Elipse Ping Driver

File Name	PingDrv.dll
Manufacturer	Elipse Software
Devices	Any device connected via IP (Internet Protocol)
Protocol	ICMP (Internet Control Message Protocol)
Version	3.0.14
Last Update	11/05/2025
Platform	Win32
Dependencies	None
Superblock Readings	No
Level	0

Introduction

The Elipse Ping Driver allows executing a **ping** command to any station of an IP (*Internet Protocol*) network, including via Internet. This Driver is useful in network applications that must check for partners on a network.

Starting with version **2.0**, this Driver allows performing **ping** commands in multiple threads, which allows decreasing real scan time in applications with a large number of Tags. If the maximum number of threads is defined as 0 (zero) on the **Ping** tab, this Driver's behavior remains the same as in version **1.0**, with all **ping** commands executed sequentially.

In version **2.0**, this Driver was ported to **Elipse Software**'s **IOKit** library. This library allows Drivers to have access to a physical layer, **Serial**, **Ethernet**, **Modem**, or **RAS**, in a standard way. This Driver, however, does not use any physical layer specified in the **IOKit** library, so users must configure the **Physical Layer** option on the **Setup** tab of **this Driver's configuration window** with the value **None**.

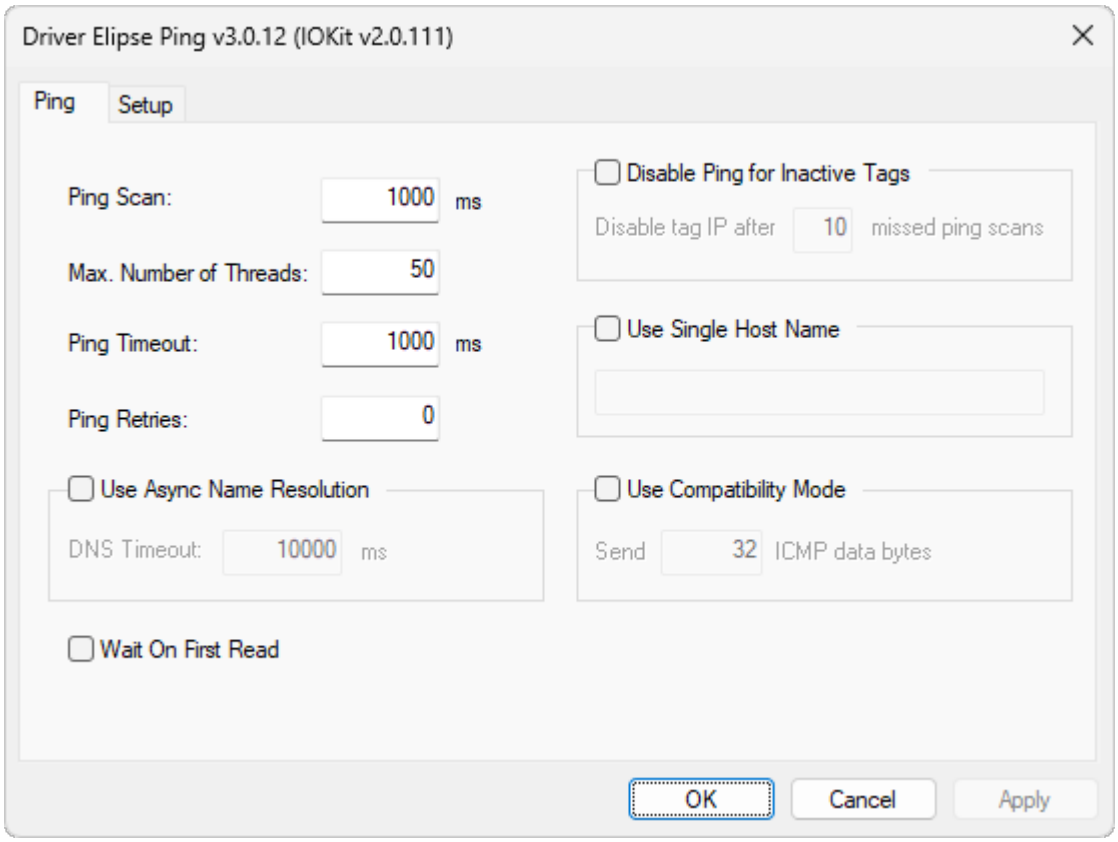
For more information about the configuration of **IOKit** library, please check topic **Documentation of I/O Interfaces**.

Driver Configuration

This Driver does not use **[P]** configuration parameters. All **IOKit** and specific configurations must be performed on this Driver's configuration window, or on **Elipse SCADA**'s extra settings. On this window, the **Ping** tab contains specific configurations for this Driver, while the **Setup** tab contains communication configurations for **Elipse Software**'s **IOKit** library. For more information about **IOKit** library configurations, please check topic **Documentation of I/O Interfaces**.

Configuring Properties

The configuration options available on the configuration window are also accessible in **Offline** mode, at run time, using the **Strings** described on topic **Configuring Properties Offline**. The next figure shows specific configurations for this Driver on the **Ping** tab.



Ping tab

This tab is specific to this Driver and its configuration options are described on this topic. On the **Setup** tab, users must select the **None** item in the **Physical Layer** option and users can also enable log generation. All other options for **IOKit** library are ignored by this Driver.

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

Available options on the Ping tab

OPTION	DESCRIPTION
Ping Scan	Defines an internal scan time for each Tag, in case of asynchronous or multi-threaded **ping** commands
Max. Number of Threads	Defines the maximum number of threads to create. If this value is greater than the number of Tags in the application, this Driver creates a thread per Tag. Otherwise, the maximum number of threads is given by this value. If this value is configured as 0 (zero), **ping** commands are executed synchronously, in the same reading thread of each Tag. Please check the next **note** for more information about this option
Ping Timeout	Time-out of a **ping** command
Ping Retries	Number of times this Driver resends **ping** requests at each occurrence of a time-out before returning 0 (zero) in the **Value** field of **Simple Ping** Tags
Use Async Name Resolution	This option enables resolving host names asynchronously, which allows configuring a time-out for that operation by loosing some performance. Use this option only in cases where a DNS (*Domain Name System*) operation may last longer or fail frequently

OPTION	DESCRIPTION
DNS Timeout	Time-out to resolve host names, used in case the **Use Async Name Resolution** option is enabled
Wait On First Read	If this option is enabled, this Driver waits to perform the first **ping** command during the first read of each Tag, so that it returns a value. If it is disabled, the first reading of a Tag only schedules a **ping** command and the operation returns no value and does not change any Tag property in an application. A Tag only starts returning values when the first **ping** command is performed by this Driver. Users must leave this option disabled preferably, because it tends to improve performance
Disable Ping for Inactive Tags	This option allows this Driver to set Tags to an inactive status in their internal scans, if an application is not requesting a reading during a configurable scan period. In an inactive status, the IP address or the host name referring to a Tag is no longer used in internal scans of this Driver. A Tag goes to an active status as soon as a new reading request is received from an application
Disable tag IP after *n* missed ping scans	Informs this Driver how many internal scans unread by an application must occur before a Tag is considered inactive
Use Single Host Name	This option must be used only for **Elipse SCADA**. Using this option limits an application to use a single Tag. Its text box allows defining a host name used for an application's single Tag, which must be defined with *N1*, *N2*, *N3*, and *N4* parameters as 0 (zero). For **Elipse E3**, **Elipse Power**, or **Elipse Water**, leave this option disabled and use the **Device** property of each Tag to define a host name. Please check the next **note** for more information about this option
Use Compatibility Mode	This option allows defining a certain number of data bytes to send in a frame of an echo request of ICMP (*Internet Control Message Protocol*) protocol. The default value of this option is disabled. Please check the next **note** for more information about this option
Send *n* ICMP data bytes	This option indicates the number of data bytes to send in a frame of an echo request of ICMP protocol, if the **Use Compatibility Mode** option is enabled. Please check the next **note** for more information about this option

NOTES

- Starting with version **2.1** of this Driver, the maximum number of threads is limited to 100. If this Driver is loaded in applications that already configured a greater number, the number of threads is limited to that maximum number of 100 and writes that limitation to its log.
- When the **Use Single Host Name** option is enabled, always use the **Max. Number of Threads** option equal to 0 (zero), that is, a synchronous reading.
- The **Use Compatibility Mode** option was implemented in this Driver because it is relatively common to find servers that do not respond correctly to an echo request without data, whether by any security policy or by implementation problems. In these cases, this Driver may not be successful with a **ping** command to servers that return success when using a Windows Command Prompt. This happens because a Windows **ping** command, by default, sends 32 data bytes in a frame of an echo request, while this Driver sends an empty **ping** command, that is, without data bytes. If this condition is detected in the servers in use, users must enable the **Use Compatibility Mode** option and define a suitable number of data bytes that can be accepted by a server in the **Send *n* ICMP data bytes** option. It is recommended to start with 32 bytes, which is the default value for a **ping** command in Windows Command Prompt and accepted for most servers. Otherwise, it is recommended to leave this option disabled, keeping a more optimized condition, without data bytes.
- To test the need to use the **Use Compatibility Mode** option, users can customize the number of data bytes sent in echo requests of a **ping** command in Windows Command Prompt by using the **-l** option, **Send buffer size**. For more information, please type **ping /?** in a Windows Command Prompt.

Configuring Properties Offline

The available parameters in the **configuration of properties** can also be accessed at run time if this Driver is started in **Offline** mode, by using **String**-type parameters described on the next table.

Parameters for offline configuration

PARAMETER	DATA TYPE
Ping.PingTimeout	Unsigned integer
Ping.MaxNumberOfThreads	Unsigned integer
Ping.PingScan	Unsigned integer
Ping.WaitOnFirstRead	**Boolean** (zero or one)
Ping.DisableInactiveTags	**Boolean** (zero or one)
Ping.MissedScans	Unsigned integer
Ping.DNSTimeout	Unsigned integer
Ping.UseAsyncDNS	**Boolean** (zero or one)
Ping.SingleHostName	**String** with the name of a host
Ping.UseSingleHostName	**Boolean** (zero or one)
Ping.UseCompatibleMode	**Boolean** (zero or one)
Ping.NumberOfDataBytes	Unsigned integer
Ping.PingDefaultRetries	Unsigned integer

For more information about the offline configuration at run time, please check topic **Documentation of I/O Interfaces**.

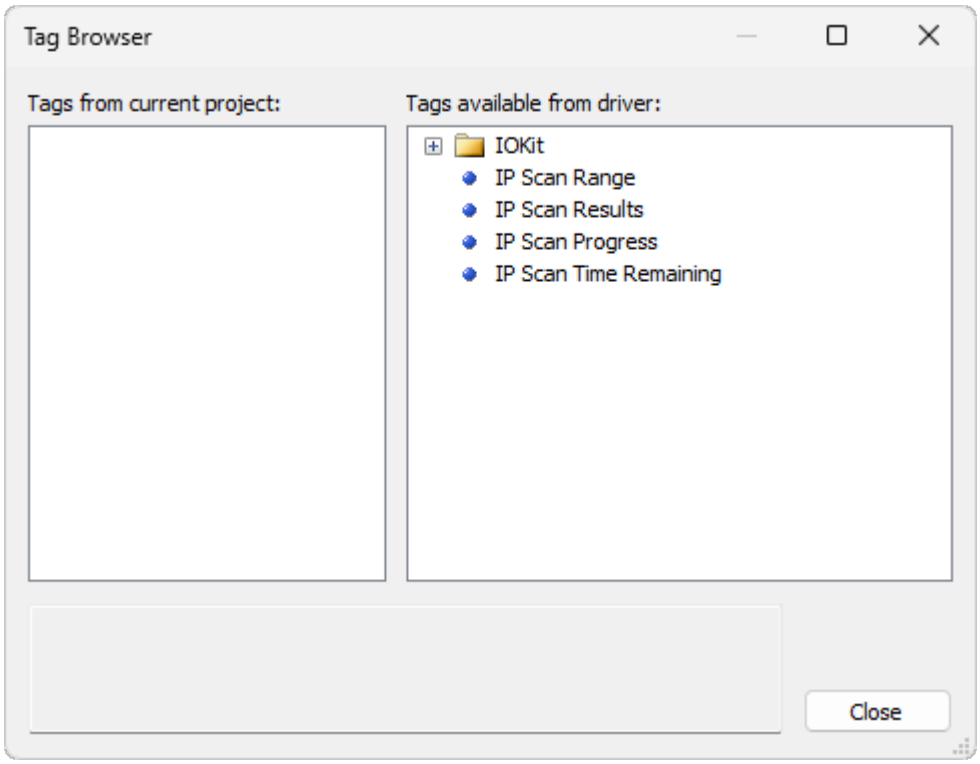
Tag Reference

This Driver contains simple **ping** Tags and also Tags to scan IP addresses.

Simple Ping Tags perform **ping** commands to unique hosts, specified using a configuration by **Strings** or by *N* parameters.

IP Scan Tags only allow configuration by **Strings**. This feature is only available in **Elipse E3**, in **Elipse Power**, and in **Elipse Water**, therefore it cannot be used in **Elipse SCADA**.

Notice that there is no need to type **Item** parameters of scan Tags, as this Driver has a Tag Browser feature in **Elipse E3**, in **Elipse Power**, or in **Elipse Water**, shown on the next figure.



Tag Browser window

To add scan Tags in an application, drag them from the **Tags available from driver** list on the right to the **Tags from current project** list on the left.

The next topics describe the different types of Tags individually.

Simple Ping Tag

Simple Ping Tags correspond to a specific host or IP address, to which a **ping** command is sent at each scan cycle of this Driver.

Simple Ping Tags are read-only and return 0 (zero) if the **ping** command fails or 1 (one) if the **ping** command is successful.

Specifying a host address can be performed by name or directly by IP address.

Detection by IP address is faster than detection by host name, because the step to resolve the name via DNS is skipped.

Configuring Tags by IP address still remains the same as in versions previous to version **2.0**. Configuring Tags by name in **Elipse E3**, in **Elipse Power**, and in **Elipse Water**, on the other hand, now allows configuring several Tags, and it is not performed anymore on the configuration window but in the **Device** field of each Tag.

N1	First segment of an IP address
N2	Second segment of an IP address
N3	Third segment of an IP address
N4	Fourth segment of an IP address

For example, to define an IP address 200.248.184.102, users must configure the *N1* parameter with value 200, the *N2* parameter with value 248, the *N3* parameter with value 184, and the *N4* parameter with value 102.

Configuration by Host Name

To configure by host name in **Elipse E3**, in **Elipse Power**, and in **Elipse Water**, add the name of a host to the **Device** property of this Tag. There is no need to change the other parameters of this Tag. Users can define several Tags, with different names.

For **Elipse SCADA**, use the **Use Single Host Name** option to define the name of a host. In this case, users can only define a single Tag in an application, which must have its *N* parameters all set to 0 (zero). This behavior is the same as the **Check by name** option on version **1.0** of this Driver.

IP Scan Tags

This topic describes Tags used to perform a scan of IP addresses, in which users can detect all valid IP addresses, that is, the ones responding to **ping** commands on a certain range of possible IP addresses.

Scan Tags are configurable only by **Strings**, a feature available only in **Elipse E3**, in **Elipse Power**, and in **Elipse Water**, therefore it does not work in **Elipse SCADA**.

The scan process starts with a writing of a **String** specifying a range of valid IP addresses in the **IP Scan Range** Tag.

During that scan, users can monitor its progress using the **IP Scan Progress** and **IP Scan Time Remaining** Tags.

After this process finishes, the list of detected IP addresses, that is, the ones that responded correctly to a **ping** command, can be retrieved by reading the **IP Scan Results** Tag, which returns that list using the feature of Tags reported by events. For more information about this feature, please check *Elipse E3 User's Manual*.

IP Scan Range

Reading and Writing

This Tag defines a range of IP addresses this Driver scans searching for valid addresses, that is, the ones responding to a **ping** command, and also starts the scan procedure.

When reading, if there is an ongoing scan, returns a **String** defining the range to scan. If there is no ongoing scan, returns an empty **String**.

This Tag is configured by **Strings** in the **Device** and **Item** fields of **Elipse E3**, **Elipse Power**, and **Elipse Water**, and the range is configured in the **Value** field. The *N3* and *N4* parameters, however, are also used. To do so, please check the list of parameters later on this topic.

NOTE

By writing any valid range, described later on this topic, in the **Value** field of this Tag immediately starts a scan process.

To cancel an ongoing scan, write an empty **String** in this Tag.

The progress of a scan can be monitored using the **IP Scan Progress** and **IP Scan Time Remaining** Tags.

After finishing a scan, the list of detected IP addresses can be retrieved using the **IP Scan Results** Tag.

The next table contains a description of the parameters of this Tag.

Parameters of the IP Scan Range Tag

PARAMETER	DESCRIPTION
N3	Determines a time-out for each **ping** command during a scan. If configured as 0 (zero), uses the default time-out configured in the **Ping Timeout** option
N4	Determines the number of retries for each IP address. If configured as 0 (zero), assumes the default value configured in the **Ping Retries** option
Device	This parameter is not used for this Tag and must be left blank
Item	This parameter must be configured with the fixed value "ScanIP.Range"
Value	Users must write in this parameter a **String** defining a range to scan in the scan process, or an empty **String** to cancel any ongoing scan, if available. Writing a valid range immediately starts a scan

The range can be defined using the formats described next.

CIDR (Classless Inter-Domain Routing) Notation

In this notation, users can add to an address in **IPv4** (*Internet Protocol version 4*) a number preceded by a slash mark, which determines the number of invariable bits, starting from the most significant ones in the provided reference IP address, according to the next examples.

- **192.168.10.0/24**: Scans 256 hosts between 192.168.10.0 and 192.168.10.255
- **192.168.0.0/16**: Scans 65536 IP addresses (hosts) between 192.168.0.0 and 192.168.255.255

Octet Range Addressing

Allows specifying different ranges for each one of the four octets of an address in **IPv4** format, by using a hyphen between minimum and maximum values of an octet. If the value on the left of that hyphen (minimum value) is omitted, assumes a value of 0 (zero). If the maximum value is omitted, assumes a value of 255, according to the next examples.

- **192.168.0-255.1-254**: Scans all IP addresses starting in "192.168", but skipping the ones finishing in 0 (zero) or 255 (fourth segment)
- **192.168.0-.1-254**: Same configuration of the previous example, assuming the maximum value for the third segment as 255
- **192.168.-255.1-254**: Same configuration of previous examples, assuming the minimum value for the third segment as 0 (zero)

- **0-255.0-255.13.37**: Scans all Internet searching for IP addresses ending in "13.37"
- **0-.0-.13.37**: Same configuration of the previous example
- **0-.-255.13.37**: Same configuration of the two previous examples
- **0-.0-.0-.-10**: Scans all Internet searching for hosts whose address in **IPv4** format ends in an octet with values between 0 (zero) and 10

IP Scan Time Remaining

Read-Only

Returns the number of seconds expected for a scan process to end. This Tag is configured by **Strings** in the **Device** and **Item** fields of **Elipse E3**, **Elipse Power**, and **Elipse Water**, and therefore it cannot be used in **Elipse SCADA**. The next table contains the description for the parameters of this Tag.

Parameters of the IP Scan Time Remaining Tag

PARAMETER	DESCRIPTION
Device	Not used, leave it blank
Item	A **String** with a fixed value "ScanIP.TimeRemainingSec"
Value	A 64-bit floating point number (**Double** or 64-bit IEEE754), corresponding to the expected remaining seconds to finish an ongoing scan. If there is no ongoing scan, or if it is already finished, returns 0 (zero)

IP Scan Progress

Read-Only

Returns a percentage of progress for an ongoing scan. This Tag is configured by **Strings** in the **Device** and **Item** fields of **Elipse E3**, **Elipse Power**, and **Elipse Water**, and therefore cannot be used in **Elipse SCADA**. The next table contains the description for the parameters of this Tag.

Parameters of the IP Scan Progress Tag

PARAMETER	DESCRIPTION
Dispositivo	Not used, leave it blank
Item	A **String** with a fixed value "ScanIP.Progress"
Valor	A 64-bit floating point number (**Double** or 64-bit IEEE754) with a value between 0 (zero) and 100, corresponding to the percentage of a scan already performed, between 0% and 100%. If there is no active scan, returns the value 100

IP Scan Results

Read-Only

This Tag is reported by events and contains a list of detected IP addresses, that is, the ones responding to a **ping** command after the last scan started by writing a valid range in the **IP Scan Range** Tag. This Tag is configured by **Strings** in the **Device** and **Item** fields of **Elipse E3**, **Elipse Power**, and **Elipse Water** and therefore cannot be used in **Elipse SCADA**. The next table contains the description for the parameters of this Tag.

Parameters of the IP Scan Results Tag

PARAMETER	DESCRIPTION
Device	Not used, leave it blank
Item	A **String** with the fixed value "ScanIP.Results"
Value	The values of this Tag are reported by events at the end of a scan process. One by one, all detected IP addresses are returned as **Strings** in **IPv4** format, with an **OnRead** event of this Tag occurring for each returned IP address. After returning all detected IP addresses, it always returns an empty **String** indicating the end of the list of IP addresses. For more information on how to handle Tags reported by events in an application, please check *Elipse E3 User's Manual*

Documentation of I/O Interfaces

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

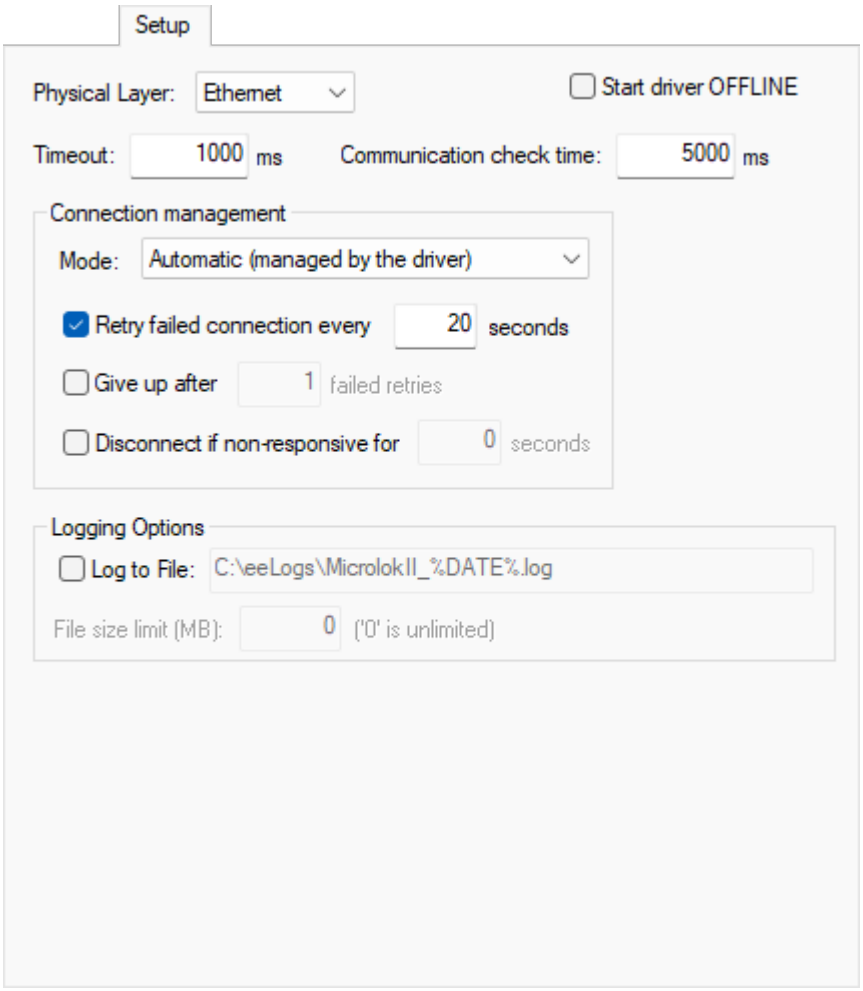
Configuration Dialog Box

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

OPTION	DESCRIPTION
	Tag in an application. This mode enables a dynamic configuration of an I/O interface at run time

Options on the Connection management group

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

Options on the Logging Options group

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

General Configurations

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

I/O Tags

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

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

IO.CommunicationStatus

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

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

IO.IOKitEvent

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

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

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
3.0.14	11/05/2025	M. Ludwig	• Fixed a GPF (*General Protection Failure*) when executing a **ping** command (*Case 39008*).
3.0.13	09/02/2025	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 37963*).
3.0.12	09/18/2019	C. Mello	• Driver ported to Visual Studio 2017 (*Case 27518*).
3.0.11	05/11/2017	A. Hertzog	• Adjustments for compatibility with multiple instances (*Case 17678*).
3.0.9	10/29/2014	A. Quites	• Implemented a scan of ranges of IP addresses (*Case 15917*). • Implemented an option to retry failed **ping** commands before returning a value of 0 (zero) (*Case 17412*). • Driver ported to **IOKit** version **2.0** (*Case 15977*).
2.1.1	09/01/2011	A. Quites	• Created a compatibility mode to avoid problems with servers not accepting **ping** commands without data bytes (*Case 12077*). • Fixed an error in which this Driver could return success for **ping** commands to non-existing servers when using high time-out values (*Case 12064*). • Implemented a limit value of 100 to the maximum number of threads (*Case 12445*). • Small performance optimizations.
2.0.1	04/24/2009	A. Quites	• Implemented the execution of **ping** commands in parallel in multiple threads (*Case 10210*). • Driver ported to **IOKit** library.
1.0.1		R. Haetinger	• All publications before 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)