Ablerex EnerSolis Driver

File Name	EnerSolis.dll
Manufacturer	Ablerex Electronics Co., Ltd.
Devices	EnerSolis ES Series 6000, 8000, 10000, 12000, and 25600
Protocol	JBus
Version	1.0.1
Last Update	08/24/2022
Platform	Win32
Dependencies	IOKit v2.00 or later
Superblock Readings	No
Level	0

Introduction

This Driver implements the JBus protocol, allowing an application developed by **Elipse Software** to communicate with EnerSolis ES Series 6000, 8000, 10000, 12000, and 25600 devices by Ablerex Electronics Co., Ltd.

Driver Configuration

This Driver does not use [P] configuration parameters.

All configurations must be performed on this Driver's configuration dialog box in **E3** or using the **Extra Settings** option in **Elipse SCADA**.

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

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

Configuring Properties

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

For **E3** or **Elipse Power** applications, the value of these settings can also be defined at run time. To do so, select the **Start driver OFFLINE** option on the **Setup** tab of the properties window to start this Driver in **Offline** mode.

This Driver's configuration options are listed on the next table.

Configuration options for Ablerex EnerSolis Driver

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
Setup	Physical Layer	IO.Type	Text	By default, use the **Ethernet** option
	Timeout	IO.TimeoutMs	Number	A time limit, in milliseconds, to receive data from a device's response. For

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
				example, the value 1000 defines a one second limit
Serial	Port	IO.Serial.Port	Number	By default, use the same value set on the device. Possible values are **1**: COM1, **2**: COM2, **3**: COM3, ..., **n**: COM*n*
	Baud rate	IO.Serial.Baudrate	Number	By default, use the same value set on the device. For example, the value 9600 defines a baud rate for serial communication
	Data bits	IO.Serial.DataBits	Number	By default, use the same value set on the device. Possible values are **7**: 7 bits or **8**: 8 bits
	Parity	IO.Serial.Parity	Number	By default, use the same value set on the device. Possible values are **0**: No Parity, **1**: Odd Parity, **2**: Even Parity, **3**: Mark Parity, or **4**: Space Parity
	Stop bits	IO.Serial.StopBits	Number	By default, use the same value set on the device. Possible values are **0**: Stop Bit 1, **1**: Stop Bit 1.5, or **2**: Stop Bit 2
EnerSolis	Use Default Slave Address	EnerSolis.UseDefaultSlaveAddress	Number	Enables or disables the use of a default slave address. Possible values are **0**: Disabled or **1**: Enabled
	Default Slave Address	EnerSolis.DefaultSlaveAddress	Number	Address (ID) of a device to be used automatically by this Driver. Valid values are in the range from 1 (one) to 255

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

Tag Reference

This section contains information about the configuration of this Driver's **[N/B]** Tags.

Reading Alarms

Read-Only

Use a PLC or Block Tag to read alarms from a device, using the parameters described on the next table.

Parameters for reading alarms

N1/B1	Address (ID) of a device
N2/B2	1000
N3/B3	Not used
N4/B4	Alarm number, ranging from 0 (zero) to 31

A device provides a reading of 32 alarms, numbered from 0 (zero) to 31, with a status value of **1** for On or **0** for Off.

Reading Multiple Alarms using a Block Tag

Use a Block Tag to read multiple alarms, in which each Element represents, sequentially, the status of an alarm, starting with the alarm number defined in the **B4** parameter. The next table contains some examples of using Block Tags.

Usage examples of Block Tags

OBJECTIVE	ELEMENTS	B1	B2	B3	B4 (INITIAL ALARM)
Reading all 32 alarms	32	Device ID	1000	Not used	0 (zero)
Reading the first 10 alarms	10	Device ID	1000	Not used	0 (zero)
Reading the last 10 alarms	10	Device ID	1000	Not used	22
Reading 5 (five) alarms, starting from alarm 10	5 (five)	Device ID	1000	Not used	10

Reading a Simple Alarm using a PLC Tag

Use a PLC Tag to individually read the status of the alarm number, between 0 (zero) and 31, defined in the **N4** parameter. The next table contains some examples of using PLC Tags.

Usage examples of PLC Tags

OBJECTIVE	N1	N2	N3	N4 (ALARM NUMBER)
Reading alarm number 14	Device ID	1000	Not used	14

OBJECTIVE	N1	N2	N3	N4 (ALARM NUMBER)
Reading alarm number 30	Device ID	1000	Not used	30

NOTE
All alarms are stored in two **Words**, in registers **0xC000** and **0xC001** of a device's memory.

Reading Errors

Read-Only

Use a PLC or Block Tag to read errors from a device, using the parameters described on the next table.

Parameters for reading errors

N1/B1	Address (ID) of a device
N2/B2	2000
N3/B3	Not used
N4/B4	Error number, ranging from 0 (zero) to 47

A device provides the reading of 48 errors, numbered from 0 (zero) to 47, with a status value of **1** for On or **0** for Off.

Reading Multiple Errors Using a Block Tag

Use a Block Tag to read multiple errors, in which each Element represent, sequentially, the status of an error, starting with the error number defined in the **B4** parameter. The next table contains some examples of using Block Tags.

Usage examples of Block Tags

OBJECTIVE	ELEMENTS	B1	B2	B3	B4 (INITIAL ERROR)
Reading all 48 errors	48	Device ID	2000	Not used	0 (zero)
Reading the first 10 errors	10	Device ID	2000	Not used	0 (zero)
Reading the last 10 errors	10	Device ID	2000	Not used	38
Reading 5 (five) errors, starting from error 10	5 (five)	Device ID	2000	Not used	10

Reading a Simple Error using a PLC Tag

Use a PLC Tag to individually read the status of the error number, between 0 (zero) and 47, defined in the **N4** parameter. The next table contains some examples of using PLC Tags.

Usage examples of PLC Tags

OBJECTIVE	N1	N2	N3	N4 (ERROR NUMBER)
Reading error number 18	Device ID	2000	Not used	18
Reading error number 45	Device ID	2000	Not used	45

NOTE

All errors are stored in three **Words**, in registers **0xC010**, **0xC011** and **0xC012** of a device's memory.

Reading Energy Values

Read-Only

Use a PLC or Block Tag to read energy values from a device, using the parameters described on the next table.

Parameters for reading energy values

N1/B1	Address (ID) of a device
N2/B2	3000
N3/B3	Not used
N4/B4	Energy value number, ranging from 0 (zero) to 31

A device provides the reading of 32 energy values, numbered from 0 (zero) to 31, as described on the next table.

Energy values

N4/B4	ENERGY VALUES
0 (zero)	Output power (kW)
1 (one)	AC voltage phase L1 (V)
2 (two)	AC voltage phase L2 (V)
3 (three)	AC voltage phase L1-L2 (V)
4 (four)	AC output current L1 (A)
5 (five)	AC output current L2 (A)
6 (six)	AC frequency L1 (Hz)
7 (seven)	DC-Bus Positive-voltage (V)
8 (eight)	DC-Bus Negative-voltage (V)
9 (nine)	Inverter internal temperature (°C)
10	Inverter Heat sink temperature (°C)
11	DC1 input voltage (V)

N4/B4	ENERGY VALUES
12	DC2 input voltage (V)
13	DC1 input current (A)
14	DC2 input current (A)
15	Input Power A (kW)
16	Input Power B (kW)
17	Total Output Power (kWh)
18	Reserve
19	Reserve
20	Reserve
21	Reserve
22	Self Test Vmin
23	Self Test Vmax
24	Self Test Fmin
25	Self Test Fmax
26	AC voltage phase L2-L3 (V)
27	AC frequency L2 (Hz)
28	AC voltage phase L3 (V)
29	AC voltage phase L3-L1 (V)
30	AC frequency L3 (Hz)
31	AC output current L3 (A)

Reading Multiple Energy Values Using a Block Tag

Use a Block Tag to read multiple energy values, in which each Element represents, sequentially, the values of energy measurements, starting with the energy value number defined in the **B4** parameter. The next table contains some examples of using Block Tags.

Usage examples of Block Tags

OBJECTIVE	ELEMENTS	B1	B2	B3	B4 (INITIAL ENERGY VALUE)
Reading all 32 energy values	32	Device ID	3000	Not used	0 (zero)
Reading the first 10 energy values	10	Device ID	3000	Not used	0 (zero)
Reading the last 10 energy values	10	Device ID	3000	Not used	22
Reading 5 (five) energy values, starting from energy value 10	5 (five)	Device ID	3000	Not used	10

Reading a Simple Energy Value Using a PLC Tag

Use a PLC Tag to individually read the energy value number, ranging from 0 (zero) to 31, defined in the **N4** parameter. The next table contains some examples of using PLC Tags.

Usage examples of PLC Tags

OBJECTIVE	N1	N2	N3	N4 (ENERGY VALUE NUMBER)
Reading energy value number 14	Device ID	3000	Not used	14
Reading energy value number 30	Device ID	3000	Not used	30

NOTE

All energy values are stored in 96 **Words**, from register **0xC020** to **0xC07F** in a device's memory.

Device Settings

Reading and Writing

Use PLC or Block Tags to read or write configuration values from a device, using the parameters described on the next table.

Parameters for reading or writing configuration values

N1/B1	Address (ID) of a device
N2/B2	Please check table **Configuration commands**
N3/B3	Not used
N4/B4	Not used

The next table contains the commands that can be used to configure a device.

Configuration commands

N2/B2	COMMAND	REGISTER	CODE
6019	Power Factor P1-P2	0xCF13	C19
6020	Power Factor PF0-PF1	0xCF14	C20
6021	Power Factor PF2-PF3	0xCF15	C21
6024	Vpset	0xC098	C24
6025	Reduce Power Rate	0xCF16	C25
6029	Enable or Disable Vpset	0xC09D	C29

NOTE
To write data via Block Tags, all Elements must be sent on a single block writing.

Reading or Writing for Power Factor P1-P2

Use a Block Tag with two Elements to read or write values, according to the next data:

- **BlockTag.Element1**: Power Factor P1 (values from 0~100, referring to 0%~100%)
- **BlockTag.Element2**: Power Factor P2 (values from 0~100, referring to 0%~100%)

Reading or Writing for Power Factor PF0-PF1

Use a Block Tag with two Elements to read or write values, according to the next data:

- **BlockTag.Element1**: Power Factor PF0, according to the next table of examples
- **BlockTag.Element2**: Power Factor PF1, according to the next table of examples

Examples for Power Factor PF0-PF1

VALUE	POWER FACTOR
125	0.75 (leading)
110	0.90 (leading)
100	1 (one)
90	0.90 (lagging)
75	0.75 (lagging)

Reading or Writing for Power Factor PF2-PF3

Use a Block Tag with two Elements to read or write values, according to the next data:

- **BlockTag.Element1**: Power Factor PF2, according to the next table of examples
- **BlockTag.Element2**: Power Factor PF3, according to the next table of examples

Examples for Power Factor PF2-PF3

VALUE	POWER FACTOR
125	0.75 (leading)
110	0.90 (leading)
100	1 (one)
90	0.90 (lagging)
75	0.75 (lagging)

Reading or Writing for Vpset

Use a PLC Tag to read or write values, according to the next data:

- **PLCTag**: Vpset (values from 231~240, referring to 231Vac~240Vac)

Reading or Writing for Reduce Power Rate

Use a PLC Tag to read or write values, according to the next data:

- **PLCTag**: Reduce Power Rate (values from 10~100, referring to 10%~100%)

Reading or Writing for Enabling or Disabling Vpset

Use a PLC Tag to read or write values, according to the next data:

- **PLCTag**: Enable or disable Vpset (**1**: Enable or **0**: Disable)

Generic Tag

Reading and Writing

Use PLC or Block Tags to read or write values directly to the memory address of a device's registers, using the parameters described on the next table.

Parameters for reading or writing memory registers

N1/B1	Address (ID) of a device
N2/B2	Please check table **Commands to manipulate data in memory registers**
N3/B3	Not used
N4/B4	Memory of a register

The next table contains the commands that can be used to manipulate data in memory registers of a device.

Commands to manipulate data in memory registers

N2/B2	COMMAND	REGISTER
310	Reading with FCode 0x03 and writing with FCode 0x10	N4/B4
364	Reading with FCode 0x03 and writing with FCode 0x64	N4/B4

Any memory register of a device can be read via FCode 0x03. Some registers can be modified by FCode 0x10 and others, more restricted, by FCode 0x64.

For reading or writing via Block Tags, each one of its Elements represents a sequential memory register, starting from the initial address defined in the **B4** parameter, according to the next table.

Usage examples of Block Tags

OBJECTIVE	ELEMENTS	B1	B2	B3	B4 (INITIAL REGISTER)	RESULT
Reading with Block Tag (FCode 0x03)	5 (five)	Device ID	310	Not used	10	Reading 5 (five) registers, from 10 to 14
Writing with Block Tag (FCode 0x10)	3 (three)	Device ID	310	Not used	32	Writing 3 (three) registers, from 32 to 34

Each memory register can store a 16-bit value (word). Please check device device's user's manual to check the table of available registers for reading and writing values.

Documentation of I/O Interfaces

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

Driver Configuration

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

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

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

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

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

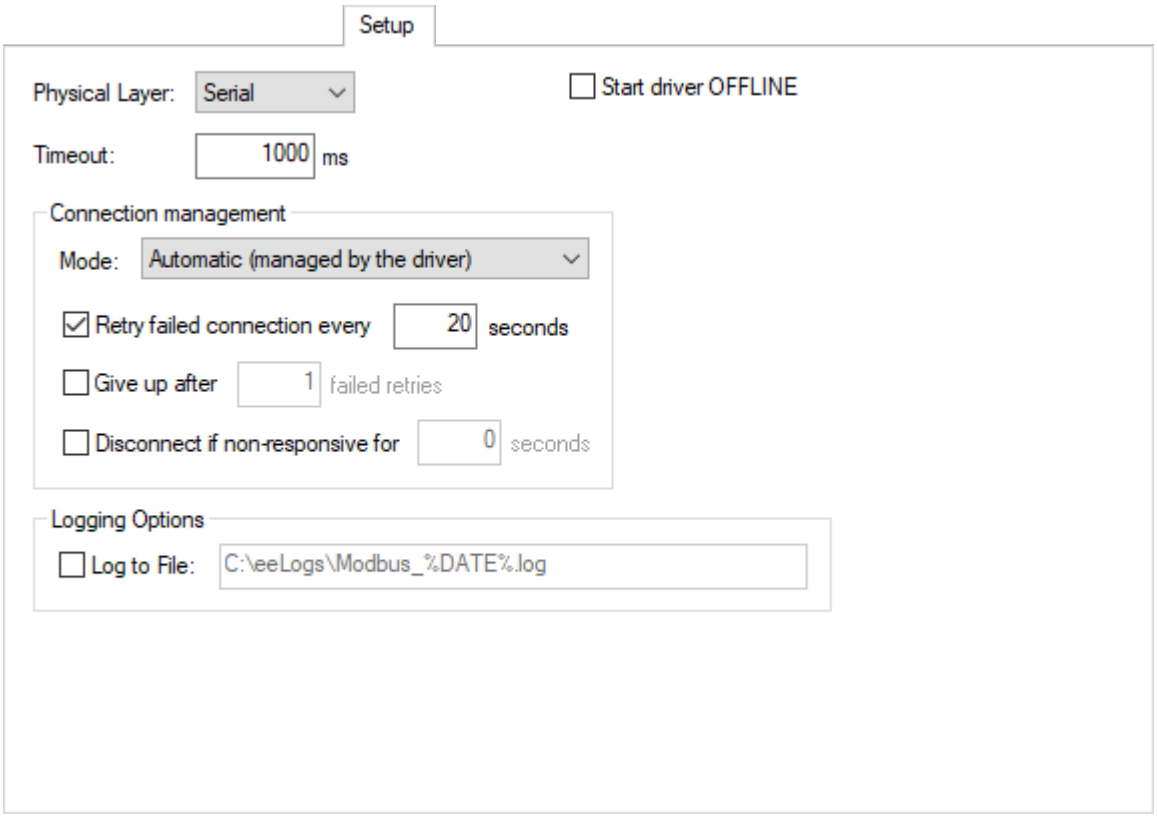
Configuration Dialog Box

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

Setup Tab

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

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



Setup tab

General options on the Setup tab

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

Options on the Connection management group

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

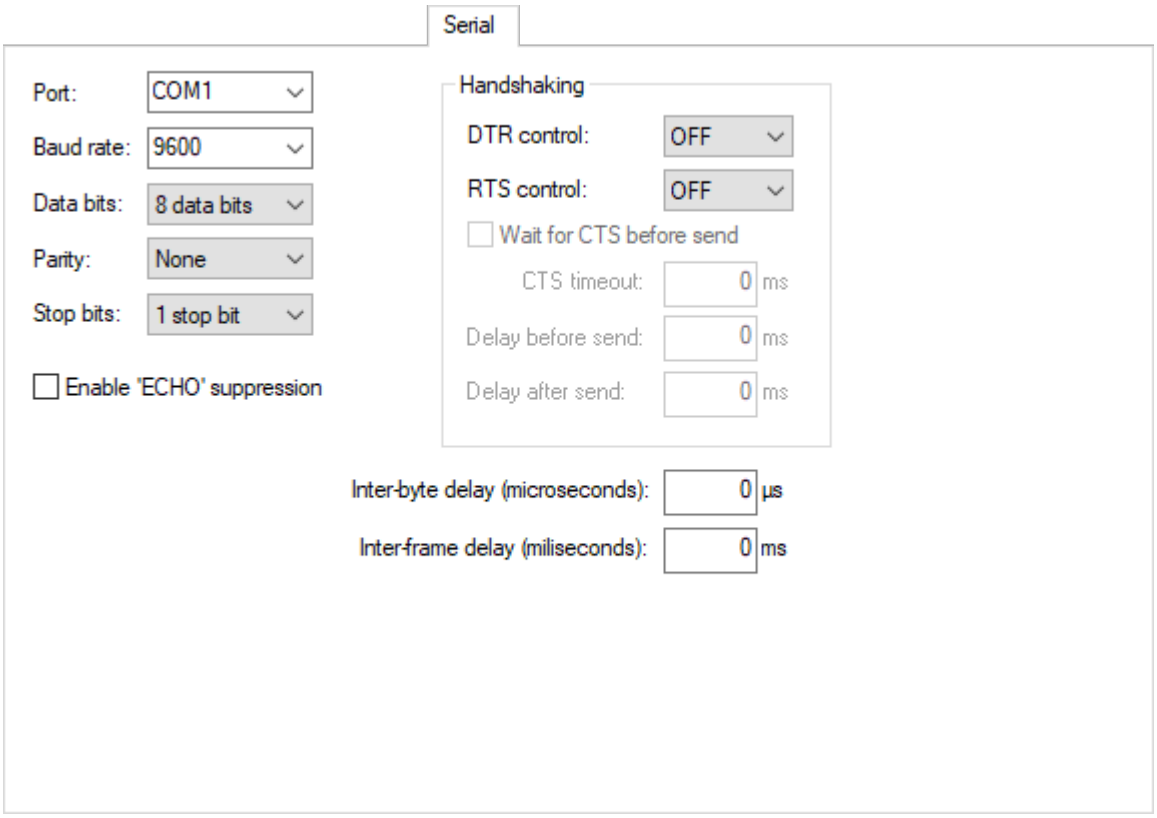
OPTION	DESCRIPTION
	manage a connection. Please check topic **Driver Statuses** for more details
Retry failed connection every ... seconds	Select this option to enable a Driver's connection retry in a certain interval, in seconds. If the **Give up after failed retries** option is not selected, the Driver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

Options on the Logging Options group

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

Serial Tab

Use this tab to configure parameters for a **Serial** Interface.



Serial tab

General options on the Serial tab

OPTION	DESCRIPTION
Port	Select a serial port on the list, from **COM1** to **COM4**, or type the name of a serial port in the format **COM*n***, such as "COM15". When typing a port's name manually, the dialog box only accepts port names starting with the expression "COM"
Baud rate	Select a baud rate on the list (**1200**, **2400**, **4800**, **9600**, **19200**, **38400**, **57600**, or **115200**) or type a baud rate, such as 600
Data bits	Select 7 (seven) or 8 (eight) data bits on the list
Parity	Select a parity on the list. The available options are **None**, **Even**, **Odd**, **Mark**, or **List**
Stop bits	Select the number of stop bits on the list. The available options are **1**, **1.5**, or **2** stop bits
Enable 'ECHO' supression	Enable this option to remove the echo received after the I/O Interface sends data via serial port. If this echo is not equal to the bytes just sent, then the I/O Interface aborts communication.
Inter-byte delay (microseconds)	Defines a delay between each byte transmitted by the I/O Interface, in millionths of a second, that is, 1000000 is equal to a second. This option must be used with small delays of less than a millisecond
Inter-frame delay (milliseconds)	Defines a delay between packets sent or received by the I/O Interface, in thousandths of a second, that is, 1000 is equal to a second. This delay is applied if the I/O Interface

OPTION	DESCRIPTION
	sends two consecutive packets, or between a received packet and the next sending

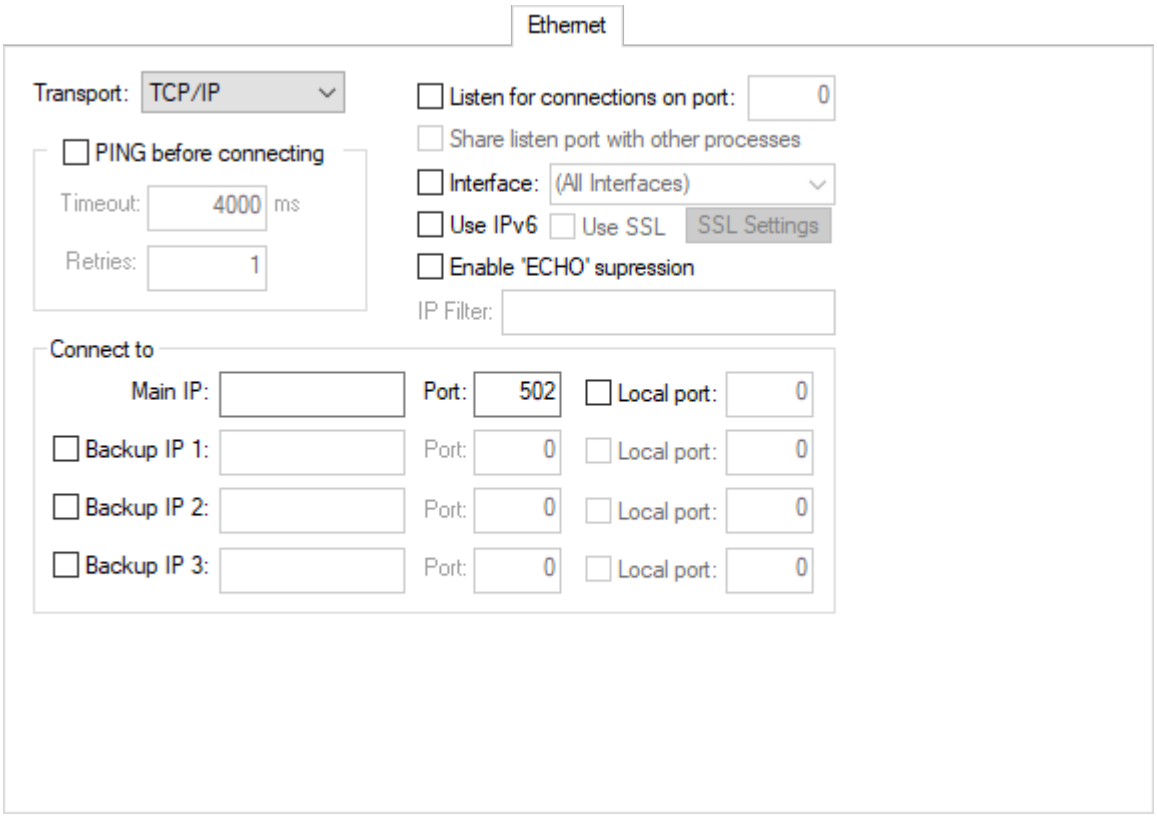
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process (controlling when data can be sent or received via serial line). Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works with RS232 serial lines as well as with RS485 serial lines.

Available options on the Handshaking group

OPTION	DESCRIPTION
DTR control	Select **ON** to keep the **DTR** signal always on while the serial port is open. Select **OFF** to turn the **DTR** signal off while the serial port is open. Some devices require the **DTR** signal always on to allow communication.
RTS control	Select **ON** to keep the **RTS** signal always on while the serial port is open. Select **OFF** to turn the **RTS** signal off while the serial port is open. Select **Toggle** to turn the **RTS** signal on while sending bytes via serial port and turn it off when not sending bytes, therefore enabling the reception.
Wait for CTS before send	Available only when the **RTS control** option is configured to **Toggle**. Use this option to force a Driver to check the **CTS** signal before sending bytes via serial port, after turning the **RTS** signal on. In this mode the **CTS** signal is handled as a permission flag for sending.
CTS timeout	Determines a maximum time, in milliseconds, that a Driver waits for the **CTS** signal after turning the **RTS** signal on. If the **CTS** signal is not turned on within this time-out, the Driver then fails the current communication and returns an error.
Delay before send	Some serial port hardware have a delay when enabling a data sending circuit after the **RTS** signal is turned on. Configure this option to wait a certain number of milliseconds after turning the **RTS** signal on and before sending the first byte. **IMPORTANT**: This delay must be used carefully, because it uses 100% of CPU resources while waiting. System's general performance degrades as this value increases.
Delay after send	This is the same effect of the **Delay before send** option, but in this case the delay is performed after sending the last byte, before turning the **RTS** signal off.

Ethernet Tab

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



Ethernet tab

Available options on Ethernet tab

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

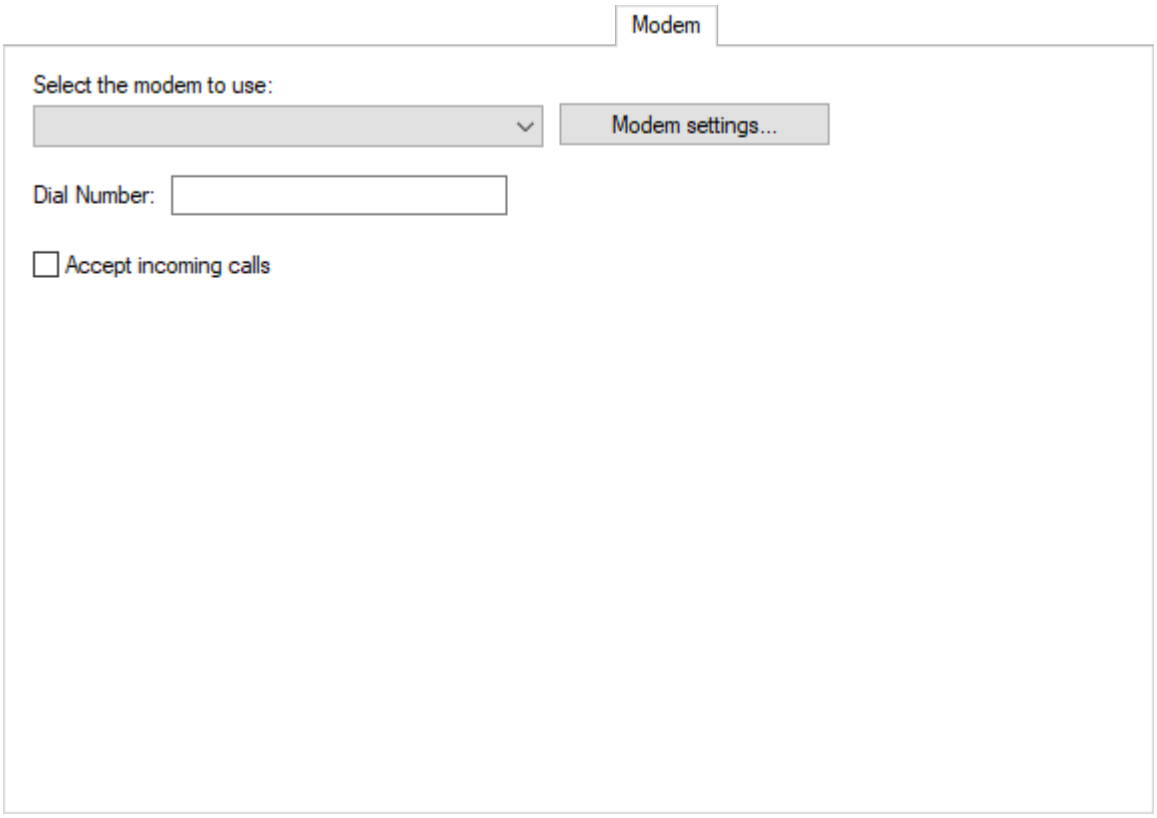
OPTION	DESCRIPTION
	open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from the **ping** command. Users must use the **ping** command to check the normal reply time, configuring this option for a value above that average. Usually this value can be configured between 1000 and 4000 milliseconds, that is, between one and four seconds • **Retries**: Number of retries of a **ping** command, not counting the first attempt. If all attempts fail, then the socket connection is aborted

Available options on the Connect to group

OPTION	DESCRIPTION
Main IP	Type the IP address of the remote device. Users can use an IP address separated by dots, as well as a URL. In case of a URL, the Driver uses the available DNS service to map that URL to an IP address, such as "192.168.0.13" or "Server1"
Port	Type the IP port of the remote device, between 0 (zero) and 65535
Local port	Select this option to use a fixed local IP port when connecting to a remote device
Backup IP 1, 2, and 3	Indicate here the IP address, the IP port, and the fixed local IP port of up to 3 (three) backup addresses of a remote device

Modem Tab

Use this tab to configure parameters of a **Modem** Interface. Some options on the **Serial** tab affect the modem configuration, therefore users must also configure the **Serial** Interface.



Modem tab

The **Modem** Interface uses the TAPI modems installed on the computer.

Available options on the Modem tab

OPTION	DESCRIPTION
Select the modem to use	Select a modem on the list of available modems on the computer. If the **Default modem** option is selected, then the first available modem is used. Selecting this option is recommended specially when an application is used on another computer
Modem settings	Click to open the configuration window of the selected modem
Dial Number	Type a default number for dialing. This value can be changed at run time. Users can use the **w** character to represent a pause or a waiting time for a dial tone. For example, "0w33313456" dials the number zero, waits, and then dials the number "33313456"
Accept incoming calls	Enable this option so that the Driver answers the phone when receiving an external call. To use this option, users must configure the **Connection management** option on the **Setup** tab to **Manual**

RAS Tab

Use this tab configure parameters of a **RAS** Interface. Users must also configure the **Ethernet** tab.

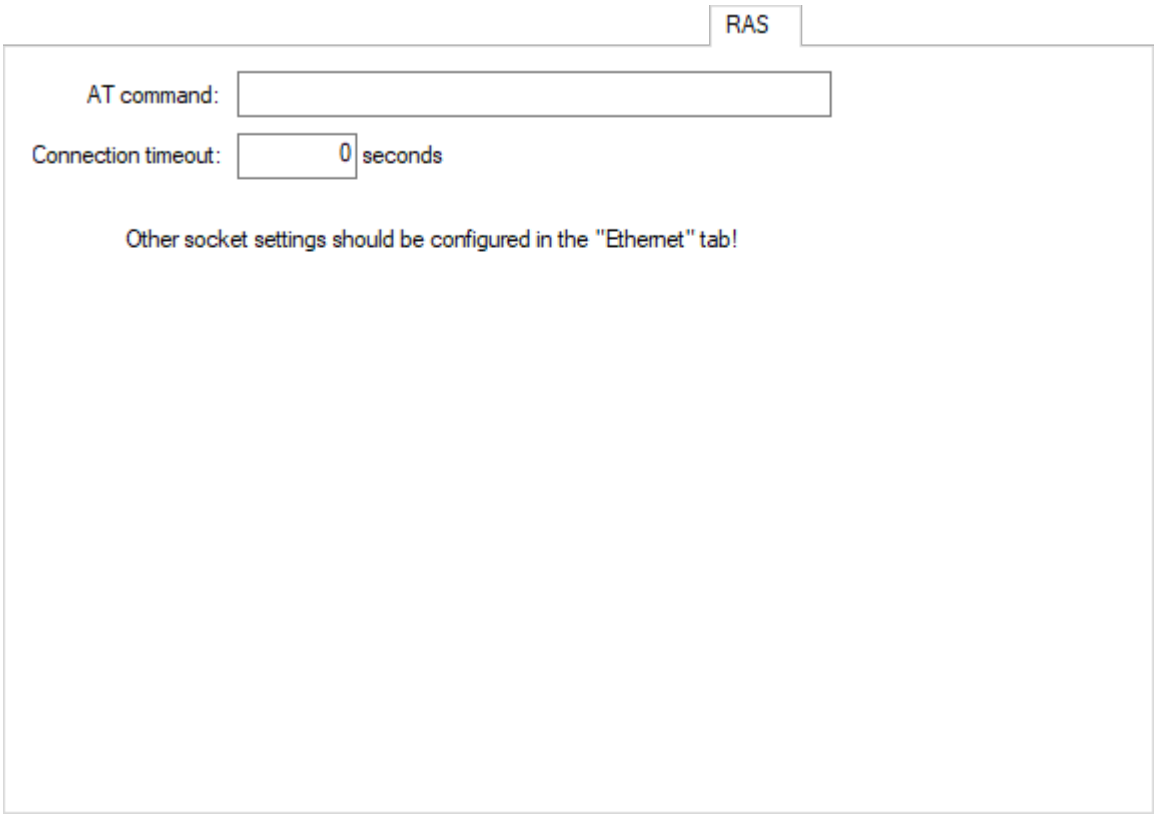
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

When connecting to a RAS device, first the I/O Interface**IOKit** connects to the socket on the IP address and port configured on the **Ethernet** tab. After opening the socket, the following initialization or connection steps are performed:

1. Clear the socket, that is, remove any TELNET greeting message received from a RAS device.
2. Send an **AT** dial message (in ASCII) in the socket.
3. Wait for a **CONNECT** reply.
4. If the time-out expires, the connection is aborted.
5. If the **CONNECT** reply is received within the time-out, the socket is available for communication with the device, that is, the connection was established.

If step 5 is successful, then the socket behaves as a normal socket, with the RAS device working as a router between the Driver and the device. Bytes sent by the Driver are received by the RAS device and sent to the destination device using a modem. Bytes received by the modem's RAS device are sent back to the Driver using the same socket.

After establishing the connection, the **RAS** interface monitors data received from a Driver. If a "NO CARRIER" **String** is found, the socket is closed. If the RAS device does not send a **NO CARRIER** signal, the **RAS** Interface cannot detect when the modem connection between the RAS device and the final I/O device fails. To recover from this failure, users are strongly advised to enable the **Disconnect if non-responsive** option on the **Setup** tab.



RAS tab

Available options on RAS tab

OPTION	DESCRIPTION
AT command	A **String** with the full **AT** command used to dial to a destination device. For example, "ATDT33313456" dials by tone to number "33313456"
Connection timeout	Number of seconds to wait for a modem's **CONNECT** reply, after sending an **AT** command

General Configurations

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

I/O Tags

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

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

IO.IOKitEvent

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

- **0**: Physical layer stopped, that is, the Driver is in **Offline** mode, the physical layer failed when initializing, or exceeded the maximum number of reconnection attempts
- **1**: Physical layer started but not connected, that is, the Driver is in **Online** mode but the physical layer is not connected. If the **Connection management** option is configured as **Automatic**, the physical layer can be connecting, disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured as **Manual**, then the physical layer remains in this status until forced to connect
- **2**: Physical layer connected, that is, the physical layer is ready for use. This **DOES NOT** mean the device is connected, only the access layer is working

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Ethernet Interface Configuration

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

I/O Tags

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

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

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

IO.Ethernet.IPSelect

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

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

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

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

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

IO.Ethernet.IPSwitch

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

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

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

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

IO.Ethernet.BackupEnable[2,3]

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

IO.Ethernet.BackupLocalPort[2,3]

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

IO.Ethernet.BackupLocalPortEnable[2,3]

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

IO.Ethernet.BackupIP[2,3]

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

IO.Ethernet.BackupPort[2,3]

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. Examples:

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

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

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

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

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

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

IO.Ethernet.ListenIP

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

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

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

IO.Ethernet.MainLocalPort

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

IO.Ethernet.MainLocalPortEnable

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

IO.Ethernet.MainPort

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

IO.Ethernet.PingEnable

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

IO.Ethernet.PingTimeoutMs

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

IO.Ethernet.PingTries

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

IO.Ethernet.ShareListenPort

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

IO.Ethernet.SupressEcho

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

IO.Ethernet.Transport

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

IO.Ethernet.UseIPv6

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

Modem Interface Configuration

This section contains information about the configuration of **I/O Tags** and **Properties** of a **Modem** (TAPI) Interface.

I/O Tags

Tags of a Modem Interface (N2/B2 = 3)

The Tags described next allow controlling and diagnosing a **Modem** (TAPI) Interface at run time.

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

IO.TAPI.ConnectionBaudRate

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	5 (five)
String Configuration	IO.TAPI.ConnectionBaudRate

Indicates a baud rate value for the current connection. If a modem is not connected, returns the value 0 (zero).

IO.TAPI.Dial

Type of Tag	I/O Tag
Type of Access	Write-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	1 (one)
String Configuration	IO.TAPI.Dial

Write any value to this Tag to force a **Modem** Interface to start a call. This is an asynchronous command, only starting the call process. Users can monitor the **IO.TAPI.IsModemConnected** Tag to detect when a call is established.

IO.TAPI.HangUp

Type of Tag	I/O Tag
Type of Access	Write-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	4 (four)
String Configuration	IO.TAPI.HangUp

Any value written to this Tag turns the current call off.

NOTE

Use this command only when managing the physical layer manually or when explicitly trying to force a Driver to restart the communication. If the physical layer is configured for automatic reconnection, the Driver immediately tries to reestablish the connection.

IO.TAPI.IsModemConnected

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	3 (three)
String Configuration	IO.TAPI.IsModemConnected

This Tag indicates the status of a modem connection. Possible values are **0**: The modem is not connected, but it may be performing or receiving an external call or **1**: The modem is connected and the Driver completed or received an external call successfully. While it is in this status, the physical layer can send or receive data.

IO.TAPI.IsModemConnecting

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	6 (six)
String Configuration	IO.TAPI.IsModemConnecting

This Tag indicates the status of a modem connection, with more details than the **IO.TAPI.IsModemConnected** Tag. Possible values are **0**: Modem is not connected, **1**: Modem is connecting, that is, performing or receiving an external call, **2**: Modem is connected. While in this status, the physical layer can send or receive data, or **3**: Modem is disconnecting the current call.

IO.TAPI.ModemStatus

Type of Tag	I/O Tag
Type of Access	Read-Only
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	2 (two)
String Configuration	IO.TAPI.ModemStatus

Returns a **String** with the current status of a modem. Possible values are the following:

- **"No status!"**: The **Modem** Interface was not open yet or was already closed
- **"Modem initialized OK!"**: The **Modem** Interface was initialized successfully
- **"Modem error at initialization!"**: Driver could not initialize modem's line. Check Driver's log file for more details
- **"Modem error at dial!"**: Driver could not start or accept a call
- **"Connecting..."**: Driver started a call successfully, and is currently processing that call
- **"Ringing..."**: Indicates that the modem is receiving an external call, but it did not accepted it yet
- **"Connected!"**: Driver connected successfully, that is, completed or accepted an external call
- **"Disconnecting..."**: Driver is turning the current call off
- **"Disconnected OK!"**: Driver turned the current call off
- **"Error: no dial tone!"**: Driver aborted a call because the available line signal was not detected
- **"Error: busy!"**: Driver aborted a call because the line was busy
- **"Error: no answer!"**: Driver aborted a call because no answer was received from the other modem
- **"Error: unknown!"**: Current call was aborted because of an unknown error

IO.TAPI.PhoneNumber

Type of Tag	I/O Tag
Type of Access	Reading or Writing
N1 Parameter	-1 (minus one)
N2 Parameter	0 (zero)
N3 Parameter	3 (three)
N4 Parameter	0 (zero)
String Configuration	IO.TAPI.PhoneNumber

This Tag is a **String** that reads or changes the telephone number used by the **IO.TAPI.Dial** Tag. When changing this Tag, the new value is used only on the next **Dial** command.

Properties

These properties control the configuration of a **Modem** (TAPI) Interface.

IO.TAPI.AcceptIncoming

9 Configure to False if a modem cannot accept external calls, that is, the Driver behaves as a master, and configure to True to enable receiving calls, that is, the Driver behaves as a slave.

IO.TAPI.ModemID

9 This is the identification number of a modem. This ID is created by Windows and used internally to identify a modem on a list of devices installed on a computer. This ID may not remain valid if a modem is reinstalled or an application is executed on another computer.

NOTE

It is advisable to configure this property as 0 (zero), indicating that a Driver must use the first available modem.

IO.TAPI.PhoneNumber

A telephone number used by **Dial** commands, such as "0w01234566", where the "w" character forces a modem to wait for a call sign.

RAS Interface Configuration

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

I/O Tags

Tags of a RAS Interface (N2/B2 = 5)

Currently, there are no Tags defined specifically to manage a **RAS** Interface at run time.

Properties

These properties control the configuration of a **RAS** Interface.

NOTE

A **RAS** Interface uses the **Ethernet** Interface, which for this reason must be also configured.

IO.RAS.ATCommand

An **AT** command to send through a socket to force a RAS device to perform a call using the current RAS channel, such as "ATDT6265545".

IO.RAS.CommandTimeoutSec

Time to wait for a **CONNECT** message in response to an **AT** command, in seconds.

Serial Interface Configuration

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

I/O Tags

Tags of a Serial Interface (N2/B2 = 2)

Currently, there are no Tags defined specifically to manage a **Serial** Interface at run time.

Properties

These properties control the configuration of a **Serial** Interface.

IO.Serial.Baudrate

Specifies a baud rate of a serial port, such as 9600.

IO.Serial.CTSTimeoutMs

Time to wait for a **CTS** signal, in milliseconds. After turning the **RTS** signal on, a timer is started to wait for a **CTS** signal. If this timer expires, the Driver aborts sending bytes through the serial port. Available only when the **IO.Serial.RTS** property is configured as **Toggle** and the **IO.Serial.WaitCTS** property is configured to True.

IO.Serial.DataBits

Specifies the number of data bits to configure a serial port. Possible values are **5**: Five data bits, **6**: Six data bits, **7**: Seven data bits, or **8**: Eight data bits.

IO.Serial.DelayAfterMs

Number of milliseconds to delay after the last byte is sent through a serial port, but before turning the **RTS** signal off. Available only when the **IO.Serial.RTS** property is configured to **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

IO.Serial.DelayBeforeMs

9 Number of milliseconds to delay after turning the **RTS** signal on, but before data is sent. Available only when the **IO.Serial.RTS** property is configured to **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

IO.Serial.DTR

A Indicates how a Driver deals with the **DTR** signal. Possible values are **OFF**: **DTR** signal is always turned off or **ON**: **DTR** signal is always turned on.

IO.Serial.InterbyteDelayUs

9 Delay time, in milliseconds (1/1000000 of a second), for each two bytes sent through a **Serial** Interface.

IO.Serial.InterframeDelayMs

9 Delay time, in milliseconds, before sending a packet after the last packet sent or received.

IO.Serial.Parity

A Specifies a parity for the configuration of a serial port. Possible values are **E or Even**: Even parity, **N or None**: No parity, **O or Odd**: Odd parity, **M or Mark**: Mark parity, or **S or Space**: Space parity.

IO.Serial.Port

9 Number of the local serial port. Possible values are **1**: Uses the COM1 port, **2**: Uses the COM2 port, **3**: Uses the COM3 port, or **n**: Uses the COM*n* port.

IO.Serial.RTS

A Indicates how a Driver deals with the **RTS** signal. Possible values are **OFF**: **RTS** signal always off, **ON**: **RTS** signal always on, or **Toggle**: Turns the **RTS** signal on when transmitting data and turns the **RTS** signal off when not transmitting data.

IO.Serial.StopBits

9 Specifies the number of stop bits for the configuration of a serial port. Possible values are **1**: One stop bit, **2**: One and a half stop bit, or **3**: Two stop bits.

IO.Serial.SupressEcho

9 Use a value different from 0 (zero) to enable suppressing the echo or 0 (zero) to disable it.

IO.Serial.WaitCTS

◩ Configure to True to force a Driver to wait for the **CTS** signal before sending bytes when the **RTS** signal is turned on. Available only when the **IO.Serial.RTS** property is configured to **Toggle**.

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.1	08/24/2022	C. Mello	• 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)