B. Braun AutoProgramming Driver

File Name	AutoProgramming.dll
Manufacturer	B. Braun
Devices	SpaceCom
Protocol	AutoProgramming
Version	1.0.2
Last Update	06/12/2025
Platform	Win32
Dependencies	IOKit version 2.0
Superblock Readings	No
Level	0

Introduction

This Driver implements the AutoProgramming feature, allowing an application developed by **Elipse Software** to communicate with SpaceCom devices by B. Braun.

Driver Operation

To use AutoProgramming properly in an application, it is important to understand how it works, because this Driver does not have the most common features of all other I/O Drivers.

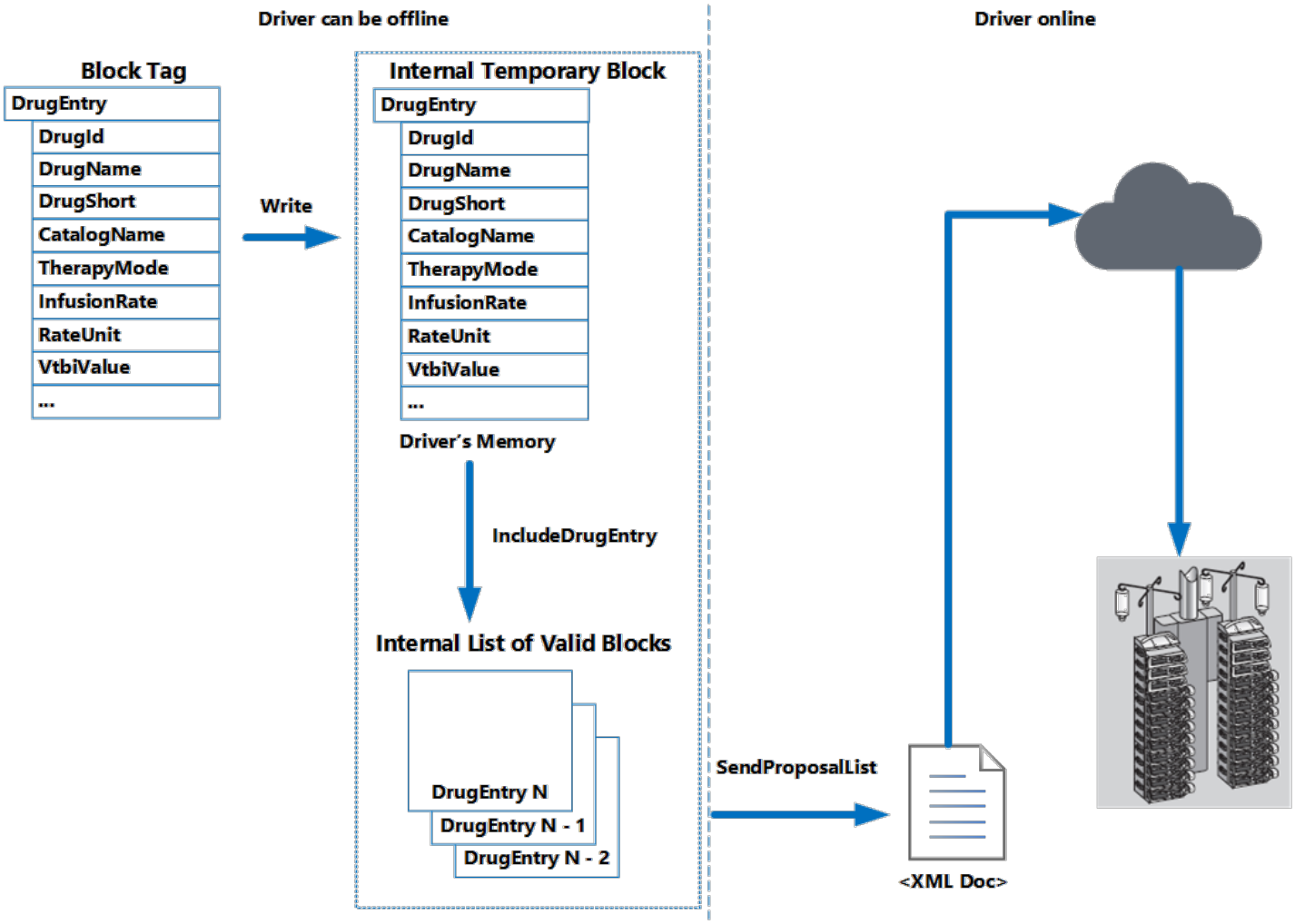
A B. Braun SpaceCom device provides a feature where an external system can receive the ability to send patient therapy configuration data sets. To do so, a device contains a TCP/IP port dedicated to receive a list of data available in **XML** (*Extensible Markup Language*) format, which is commonly known as an **XML Document**. This XML document can consist of a list of up to 24 therapeutic configurations, and each one of the items on this list can receive a certain set of parameters that determines this configuration. To build an XML document of therapy settings, therefore a certain number of values must be configured by some sort of form editing. An **Elipse Software** application working with this Driver can then become a management system of patient's therapies, by editing forms and applying those values to create an XML document to send to a certain therapy unit.

Due to the nature of this AutoProgramming feature of B. Braun devices, basically most of the processing activity of this Driver concentrates on the creation of an XML document to send than in the communication process itself.

This Driver's operation can be reduced to the operation of a sequence of three essential ordered writings, that is, a writing of a **DrugEntry** Block Tag, a writing of an **IncludeDrugEntry** operation, using a **DrugEntryOperation** Tag, and a writing of a **SendProposalList** Tag to command forming and sending an XML document to a remote device. Up to the moment previous to a call to the **SendProposalList** command, this Driver can be started in offline mode, that is, without starting the communication interface and connecting to the remote device, because before sending data this Driver works only with data in memory. However, by the time the **SendProposalList** command is called, this Driver must be already in an online status, that is, connected to a remote device.

On this Driver's internal memory there are two types of data containers to form an XML document. First, there is a temporary block that is memorized by writing a **DrugEntry** Block Tag. Assuming that this Block Tag is correctly filled, it can be copied to a second container that holds, by writing order, a list of valid blocks after each **IncludeDrugEntry** operation. This way, an XML document can be composed by up to 24 blocks, performing those writing **DrugEntry** block operations and an **IncludeDrugEntry** operation as may times as needed to finish an XML document, and then execute a writing of a **SendProposalList** Tag. After executing this writing, this Driver communicates the set of therapy settings to the device remotely connected by the Ethernet TCP/IP connection.

The next figure illustrates these elementary operations on how this Driver works. Other more secondary operations can be performed by other Tags not yet mentioned, but that can be useful according to a designer's intentions.



Operation of B. Braun AutoProgramming Driver

Driver Configuration

This Driver's **[P]** parameters are not used. All configurations are performed on this Driver's configuration dialog box. For more information about the configuration tabs, please check topic **Documentation of I/O Interfaces**.

Configuring Properties

All settings are executed on this Driver's configuration dialog box. On this dialog box, the most relevant parameters are described on the next table.

Configuration options of B. Braun AutoProgramming Driver

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
Setup	Physical Layer	IO.Type	Text	By default, use the **Ethernet** option
	Timeout	IO.TimeoutMs	Number	Time limit, in milliseconds, to receive data on a device's answer. For example, a value of 1000 defines a time-out of 1 (one) second

TAB	PARAMETER	OFFLINE STRING	DATA TYPE	DESCRIPTION
	Logging options / Log to File	IO.Log.Enable	Number	Enable log generation and a log file name to generate for debugging purposes on communication. A value of 0 (zero) disables log generation and any value different from 0 (zero) enables it
		IO.Log.Filename	Text	
Ethernet	Transport	IO.Ethernet.Transport	Text	By default, use the **TCP** option
	Main IP	IO.Ethernet.MainIP	Text	Device's IP address, in the format **[0-255].[0-255].[0-255].[0-255]**
	Port	IO.Ethernet.MainPort	Number	By default, use the value 4002

In addition to this Properties Window, these settings can also be defined at run time in **Elipse E3**, **Elipse Power** or **Elipse Water** applications. For this, initialize this Driver in **Offline** mode, that is, execute an application with the **Start driver OFFLINE** option enabled, configurable on the **Setup** tab of the Properties Window, described previously.

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

Tag Reference

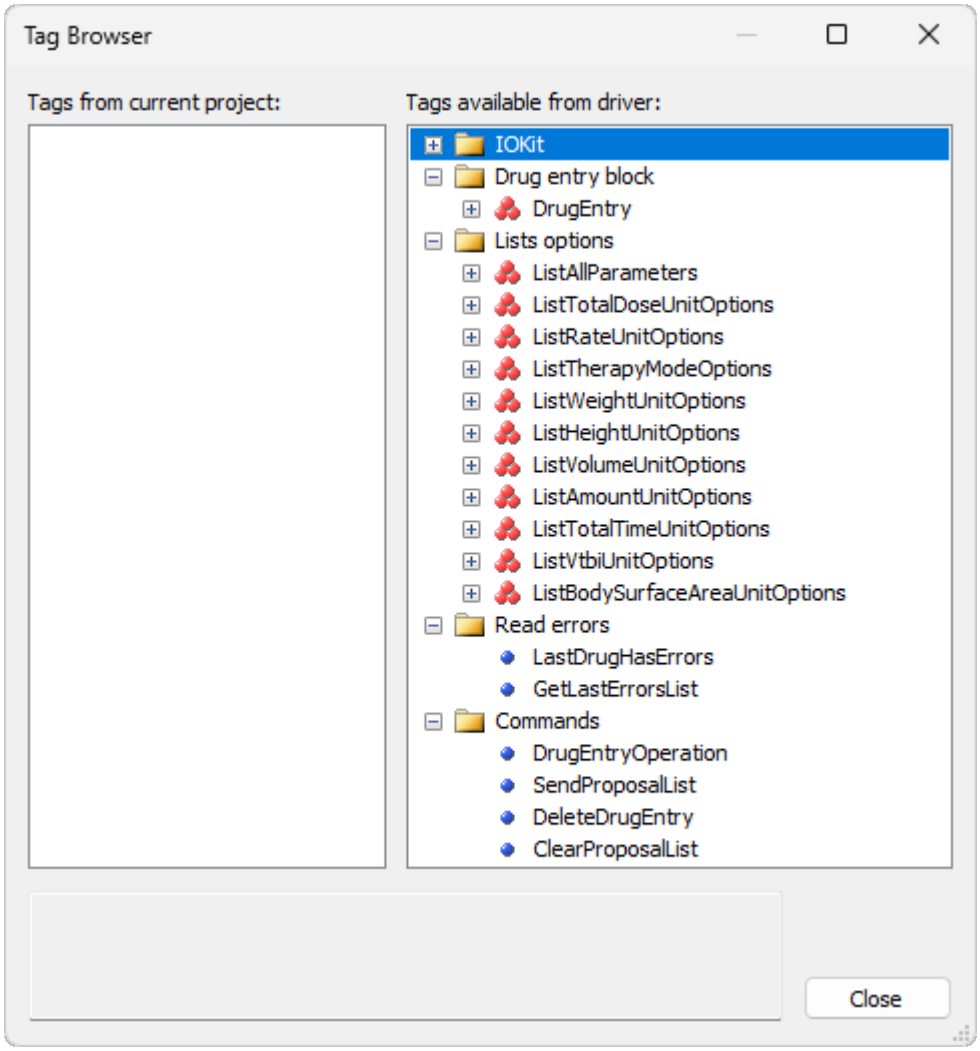
This section contains information about the configuration of this Driver's Tags. *N* or *B* parameters are not used. Only the *Item* parameter is used to refer which Tag is executed.

Any word not expected as a Tag name in the *Item* parameter results in a fill error. *N* or *B* parameters filled with values different from 0 (zero) also result in a fill error.

Using Tag Browser

Elipse E3 version **2.0**, **Elipse Power**, and **Elipse Water** have a tool called **Tag Browser**, which allows a Driver to help users on creating and configuring Tags.

Tag Browser aims to automate Tag configuration by automatically filling the **Item** field and the proper size of Elements, if it is a Block Tag.



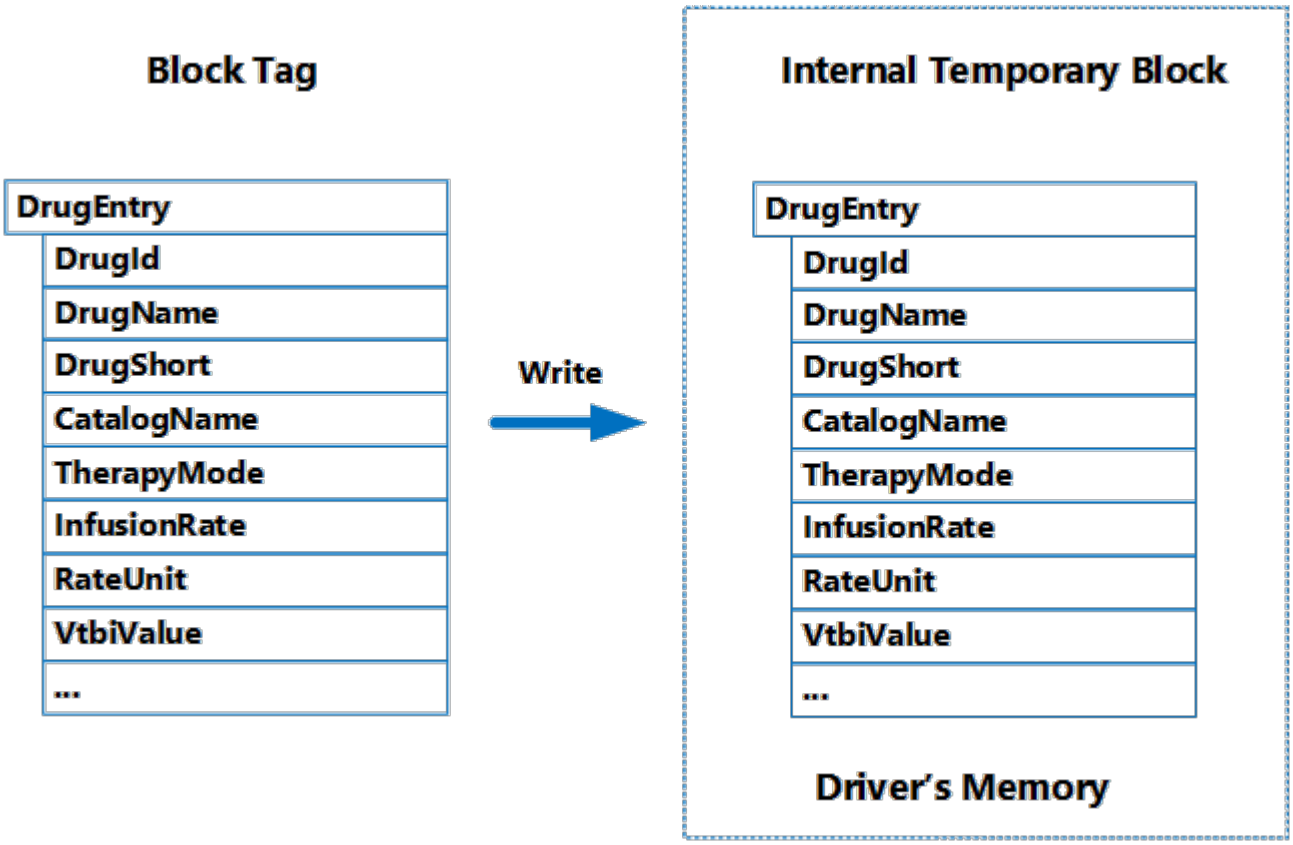
Tag Browser window

The **Tags from current project** list displays Tags and folders already in the current project. The **Tags available from driver** list displays a tree with Tags provided by this Driver. To create a new Tag in an application, drag one of the Driver-defined Tags to the desired folder on the current directory. B. Braun AutoProgramming Driver provides the following nodes on this tree view:

- The **IOKit** node displays Tags available on the **IOKit** library, grouped into the following categories:
  - **General**: Tags for general use
  - **Modem**: Tags for manipulating communication via modem
  - **Ethernet**: Tags for manipulating communication via Ethernet network
  - **Parameters**: Tags for configuring **IOKit** parameters
- The **Drug entry block** node contains a **DrugEntry** Block Tag for reading and writing
- The **Lists options** node contains all Block Tags for **listing options**
- The **Read errors** node contains all PLC Tags for **reading configuration errors**
- The **Commands** node contains all PLC Tags for **writing commands**

Therapy Settings Tag

A **DrugEntry** Block Tag performs a therapy configuration and writing to this Tag inserts values into this Driver's memory as a new configuration candidate to add to the internal list of therapy configurations.



DrugEntry Block Tag

DrugEntry

Read and Write

This Block Tag must contain 27 Elements. All names and fill rules for each one of these Elements are described on the next table. Depending on the operation performed, reading or writing, this Tag follows the set of behaviors described next.

Write

Writing must be performed as an entire block operation, that is, writing only Block Elements is not allowed. This operation stores in an internal memory a candidate for a new **Drug** item, which is processed according to the command selected in the operations of the **DrugEntryOperation** Tag.

Read

Displays all **Drug** items stored in this Driver's internal memory list. This reading returns a list of *N* blocks of **Drug** items, where *N* is the number of items in memory.

Format of a DrugEntry Block Tag

ELEMENT	PARAMETER	FORMAT	RULE
Element 1	DrugId	**String**	Optional
Element 2	DrugName	**String**	Please check **Rule 1** (one)
Element 3	DrugShort	**String** with a maximum of 8 (eight) characters	Please check **Rule 1** (one)
Element 4	CatalogName	**String**	Optional

ELEMENT	PARAMETER	FORMAT	RULE
Element 5	TherapyMode	**ListTherapyModeOptions**	Optional
Element 6	InfusionRate	Number with a decimal point	Optional
Element 7	RateUnit	**ListRateUnitOptions**	Please check **Rule 2** (two)
Element 8	VtbiValue	Number with a decimal point	Please check **Rule 3** (three)
Element 9	VtbiUnit	**ListVtbiUnitOptions**	Please check **Rule 2** (two)
Element 10	AmountValue	Number with a decimal point	Optional
Element 11	AmountUnit	**ListAmountUnitOptions**	Please check **Rule 2** (two)
Element 12	VolumeValue	Positive integer	Please check **Rule 4** (four)
Element 13	VolumeUnit	**ListVolumeUnitOptions**	Please check **Rule 2** (two)
Element 14	TotalDose	Number with a decimal point	Optional
Element 15	TotalDoseUnit	**ListTotalDoseUnitOptions**	Please check **Rule 2** (two)
Element 16	TotalTime	Positive integer	Optional
Element 17	TotalTimeUnit	**ListTotalTimeUnitOptions**	Please check **Rule 2** (two)
Element 18	PatientID	**String**	Optional
Element 19	NurseID	**String**	Optional
Element 20	HeightValue	Number with a decimal point	Optional
Element 21	HeightUnit	**ListHeightUnitOptions**	Please check **Rule 2** (two)
Element 22	WeightValue	Number with a decimal point	Optional
Element 23	WeightUnit	**ListWeightUnitOptions**	Please check **Rule 2** (two)
Element 24	BodySurfaceAreaValue	Number with a decimal point	Optional
Element 25	BodySurfaceAreaUnit	**ListBodySurfaceAreaUnitOptions**	Please check **Rule 2** (two)
Element 26	OrderNumber	**String** with a maximum of 19 number characters	Optional
Element 27	PumpId	**String**	Please check **Rule 5** (five)

Format Types

All Elements must be filled as Strings obeying the format described on the previous table.

- **Number format with a decimal point**: It may have decimal places after a point. For example, 10.0 or 10 are valid values
- **Positive integer format**: Without decimal places. For example, 10, 20, or 1000 are valid values

Fill Rules

There is no need to fill all Elements, because several of them are optional, but users must follow these rules:

- **Rule 1**: **DrugName** or **DrugShort** Elements must be filled. Only one of these Elements must be filled and the other one must be omitted

- **Rule 2**: It is mandatory to fill this Element if the previous Element is filled, otherwise it must be omitted
- **Rule 3**: It is optional, but if the **TotalTime** Element is filled, it must be omitted
- **Rule 4**: It is mandatory to fill if the **AmountValue** and **AmountUnit** Elements are filled, otherwise it must be omitted
- **Rule 5**: It is optional, but when filling it, users must obey the ISP or PSP prefixes, followed by decimal number digits

Writing this Block Tag by Script

The next script is an example on how to fill an Array and write it to this Block Tag.

```
Dim Arr(26) 'Array DrugList
Arr(1)  = "ampicillin"    'DrugName
Arr(3)  = "2749283"       'CatalogName
Arr(4)  = "Intermittent" 'TherapyMode
Arr(5)  = "200.0"         'InfusionRate
Arr(6)  = "ml/h"          'RateUnit
Arr(7)  = "100.0"         'VtbiValue
Arr(8)  = "ml"            'VtbiUnit
Arr(9)  = "1000.0"        'AmountValue
Arr(10) = "mg"            'AmountUnit
Arr(11) = "100.0"         'VolumeValue
Arr(12) = "ml"            'VolumeUnit
Arr(13) = "1000.0"        'TotalDose
Arr(14) = "mg"            'TotalDoseUnit
Arr(25) = "3488817"       'OrderNumber
Arr(26) = "ISP154133"     'PumpId
Set Driver = Application.GetObject("DriverAutoProgramming")
'Write DrugList
Driver.Item("DrugEntry").WriteEx Arr
```

Writing Commands Tags

These write-only PLC Tags perform important operations to manipulate internal memory, especially the **DrugEntryOperation** Tag, and to finish and send a document in **XML** format, **SendProposalList**.

ClearProposalList

Write-Only

This PLC Tag can be used to clear all combinations of therapy settings, or **Drug** items, memorized internally on this Driver before send them to a device using the **SendProposalList** Tag. Any value can be written to this Tag to execute it.

DeleteDrugEntry

Write-Only

A PLC Tag that receives an integer value to command the erasing of one item stored in this Driver's internal memory. This number value must match the position stored in memory. The first item included by the **IncludeDrugEntry** command is 0 (zero), the second one is 1 (one), and so on. When deleting an item from the internal list, the position of all other items changes, therefore the value to use on a new deletion must be reviewed according to the current situation of the stored list.

DrugEntryOperation

Write-Only

This Tag receives an integer value that determines which operation to perform with the last **DrugEntry** Block Tag stored on this Driver's internal memory. Values outside the expected value range return a writing error.

VALUE	OPERATION	COMMAND
1	ValidateDrugEntry	Validation of a new therapy configuration, a **Drug** item, which was assigned on the last writing of a **DrugEntry** Block Tag. If there are fill errors, this Driver indicates them in the Tags **LastDrugHasErrors** and **GetLastErrorsList**, and in the *WriteStatus* return parameter
2	IncludeDrugEntry	Inclusion of a new therapy configuration on this Driver's internal memory, a **Drug** item, which was assigned on the last writing of a **DrugEntry** Block Tag. Invalid values cannot be added and any attempt to include invalid values causes a writing error

WriteStatus Parameter

If this Tag is called to perform a **ValidateDrugEntry** operation using a script, the *WriteStatus* parameter can be read to check the validity of the last **DrugEntry** Block Tag, that is, a value of 0 (zero) contains no errors and a value of 1 (one) contains errors. If this feature is used, there is no need to read the **LastDrugHasErrors** Tag to find out the validity of a Block Tag. The next example demonstrates how to use this feature.

```
'Write DrugList for validation
Driver.Item("DrugEntryOperation").WriteEx 1, Now, 192, WriteStatus
'Checks whether there is a fill error
If WriteStatus Then
  MsgBox "Fill error"
Else
  MsgBox "No errors"
End If
```

SendProposalList

Write-Only

Use this Tag to command the processing of a document in **XML** format with each **Drug** item internally memorized by this Driver and then send to a device. Any problems during the assembly of this document returns an error and does not send it. This Tag waits for a device's return and, depending on that response, this Tag can return an error or a writing success. Any value can be written to this Tag to execute it.

NOTES

- Writing to this Tag without any **Drug** item on this Driver's internal memory sends an empty document, which indicates a delete command of all **ProposalLists** of a SpaceCom device.
- Writing to this Tag does not automatically erase the content of the list of valid blocks on this Driver's memory. If users want to submit new therapy settings, they must execute a writing to the **ClearProposalList** Tag.
- This is the only Tag of this Driver that establishes a communication with a device. This means this Driver can stay offline throughout its execution until the moment of executing this Tag. When executing a writing to this Tag, its online status must have been already started and the connection to a remote device must have been already started. This Driver can return to its offline status as soon as this Tag finishes executing.

Error Reading Configuration Tags

These are useful PLC Tags for validating a **DrugEntry** Block Tag and, in case it is not valid one, for finding which are the fill errors.

The values contained in these PLC Tags always refer to the last **DrugEntry** Block Tag written and after an operation of a **DrugEntryOperation** Tag, that is, while a new operation of a **DrugEntryOperation** Tag is not performed, the content of these Tags refer to the content of the previous operation. The content of these PLC Tags indicates no errors until a **DrugEntryOperation** Tag operation is performed for the first time.

GetLastErrorsList

Read-Only

Returns a single **String** containing all filling errors of parameters after performing the last writing of the **DrugEntryOperation** Tag.

LastDrugHasErrors

Read-Only

Returns 0 (zero) if there is no error after performing the last writing of a **DrugEntryOperation** Tag. If there is any error, it returns a value other than 0 (zero).

List Options Tags

These read-only Tags are useful for knowing which **Strings** are valid for some fields of a **DrugEntry** Tag. These are Block Tags that depend on the number of options in each parameter. These Tags are not fundamental to the operation of this Driver, but they can help filling fields of a **DrugEntry** Tag, according to a designer's convenience.

ListAllParameters

Read-Only

Block Tag to list all parameter names to fill for each Element of a **DrugEntry** Block Tag.

PARAMETER	VALUE
Item	ListAllParameters
Size	27

ListAmountUnitOptions

Read-Only

Lists all fill options of an **AmountUnit** Element.

PARAMETER	VALUE
Item	ListAmountUnitOptions
Size	12

ListBodySurfaceAreaUnitOptions

Read-Only

Lists all fill options of a **BodySurfaceAreaUnit** Element.

PARAMETER	VALUE
Item	ListBodySurfaceAreaUnitOptions
Size	1 (one)

ListHeightUnitOptions

Read-Only

Lists all fill options of a **HeightUnit** Element.

PARAMETER	VALUE
Item	ListHeightUnitOptions
Size	3 (three)

ListRateUnitOptions

Read-Only

Lists all fill options of a **RateUnit** Element.

PARAMETER	VALUE
Item	ListRateUnitOptions
Size	79

ListTherapyModeOptions

Read-Only

Lists all fill options of a **TherapyMode** Element.

PARAMETER	VALUE
Item	ListTherapyModeOptions
Size	2 (two)

ListTotalDoseUnitOptions

Read-Only

Lists all fill options of a **TotalDoseUnit** Element.

PARAMETER	VALUE
Item	ListTotalDoseUnitOptions
Size	32

ListTotalTimeUnitOptions

Read-Only

Lists all fill options of a **TotalTimeUnit** Element.

PARAMETER	VALUE
Item	ListTotalTimeUnitOptions
Size	1 (one)

ListVolumeUnitOptions

Read-Only

Lists all fill options of a **VolumeUnit** Element.

PARAMETER	VALUE
Item	ListVolumeUnitOptions
Size	1 (one)

ListVtbiUnitOptions

Read-Only

Lists all fill options of a **VtbiUnit** Element.

PARAMETER	VALUE
Item	ListVtbiUnitOptions
Size	1 (one)

ListWeightUnitOptions

Read-Only

Lists all fill options of a **WeightUnit** Element.

PARAMETER	VALUE
Item	ListWeightUnitOptions
Size	3 (three)

Documentation of I/O Interfaces

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

Configuration of a Driver

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

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

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

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

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

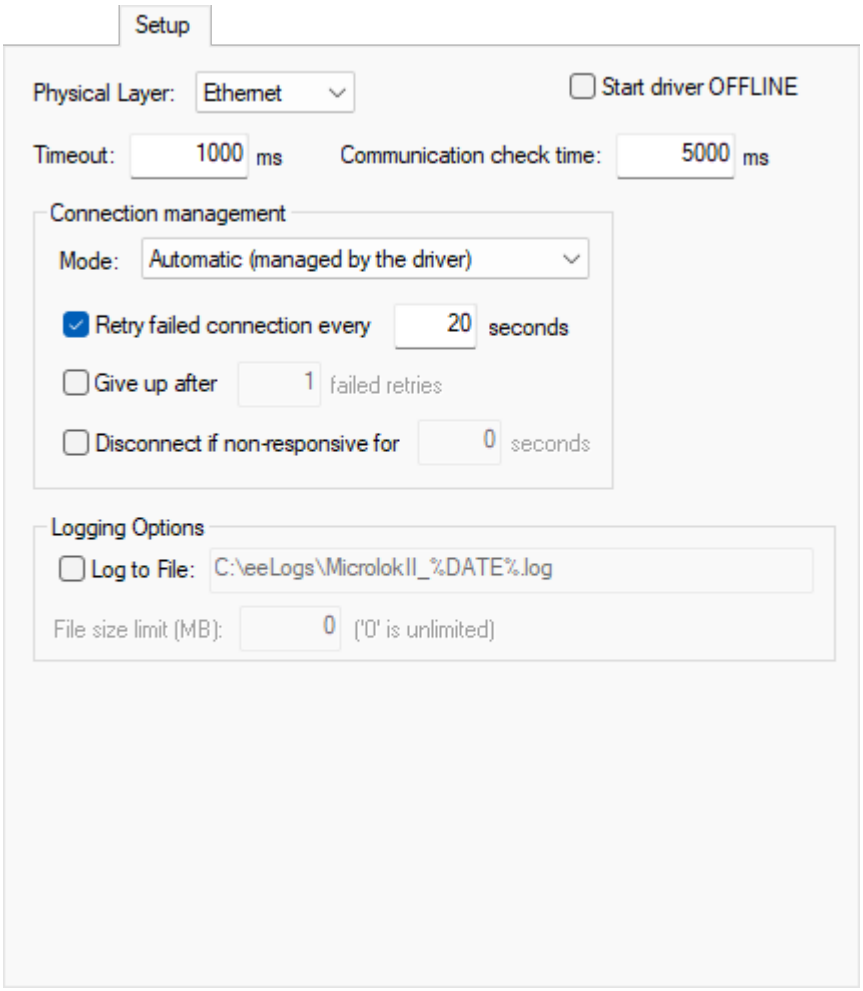
Configuration Dialog Box

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

Setup Tab

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

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



Setup tab

General options on the Setup tab

OPTION	DESCRIPTION
Physical Layer	Select the physical layer on a list. Available options are **Serial**, **Ethernet**, **Modem**, and **RAS**. The selected interface must be configured on its specific tab
Timeout	Configure a time-out, in milliseconds, for the physical layer. This is the amount of time an I/O interface waits to receive any byte from the reception's buffer
Communication check time	Set the time, in milliseconds, to define the interval at which communication is considered to be in an inactive state. As long as an I/O Driver receives valid data, its communication state is considered active. However, if during operation an I/O Driver does not receive valid data inside this period of time, the state is considered inactive. The communication state is shown in the **IO.CommunicationStatus** Tag
Start driver OFFLINE	Select this option so that a Driver starts in **Offline** mode or stopped. This means that the I/O interface is not created until this Driver is configured to **Online** mode by using a Tag in an application. This mode enables a dynamic configuration of an I/O interface at run time

Options on the Connection management group

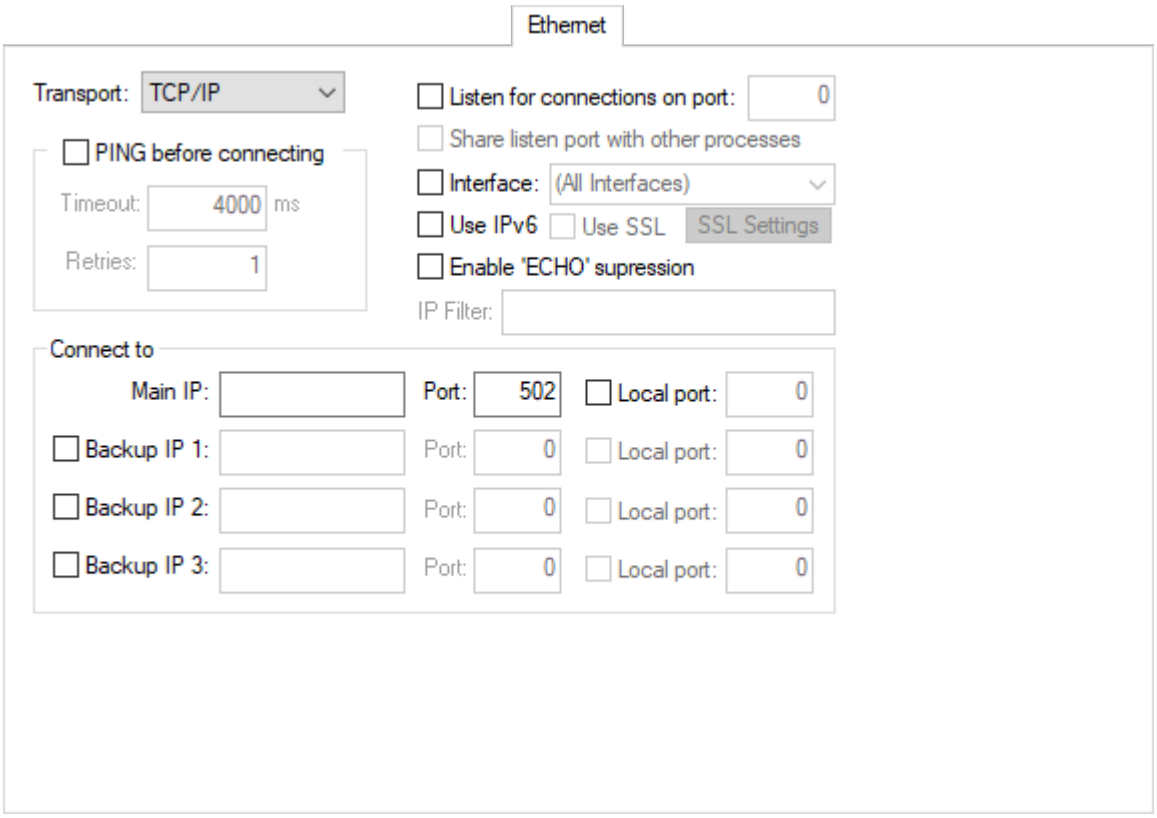
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every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

Options on the Logging Options group

OPTION	DESCRIPTION
Log to File	Enable this option and configure the name of a file to write a log. Log files can be large, so use this option for short periods of time, only for testing and debugging purposes. If the **%PROCESS%** macro is used in the log file name, it is replaced by the identifier of the current process. This option is particularly useful when using several instances of the same Driver in **Elipse E3**, thus allowing each instance to generate a separate log file. For example, when configuring this option with value "c:\e3logs\drivers\sim_%PROCESS%.log", it generates a file named c:\e3logs\drivers\sim_00000FDA.log for process **0FDAh**. Users can also use the **%DATE%** macro in the file name. In this case a log file is generated every day, in the format **aaaa_mm_dd**. For example, when configuring this option with value "c:\e3logs\drivers\sim_%DATE%.log", it generates a file named c:\e3logs\drivers\sim_2005_12_31.log in 12/31/2005 and a file named c:\e3logs\drivers\sim_2006_01_01.log in 01/01/2006. Similarly, the **%DATE_HOUR%** macro generates one log file per hour, in the format **aaaa_mm_dd_hh**
File size limit (MB)	Configure the log file size limit, in megabytes. A value equal to 0 (zero) means that there is no size limit for the log file

Ethernet Tab

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



Ethernet tab

Available options on the Ethernet tab

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

OPTION	DESCRIPTION
	of determining a successful connection before trying to open a socket with a device. The time-out of a connection with a socket can be very high. The available options are: • **Timeout**: Specify the number of milliseconds to wait for a reply from a **ping** command. Users must use a **ping** command to check the normal reply time, configuring this option for a value above that average. Usually this value can be configured between 1000 and 4000 milliseconds, that is, between 1 (one) and 4 (four) seconds • **Retries**: Number of retries of a **ping** command, not counting the first attempt. If all attempts fail, then the socket connection is aborted

Available options on the Connect to group

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

General Configurations

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

I/O Tags

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

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

IO.CommunicationStatus

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

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

IO.IOKitEvent

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

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

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Ethernet Interface Configuration

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

I/O Tags

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

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

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

IO.Ethernet.IPSelect

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

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

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

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

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

IO.Ethernet.IPSwitch

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

Any value written to this Tag forces a manual switchover. If the main IP address is active, then the first alternative or backup IP address is activated, and so on for all alternative IP addresses and returning to the main address until a connection is established.

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

IO.Ethernet.SocketState

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

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

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

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

IO.Ethernet.BackupEnable[2,3]

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

IO.Ethernet.BackupIP[2,3]

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

IO.Ethernet.BackupLocalPort[2,3]

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

IO.Ethernet.BackupLocalPortEnable[2,3]

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

IO.Ethernet.BackupPort[2,3]

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

IO.Ethernet.IPFilter

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

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

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

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

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

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

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

IO.Ethernet.ListenIP

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

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

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

IO.Ethernet.MainLocalPort

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

IO.Ethernet.MainLocalPortEnable

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

IO.Ethernet.MainPort

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

IO.Ethernet.PingEnable

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

IO.Ethernet.PingTimeoutMs

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

IO.Ethernet.PingTries

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

IO.Ethernet.ShareListenPort

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

IO.Ethernet.SupressEcho

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

IO.Ethernet.Transport

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

IO.Ethernet.UseIPv6

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.2	06/12/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37942*).
1.0.1	06/05/2020	M. Ludwig	• First version of this Driver.

Headquarters
Rua Mostardeiro, 322/Cj. 902, 1001 e 1002
90510-002 — Porto Alegre — RS
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

Branch in Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist.
807 — Kaohsiung City — Taiwan
Phone: (+886 7) 323-8468
Fax: (+886 7) 323-9656
E-mail: evan@elipse.com.br

Check our website for information about a representative in your country.

www.elipse.com.br

kb.elipse.com.br
forum.elipse.com.br
www.youtube.com/elipsesoftware
elipse@elipse.com.br

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)