configure to False to disable its usage.

IO.Ethernet.BackupIP[2,3]

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.BackupLocalPort[2,3]

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 or configure to False to use any available local port.

IO.Ethernet.BackupPort[2,3]

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

IO.Ethernet.IPFilter

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, according to the next 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 in the interval between 192.168.0.41 and 192.168.0.50
- **192.168.0.***: Accepts connections from IPv4 addresses in the interval between 192.168.0.0 and 192.168.0.255
- **fe80:3bf:877::*:* (expands to fe80:03bf:0877:0000:0000:0000:*:*)**: Accepts connections from IPv6 addresses in the interval between fe80:03bf:0877:0000:0000:0000:0000:0000 and 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 in the interval between 192.168.0.0 and 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 listening mode, in which 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 accept connections. Leave this property empty to establish and accepts connections using 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 sent to a device.

IO.Ethernet.Transport

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
2.0.9	09/03/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37974*).
2.0.8	10/07/2024	M. Salvador	• Added support to UDT (*User Data Types*).
2.0.7	07/12/2021	C. Mello	• Compatibility adjustments for reading superblocks via **E3**'s **EnableReadGrouping** property (*Case 31113*).
2.0.6	11/04/2020	C. Mello	• Added support for **Double 64-bits IEEE-754 Floating point** data types for **%R**, **%L**, and **%W** registers (*Case 29410*). • Changed the configuration of **Unsigned 16-bit Word** data types for the **%W** register from the *N2* or *B2* parameters equal to **29** to the *N2* or *B2* parameters equal to **40**.
2.0.5	08/01/2019	M. Ludwig	• Driver ported to Visual Studio 2017 (*Case 27187*).
2.0.4	10/31/2017	C. Mello	• Fixed internal inconsistencies related to

VERSION	DATE	AUTHOR	COMMENTS
			the update of Tags via Superblocks (*Case 23555*).
2.0.3	07/25/2017	M. Ludwig	• Fixed an eventual invalid access when performing a grouped reading with the *N3* parameter greater than zero (*Case 23079*).
2.0.2	02/06/2017	C. Mello	• Added an option to configure the **Source ID** and **Destination ID** fields on this Driver's configuration window (*Case 18507*).
2.0.1	12/16/2016	C. Mello	• Added support for reading with Superblocks plus callbacks for **%L** registers (*Case 21895*). • Adjustments for syncing values changed by the constant scan mode of a few controllers (*Case 18018*). • Adjustments for writing Block Tags involving a **String** data type (*Case 17902*). • Adjustments for reading and writing Block Tags involving **Bit** and **Byte** data types (*Case 17415*). • Added a command for querying the last **PLC Status Word** informed by a device (*Case 17380*). • Added new interpretations of values for **%W** memory registers (*Case 16545*). • Added new interpretations of values for the **Program Block Memory** region (**%L** registers) of devices from series 90-70 (*Case 15904*). • Added support for reading Tags by callbacks and the Superblock service was moved to this Driver's internal control and processing (*Case 14555*). • Driver ported to the new standard of **IOKit** library version **2.0** (*Case 14582*).
1.10.1	01/19/2012	C. Mello	• Added support for interpreting error codes

VERSION	DATE	AUTHOR	COMMENTS
			defined in the protocol (*Case 12677*).
1.9.1	06/27/2011	C. Mello	• Added support for the **Local Memory** (**%L**) data type (the *B2* parameter equal to 30) (*Case 11886*).
1.8.1	06/17/2010	C. Mello	• Added support for the **Word Memory** (**%W**) data type (the *B2* parameter equal to 29) and also support for PAC System RX3i event collecting, Function 400 (*Case 10854*). • Added information about querying the status of a SER PLC event collecting (*Case 11232*).
1.7.1	03/16/2009	C. Mello	• Adjustments to prevent a break on the transaction control of messages (*Case 10187*).
1.6.1	06/06/2008	C. Mello	• Adjustments to prevent an error after physically losing an Ethernet connection (*Case 9571*).
1.5.1	05/18/2007	F. Englert	• Adjustments for the initialization of the communication protocol whenever a loss of physical connection happens (*Case 8274*).
1.4.1	11/24/2006	C. Mello	• Added a **Collecting of Horner SER300/CDC300 Events - Function 300** (*Case 6776*). • Added a **Collecting of Event History - Function 200** (*Case 6306*). • Adjustments to change the active IP address (*Case 7255*). • Adjustments on the TNS counter to become aware of a context of automatic reconnection (*Case 6796*).
		F. Englert	• Support for Superblocks updated to **Elipse E3** version **2.0** (*Case 6792*). • Fixed an error when reading and writing to a Block

VERSION	DATE	AUTHOR	COMMENTS
			Element with the *B2* parameter equal to 24, 25, or 26 that uses an incorrect offset (*Case 6793*).
1.3.1	10/27/2005	M. Salvador	• Added support for Superblocks, only compatible with **Elipse E3** version **1.23**, unofficial. • Implemented a control for automatic reconnection if there is a constant failure on communication.
1.2.1	04/25/2005	C. Mello	• Full revision of source code.
		M. Salvador	• Disabled writings to **%SA**, **%SB**, and **%SC** data types, because they do not work on some PLC models.
1.1.1	04/24/2004	L. Silva	• Added a new option for this Driver's **200** function to control timestamps and milliseconds (*Case 3524*).
1.0.1	01/28/2005	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)