SNMP Manager (50 Connections) Driver

File Name	SNMPManager.dll
Manufacturer	Simple Network Management Protocol
Devices	
Protocol	SNMP V1 and V2C
Version	2.0.28
Last Update	01/28/2026
Platform	Win32 and Elipse E3 version 3.2 or later
Dependencies	IOKit version 2.0 or later
Superblock Readings	No
Level	31301

Introduction

The SNMP Manager (50 Connections) Driver communicates with SNMP Agents using the SNMP (*Simple Network Management Protocol*) protocol. According to SNMP nomenclature, **Agents** are network devices, such as switches, routers, multiplexers, and no-breaks, among others, that produce information to manage, which are therefore sent to a **Manager** system, in this case the SNMP Manager (50 Connections) Driver.

This Driver currently supports versions **V1**, **V2C**, and **V3** of SNMP specification, allowing the following resources:

- Communication with up to 50 devices using a single Driver (DLL)
- Importing or creating Tags from devices using files in **MIB** (*Management Information Base*) format
- Asynchronously reading variables by grouping several requests into a single query (polling)
- Synchronously writing parameters
- Supporting unsolicited messages, that is, receiving SNMP traps

NOTE

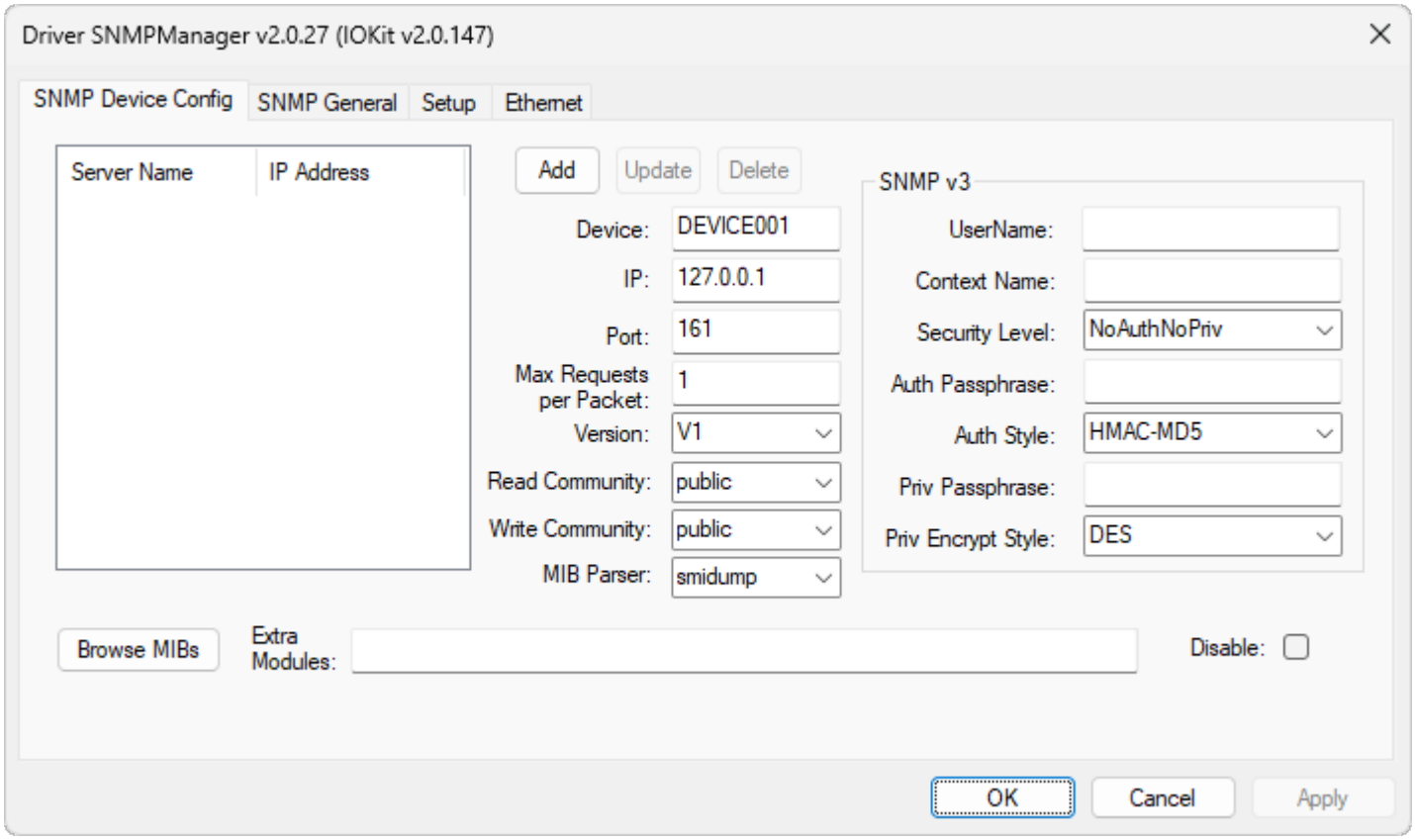
When reaching this limit of 50 devices, this Driver does not create more instances. If there are Tags configured for these hosts, the values for these Tags are invalid. To address more Agents, use a new instance of this Driver.

Configuration Parameters

The **[P]** parameters are not used by this Driver. These configurations are the same as the ones in **Elipse Software**'s **IOKit** library, adding specific property tabs, the **SNMP Device Config** tab, which configures devices or Agents with which communication occurs, and the **SNMP General** tab, which configures all other parameters of this Driver. For more information about the other tabs of this Driver, please check topic **Documentation of I/O Interfaces**.

SNMP Device Config Tab

Select the **SNMP Device Config** tab to configure Agents or devices, according to the next figure.



SNMP Device Config tab

The available options on this tab are described on the next table.

Available options on the SNMP Device Config tab

OPTION	DESCRIPTION
Add	Adds a device or Agent
Update	Updates the configuration of the selected device or Agent. Changes performed without confirmation using this option are discarded
Delete	Removes the selected device or Agent
Device	Symbolic name of a device or Agent. This name is used only to identify a device for this Driver, mapping it to the IP address and UDP/IP port configured in the **IP** and **Port** options, respectively, in addition to other settings
IP	Informs the IP address of an Agent
Port	Informs the UDP/IP port in which an Agent expects for a connection. The default UDP/IP port is 161, but some devices or simulators can use other UDP/IP ports
Max Requests per Packet	Maximum number of requests for variables, or OIDs, to perform in a single message for an Agent. The more requests can be performed at each communication, the quicker and more efficient is the general scan of variables configured for a device. **NOTE**: Variables not declared for exclusive reception of Traps use a polling method, always respecting the maximum number of variables for each request. Please check topic **Tag Reference** for more information

OPTION	DESCRIPTION
Version	Version of the SNMP protocol used by an Agent. Currently this Driver supports versions **V1**, **V2C** and **V3**
Read Community	Text used as a parameter on SNMP messages to indicate the group or community of variables to which this request belongs. The most common community is public, but there can also be variables from a private community or other communities created by other companies or users. In this case, type the name of a community. In case of using more than one community, their names must be separated by semicolons, such as "public;communityA;communityB". The community used for each Tag, which can only be one, is defined by the *N2* parameter, in which the value 0 (zero) is the first community on that list, the value 1 (one) is the second community on that list, and so on
Write Community	Inform the default community used for writing. Only one community value is accepted for writing
MIB Parser	Up to version **2.0.21**, this Driver used only the **smidump** interpreter for files in **MIB** format, generating a view of Tags that can be imported to an application using Tag Browser. Starting with version **2.0.22**, users can also use the **snmptranslate** interpreter, which can generate different results depending on the file in **MIB** format
Browse MIBs	Opens a dialog box to select files in **MIB** format to import, which provide patterns for Tags
Extra Modules	List of modules imported using the **Browse MIBs** option
Disable	Disables this device, not establishing communication with this Driver when it starts, and also hides an Agent from the list of devices on the **Tag Browser** window
SNMP v3 - UserName	Name of a user to use when the selected version is **V3**. It must be compatible with the name of a user defined in an Agent
SNMP v3 - Context Name	Name of a context to use when the selected version is **V3**. It must be compatible with the name of a context defined in an Agent
SNMP v3 - Security Level	Defines options for authentication and privacy. The authentication option uses a hash mechanism to check whether a message was changed in the transmission process. The privacy option means that, in addition to authentication, data is encrypted and therefore invisible to an external observer. Privacy is only available with authentication, therefore the available options are **NoAuthNoPriv**: There is no security functionality and the protocol behaves as in version **V2C**, **AuthNoPriv**: Uses authentication without privacy, or **AuthPriv**: Uses authentication and privacy
SNMP v3 - Auth Passphrase	Informs a password used for authentication. This password must match the password defined in an Agent
SNMP v3 - Auth Style	Algorithm for authentication. This algorithm is also used to transform authentication and privacy passwords into a private key, thus not exposing this password in plain text.

OPTION	DESCRIPTION
	The available options are **HMAC-MD5**, **HMAC-SHA1-96**, or **HMAC-SHA2-256**
SNMP v3 - Priv Passphrase	Informs a password used for privacy. The password must match the password defined in an Agent
SNMP v3 - Priv Style	Algorithm for privacy. The available options are **DES**, **3DES**, **AES128**, **AES196**, or **AES256**

Important Information about Authentication and Cryptography Algorithms in SNMP Protocol V3

The current official specification of the SNMP protocol only defines the **MD5** and **SHA1-96** algorithms for authentication and **DES** and **AES128** algorithms for cryptography. However, over time new algorithms have been developed and use longer keys to prevent breaching communication, such as using a brute force method, and many manufacturers adopt these new algorithms on their own.

In case of selecting an authentication option that generates a key shorter than the minimum needed for a cryptography algorithm, this Drivers uses a method for extending a key, described on document *Extension to the User-Based Security Model (USM) to Support Triple-DES EDE in "Outside" CBC Mode* and used by Cisco and other manufacturers. Notice that this mechanism is not used when authentication is equal to **SHA2-256**, because this authentication already produces a key with the right size. The next tables contain the default sizes of keys for each cryptography algorithm.

Sizes for authentication keys

AUTHENTICATION	SIZE
MD5	16 bytes
SHA1-96	20 bytes
SHA2-256	32 bytes

Sizes for cryptography keys

CRYPTHOGRAPHY	SIZE
DES	Needs a key with 16 bytes
3DES	Needs a key with 32 bytes
AES128, AES192 e AES256	Needs keys with 16, 24, and 32 bytes, respectively

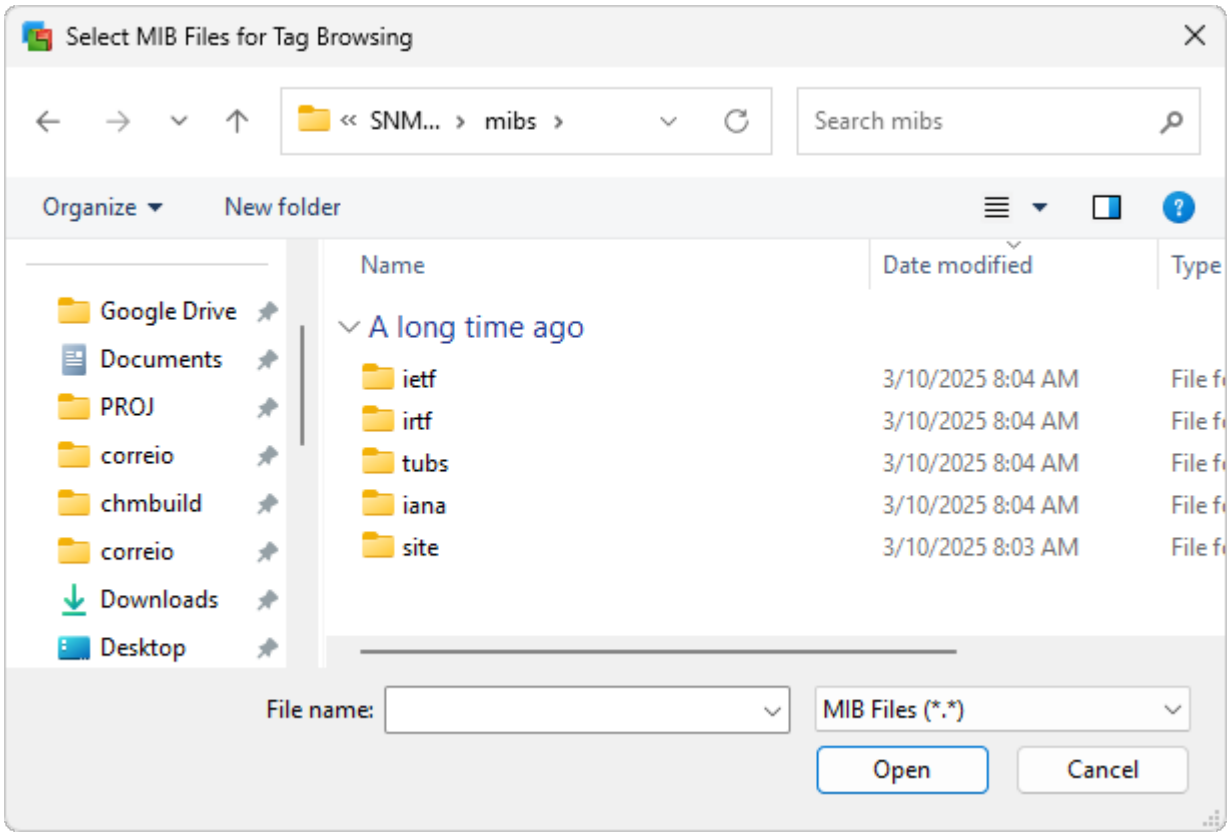
Therefore, when selecting the **MD5** authentication option and the **3DES** cryptography option, the algorithm automatically extends the key from 16 to 32 bytes.

MIB Files

The Tags of an Agent can be created by importing files in **MIB** (*Management Information Base*) format, retrieved from public domain and standardized according to IETF, IRTF, and IANA standards, among others, or provided by Agent's or device's manufacturers.

Click **Browse MIBs** to open a dialog box to select files in **MIB** format for importing. After selecting the files, an entry is created in the declaration section of devices corresponding to an Agent, in addition to a folder on the Tag Browser window containing all Tags identified.

Together with this Driver, there is a **mibs** folder with a very complete and updated set of files in standard **MIB** format, as shown on the next figure.



Window to select MIB files

If users want to import files in **MIB** format provided by manufacturers of devices or third-party manufacturers, these files must be copied to the **mibs/site** folder.

For this import process to succeed, each file in **MIB** format must have the exact same name of the module contained in this file, that is, the name found before the keyword **DEFINITIONS** at the beginning of this file, without extension or additional characters.

After selecting the files in **MIB** format, the import process is executed and, if successful, the **Extra Modules** list of the selected device is filled with the new modules, as a list separated by semicolons. This operation is equivalent of manually editing this list and clicking **Update**.

These modules are available on the Tag Browser window of **Elipse E3**, **Elipse Power**, or **Elipse Water** as folders containing all imported objects and maintaining the hierarchy of modules, nodes, tables, rows. columns, groups, notifications, and scalars from each file in **MIB** format in each device.

NOTE
To add the same file in **MIB** format to several devices, add this file in the **Extra Modules** option for each one of these Agents.

The import process provides the Tags with the parameters described on the next table.

Tag parameters from the import process

PARAMETER	DESCRIPTION
Name	Default symbolic name of a scalar or column of a table, as retrieved from a file in **MIB** format

PARAMETER	DESCRIPTION
ParamDevice	Name of a user-defined Agent on the list of devices or Agents
ParamItem	Default number of an OID (*Object Identifier*) address already in **ASN.1** (*Abstract Syntax Notation*) format, used by the SNMP protocol and retrieved from a file in **MIB** format

When a Tag references a scalar object, that is, a simple instance, the OID address ends with ".0". When a Tag represents a column of a table, this Tag is created with a default index 1 (one), indicated by brackets, that is, [1].

Tables in the SNMP protocol may contain n rows. Tags, which are the columns, are therefore created with the address of row 1 (one) by default. In case of Tags for other rows, change the index or create copies of a Tag pointing to other indexes, according to the next examples.

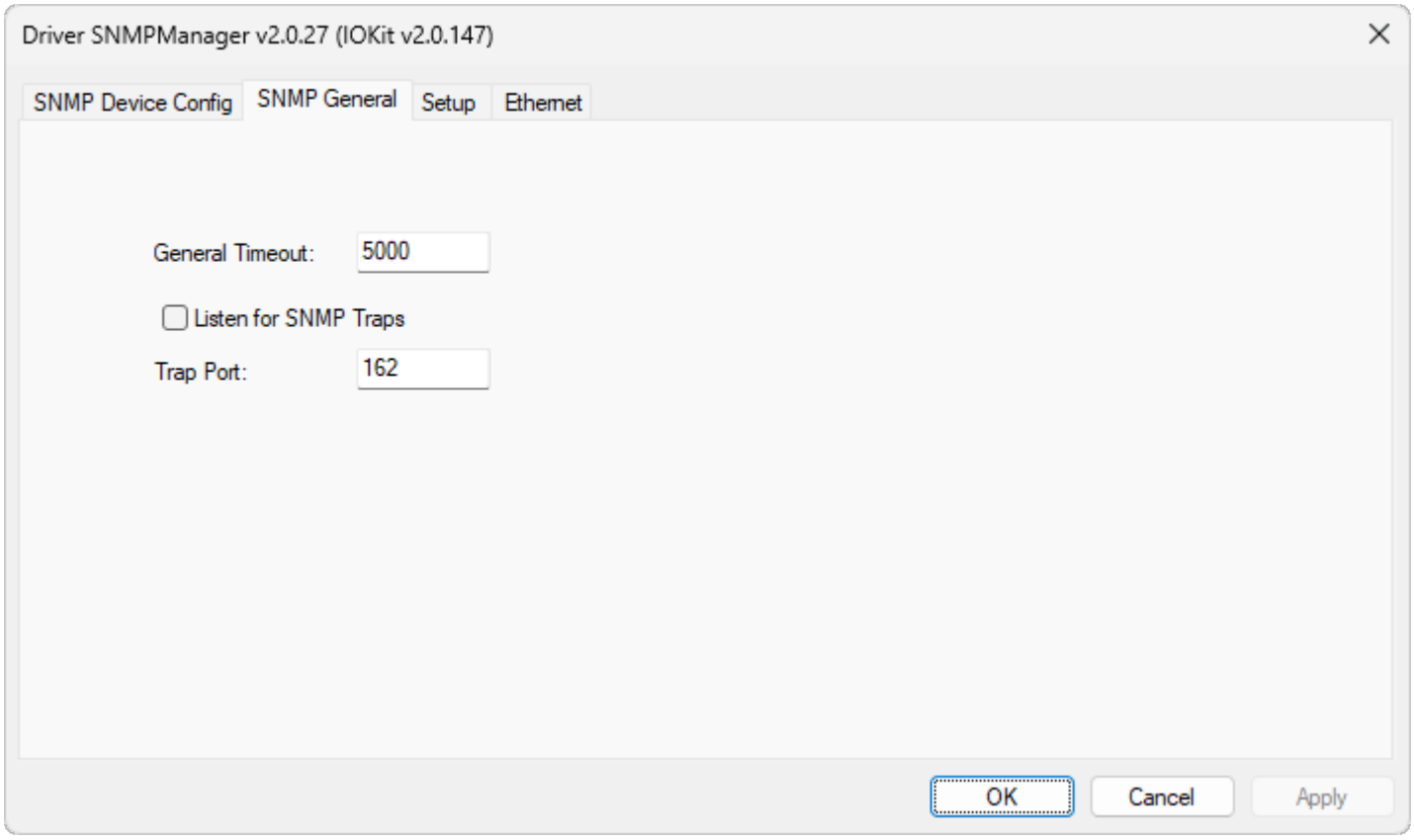
```
'Default Tag for a column called lldpPortNum:
lldpPortNum.1 = 1.0.8802.1.1.2.1.3.7.1.1[1]

'For Tags to other rows, copies can be created:
lldpPortNum.0 = 1.0.8802.1.1.2.1.3.7.1.1[0]
lldpPortNum.2 = 1.0.8802.1.1.2.1.3.7.1.1[2]
lldpPortNum.3 = 1.0.8802.1.1.2.1.3.7.1.1[3]
```

If a table does not contain a certain cell, the quality of this Tag is configured as **Bad**.

SNMP General Tab

Select the **SNMP General** tab to configure general options of this Driver.



SNMP General tab

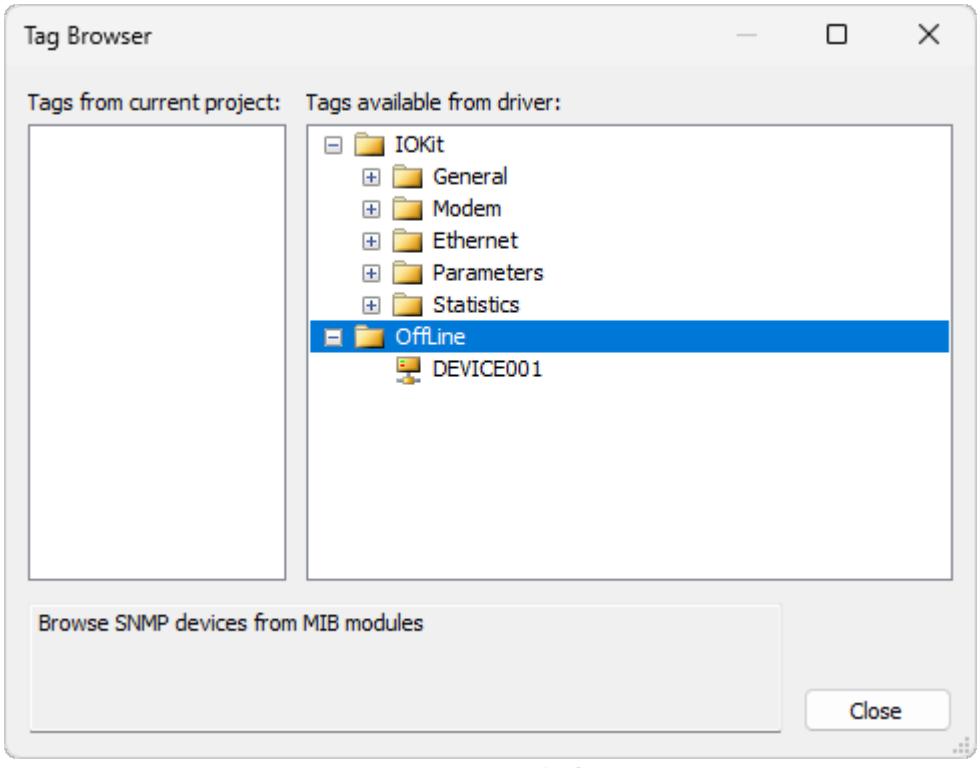
The available options on this tab are described on the next table.

Available options on the SNMP General tab

OPTION	DESCRIPTION
General Timeout	Time-out, in milliseconds, to answer a reading (*SNMP Get*) or writing (*SNMP Set*). A byte-per-byte time-out is defined on the **Setup** tab of **IOKit** library. Both time-outs must be configured, observing that the value of this option must always be higher than **IOKit** library's time-out. This option can be changed at run time by using a configuration Tag called **SNMP.Timeout**. For more information, please check topic **Documentation of I/O Interfaces**
Listen for SNMP Traps	Instructs this Driver to create a specific process to handle unsolicited messages (*SNMP Trap*) sent during the occurrence of specific events that can be configured in some Agents. Information retrieved on trap messages are sent to Tags with the same **Device** and **Item** described in that message. This option can be changed at run time by using a configuration Tag called **SNMP.UseTraps**. For more information, please check topic **Documentation of I/O Interfaces**
Trap Port	Indicates the local UDP/IP port in which trap messages are received. The default UDP/IP port is 162. This option can be changed at run time by using a configuration Tag called **SNMP.TrapPort**. For more information, please check topic **Documentation of I/O Interfaces**

Tag Browser Window

Click **Tag Browser** in **Elipse E3**, **Elipse Power**, or **Elipse Water** to open the window on the next figure, which allows expanding, listing, and dragging to this Driver all Tags identified in an Agent. Select, in the **MIB Parser** option of each device, the interpreter used, **smidump** or **snmptranslate**.



Tag Browser window

This window contains an **IOKit** folder, which contains default Tags from **IOKit** library that allow reading and writing general or status parameters, for example, and an **Offline** folder, which contains Tags from Agents, according to the modules imported from files in **MIB** format.

To use these Tags in an application, drag a Tag or folder from the **Tags available from driver** list to the **Tags from current project** list.

Tag Reference

The next parameters are used for Tags:

- **Device**: Name of an Agent registered on the **SNMP Device Config** tab
- **Item**: Identifier of an object or OID. This parameter can also have the next formats:
  - "ReadCommunity:OID": Information about a specific community for reading
  - "TableOID[Row]": When using a block pointing to an OID identifier of a table, users can specify with brackets a specific row from this table to return when reading, discarding other rows found
  - "WalkTable:TableOID": Instruction to retrieve all cells from an SNMP table to a single Tag
  - EnterpriseOID.<suffix> = When used with Trap messages, there are four possible suffixes to extract specific metadata from the received Trap:
    - EnterpriseOID.Source: Returns the IP address of the agent that sent the Trap
    - EnterpriseOID.Generic: Returns the Generic-Trap code (generic SNMP v1 trap type: 0–6)
    - EnterpriseOID.Specific: Returns the Specific-Trap code (vendor-defined specific trap type)
    - EnterpriseOID.Timestamp: Returns the timestamp (sysUpTime) in seconds at the moment the Trap was generated
- **N1**: If this parameter is different from 0 (zero), this indicates that a Tag is only updated using a trap message, it does not perform a communication by polling using the scan rate defined in this Tag
- **N2**: Allows indicating one of the communities declared in the **Read Community** option of a device, using an index starting from 0 (zero) for reading. For example, when declaring in a device the communities "A;B;C" and configuring in a Tag the *N2* parameter equal to 2 (two), then the "C" community is used in this Tag. This parameter is only used for reading, and the community for writing is defined in the **Write Community** option

Available options for Tag parameters

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
Name of an Agent, such as "router-001"	OID, such as 1.3.6.1.2.1.37.1.2.1.1.1.0	Reading and writing	Any Tag read according to the configured scan. If the *N1* parameter is different from 0 (zero), this Tag is only updated when receiving a trap message with the indicated variable. The OID identifier ending with ".0" represents a scalar variable. The OID identifier specified with an index between brackets corresponds to a column of a table, with the index between brackets

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
			representing a row. In this case, a search is performed in all OIDs lexicographically found on that table's column, returning the result the same way of the **WalkTable** option described next
Same as previous	Community:OID	Read-only	Indicate the name of a community to be used in this reading, such as "public:1.3.6.1.2.1.37.1.1.0"
Same as previous	TableOID	Read-only	The Tag Browser window offers **Table**-type objects from the SNMP protocol as Block Tags, in which each Element corresponds to a column from a table. The **Item** parameter is configured with the OID identifier of that table. When performing a reading of a Block Tag, this Driver searches and returns all rows found on that table. Users can handle the content of each row, with the value of each one of the respective columns, using this Block Tag's **OnRead** event. Therefore, if this table contains 5 (five) rows, this Block Tag's **OnRead** event is triggered 5 (five) times
Same as previous	TableOID[Row]	Read-only	Similar to the previous item, when adding to the **Item** parameter of a Block Tag linked to a table an index between brackets, this reading returns only the selected row, if it exists on that table
Same as previous	WalkTable:OID	Read-only	This instruction indicates that this Driver must scan all OID identifiers lexicographically found starting with the initial OID identifier informed in the **Item** parameter, by using SNMP's **GetNextRequest** command. The result is returned in a Tag as a **String** composed by a sequence of "OID = value" combinations

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
			separated by semicolons, such as "1.3.6.1.2.1.37.1.2.1.33.1 = 1260; 1.3.6.1.2.1.37.1.2.1.33.2 = 'Fast Ethernet Switch'"
Same as previous	EnterpriseOID.Source	Read-only	Exclusive use with Traps (N1 ≠ 0) Returns the IP address of the agent that sent the Trap message. The EnterpriseOID refers to the organization/manufacturer identifier configured on the device. This suffix extracts metadata from the received Trap, not the variable value itself.
Same as previous	EnterpriseOID.Generic	Read-only	Exclusive use with Traps (N1 ≠ 0) Returns the SNMP v1 Generic-Trap code. Possible values: 0 = coldStart 1 = warmStart 2 = linkDown 3 = linkUp 4 = authenticationFailure 5 = egpNeighborLoss 6 = enterpriseSpecific
Same as previous	EnterpriseOID.Specific	Read-only	Exclusive use with Traps (N1 ≠ 0) Returns the manufacturer-defined Specific-Trap code. This value identifies equipment-specific event types when Generic-Trap = 6 (enterpriseSpecific).
Same as previous	EnterpriseOID.Timestamp	Read-only	Exclusive use with Traps (N1 ≠ 0) Returns the timestamp (sysUpTime) in seconds from the moment the Trap event was generated by the agent. It represents the elapsed time since the last device reboot.
Same as previous	ServerStatus	Read-only	Returns the internal status of the connection with an Agent. Possible values are **0**:

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
			No communication or **1**: Communicating normally
Any name, which cannot be an empty String	UpdateConfig	Write-only	Recreates all connections with each device, using the current definition of parameters, which can be changed by writing an array of parameters and values. For more information, please check topic **Updating Parameters At Run Time**

Updating Parameters At Run Time

To add, remove, or change any parameter at run time from a device or Agent, execute the next procedures:

1. Create a Tag named "UpdateConfig" and inform in its **ParamDevice** property the name of any device, such as "Test", because this value cannot be an empty **String**. Configure the **ParamItem** property with the value "UpdateConfig".

2. Create, in an application in **Elipse E3**, **Elipse Power**, or **Elipse Water**, a script to write the parameters to update using the **SetConfigurationParameters** Tag from **IOKit** library. For more information, please check topic **Documentation of I/O Interfaces**.

3. The names of parameters used by this Driver are described on the next table and they store the result of the configuration performed on the **SNMP Device Config** tab of the configuration window.

Parameters used by this Driver

PARAMETER	DESCRIPTION
SNMP.DeviceCount	Indicates the number of devices on a list, in which each device is addressed by an index starting at 0 (zero). This index is needed for all other properties
SNMP.Device[Index].Name	Name of a device according to the index, such as SNMP.Device[0].Name = "MainSwitch"
SNMP.Device[Index].IP	IP address
SNMP.Device[Index].Port	UDP/IP port
SNMP.Device[Index].Community	Reading community
SNMP.Device[Index].WCommunity	Writing community
SNMP.Device[Index].Version	Version of SNMP protocol
SNMP.Device[Index].MaxReq	Maximum number of simultaneous requests
SNMP.Device[Index].Modules	List of MIB modules
SNMP.Device[Index].Disable	Possible values are **1**: Disables communication with this device or **0**: Enables communication with this device

In the next example script, communication is enabled with a device with index 1 (one) and configured the IP address of this device. After sending these parameters by using a writing to the **SetConfigurationParameters** Tag, a communication restart is requested by writing to the **UpdateConfig** Tag.

```
Dim CommArr(1)
CommArr(0) = Array("SNMP.Device[1].Disable",0)
CommArr(1) = Array("SNMP.Device[1].IP","192.168.100.2")
Item("DriverSNMP").Write -1,0,0,3,CommArr
Item("DriverSNMP").Item("UpdateConfig").WriteEx(1)
```

Additional Information about the SNMP Protocol

SNMP is a protocol for typical management of TCP/IP networks, from the application layer, which makes it easy to exchange information among network devices such as routers, switches, and hosts. This protocol allows network administrators to manage network connectivity and performance, find and solve any network problem, and provide information for planning network expansion, among other possibilities.

The network management application does not follow a conventional Client-Server model, because for some operations the management station behaves as a client while the network device to be analyzed or monitored behaves as a server, while in a Trap operation the opposite occurs, because when sending alarms the managed device starts the communication. Because of that, network management systems avoid using the terms client and server, opting for **Manager** for the application executing in the management station and **Agent** for the application executing in the network device.

NTCIP Profile

When a device supports the NTCIP (*National Transportation Communications for Intelligent Transportation System*) protocol, some special functions are available. This Driver currently supports DMS (*Dynamic Message Sign*) functions only.

When configuring a device, if users inform in the **Extra Modules** option the name of **MIB** modules starting with the word "NTCIP", then this is an **NTCIP**-type device.

The special Tags from NTCIP protocol are available on the Tag Browser window, which displays both normal Tags from SNMP protocol and special Tags on the **NTCIP** folder, according to the next figure.

NTCIP Tags

The next table contains the description of Tags supported by this Driver for the **DMS** profile of NTCIP protocol.

Supported Tags for the DMS profile of NTCIP protocol

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
Name of an Agent, such as "device-001"	NTCIP.DMS.GetCurrentMessage	Read-only	Retrieves the current message displayed on a panel. This Block Tag contains 6 (six) Elements. These Elements are **Element 1**: Text displayed in **MULTI String** format, **Element 2**: Remaining time, in minutes (the value 65535 indicates that there is no time limit), **Element 3**: Priority, between 1 (one) and 255, **Element 4**: Status. Possible values are **1**: Not used, **2**: Modifying, **3**: Validating, **4**: Valid, **5**: Error,

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
			6: ModifyReq, **7**: ValidateReq, or **8**: NotUsedReq, **Element 5**: Name of the user who added a message, and **Element 6**: Source of a message. Possible values are **1**: SourceModeOther, **2**: SourceModeLocal, **3**: SourceModeExternal, **8**: SourceModeCentral, **9**: SourceModeTimeBasedScheduler, **10**: SourceModePowerRecovery, **11**: SourceModeReset, **12**: SourceModeCommLoss, **13**: SourceModePowerLoss, or **14**: SourceModeEndDuration
Same as previous	NTCIP.DMS.GetMessage	Read-only	Retrieves any message, in which the *B1* parameter is the type of memory and the *B2* parameter is an index, between 1 (one) and 255. The available types of memory are **2**: MemoryTypePermanent, **3**: MemoryTypeChangeable, **4**: MemoryTypeVolatile, **5**: emoryTypeCurrentBuffer, **6**: MemoryTypeSchedule, or **7**: MemoryTypeBlank. This Block Tag contains 4 (four) Elements. These Elements are **Element 1**: Text displayed in **MULTI String** format, **Element 2**: Priority, between 1 (one) and 255, **Element 3**: Status. Possible values are **1**: Not Used, **2**: Modifying, **3**: Validating, **4**: Valid, **5**: Error, **6**: ModifyReq, **7**: ValidateReq, or **8**: NotUsedReq, and **Element 4**: Name of the user who added a message
Same as previous	NTCIP.DMS.DefineMessage	Write-only	Defines a message that later can be activated. This Block Tag contains 7 (seven) Elements. These Elements are **Element 1**: Type of memory. Possible values are **2**: MemoryTypePermanent, **3**: MemoryTypeChangeable, or **4**: MemoryTypeVolatile, **Element 2**: Index of a

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
			message, between 1 (one) and 255, **Element 3**: Text displayed in **MULTI String** format, **Element 4**: Priority, between 1 (one) amd 255, **Element 5**: Name of the user who added a message, **Element 6**: Indicates whether beacons are triggered or not. Possible values are 0 (zero, does not trigger) or 1 (one, triggers), **Element 7**: Indicates whether the pixel service is triggered or not. Possible values are 0 (zero, does not trigger) or 1 (one, triggers)
Same as previous	NTCIP.DMS.ActivateMessage	Write-only	Activates an already existing message. This Block Tag contains 4 (four) Elements. These Elements are **Element 1**: Type of memory. Possible values are **2**: MemoryTypePermanent, **3**: MemoryTypeChangeable, or **4**: MemoryTypeVolatile, **Element 2**: Index of a message, between 1 (one) and 255, **Element 3**: Priority, between 1 (one) and 255, and **Element 4**: Duration, in minutes (use the value 65535 for a permanent message)
Same as previous	NTCIP.DMS.LongPowerMessage	Reading or writing	Allows reading or defining a message activated after a power failure. The *N1* and *N2* parameters are used only when writing, in which the *N1* parameter is the type of memory and the *N2* parameter is an index between 1 (one) and 255. Possible values for memory types are **2**: MemoryTypePermanent, **3**: MemoryTypeChangeable, or **4**: MemoryTypeVolatile. When writing, this Tag accepts any value to start a writing operation. When reading, this Tag returns a value in the format **MMXXXXCCCC**, in which **MM** is a type of memory with 8 (eight) bits, **XXXX** is

PARAMDEVICE	PARAMITEM	OPERATION	DESCRIPTION
			the memory index with 16 bits, and **CCCC** is a CRC (*Cyclic Redundancy Code*) code of a message
Same as previous	NTCIP.DMS.CommLostMessage	Reading or writing	Allows reading or defining which message is activated after a communication failure. The *N1* and *N2* parameters are used only when writing, in which the *N1* parameter is a type of memory and the *N2* parameter is an index between 1 (one) and 255. Possible values for memory types are **2**: MemoryTypePermanent, **3**: MemoryTypeChangeable, or **4**: MemoryTypeVolatile. When writing, this Tag accepts any value to start a writing operation. When reading, this Tag returns a value in the format **MMXXXXCCCC**, in which **MM** is a type of memory with 8 (eight) bits, **XXXX** is the memory index with 16 bits, and **CCCC** is a CRC (*Cyclic Redundancy Code*) code of a message

MULTI String Format

The text to display on a DMS panel obeys the **MULTI String** (*Markup Language for Transportation Information*) format. This language is similar to the HTML language, in which a text is transmitted and Tags define how this text is displayed. Tags are placed between delimiters, contain an identifier with 1 (one) or more characters, and any optional parameters for this Tag.

Each Tag in **MULTI** format starts with a left bracket ([) and ends with a right bracket (]). The identifier of a Tag appears after the left bracket and contains 1 (one) or more character, case-insensitive. If a Tag contains parameters, then these parameters immediately follow after the identifier of this Tag and are also case-insensitive, except when specified. No spaces or other separator characters can be inserted between an identifier and the parameters. Some Tags can operate in pairs and the default notation of a Tag is defined in the opening Tag. An opening Tag defines where this Tag's functionality starts. A closing Tag defines where this Tag's functionality ends, and it is defined as an opening Tag with a left bar preceding the identifier, that is, if a flash opening Tag is equal to **[fl]**, then the flash closing Tags is equal to **[/fl]**. The next code contains an example of a message in **MULTI** format:

```
[pt50o0][fo6][jl3]This[nl20][jl3]is[nl20][jl3]a Test
```

Available Tags in MULTI String format

OPENING TAG	CLOSING TAG	DESCRIPTION
cbx		Background color of a message

OPENING TAG	CLOSING TAG	DESCRIPTION
pbz or pbr,g,b		Background color of a page
cfx or cfr,g,b		Text color of a message
crx,y,w,h,r,g, b or crx,y,w,h,z		Color of a rectangular are on the current page of a message
fx,y		Field incorporated into a message based on data from a device, such as a clock, a temperature sensor, or a detector, among others
fltxoy or floytx	**/fl**	Activates the blinking of a text, defining the number of times the text blinks and the blinking order (turn on and off or turn off and on)
fox or fox,cccc		Allows selecting the code of a font to view a message. The optional *cccc* parameter indicates the version of that font
gn or gn,x,y or gn,x,y,cccc		Selects an image to insert in a message. An image is handled as a single printable character. It may require some pixels or a complete sign to be displayed. The optional *cccc* parameter indicates the identifier of a message
hcx		Hexadecimal value of a character to display
jlx		Specifies the alignment of a text. Possible values are **left**: Alignment to the left, **center**: Centered text, **right**: Alignment to the right, or **full**: Justified text
jpx		Specifies the alignment of a page. Possible values are **top**: Alignment by the top, **middle**: Alignment by the center, or **bottom**: Alignment by the base
msx,y	**/msx,y**	Specifies a specific Tag from a manufacturer
mvtdw,s,r,text		Specifies the parameters for a text that can be scrolled
nlx		Specifies the start of a new line
np		Specifies the start of a new page
ptxoy		Specifies the time values of a page
scx /sc		Specifies the spacing between characters
trx,y,w,h		Specifies the position of a text on a display

Documentation of I/O Interfaces

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

Configuration of a Driver

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

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

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

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

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

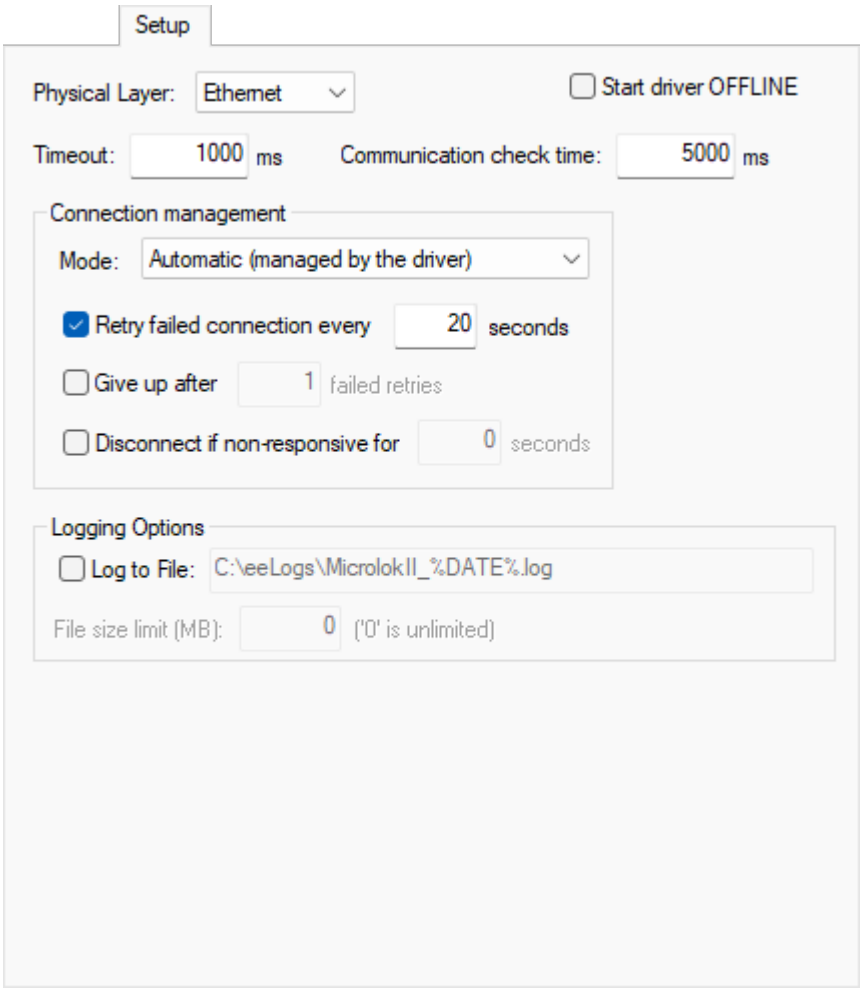
Configuration Dialog Box

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

Setup Tab

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

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



Setup tab

General options on the Setup tab

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

Options on the Connection management group

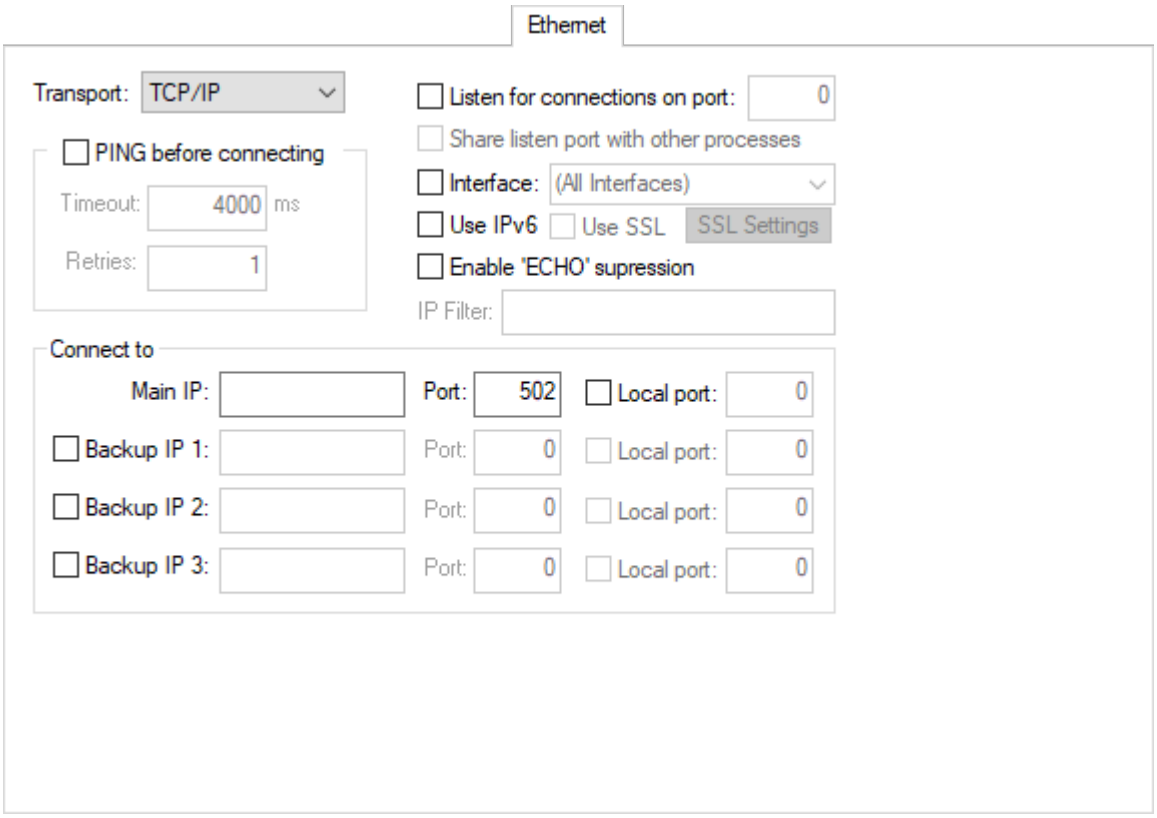
OPTION	DESCRIPTION
Mode	Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection
Retry failed connection every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

Options on the Logging Options group

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

Ethernet Tab

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



Ethernet tab

Available options on the Ethernet tab

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

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

Available options on the Connect to group

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

General Configurations

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

I/O Tags

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

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

IO.CommunicationStatus

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

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

IO.IOKitEvent

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

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

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Ethernet Interface Configuration

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

I/O Tags

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

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

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

IO.Ethernet.IPSelect

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

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

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

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

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

IO.Ethernet.IPSwitch

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

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

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

IO.Ethernet.SocketState

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

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

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

Properties

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

NOTE

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

IO.Ethernet.AcceptConnection

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

IO.Ethernet.BackupEnable[2,3]

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

IO.Ethernet.BackupIP[2,3]

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

IO.Ethernet.BackupLocalPort[2,3]

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

IO.Ethernet.BackupLocalPortEnable[2,3]

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

IO.Ethernet.BackupPort[2,3]

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

IO.Ethernet.IPFilter

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

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

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

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

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

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

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

IO.Ethernet.ListenIP

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

IO.Ethernet.ListenPort

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

IO.Ethernet.MainIP

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

IO.Ethernet.MainLocalPort

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

IO.Ethernet.MainLocalPortEnable

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

IO.Ethernet.MainPort

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

IO.Ethernet.PingEnable

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

IO.Ethernet.PingTimeoutMs

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

IO.Ethernet.PingTries

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

IO.Ethernet.ShareListenPort

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

IO.Ethernet.SupressEcho

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

IO.Ethernet.Transport

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

IO.Ethernet.UseIPv6

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
2.0.28	01/28/2026	A. Fetzner M. Ludwig	• Support for reading SNMP v1 Trap metadata has been implemented (Case 39340). • Fixed a buffer overflow when decoding an *Object ID* to a **String** (*Case 39319*). • Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38020*).
2.0.27	03/10/2025	M. Salvador	• Fixed writings or comands (*Case 37437*).
2.0.26	03/21/2024	M. Salvador	• Fixed the reading of tables (*Case 35438*).
2.0.24	06/26/2023	M. Salvador	• Implemented a second MIB Parser option per device. Now users can select between **smidump** and **snmptranslate** (*Case 33723*).
2.0.23	06/26/2023	M. Salvador	• Implemented the NTCIP DMS Profile (*Dynamic Message Sign*), used in roads or speedways (*Case 33727*).
2.0.19	11/11/2021	M. Salvador	• Added support to SNMP protocol version **3** (*Case*

VERSION	DATE	AUTHOR	COMMENTS
			18316).
2.0.16	08/05/2019	M. Ludwig	• Driver ported to Visual Studio 2017 (*Case 27093*).
2.0.15	04/03/2019	M. Ludwig F. Englert M. Salvador	• Improved error diagnostic when loading MIB files (*Case 23007*). • Implemented several improvements when importing Tags using MIB files (*Case 21919*). • Browsing Tags of user-selected MIB files does not show empty folders anymore (*Case 22832*). • Fixed a situation that prevented the configuration of this Driver at run time by using the **UpdateConfig** Tag (*Case 20336*).
2.0.4	12/22/2015	M. Salvador	• Added support for the new implementation of UDP Sockets from **IOKit** library (*Case 18185*). • Driver ported to **IOKit** library version **2.0** (*Case 16423*).
1.1.1	01/28/2014	M. Salvador	• Added a limit of 50 devices per Driver.
1.0.1	01/20/2010	M. Salvador M. Bihre	• Initial version of this Driver and its documentation.

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)