Introduction

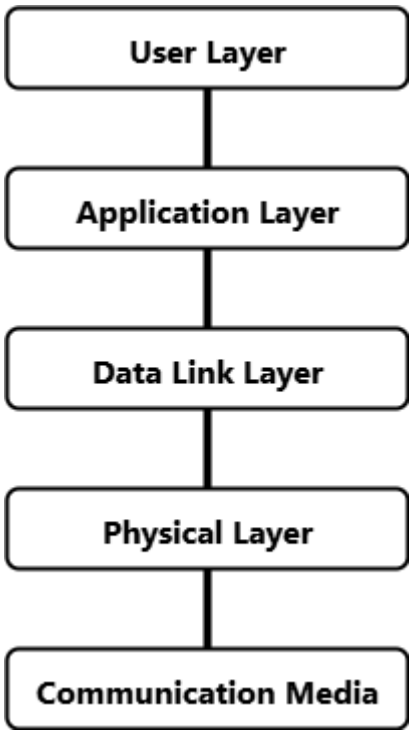
File Name	DNPMaster.dll
Manufacturer	DNP (Distributed Network Protocol)
Devices	
Protocol	DNP 3.0
Version	4.0.58
Last Update	09/02/2025
Platform	Win32
Dependency	IOKit version 2.0 or later
Superblock Readings	No
Level	31201

Introduction

This Driver implements protocol DNP 3.0 in **Master** mode, according to level two and some level three functionality.

DNP (*Distributed Network Protocol*) is an open and non-proprietary communication protocol, based on IEC (*International Eletrotechnical Commission*) specifications, adapted for usage on highly secure applications, with moderated speed and amount of data. It is extremely flexible and can be used in any hardware platform.

The model specified by ISO OSI (*International Standards Organization - Open System Interconnection*) establishes seven layers for a network protocol. IEC, on the other hand, specifies a simplified model, which consists only on physical, data link, and application layers. Such model is called EPA (*Enhanced Performance Architecture*). The next figure shows an EPA structure and its communication system.



EPA architecture

A User Layer can be defined as a location where users manipulate data, after all communication. In **Elipse Software** products, it is represented by a user application. The User Layer uses this Driver's Application Layer to send and receive full messages to or from a station.

An Application Layer is responsible for specifying in details the requests from the User Layer, and back to it when the message comes from the Data Link Layer. In other words, it assembles messages from User Layer, called fragments, into a multiple-fragment message with full information to be processed and sent to a station using the Data Link Layer.

A Data Link Layer is used to pass messages between primary, or origin, and secondary, or destination, stations. It also packs data, checks transmission errors, and sends them to a TCP/IP network.

DNP protocol can be configured to switch messages via polling (constant communication) or via integrity or changes (more efficient). Sending changes, also known as RBE (*Report by Exception*), can occur spontaneously, or unsolicited, or non-spontaneously by an explicit request from a Master for these changes. It is recommended to use the following configurations:

- **Enable Class 0 (zero) during startup and at regular intervals**: This way all Tags have a value when starting an application
- **Use unsolicited messages or event scanning at regular intervals**: Data updates, as changes occur, can be sent in an unsolicited way by a Slave or using Classes 1 (one), 2 (two), and 3 (three) event requests automatically by this Driver
- **Configure Tags using event objects instead of static objects**: Tags configured as static objects generate communication by polling (constant exchange of messages), which generates unnecessary traffic. Tags configured as events do not perform communication and are updated automatically as integrity or change messages arrive, as exposed previously

Function Codes

A function code identifies the purpose of a message. There are two groups of functions, one for requests and another one for responses. There are several types of functions for requests, as shown on the next table. Response functions are used internally and not open to users.

Function Codes

CODE	FUNCTION	DESCRIPTION
1	Read	Requests objects specified by a remote station and responds with the requested objects available
2	Write	Stores the objects specified by a remote station and responds with an operation status
3	Select	Selects or operates output points but does not configure or produce any action (controls, SetPoints, or analog outputs) and responds with an operation status. The **Operate** function must be used to activate these outputs.
4	Operate	Configures or produces actions on outputs or points previously selected with the **Select** function
5	Direct Operate	Selects and operates the specified outputs and responds with the status of control points

CODE	FUNCTION	DESCRIPTION
6	Direct Operate NO ACK	Selects and operates the specified outputs but does not send a response, but takes less time
7	Immediate Freeze	Copies the specified objects to a freezing buffer
9	Freeze and Clear	Copies the specified objects to a freezing buffer and then zeroes these objects
10	Freeze and Clear No ACK	Copies the specified objects to a freezing buffer and then zeroes these objects, but does not send a confirmation response
11	Freeze with Time	Copies the selected objects to a freezing buffer at the specified moment, responding with an operation status
13	Cold Restart	Performs a reset sequence
14	Warm Restart	Performs a partial reset sequence
20	Enable Unsolicited Messages	Enables a spontaneous notification for the specified objects
21	Disable Unsolicited Messages	Disables a spontaneous notification for the specified objects
22	Assign Classes	Links the specified objects to a class
23	Delay Measurement	Allows an application to calculate a delay, or propagation time, to a specific station

After specifying the purpose of a message (APCI), there is a second part of that message, if needed, called ASDU. Every ASDU consists of one or more data identifiers (DUI), object headers, object information, or data fields.

Object Headers

An object header of a message identifies data objects inside this message or the ones that are used in a response for this message. It is composed basically by the elements shown on the next figure.

Object header

Objects

Smart devices using DNP protocol's Application Layer can monitor, control, or produce a large amount of data. This data, called information elements, is processed and stored as an information object, which is standardized to produce ways to describe it and represent it in a unique way. The types of categories for data objects are the following:

- **Static Objects**: These are objects that reflect the current value of a field or internal variable

- **Event Objects**: These are objects generated as a result of a value change or other trigger. These are historical objects, that is, they reflect a data value at an instant in the past
- **Frozen Static Objects**: Reflect a current frozen value of a field or internal variable. Data is frozen as a result of a data-freezing request
- **Frozen Event Objects**: These objects are the result of a change in a frozen value or other trigger. These are historical objects, that is, they reflect a data value at an instant in the past

Every category is represented by a different object, as described on the next table.

Categories of objects

OBJECT	DESCRIPTION
Digital Inputs	A group of digital inputs contains all objects that represent binary inputs (status or Boolean attributes). Ranges from 1 (one) to 9 (nine)
Digital Outputs	A group of digital outputs contains all objects that represent binary outputs or control information on relays. Ranges from 10 to 19
Counters	This group contains all counter objects. Ranges from 20 to 29
Analog Inputs	Contains all analog inputs. Ranges from 30 to 39
Analog Outputs	Contains all analog outputs. Ranges from 40 to 49
Time	Contains all objects that represent time in an absolute or relative form. Ranges from 50 to 59
Classes	This group contains all objects that represent data classes or data priorities. Ranges from 60 to 69
Files	Files or file system. Ranges from 70 to 79
Devices	Ranges from 80 to 89
Applications	Objects that represent operating system applications or processes. Ranges from 90 to 99
Alternative numbers	Custom number representations. Ranges from 100 to 109

It is important to notice that a static object, when suffering a variation, is capable of generating an event object indicating that change. However, both represent the same object.

Variation

These are changes or sub-types that may occur on objects. As an example, a digital input can be represented by a single bit (zero or one), by a status word (one byte), or yet contain or not a time information (timestamp). Thus, a combination of an object plus a variation completely describes an information, according to the next examples.

- **Object 01 - Variation 01**: Represents a digital input without status (a single bit)
- **Object 01 - Variation 02**: Represents a digital input with status (one byte)
- **Object 02 - Variation 01**: Represents a digital input change without time information
- **Object 02 - Variation 02**: Represents a digital input change with time information

Although both objects are linked to the same digital input, this information can be represented in different ways.

Qualifier

Specifies the meaning of the **Range** field, described next.

Range

Indicates the number of objects, starting and ending indexes, or identifiers for the objects.

Classes

Objects declared in a system or device that implements DNP protocol in **Slave** mode can be grouped into classes. DNP protocol defines the following classes:

- **Class 0**: It means all objects, that is, when starting this Driver, on the Master side, can perform a class 0 (zero) request, and as a response the Slave side sends the current value of all declared objects
- **Classes 1 to 3**: These are entities that temporarily store lists of events or changes in objects. Each object must be configured on the Slave side to generate events when changes occur, and usually there is a pattern among DPN users to reserve class 1 for digital events, class 2 for analog events, and class 3 for counters

Driver Configuration

This section contains information about the configuration of DNP 3.0 Master Driver.

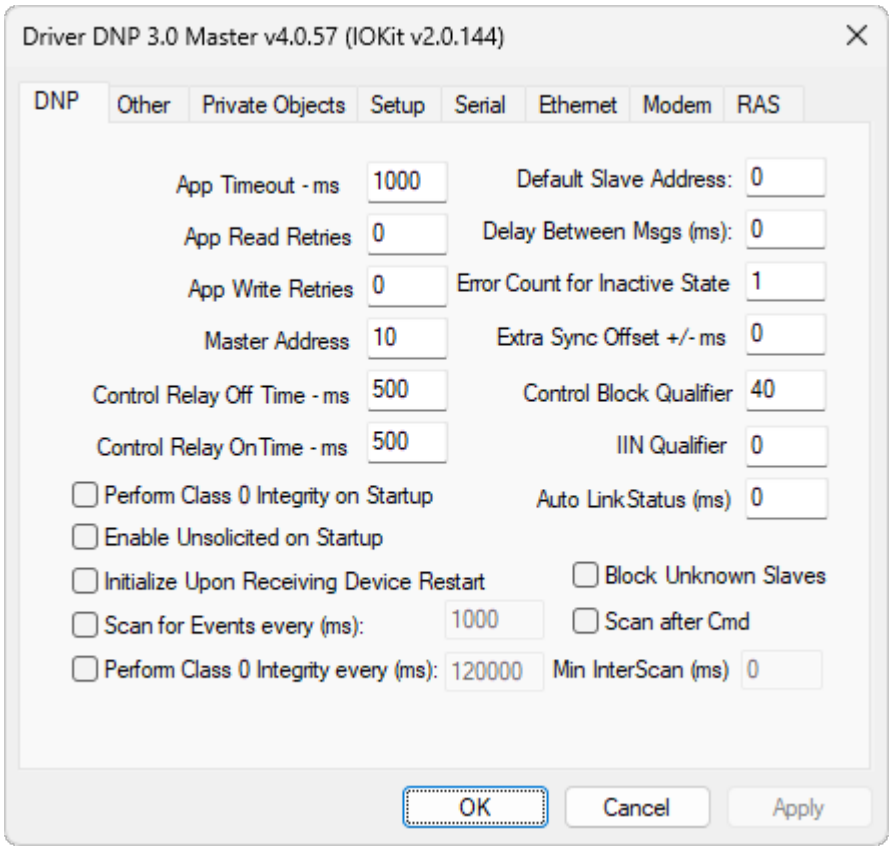
P Configuration Parameters

P1	Not used, keep its value in 0 (zero)
P2	Not used, keep its value in 0 (zero)
P3	Not used, keep its value in 0 (zero)
P4	Keep its value in 0 (zero). When using a Toshiba Regulator device, configure it as 1 (one)

Extra Configurations

This section contains configurations for this Driver contained on **DNP**, **Other**, and **Private Objects** tabs, described on the next topics.

DNP Tab



DNP tab

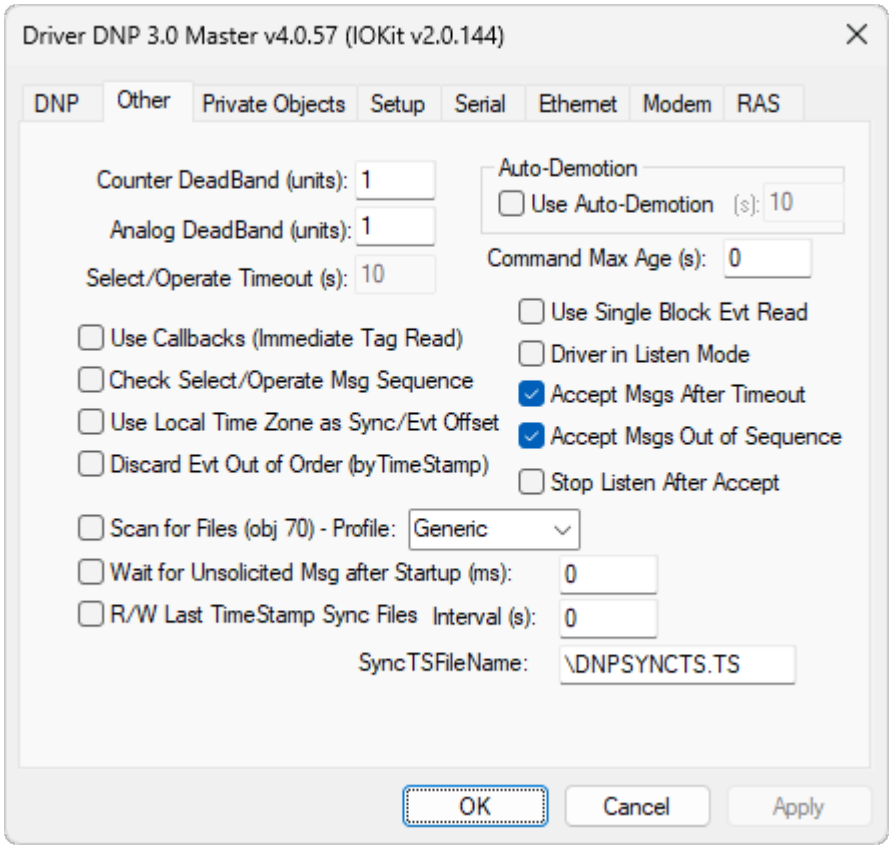
Available options on the DNP tab

OPTION	DESCRIPTION
App Timeout (ms)	This is the maximum time the application layer waits for a full response from Data Link layer. If receiving a request is ongoing by the Data Link layer, this time is automatically extended up to the end of the reception from the Data Link layer. The default value of this option depends on the communication rate used, but it is recommended that this value be equal or greater than **IOKit** library's time-out, on the **Setup** tab. This value represents a message's byte-to-byte time-out, while the **App Timeout (ms)** option represents one or more full Data Link messages
App Read Retries	Number of communication retries performed by the application layer in case of a reading error. Default value of this option is 0 (zero)
App Write Retries	Number of communication retries performed by the application layer in case of a writing error. Default value of this option is 0 (zero)
Master Address	This is the address of the Master station (PC)
Control Relay Off-Time	When relay commands are sent (Control Relay Commands) with individual Tags, this option indicates a normal turned-off time for **Pulse On/Off** or **Latch On/Off** commands
Control Relay On-Time	When relay commands are sent (Control Relay Commands) with individual Tags, this option indicates a normal turned-on time for **Pulse On/Off** or **Latch On/Off** commands

OPTION	DESCRIPTION
Default Slave Address	Device's default DNP address, to use when the *N1* parameter of each Tag is configured with 0 (zero)
Delay Between Messages	Delay time to apply between each message sent by this Driver, in milliseconds
Error Count for Inactive State	Indicates how many consecutive errors this Driver must consider to switch a device to an **Inactive** status. This Driver tries to communicate with this device at the next scan of any Tag on the same address or only at the time informed in the **Demotion Time** property, if this resource is used
Extra Sync Offset	Additional time, positive or negative, to add to syncing commands
Control Block Qualifier	When performing a writing command of a control block, such as **Direct Operate**, **Select**, or **Operate** digital output commands, users must send a qualifier, which may vary from device to device. Possible values for this option are 39 (27h) for the **Range** field with a byte and a prefixed object with two bytes, or 40 (28h) for the **Range** field with two bytes and a prefixed object with two bytes. Please check the equipment's device profile for a correct qualifier
IIN Qualifier	When a Slave device informs a restart, this Driver performs a writing to an Internal Indications Object informing the acknowledgment of this event. Use this option to inform a qualifier, which is on the Slave Device Profile document of the slave device. The most common values are 0 (zero, one starting and ending index byte) and 1 (one, two starting and ending index byte)
Auto LinkStatus (ms)	Interval, in milliseconds, to send a LinkStatus Request message, which must receive a LinkStatus Response. The lack of reception puts this address in an Inactive state. To disable this option, configure its value as 0 (zero)
Perform Class 0 Integrity On Startup	Performs a Class 0 (zero) request when starting this Driver, which gets the whole device's database, thus updating all Tags. Event Tags, in this case, have their **TimeStamp** property as the time of data reception and the **Quality** property equal to 216
Enable Unsolicited on Startup	Indicates whether this Driver must send a command to enable unsolicited messages when starting communication
Initialize Upon Receiving Device Restart	Indicates whether this Driver must send a command to start a device (reset link, Class 0, and enable unsolicited, if enabled) when receiving a message that this device was restarted
Scan for Events Every X ms	Sends a Class 1 (one), 2 (two), or 3 (three) reading command to check if there are events on a device. Indicates a time interval in which this command must be repeated.
Min InterScan ms	If after any communication, or even after an event scan, there are pending events on the Slave, which is verified by IIN bits, this Driver requests new events from the pending class or classes. To limit this communication and thus

OPTION	DESCRIPTION
	avoiding from happening continuously, users can define a minimum interscan interval in this option. The default value is 0 (zero), so that if there are still pending events, they are requested immediately, keeping the behavior of previous versions
Perform Class 0 Integrity every X ms	Performs a Class 0 (zero) request cyclically, only to check if this Driver's database is equal to device's database. The default time is 10 or 15 minutes
Block Unknown Slaves	Blocks the creation of new Slaves upon receiving unsolicited messages from addresses undeclared in any application Tag
Scan After Cmd	If the **Scan for Events Every X ms** option is enabled, this option enables requesting a new scan immediately after a command, so that users can get a state change faster

Other Tab



Other tab

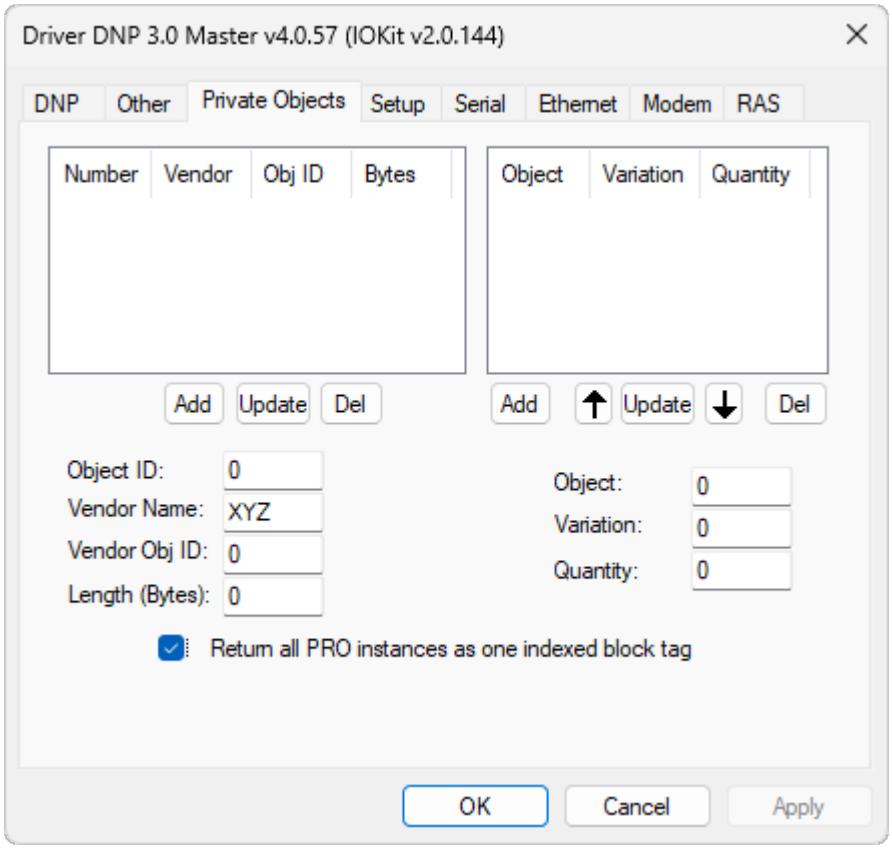
Available options on the Other tab

OPTION	DESCRIPTION
Counter DeadBand (units)	Dead band in units, to verify when checking new events for counters
Analog DeadBand (units)	Dead band in units, to verify when checking new events for analog points
Select/Operate Timeout (s)	Maximum time for blocking other messages, except an **Operate** command right after a **Select** command

OPTION	DESCRIPTION
Use Auto Demotion	Enables an auto-demotion system, which automatically removes and inserts from communication the Slaves in an **Inactive** status, that is, with communication errors. This procedure is used when two or more Slaves use the same channel, thus avoiding that a Slave in this status takes over the channel
Demotion Time	Time, in seconds, that this Driver tries to communicate with every Slave in an **Inactive** status, to check if communication is correct, setting it back to an **Active** status
Command Max Age (s)	Verifies if the timestamp of the requested command is greater than local timestamp minus the number of seconds defined in htis option, to avoid processing old commands that for any reason remain pending
Use Callbacks	This Driver can be configured to use callbacks or asynchronous calls to **Elipse E3**, **Elipse Power**, or **Elipse Water**, which immediately receives values from objects arriving on the communication, and in this case the scan rate is not used. If this option is disabled, Tags take more time to update, more than the defined scan time in average
Check Select/Operate Message Sequence	Allows using **Operate** commands to control a message's sequence number, so that it is immediately subsequent to a **Select** command. When selecting this option, this Driver does not send a command, except for an **Operate** command, until a maximum time for this operation is exceeded
Use Local Time Zone as Sync/Event Time Offset	Informs this Driver to consider the difference between Windows official time and UTC/GMT time for all events and time-syncing requests. This function is useful when a DNP Server is configured as UTC but users want to inform a local time to an application
Discard Events out of Order (by TimeStamp Info)	For objects with timestamp information, this option forces events whose timestamp is older than the last one processed to be discarded
Scan for Files (obj 70)	Requests a file reading using DNP's object 70, which is deprecated by the standard but still implemented in some devices. The available profiles are Pextron and Generic. For more information, please check topic **Collecting and Storing Files in COMTRADE Standard**
Wait for Unsolicited Messages after Startup (ms)	Indicates how long this Driver must wait for unsolicited messages after connecting, to get recent events before performing an integrity request, or Class 0 (zero). This way, points with events get a quality of 192 and a correct timestamp, as in integrity responses objects do not have a timestamp
R/W Last TimeStamp Sync Files	With this option, the last timestamp of each Tag is saved to a file in pre-defined intervals, starting when a new event is received. When starting, this Driver tries to read this file to initialize timestamps and discard older events. The generated file can be copied among redundant stations.

OPTION	DESCRIPTION
	File name and extension can be changed by users, and its content is in **ASCII** format
Use Single Block Evt Read	By using this option, all events sent by a device are reported in a single Block Tag. This option is useful to concentrate all events and generate specific event logs, for example. **NOTE**: Users must create a Block Tag as described on topic **Single Block Events** to remove received events
Driver in Listen Mode	In this mode, this Driver does not send anything, but it may interpret responses, received from a communication sniffer between other Masters and a Slave
Accept Msgs After Timeout	Indicates whether this Driver must accept a response to a message requested after a time-out defined in the **App Timeout (ms)** option. The default value of this option is False to keep compatibility with previous versions
Accept Msgs Out of Sequence	Indicates whether this Driver must accept a response to a requested message if the application's sequence number is different from expected

Private Objects Tab



Private Objects tab

Private objects are declarations that can be created by every manufacturer and reflect composed data structures of basic data types of DNP protocol.

The Private Objects tab displays a list on the left where objects are declared, composed by an index (**Number**), a code with up to four characters for the manufacturer (**Vendor**), an object's identification number for the manufacturer (**Object ID**), and the object's size in bytes (**Length**). This list can be edited using the **Add**, **Update**, and **Del** options.

Click an object to insert DNP objects, which are part of each PROs (*Private Registration Objects*) on the list on the right. Users must inform an object's code (**Object**), a **Variation**, and a **Quantity**. The order in which these objects appear is also important. This can be changed by using the arrow keys or using the **Add**, **Update**, and **Del** options.

When using a declaration of time objects, that is, type **50**, this declaration is used as a timestamp for that PRO object.

The **Return all PRO instances as one indexed block tag** option allows all PRO objects to be sent to the same Block Tag, as long as it has the same DNP address. In this case, the Block Tag must be created with one more Element than the PRO object declaration, because the first Element contains the object's index, followed by the other Elements, each one with one of the variables declared in the PRO object.

N-Addressing Parameters of PLC Tags

N1	Slave address or 0 (zero) to use a **Default Slave Address**
N2	Code of a function to perform. For more information, please check table **Supported Function Codes**
N3	Code of an object and variation. For more information, please check table **Supported Objects**
N4	Address or number of a variable

NOTES

- The *N3* parameter must be informed as a formula, *Object Code* × *100* + *Variation*. *Object Code* is the type of object, such as *Binary Inputs*, and *Variation* is a sub-type. Please check table **Supported Objects** for information about supported objects and variations.
- The *N4* parameter is an address or number of a variable, regardless if this is a physical or logical point.

Function Codes

Supported function codes

N2	READING OR WRITING	OPERATION
-34	Write-only	Data for analog channels of a COMTRADE file (Pextron profile). The N3 parameter indicates a channel, from 0 (zero) to 7 (seven). Users must use a Block with up to eight Elements. The Elements must be **0**: Channel's Name, **1**: A Constant, **2**: B Constant, **3**: Maximum Channel's Value, **4**: Minimum Channel's Value, **5**: RPV Constant, **6**: RSV Constant, and **7**: Primary or Secondary Circuit (can be **p**, **P**, **s**, or **S**)
-33	Reading and writing	Data for Pextron profile transfer. The values for the *N3* parameter can be **0**: Indicates that a file was just recorded (pulse in one), **1**: Number of files left for transmission, **2**: Requests a collecting on the Pextron file (instantaneous), or **3**: Requests the start of a random counting to collect a

N2	READING OR WRITING	OPERATION
		Pextron file (counts the time to request that collecting)
-32	Reading and writing	Base name of a COMTRADE file
-31	Reading and writing	Circuit name for COMTRADE files
-30	Reading and writing	Directory to store files
-22	Read-only	Returns a **Word** containing **Internal Indications**, **IIN1** on **Low** byte and **IIN2** on **High** byte. Possible values for bits are **0**: All station messages received, **1**: Class 1 data available, **2**: Class 2 data available, **3**: Class 3 data available, **4**: Time sync required, **5**: Points in local status, **6**: Device trouble, **7**: Device restart, **8**: Not implemented, **9**: Requested objects unknown, **10**: Parameters not valid, **11**: Buffer overflow, **12**: Operation already executing, **13**: Configuration corrupted, **14**: Not used, and **15**: Not used
-21	Read-only	Address indicated in the *N1* parameter is active (one) or inactive (zero)
-20	Reading and writing	Address indicated in the *N1* parameter is turned on (one) or turned off (zero)
-15	Write-only	Requests a **Link Status** command to be sent. The result is displayed in a Tag with the *N2* parameter equal to -10
-14	Write-only	Requests a restart (Reset Link, Class 0, etc.)
-13	Reading and writing	Communication statistics. Possible values for the *N4* parameter are **0**: Clears statistics (read-only), **1**: Frames sent, **2**: Frames without response, **3**: CRC errors in response format, **4**: Sending errors, **5**: Transmission retries, **6**: Frames received, **7**: Bytes sent, or **8**: Bytes received
-11	Reading and writing	Requests Class 0 (Integrity)
-10	Read-only	Link Status
-7	Write-only	Time syncing (Ethernet procedure)
-6	Write-only	Time syncing. Writes computer's time to a remote station
0	Read-only	**Read from Cache**. Removes all events available for a Tag, leaving the last event as the current Tag's value. For static objects, performs a reading communication before (**Read** function)

N2	READING OR WRITING	OPERATION
1	Read-only	**Read Events**. Returns all events available for a Tag. For static objects, performs a reading communication before (**Read** function)
2	Write-only	Write
3	Write-only	Select
4	Write-only	Operate
5	Write-only	Direct Operate
6	Write-only	Direct Operate No ACK
7	Write-only	Immediate Freeze
9	Write-only	Freeze and Clear
10	Write-only	Freeze and Clear No ACK
11	Write-only	Freeze with Time
13	Write-only	Cold Restart
14	Write-only	Warm Restart
20	Write-only	Enable Unsolicited Messages
21	Write-only	Disable Unsolicited Messages
22	Write-only	Assign Classes
23	Read-only	Delay Measurement
50	Read-only	Single Block Event
51	Read-only	**Auto Zero Bad Read**. Function used when a Tag receives a ramp up-only change event (from zero to one) and never a ramp down (from one to zero). In this case, this Driver inserts an event with a value of 0 (zero) before returning each event in 1 (one)
101	Read-only	**Read** (Integrity). Performs an integrity operation, requesting all points with object type × 100 + Variation declared in the *N3* parameter. Users can also use a 0 (zero) variation, corresponding to any informed object, regardless of their variation. This function performs a communication (integrity) each time a Tag is read

Supported Objects

Supported objects, variations, qualifiers, or functions

OBJECT	VARIATION	NAME	FUNCTION CODE (N2)
1 (Static)	1	Binary Input without Status	0, 1
1 (Static)	2	Binary Input with Status	0, 1

OBJECT	VARIATION	NAME	FUNCTION CODE (N2)
2 (Event)	1	Binary Input Change without Time	1
2 (Event)	2	Binary Input Change with Time	1
2 (Event)	3	Binary Input Change with Relative Time	1
3 (Static)	1	Double bit binary Input without Status	0, 1
3 (Static)	2	Double bit binary Input with Status	0, 1
4 (Event)	1	Double bit binary Input Change without Time	1
4 (Event)	2	Double bit binary Input Change with Time	1
10 (Static)	1	Binary Output	0, 1
10 (Static)	2	Binary Output Status	0, 1
11 (Event)	1	Binary Output Event without Time	1
11 (Event)	2	Binary Output Event with Time	1
12	1	Control Relay Output Block	3, 4, 5, 6
20 (Static)	1	32-bit Counter	1, 7, 9, 10, 11
20 (Static)	2	16-bit Binary Counter	1, 7, 9, 10, 11
20 (Static)	3	32-bit Delta Counter	1, 7, 9, 10, 11
20 (Static)	4	16-bit Delta Counter	1, 7, 9, 10, 11
20 (Static)	5	32-bit Counter without Flag	1, 7, 9, 10, 11
20 (Static)	6	16-bit Counter without Flag	1, 7, 9, 10, 11
20 (Static)	7	32-bit Delta Counter without Flag	1, 7, 9, 10, 11
20 (Static)	8	16-bit Delta Counter without Flag	1, 7, 9, 10, 11
21 (Static)	1	32-bit Frozen Counter	1, 7, 9, 10, 11
21 (Static)	2	16-bit Frozen Counter	1, 7, 9, 10, 11
21 (Static)	3	32-bit Frozen Delta Counter	1, 7, 9, 10, 11
21 (Static)	4	16-bit Frozen Delta Counter	1, 7, 9, 10, 11
21 (Static)	5	32-bit Frozen Counter with Time Of Freeze	1, 7, 9, 10, 11
21 (Static)	6	16-bit Frozen Counter with Time Of Freeze	1, 7, 9, 10, 11
21 (Static)	7	32-bit Frozen Delta Counter with Time Of Freeze	1, 7, 9, 10, 11

OBJECT	VARIATION	NAME	FUNCTION CODE (N2)
21 (Static)	8	16-bit Frozen Delta Counter with Time Of Freeze	1, 7, 9, 10, 11
21 (Static)	9	32-bit Frozen Counter without Flag	1, 7, 9, 10, 11
21 (Static)	10	16-bit Frozen Counter without Flag	1, 7, 9, 10, 11
21 (Static)	11	32-bit Frozen Delta Counter without Flag	1, 7, 9, 10, 11
21 (Static)	12	16-bit Frozen Delta Counter without Flag	1, 7, 9, 10, 11
22 (Event)	1	32-bit Counter Change Event without Time	1
22 (Event)	2	16-bit Counter Change Event without Time	1
22 (Event)	3	32-bit Delta Counter Change Event without Time	1
22 (Event)	4	16-bit Delta Counter Change Event without Time	1
22 (Event)	5	32-bit Counter Change Event with Time	1
22 (Event)	6	16-bit Counter Change Event with Time	1
22 (Event)	7	32-bit Delta Counter Change Event with Time	1
22 (Event)	8	16-bit Delta Counter Change Event with Time	1
23 (Event)	1	32-bit Counter Change Event without Time	1
23 (Event)	2	16-bit Frozen Counter Event without Time	1
23 (Event)	3	32-bit Frozen Delta Counter Event without Time	1
23 (Event)	4	16-bit Frozen Delta Counter without Time	1
23 (Event)	5	32-bit Frozen Counter Event with Time	1
23 (Event)	6	16-bit Frozen Counter Event with Time	1
23 (Event)	7	32-bit Frozen Delta Counter Event with Time	1
23 (Event)	8	16-bit Frozen Delta Counter Event with Time	1
30 (Static)	1	32-bit Analog Input	0, 1
30 (Static)	2	16-bit Analog Input	0, 1

OBJECT	VARIATION	NAME	FUNCTION CODE (N2)
30 (Static)	3	32-bit Analog Input without Flag	0, 1
30 (Static)	4	16-bit Analog Input without Flag	0, 1
30 (Static)	5	32-bit Analog Input Floating Point	0, 1
31 (Static)	1	32-bit Frozen Analog Input	0, 1
31 (Static)	2	16-bit Frozen Analog Input	0, 1
31 (Static)	3	32-bit Frozen Analog Input with Time Of Freeze	0, 1
31 (Static)	4	16-bit Frozen Analog Input with Time Of Freeze	0, 1
31 (Static)	5	32-bit Frozen Analog Input without Flag	0, 1
31 (Static)	6	16-bit Frozen Analog Input without Flag	0, 1
31 (Static)	7	32-bit Frozen Analog Input Floating Point	0, 1
32 (Event)	1	32-bit Change Event without Time	1
32 (Event)	2	16-bit Change Event without Time	1
32 (Event)	3	32-bit Analog Change with Time	1
32 (Event)	4	16-bit Analog Change Event with Time	1
32 (Event)	5	32-bit Analog Change Floating Point without Time	1
32 (Event)	7	32-bit Analog Change Floating Point with Time	1
33 (Event)	1	32-bit Frozen Analog Event without Time	1
33 (Event)	2	16-bit Frozen Analog Event without Time	1
33 (Event)	3	32-bit Frozen Analog Event with Time	1
33 (Event)	4	16-bit Frozen Analog Event with Time	1
33 (Event)	5	32-bit Frozen Analog Floating Point without Time	1
33 (Event)	7	32-bit Frozen Analog Floating Point with Time	1
34 (Static)	1	16-bit Analog Input Dead Band	1

OBJECT	VARIATION	NAME	FUNCTION CODE (N2)
34 (Static)	2	32-bit Analog Input Dead Band	1
34 (Static)	3	32-bit Analog Input Floating Point Dead Band	1
40 (Static)	1	32-bit Analog Output Status	1
40 (Static)	2	16-bit Analog Output Status	1
40 (Static)	3	32-bit Analog Output Status Floating Point	1
41	1	32-bit Analog Output Block	2, 3, 4, 5, 6
41	2	16-bit Analog Output Block	2, 3, 4, 5, 6
41	3	32-bit Floating Point Analog Output Block	2, 3, 4, 5, 6
42 (Event)	1	32-bit Analog Output Event without Time	1
42 (Event)	2	16-bit Analog Output Event without Time	1
42 (Event)	3	32-bit Analog Output Event with Time	1
42 (Event)	4	16-bit Analog Output Event with Time	1
42 (Event)	5	32-bit Floating Point Analog Output Event without Time	1
42 (Event)	7	32-bit Floating Point Analog Output Event with Time	1
50 (Static)	1	Time and Date	1, 2
51 (Static)	1	Time and Date CTO (Common Time of Occurrence)	1
51 (Static)	2	Not synced Time and CTO (Common Time of Occurrence)	1
52 (Static)	1	Time Delay Coarse	1
52 (Static)	2	Time Delay Fine	1
60 (Static)	1	Class 0 Data	1
60 (Event)	2	Class 1 Data	1
60 (Event)	3	Class 2 Data	1
60 (Event)	4	Class 3 Data	1
80	1	Internal Indications	1, 2
83 (Event)	1	Pro	1
110 (Static)	X	Octet String	1
111 (Event)	X	Octet String Event	1

Supported Qualifiers

Supported Qualifiers

OBJECT OR VARIATION	SENDING QUALIFIER	RECEIVING QUALIFIER
Static objects in general (Reading)	1	0, 1, 7, 8, 17, 18, 27, 28, 47, 58
Object 34 (Writing)	1	0, 1, 7, 8, 17, 18, 27, 28, 47, 58
Object 60 (Classes) or General Integrity Requests (Reading)	6	0, 1, 7, 8, 17, 18, 27, 28, 47, 58
Object 50 (Writing)	7	0, 1, 7, 8, 17, 18, 27, 28, 47, 58
Object 80 (Writing)	User-defined	0, 1, 7, 8, 17, 18, 27, 28, 47, 58
Object 12, Object 41 (SELECT and OPERATE, among others)	User-defined	0, 1, 7, 8, 17, 18, 27, 28, 47, 58

B-Addressing Parameters of Block Tags

B1	Slave address
B2	Code of a function to perform. For more information, please check table **Function Codes**
B3	Code of an object and variation (*Object* × *100* + *Variation*)
B4	Initial address or number of a variable

Using Block Tags to read events is not allowed, because all Block Elements share a single timestamp, which cannot be accepted as each event has its own occurrence time. Therefore, when reading the *B2* parameter must be equal to 0 (zero, **Read from cache**). The only exception is when using **Single Block Events** because, in this case, the timestamp of each event is reported in a fixed Element of a Block Tag.

Examples of Tag Configuration

Assuming that a device is configured with a DNP address equal to 3 (three), these are examples of *N1*, *N2*, *N3*, and *N4* parameters, in **N1.N2.N3.N4** format.

```
Digital Input 100, Object 1 Variation 2: 3.1.102.100 (configured as a static object)
Digital Input 100, Object 2 Variation 2: 3.1.202.100 (configured as an event)
```

In the previous example, both Tags reference the same variable (Digital input 100), but the first Tag remains continually requesting the current value (polling), while the second one only receives notifications when the point changes its value. In addition, as an **Event** the timestamp of the change from the device is kept. In the **Static** mode, on the other hand, as there is no timestamp, it is generated with the computer's time at event's arrival.

General Comments

This section contains additional information about DNP 3.0 Master Driver.

Data and Event Integrity

An integrity command performs a request of all data configured in a Slave, for all Class 0 (zero) objects or for specific objects.

Due to a peculiarity of DNP protocol sending together with integrity the current value of variables, therefore as static objects and not as events, this Driver must process this information to unify the static value with the event's value, so that an application has a single Tag for the point.

This way, if the application uses Tags as events, as recommended in this document, then during start up, when receiving an integrity (Class 0) with a point's static value, a temporary event is generated with its **Quality** property equal to 216, indicating that the timestamp used was not generated by the device, but rather by the computer's local time, due to the lack of this information in the integrity.

As soon as there is a reception of an event for this Tag, then its quality changes to 192 and its **Timestamp** property reflects the timestamp received from the event, if the object contains one.

NOTE

Quality values mentioned previously assume that object statuses are informing a normal situation, otherwise quality values reflect a point's bad status, as described on topic **Quality Information**.

Event Reading

Receiving events using this Driver is fundamental for Tags configured as events to receive their values. This Driver only discovers that there is data available for one of the classes of events (one, two, or three) if an application uses at least one of the following options:

- The **Scan for Events Every X ms** option on this Driver's configuration is enabled. This informs this Driver to request Classes 1 (one), 2 (two), or 3 (three) at fixed intervals, receiving an indication of events
- A device sends events spontaneously through unsolicited messages
- The application contains at least one Tag configured for reading any static variable, such as digital or analog input or output and counters, among others

Commands

When using a **Control Relay Output Block** (Object 12, Variation 1) object, a series of data is sent by this Driver, as described next.

Byte 0: Control Code

The value of a Tag configured in the application is copied to this byte, which specifies details about the command's operation. This field is subdivided as follows.

Bit	7	6	5	4	3	2	1	0
Meaning	Trip or Close		Clear	Queue	Code			

- **Trip or Close**: This field determines which control relay is activated in a system where a pair of Trip and Close relays is used to energize and de-energize points in the field. Possible values are, in binary format, **00**: NUL, **01**: Close, and **10**: Trip. The **NUL** value can be used to activate the selection relay without activating the Trip or Close relays. In a system without selection relays, the **NUL** value does not perform any operation. In a system without Trip or Close relays, on the other hand, this field must be always **NUL** to indicate a normal operation of digital control, where the exact control point is implicit or fully known. This field does not support Trip and Close commands simultaneously, which is considered an illegal operation

- **Clear**: If the command contains this field set as 1 (one, turned on), all control operations are removed from the queue, including the command currently executing, and this control operation is performed
- **Queue**: Indicates placing a command in a command queue in the device. If this field is equal to 0 (zero, **NUL**) then no operation is added to the queue and this queue is cleared from all controls, including the command that is executing if the **Clear** field is turned on. When the control function is executed and completed, it is removed from the queue. If this command contains a **Queue** attribute turned on, then this operation returns to the queue (at the end) for that point
- **Code**: This field specifies a type of operation. This command can be used with devices that support queuing commands, point-to-point, or other control mechanisms. In the first type, any control command must be queued for the point. In the second type, each control is performed until completed before the next command be accepted for that point.

Possible values for Code in binary format

VALUE	OPERATION	DESCRIPTION
0000	NUL	No operation is performed
0001	PULSE ON	Points are connected by the time specified in **On-Time**, turned off by the time specified in **Off-Time**, and kept in an **OFF** status
0010	PULSE OFF	Points are disconnected by the time specified in **Off-Time**, turned on by the time specified in **On-Time**, and kept in an **ON** status
0011	LATCH ON	Keeps all points in an **ON** status
0100	LATCH OFF	Keeps all points in an **OFF** status

Other values outside this table are not defined.

Byte 1: Count

This byte indicates how many times an operation is executed. This value is kept fixed in 1 (one) by this Driver.

Bytes 2 a 5: On-Time

On-time, in milliseconds. It is defined on Driver's extra configurations window and it is fixed for all commands.

Bytes 6 a 9: Off-Time

Off-time, in milliseconds. It is defined on Driver's extra configurations window and it is fixed for all commands.

Byte 10: Status

Status of the operation returned by this Driver if that operation was successful. Status is only interpreted in the response, and it can be used by an application to check if the command was performed successfully. The available codes are the following:

- **0**: Command correctly executed, including **Select** and **Operate** operations
- **1**: **Operate** command sent after a maximum **Select** time defined by the slave
- **2**: **Operate** command was sent without a previous **Select** command
- **3**: Formatting errors in the message
- **4**: Operation not supported for this point
- **5**: Queue is full or point is already active
- **6**: Hardware problems
- **Others**: Non-standard error codes

The value of the **Status** field can be obtained by an **Elipse E3**, **Elipse Power**, or **Elipse Water** application using the *WStatus* parameter of the **WriteEx** method of I/O Tags or Block Tags, as described on the next examples.

To send commands, users can use a PLC Tag (IOTag) as well as a Block Tag. When using a PLC Tag, it must be set with a number between 0 (zero) and 255, which corresponds to the Control Code (message's byte zero). Remaining bytes are retrieved from this Driver's default configuration, defined on extra configurations window.

For a Block Tag, users must use a script that executes Block Tag's **Write** method. To do so, this Block Tag must have only four Elements, which must have their individual writing properties disabled:

- **Element 0**: Control Code
- **Element 1**: Count
- **Element 2**: Relay On-Time
- **Element 3**: Relay Off-Time

This feature can be used if individual command programming is needed for each point, ignoring default **On-Time** and **Off-Time** timings.

Example of a script in Elipse SCADA using a Block Tag with four Elements

```
//Configurations: B2=5 (DIRECT OPERATE), B3 = 1201
Block1.Element0 = 65  // Operation Code
Block1.Element1 = 1   // Count
Block1.Element2 = 500 // On Time
Block1.Element3 = 500 // Off Time
Block1.Write()
```

Example of a script in Elipse E3, Elipse Power, or Elipse Water using a simple command Tag

```
Value = 65 // Operation Code
msg = MsgBox("Send this command?", 292, "Command")
If msg = 7 Then
  MsgBox "Command aborted", 48, "Quit"
ElseIf msg = 6 Then
  If MsgBox("Confirm sending this command?", 292, "Warning") = 6 Then
    Set Tag = Application.GetObject("DNPDriver.TagName")
    If Tag.WriteEx(Value, , , WStatus) = False Then
      MsgBox "Error sending this command", 48, "Error"
    Else
      If WStatus <> 0 Then
        MsgTrack = "Command not executed,"
        EndText = ""
        Select Case WStatus
          Case 1
            EndText = " Operate received after a Selection time-out"
          Case 2
            EndText = " No previous Selection message"
          Case 3
            EndText = " Formatting error in this Command"
          Case 4
            EndText = " Operation not supported for this point"
          Case 5
            EndText = " Queue is full or point is already active"
          Case 6
            EndText = " Hardware problems"
          Case Else
            EndText = " Undefined problem"
        End Select
        MsgBox MsgTrack & EndText, 48, "Error"
    End If
  End If
End If
```

Single Block Events

If the **Use Single Block Read for All Events** option is enabled, users can use a Block Tag to receive all events at once. This Block Tag must have the following configuration:

- **N1**: DNP address
- **N2**: 50
- **N3 and N4**: Not used
- **Size**: Can have up to seven Elements:
  - **Element 0**: Object
  - **Element 1**: Variation
  - **Element 2**: Index
  - **Element 3**: Status
  - **Element 4**: Value
  - **Element 5**: Timestamp
  - **Element 6**: Quality

NOTE

A Single Block Event cannot be used together with event Tags in an application.

Quality Information

For objects with a status indication (most of them), this Driver performs a mapping of the status to OPC standard used in **Elipse E3**, **Elipse Power**, or **Elipse Water**, as explained next.

For objects 1, 2, and 10

Bit	7	6	5	4	3	2	1	0
Meaning	XX	XX	CF	LF	RF	CL	RS	OL

- **OL (Online)**: In zero, Q = 12 (*Bad, device failure*)
- **CL (Comm lost)**: In one, Q = 12 (*Bad, device failure*)
- **RS (Restart)**: In one, Q = 68 (*Uncertain*)
- **CF (Chatter filter)**: In one, Q = 80 (*Uncertain*)
- **RF (Remote forced)**: In one, Q = 216 (*Good, local override*)
- **LF (Local forced)**: In one, Q = 216 (*Good, local override*)

For object 40

Bit	7	6	5	4	3	2	1	0
Meaning	0	RE	OR	LF	RF	CL	RS	OL

- **OL (Online)**: In zero, Q = 12 (*Bad, device failure*)
- **CL (Comm lost)**: In one, Q = 12 (*Bad, device failure*)
- **RS (Restart)**: In one, Q = 68 (*Uncertain*)
- **RE (Reference error)**: In one, Q = 80 (*Uncertain*)
- **OR (Over range)**: In one, Q = 66 (*Uncertain*)
- **RF (Remote forced)**: In one, Q = 216 (*Good, local override*)
- **LF (Local forced)**: In one, Q = 216 (*Good, local override*)

Other objects

Bit	7	6	5	4	3	2	1	0
Meaning	0	XX	XX	LF	RF	CL	RS	OL

- **OL (Online)**: In zero, Q = 12 (*Bad, device failure*)
- **CL (Comm lost)**: In one, Q = 12 (*Bad, device failure*)

- **RS (Restart)**: In one, Q = 68 (*Uncertain*)
- **Bits 5 or 6**: In one, Q = 80 (*Uncertain*)
- **RF (Remote forced)**: In one, Q = 216 (*Good, local override*)
- **LF (Local forced)**: In one, Q = 216 (*Good, local override*)

NOTE

According to OPC standard, a quality greater or equal to 192 is considered **Good**.

Collecting and Storing Files in COMTRADE Standard

If the process of collecting files is enabled in a **Pextron** profile, this Driver follows these steps:

1. When starting this Driver, a random time between 0 (zero) and 13 minutes starts counting, and then tries to perform a collect process. If this collect process is successful, this file is saved and its existence is indicated by the **DNPADD.-33.0.X** Tag. The number of files left for transfer is indicated by the **DNPADD.-33.1.X** Tag.
2. The application can write to the **DNPADD.-33.2.X** Tag to request a new immediate collect process or to **DNPADD.-33.3.X** Tag to request a new collect process after a random time. In both cases the collect process is performed regardless of **DNPADD.-33.1.X** Tag indicating 0 (zero) or more collect processes to transfer.
3. If no Tag is written, this Driver does not perform a new collect process.

Configuration of channels, directory, and file name must be performed before starting a collect process.

Documentation of I/O Interfaces

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

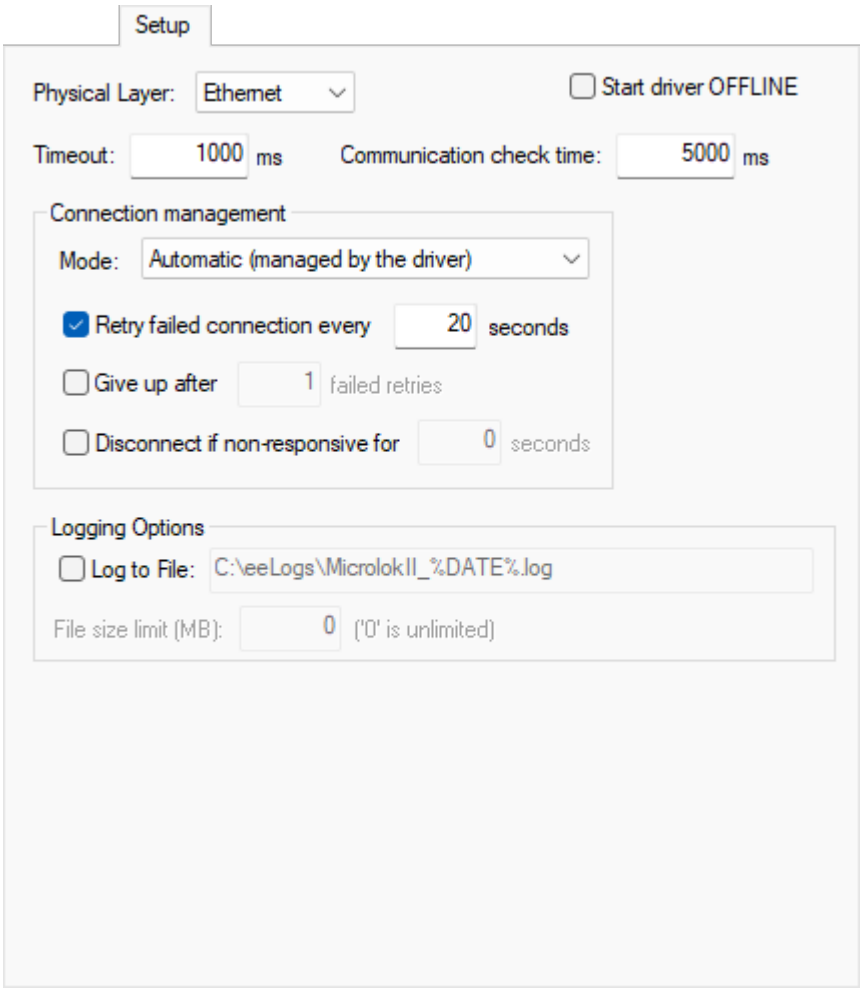
Configuration Dialog Box

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

OPTION	DESCRIPTION
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

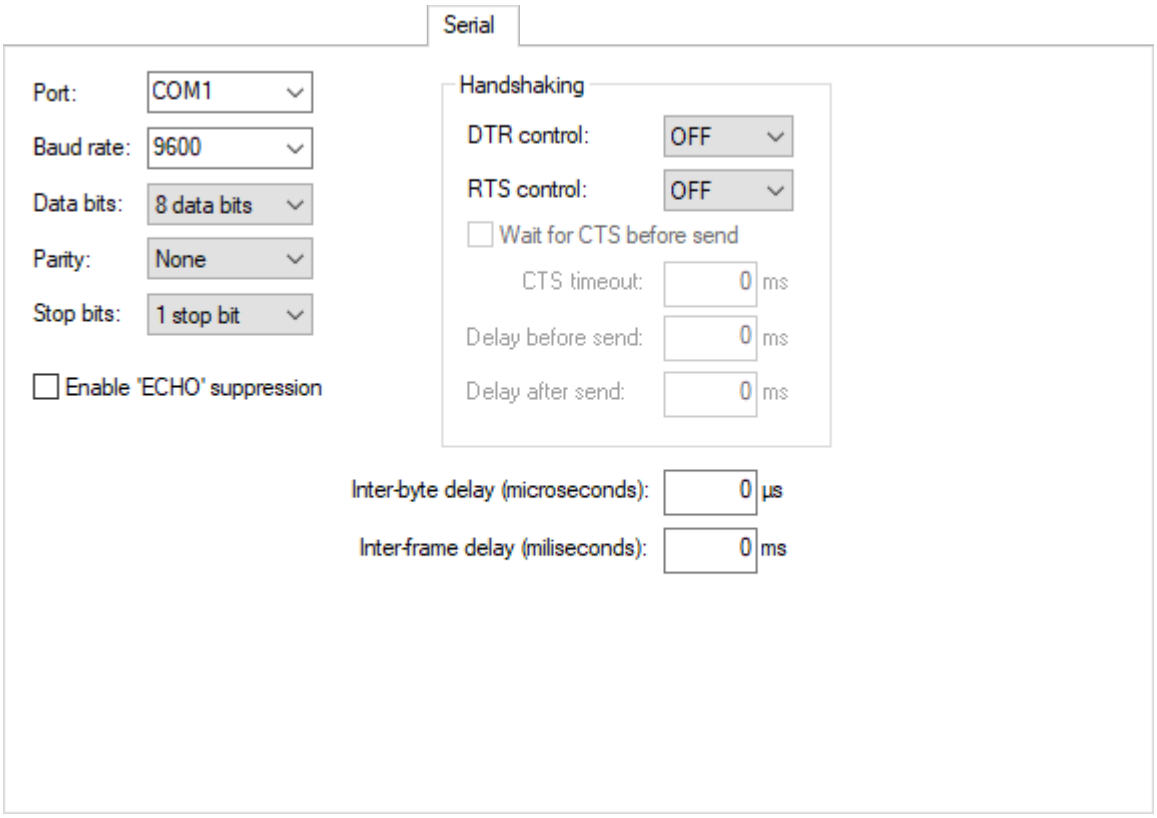
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every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

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 the name of a serial port manually, the dialog box only accepts names of serial ports 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

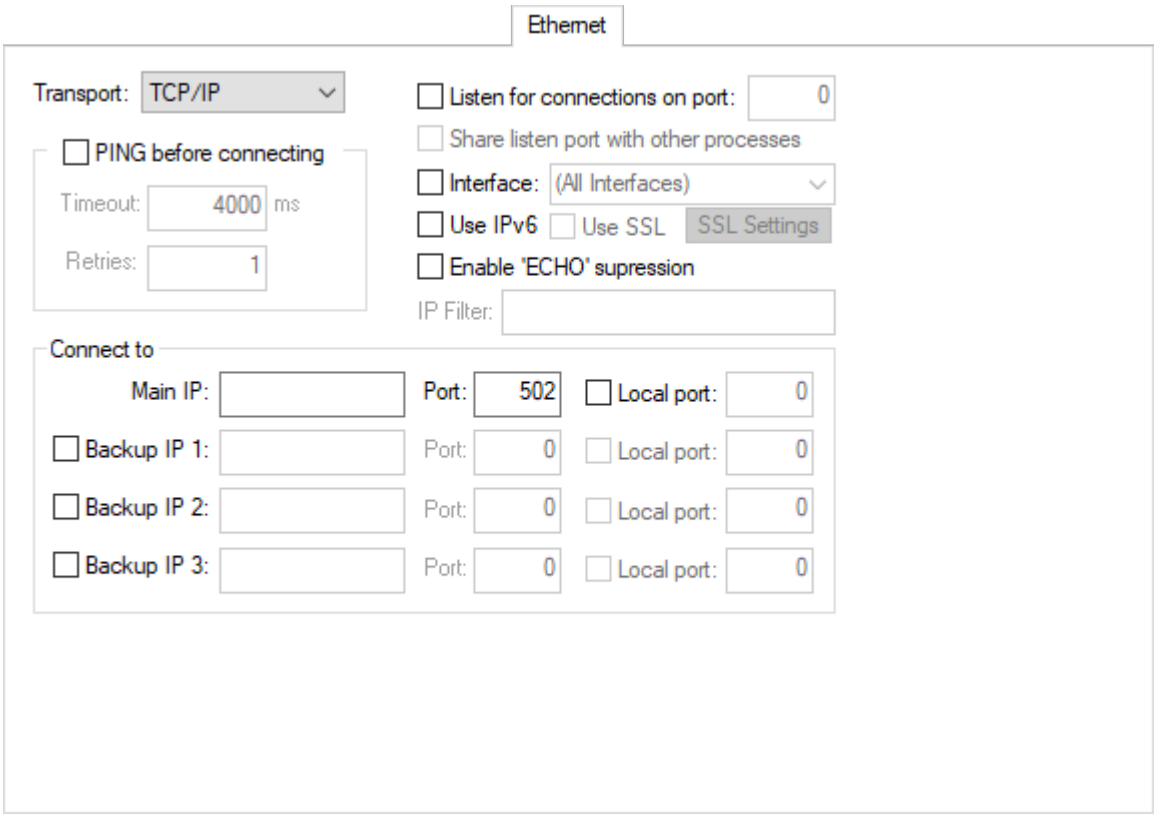
The **Handshaking** group configures the usage of **RTS**, **CTS**, and **DTR** signals in the handshaking process, that is, it controls when data can be sent or received via serial line. Most of the time, configuring the **DTR control** option to **ON** and the **RTS control** option to **Toggle** works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

Available options on the Handshaking group

OPTION	DESCRIPTION
DTR control	Select the value **ON** to keep the **DTR** signal always on while the serial port is open. Select the value **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 the value **ON** to keep the **RTS** signal always on while the serial port is open. Select the value **OFF** to turn the **RTS** signal off while the serial port is open. Select the value **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 with the value **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, that Driver then fails the current communication and returns an error
Delay before send	Some serial port devices 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** 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

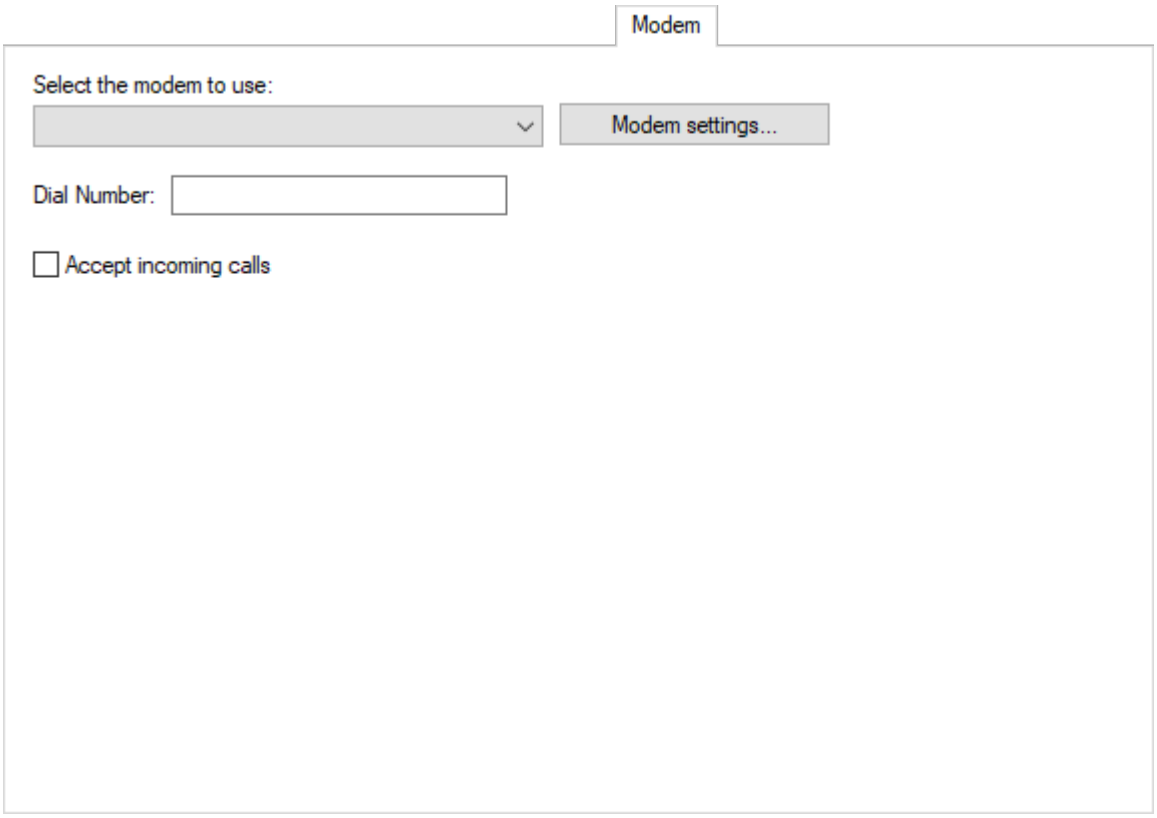
OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use 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

Modem Tab

Use this tab to configure parameters of a **Modem** Interface. Some options on the **Serial** tab affect the configuration of a modem, 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 value **Default modem** 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 0 (zero), waits, and then dials the number "33313456"
Accept incoming calls	Enable this option so that a 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 the value **Manual**

RAS Tab

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

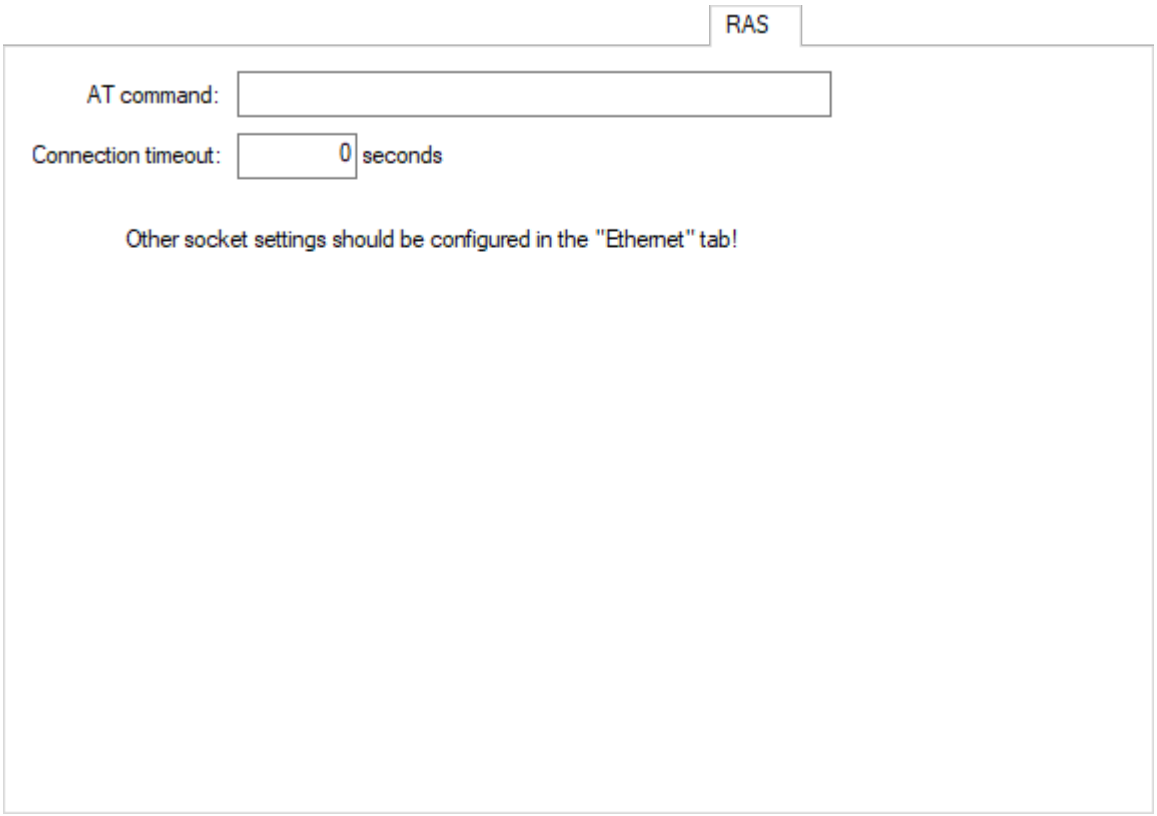
A **RAS** Interface opens a socket connection with a RAS device. A RAS device is a server of modems available through TCP/IP, waiting for socket connections on an IP port. For each connection accepted on this port, users have access to one modem.

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. Clears the socket, that is, removes any **TELNET** greeting message received from a RAS device.
2. Sends an **AT** dial message, in **ASCII** format, in the socket.
3. Waits 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 a device, that is, the connection was established.

If step 5 (five) is successful, then the socket behaves as a normal socket, with the RAS device working as a router between a Driver and the device. Bytes sent by a 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 a Driver using the same socket.

After establishing a connection, the **RAS** interface monitors data received by 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.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

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

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

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]

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

IO.Ethernet.BackupLocalPort[2,3]

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

IO.Ethernet.BackupLocalPortEnable[2,3]

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

IO.Ethernet.BackupPort[2,3]

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

IO.Ethernet.IPFilter

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

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

IO.Ethernet.PingTries

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).

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 a 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 hangs the current call up.

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, a 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 a 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!"**: A Driver could not initialize modem's line. Check that Driver's log file for more details
- **"Modem error at dial!"**: A Driver could not start or accept a call
- **"Connecting..."**: A 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!"**: A Driver connected successfully, that is, completed or accepted an external call
- **"Disconnecting..."**: A Driver is turning the current call off
- **"Disconnected OK!"**: A Driver turned the current call off
- **"Error: no dial tone!"**: A Driver aborted a call because the available line signal was not detected
- **"Error: busy!"**: A Driver aborted a call because the line was busy
- **"Error: no answer!"**: A 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, if a Driver behaves as a master, and configure to True to enable receiving calls, that is, if a 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 A telephone number used by **Dial** commands, such as "0w01234566", in which 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 therefore must be also configured.

IO.RAS.ATCommand

A 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, a Driver aborts sending bytes through the serial port. Available only when the **IO.Serial.RTS** property is configured with the value **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 with the value **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

IO.Serial.DelayBeforeMs

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 with the value **Toggle** and the **IO.Serial.WaitCTS** property is configured to False.

IO.Serial.DTR

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

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

IO.Serial.InterframeDelayMs

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

IO.Serial.Parity

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

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

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

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

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 with the value **Toggle**.

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
4.0.58	09/02/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37954*).
4.0.57	10/12/2024	M. Salvador	• Created a **Stop Listen After Accept** option, which is used if the **Listen for connections on port** option is enabled on the **Ethernet** tab. Then, this Driver no longer remains in **Listen** mode after receiving a connection (*Case 36661*). • This Driver now interrupts the waiting for responses if it is being stopped (*Case 36477*).

VERSION	DATE	AUTHOR	COMMENTS
4.0.53	20/03/2024	M. Salvador	• **String** readings (objects 110 and 111) now work correctly (*Case 34740*). • Class 0 was not being performed sporadically after a device restart (*Case 33970*). • Added a protection in case this Driver receives a message with a size smaller than expected (*Case 33840*). • Resolved a GPF (*General Protection Failure*) that could occur when adding and removing events very quickly (*Case 35489*).
4.0.45	29/03/2022	M. Salvador	• Implemented variations or objects 3006, 3206, and 3208 (64 bits) (*Case 32404*). • Now, when a polling Tag or event request (class 1, 2, and 3) cannot access the communication channel because it is occupied with another communication, in case of an unsolicited one, this is not considered as a communication error. Furthermore, this Driver now waits for the completion of fragment reception before allowing new messages to be sent, always prioritizing the handling and confirmation of unsolicited ones, which was already performed before, only a new collision protection was added (*Case 31112*). • **Link Status** request can now be correctly disabled (*Case 30482*). • Implemented support for object 5201 (*Time Delay Coarse*) (*Case 30148*). • Fixed overlapping texts in the configuration dialog box (*Case 29115*).
4.0.40	27/11/2019	M. Salvador	• Fixed the handling and attempts to clear **Device Restart - IIN1 Bit 7** bit (*Case 27925*).

VERSION	DATE	AUTHOR	COMMENTS
		C. Mello	• Driver ported to Visual Studio 2017 (*Case 27493*).
4.0.39	04/23/2019	M. Salvador	• Added the **Accept Msgs After Timeout** and **Accept Msgs Out of Order** options on the **Other** tab (*Case 26541*). • Adjustments to the quality value after receiving an integrity (*Case 26525*). • Adjustments to the quality value after changing the IP address of a redundant Slave (*Case 25830*).
4.0.36	08/08/2018	M. Salvador	• Added the **Command Max Age (s)** option to the **Other** tab (*Case 24231*). • Adjustments in the sequence of **Select** and **Operate** messages (*Case 24230*). • Fixed the handling of Data Link messages initiated by a Slave (*Case 22797*). • Added the **Block Unknown Slaves** option to prevent sending messages to undeclared devices (*Case 22581*). • Added support for requesting integrity and changes in alphabetical order from multiple Slaves (*Case 22401*). • Added the **Scan After Cmd** option to scan events after sending commands (Case 22219). • Improved the accuracy of the minimum interval of -6 and -7 sync Tags (*Case 22105*). • Added the **Min Delay between Messages** option for communication (*Case 21534*). • Improvements in handling messages coming from a Slave as primary (*Case 21813*). • Adjustments in handling Class 0 to prevent conflicts

VERSION	DATE	AUTHOR	COMMENTS
			with unsolicited messages (*Case 21627*). • Added support for object 41 variation 4 (*Case 21184*). • Fixed the operation of object 100 variation 1 (*Case 20963*). • Improvements in the process of physically reconnecting communication (*Case 20962*). • Improvements in event processing (*Case 20883*). • Adjustments in updates of status Tags for the **Offline** mode (*Case 19395*). • Added a writing command via Tag with the *N2* parameter equal to -15 for asynchronous requests for a Status Link message to Slaves (*Case 19155*). • Improved control of time-out interval and integrity request for Class 0 (*Case 18956*). • Added a minimum time parameter between scans of pending events (*Case 18070*). • Adjustments to the Link Status Tag, *N2* parameter equal to -10, for asynchronous requests of a Status Link message to Slaves (*Case 17926*).
4.0.21	10/27/2014	M. Salvador	• Added an option for syncing via Ethernet in a Tag with the *N2* parameter equal to -7 (*Case 17569*). • Adjustments in the processing of Tags with multiple Slaves (*Case 17518*). • Added support for **String**-type objects 110 and 111 (*Case 16920*). • Fixed problems with the addressing of Slaves for event Tags (*Case 16919*).

VERSION	DATE	AUTHOR	COMMENTS
			• Fixed a vulnerability from ICS-CERT VU-028282 notification, according to tests with an emulator for formatting errors in DNP Aegis Fuzzer messages (*Case 16590*).
4.0.17	05/29/2014	M. Salvador	• Added support for objects 4003 and 4103 (*Case 15232*). • Migration to Visual Studio 2013 (*Case 16392*). • Migration to **IOKit** library version **2.0** (*Case 13516*). • Allowed negative values in the **Extra Sync Offset** option on the **DNP** tab. • Fixed timers for checking events and Class 0 when computer's time moves back in time, that is, does not use the **etGetTimeDouble** method anymore. • Generation of syncing files of the last timestamp (*Case 13528*). • Performance improvements when using thousands of simultaneous Drivers (*Case 13117*). • Fixed the behavior when this Driver only contains event Tags with their callback option enabled (*Cases 14040 and 16088*). • **Bypass Reset of Remote Link** on the first unsolicited communication from a Slave (*Case 15233*).
3.2.1	05/31/2012	M. Salvador	• **Beta 1**: Fixed Single Block Events. • **Beta 2**: Added support for reading object 50 variation 01. Fixed qualifiers 7 (seven) and 8 (eight). When connecting, event request is performed before Class 0 (zero). • **Beta 3**: Fixed a check for out-of-order events.

VERSION	DATE	AUTHOR	COMMENTS
			• **Beta 4**: Delay time for unsolicited messages, in milliseconds. • **Beta 5**: Checking out-of-order events is individual by Tag. • **Beta 6**: Fixed a log message when there is a CRC error. • **Beta 7**: Fixed the quality information of Single Block Events. • **Betas 8 and 9**: Fixed the expected sequence number on the application layer. • **Beta 10**: Fixed double points on objects 3 (three) and 4 (four). • **Beta 11**: Tag indicating the number of frames received. • **Beta 12**: Writing command on object 10 variation 1. • **Beta 13**: Decreasing on the delay time of transmission when not connected. • **Beta 14**: Tag with the *N2* parameter equal to -21 only informs Active after receiving a response from a Slave's **Reset of Remote Link**. • **Beta 15**: Handling a duplicated **05** at the beginning of a frame, such as **05 05 64**. • **Beta 16**: Separated processes for sending, receiving, and checking. • **Beta 17**: Sending process now is synchronous with application layer. • **Beta 18**: Option to apply a Local Time Zone to events and syncing. • **Beta 19**: Option to discard out-of-order events was always applied. • **Beta 20**: Statistics of bytes sent and received.

VERSION	DATE	AUTHOR	COMMENTS
			• **Beta 21**: Performance improvements when using many Tags with callbacks. • **Beta 22**: Fixed the handling of object 10 variation 1. • **Beta 23**: Released confirmation of application messages even without a **Reset of Remote Link**, to allow processing unsolicited messages received right after a connection. • **Beta 24**: Added a Tag for IIN indication (-22).
3.1.1	12/14/2009	M. Salvador	• Fixed case 11020, in which a connection was considered inactive as soon as it entered an active state, due to an internal error. • Changed the library name to DNPMaster.dll.
3.0.1	11/30/2009	M. Salvador	• Implemented the **Read from Cache** and **Read Events** options, which turns this Driver's behavior similar to IEC 60870-101/104 Driver. • Revised documentation.
2.29.1	11/30/2008	M. Salvador	• Fixed syncing for more than one Slave on the same link. • Fixed a re-validation of points after shutting down a connection. • Fixed Block readings with a **Float** data type. • Fixed the message for checking events, in which the same class requested more than once in the same message. • Fixed a race condition among processes when in **Select**. • Option to start on Device Restart. • File collecting.
2.28.1	10/29/2006	M. Salvador	• Independent reading and writing retries. • Single Block Events.

VERSION	DATE	AUTHOR	COMMENTS
			• Enabled unsolicited messages when starting.
2.20.1	12/20/2005	M. Salvador	• On or Off control for each address. • Active or Inactive control for each address. • Automatic Demotion control.
2.19.1	10/11/2005	M. Salvador	• Improvements on the database update process. • Event reading based on class requests instead of a clock-reading command.
2.18.1	09/30/2005	M. Salvador	• Internal integrity and polling system. • Fixed a freezing on values.
2.17.1	06/24/2005	M. Salvador	• Added support for Toshiba Regulator devices (*Case 5768*).
1.0.1	08/03/2004	M. Salvador	• All publications previous to revision control.

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)