# Elipse SuperDriver Driver

| | |
|---|---|
| **File Name** | SuperDriver.dll |
| **Manufacturer** | Elipse Software |
| **Devices** | Any device communicating with Elipse Software Drivers developed using IOKit library and communicating via Ethernet TCP/IP |
| **Protocol** | Any protocol over TCP/IP implemented on Elipse Software Drivers developed using IOKit library version 2.0 and communicating via Ethernet TCP/IP |
| **Version** | 2.1.10 |
| **Last Update** | 08/06/2026 |
| **Platform** | Win32 |
| **Dependencies** | N/A |
| **Superblock Readings** | No |
| **Level** | 31302 |

## Introduction

The primary goal of Elipse SuperDriver Driver is to communicate via other **Elipse Software** Drivers, each one of these Drivers might be specialized in some protocol. Multiple Drivers are associated to Elipse SuperDriver Driver, generating items called **Slaves**, whose configuration definitions are isolated by **Driver Profiles**. Therefore, the existing I/O Tags in an application, referring to these **Slaves** by some configuration field, have their parameters sent to their respective instance of a Driver.

**Elipse Software** Drivers that can be subordinated to Elipse SuperDriver Driver must be the ones developed for the **IOKit** library, and their topology must provide as an immediate physical layer for communication with the application the **Ethernet** interface, and **TCP/IP** as its network transport protocol.

Elipse SuperDriver Driver is not compatible with **Elipse SCADA**.

# SuperDriver Functionality

For a proper usage of Elipse SuperDriver Driver in an application, it is really important to understand how this Driver works.

# Definitions

First, users must be acquainted with some concepts used later in this document, for a good understanding of the operation of Elipse SuperDriver Driver.

## Driver Profile

A **Driver Profile** is an object from a data structure that contains a name and a path to a dynamic link library file (DLL) from an **Elipse Software** Driver, in addition to a configuration declaration for that Driver. Several Driver Profiles can share the same Driver file, each one with a different configuration and name.

On the **SuperDriver** tab there is a list of Profile declarations, which is explained on a later chapter. For a proper usage of Elipse SuperDriver Driver, a list of Driver Profiles must be configured with at least 1 (one) item.

## Status of an In Advise Tag

The **In Advise** status of a Tag is defined by keeping it in constant update, at time intervals defined by its **Scan** property, or scan periods.

The way to change a Tag status to **In Advise** is by configuring its **AllowRead** property to True and by one of the following conditions:

- A Tag with the value of the **AdviseType** property equal to 0 (zero), that is, equal to **AlwaysInAdvise**
- A Tag with the value of the **AdviseType** property equal to 1 (one), that is, equal to **AdviseWhenLinked**, and linked to an active object on an open Screen

If one of these conditions is not available, an I/O Tag does not have an **In Advise** status, that is, it is not read or written.

For a writing service via Elipse SuperDriver Driver, it is not enough to enable the **AllowWrite** property of a Tag, users must also have at least one I/O Tag with an **In Advise** status, according to the previous definition, active in their respective Driver Profile.

Only in this condition that a writing Tag can be added to the cache list of a **Slave** for further processing within the scan periods.

Therefore, users cannot perform a Tag writing directly or individually. A writing is always processed via cache list of a **Slave**, together with the other **In Advise** Tags.

## Slave

A **Slave** is an object dynamically created by Elipse SuperDriver Driver, which carries an independent Tag cache (repository) and memory resources, and it is related to an I/O Driver Profile. **Slaves** are created by Elipse SuperDriver Driver by supplying pairs of Profiles and Ports by the declared Tags.

A pair of **Profile** and **Port** is supplied by the content of the **Device** field. The **Device** field of each Tag must contain the name of one of those declared Profiles and a TCP/IP port number, separated by a colon character. This syntax is presented later.

There is a **Slave** for each one of these different pairs declared in a set of Tags linked to a Elipse SuperDriver Driver object in an application. Ten hundred **Slaves** is the maximum value supported by Elipse SuperDriver Driver.

# Operation Modes

An Elipse SuperDriver Driver can operate in **General Scan** or **Individual Scan** modes. The **General Scan** mode is always executed, for all **Slaves** with **In Advise** Tags, and the **Individual Scan** mode of a **Slave** is only executed if it is enabled.

## General Scan

Elipse SuperDriver Driver uses the **General Scan** mode, the existing general scan time on its configuration tab, to perform a reading and writing of all **In Advise** Tags from all **Slaves**. Each time the **General Scan** mode of a **Slave** expires, that **Slave** starts reading and writing all its Tags, obeying the following connection strategy:

1. The **Slave** starts its respective Driver object, without enabling an automatic connection to a backup IP address.
2. The **Slave** tries to process all its Tags. As Tags are successfully processed, they are removed from the cache list.
3. At the end of a service iteration, the Driver object is then stopped and the resources is released, that is, the TCP/IP port, if there is another **Slave** requesting that resource.
4. If there are Tags on the cache list at the end of the first service iteration, reading or writing is retried up to *n* times, according to the number of retries configured on the properties tab of Elipse SuperDriver Driver.

5. If there are still Tags on the cache list at the end of all retries, and the number of I/O Tags remains equal to the number initially requested, Elipse SuperDriver Driver takes the backup IP address as its main IP address and retries again the cycle of services using that IP address. This is obviously linked to what was configured on the properties tab. If users do not want a backup IP address, there is no such retry.

6. If the number of I/O Tags is less than the originally requested, that is, if at least one Tag is successful, this processing ends without trying the backup IP address.

**NOTE**

If a destination IP address refuses a connection, both for its main IP address as for its backup IP address, a device is considered as offline, and therefore the programmed retries are not performed.

## Individual Scan

As there is a **General Scan** mode, all **Slave** Tags have their individual scans respected if there is an **In Advise** of an **Individual Scan** Tag for its respective Profile. Readings and writings performed by these scan times do not influence reading and writing scan times of the **General Scan** mode, that is, the service using an **Individual Scan** does not restart the remaining time for the **General Scan** service. This individual scan without influencing the **General Scan** mode is the desired behavior for readings and writings provoked during the interval among general scans, which would de-sync the expiration of all Tags at the same time. The **General Scan** mode handles, therefore, updates of all **In Advise** Tags of all Profiles on events among equal intervals for all. If there is one or more Tags with an expired individual scan, these ones are grouped and added to a cache list, obeying the following connection strategy:

1. The **Slave** starts its respective Driver object, by enabling an automatic connection to a backup IP address.

2. The **IOKit** module of the respective Driver object of the **Slave** is automatically in charge of trying a main or a backup IP address, according to the configuration for automatic connection management.

3. The **Slave** performs the processing of all Tags on the cache list. As Tags are successfully processed, they are removed from that cache list.

4. This list of Tags in **Individual Scan** mode also retries *n* times, as configured on the properties tab, while there are Tags on that cache list. For each retry, the Profile releases the resource if there is other concurrent Profile to it.

5. The processing of individual Tags ends with the success of the whole group or by reaching the end of retries.

**NOTE**

If a destination IP address refuses a connection, both for its main IP address as for its backup IP address, a device is considered as offline, and therefore the programmed retries are not performed.

# Execution

When executed, Elipse SuperDriver Driver immediately creates its **Slave** objects. When a **Slave** is created by Elipse SuperDriver Driver, this **Slave** remains in a standby status until a Tag enters the **In Advise** mode and is placed on its cache. Elipse SuperDriver Driver then performs a management of all declared Tags subordinated to it and that are **In Advise**. The management algorithm of Tags of Elipse SuperDriver Driver separates and adds Tags to caches of their respective **Slaves**, as these Tags enter the **In Advise** mode. Exiting an **In Advise** status leads to removal of a Tag from the cache. Therefore, having an **In Advise** status is a condition for a Tag to enter the reading and writing cache of a **Slave**.

As soon as there is one or more Tags on the cache of a **Slave**, this **Slave** immediately switches to a reading or writing status and processes a list of Tags that must be read or written. This list of Tags is processed according to the next steps:

1. If there are I/O Tags:

1. The Driver of a **Slave** object starts and connects to a remote device.
2. The Driver processes its I/O Tags, one by one.
3. The Driver processes special Tags, one by one, which are stored in the memory of the **Slave**.
4. The Driver then disconnects and stops.

2. If there are special Tags, these Tags are updated with the content stored in the memory of the **Slave**.
3. If there are service retries, returns to step 1 (one).

Tags **In Advise** status are always read or written after each expiration time of the **General Scan** mode, and its scan period is defined by a general value configured on the **SuperDriver** tab, that is, the management of Elipse SuperDriver Driver services does not take into account those scan period values, that is, the **Scan** property of each one of the Tags, unless the individual scan mode is enabled for the respective Profile of these Tags. **In Advise** Tags are only read or written by respecting their individual scans when the **Individual Scan** mode is enabled.

A **Slave** returns to its standby status while waiting until the next Tag expires its scan period. When this situation happens, this **Slave** builds a new list of Tags to be read or written.

This execution process described previously is performed for any **Slave** subordinated to Elipse SuperDriver Driver. It is not allowed that **Slaves** with the same TCP/IP ports have connections performed at the same time. A **Slave** that needs a busy TCP port remains in standby mode until that resource is released.

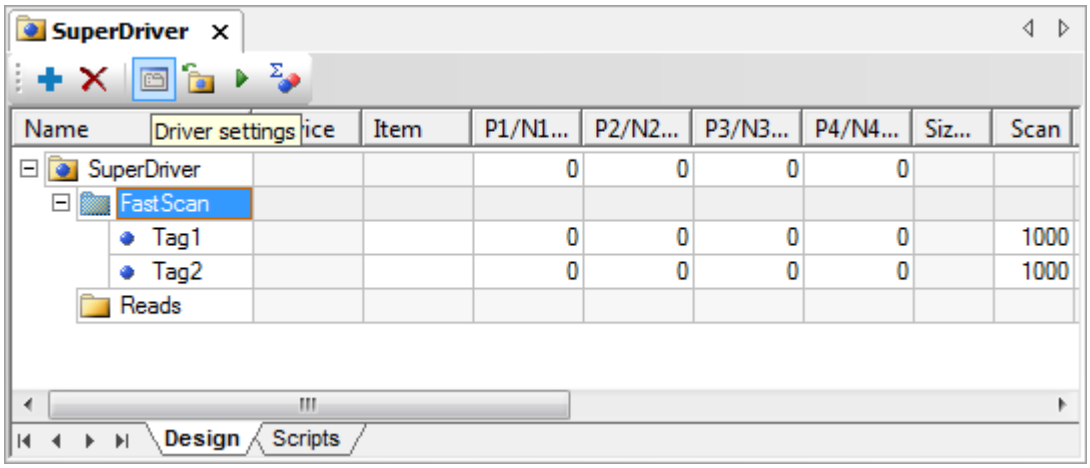
# Concurrent Communication

Elipse SuperDriver Driver, when in execution, manages the expiration time of **In Advise** Tags of all **Slaves**. When detecting an expiration time of one or more Tags, Elipse SuperDriver Driver saves a list of **Slaves** with Tags with expired times to perform readings as soon as possible. This list may have the maximum possible size of **Slaves**. However, I/O tasks of all **Slaves** may not always occur simultaneously.

Every I/O task that occurs on a **Slave** consumes processing resources, or CPU usage, and at least one thread is created for each communication. The operating system limits the maximum number of threads per process. For this reason, Elipse SuperDriver Driver has a maximum limitation of communications occurring at the same time. When this limit is reached, all **Slaves** that need to communicate remain on a waiting list, and start communicating as soon as the current communications end.

# Driver Configuration

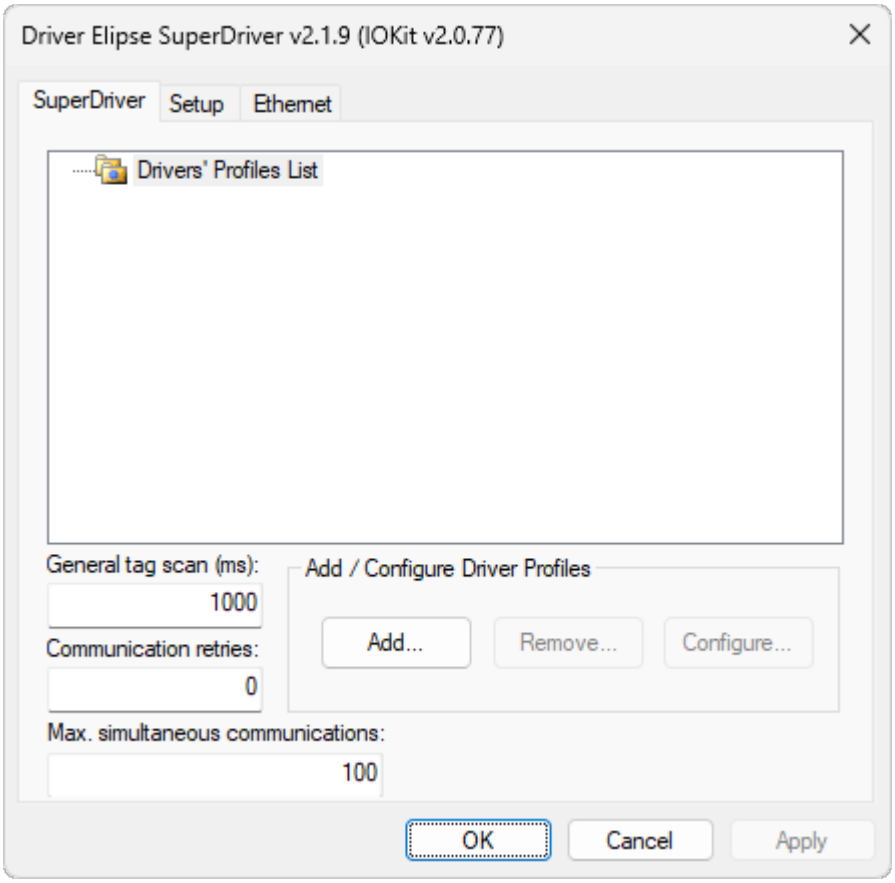
Elipse SuperDriver Driver does not use **[P]** parameters for configuration. All configurations are performed on the properties window, available on the **Design** tab of an I/O Driver object. To do so, click **Driver Settings**  , according to the next figure.




**Driver Settings option**

# SuperDriver Tab

When clicking **Driver Settings**, the window on the next figure is opened and the **SuperDriver** tab is selected.




**SuperDriver tab**

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

**Available options on the SuperDriver tab**

| OPTION | DESCRIÇÃO |
|---|---|
| **Drivers' Profiles List** | A tree structure aiming to display the list of Profiles of Drivers |
| **General tag scan (ms)** | Time, in milliseconds, of a scan period relative to all Tags on the reading list and universal to all **Slave** objects. This means that all **Slaves** obey a cyclic reading of each one of |

| OPTION | DESCRIÇÃO |
|---|---|
| | the Tags by this value. The expiration of the general scan time always generates a reading request to all Tags on the list of each **Slave** |
| **Communication retries** | Configures the number of communication retries. A **Slave** object retries communication while there are Tags with unsuccessful readings. In each operation mode, Elipse SuperDriver Driver works in a different way relative to this value. For more information, please check topic **Operation Modes**. The maximum allowed number of retries is 10 |
| **Max. simultaneous communications** | Configures the maximum limit of communications occurring at the same time. For more information, please check topic **Concurrent Communication**. When reaching this value, all **Slaves** that must communicate remain on a waiting list and start communicating as the current communications are finished. This value must be taken into account by the physical limitations of the computer's processor that executes a system. The minimum configurable value is 1 (one) and the maximum value is 750. The best value to configure varies according to the capacity of the system's processor when performing parallel tasks, therefore the maximum possible value is not always the optimal value for a system, as in a directly proportional relationship. The optimal value can be obtained through testing with concurrent readings of a number of **Slaves** greater than the configured value |
| **Add/Configure Driver Profiles** | Allows adding, removing, and configuring Driver Profiles. For more information, please check topic **Profile Configuration** |

Click **OK** to confirm the listed configurations and close this window.

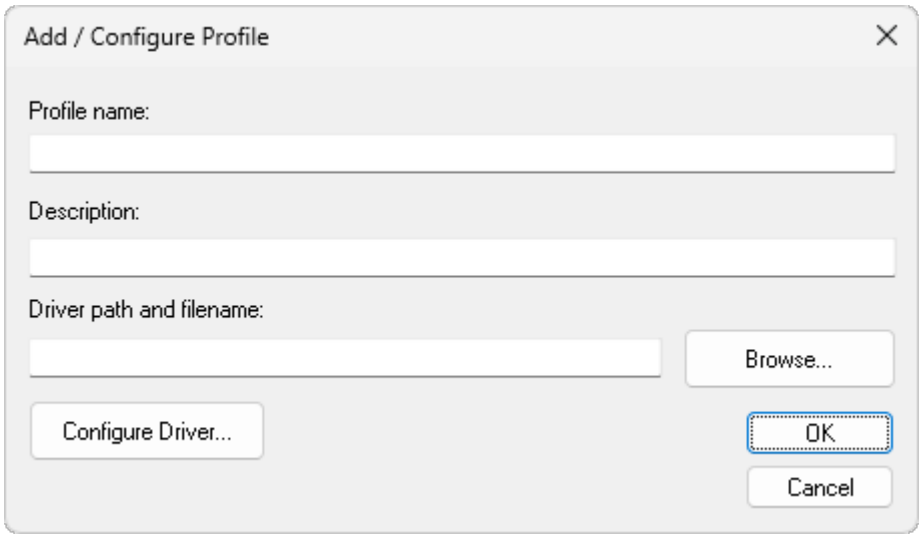
# Setup and Ethernet Tabs

**Setup** and **Ethernet** tabs are inherited from **IOKit** library and are the only ones that must be configured. Please check topic **Documentation of I/O Interfaces** for more information about these tabs.

Several parameters configured on these tabs are passed to **Slave** Drivers. The topic **Profile Parameters Inherited from SuperDriver** specifies each one of these parameters and must be checked for more information about the necessary configurations in each Profile.

# Profile Configuration

Click **Add** or **Configure** on the **SuperDriver** tab to open the window on the next figure and add or configure a Profile.




**Add/Configure Profile window**

The available option on this window are described on the next table.

**Available options on the Add/Configure Profile window**

| OPTION | DESCRIPTION |
|---|---|
| **Profile name** | Configures the name of a Profile. Select a name that represents this Driver Profile when referencing Tags |
| **Description** | Configures the description of a Profiles aiming documentation. Although it does not impact the functionality of a Profile, this is a mandatory option that can be filled with any keyboard-input content |
| **Driver path and filename** | Path and name of a Driver file selected for a Profile. Click **Browse** to open the Windows default file selection window and select an existing file |
| **Configure Driver** | Opens the configuration window of the Driver indicated in the **Driver path and filename** option. As each Driver has their own peculiarities, this configuration window must be filled according to the User's Manual of that Driver. Please check topic **Profile Parameters Inherited from SuperDriver** to avoid unnecessary configurations |

# Profile Parameters Inherited from SuperDriver

The next parameters are copied from the configuration of Elipse SuperDriver Driver to all its subordinated Profiles, therefore they do not need to be filled in all Profiles.

- **Setup** tab:
  - **Physical layer**: Fixed as **Ethernet** and does not expect communication using other interfaces
  - **Connection management mode**: **Automatic** or **Manual**
  - **Retry failed connection every *n* seconds**
  - **Give up after *n* failed retries**
  - **Disconnect if non-responsive for *n* seconds**

- **Activation and Log file name**: Generates a log file for each **Slave**, whose file name is formed by the name configured in Elipse SuperDriver Driver and by a suffix formed by the Profile name and TCP/IP Port

- **Ethernet** tab:
  - **Transport (protocol)**: Fixed as **TCP** and does not expect another transport protocol
  - **Main IP address**
  - **Backup IP address**
  - **Local port specification (for main IP, as well as for backup IP)**
  - **Ping**: Enabling, time-out, and retries

**NOTES**

- The other settings on the **Setup** and **Ethernet** tabs of Elipse SuperDriver Driver are ignored and each Profile apply their own settings.
- The TCP/IP port on the **Ethernet** tab is not considered. This parameter is specified in the Tags.

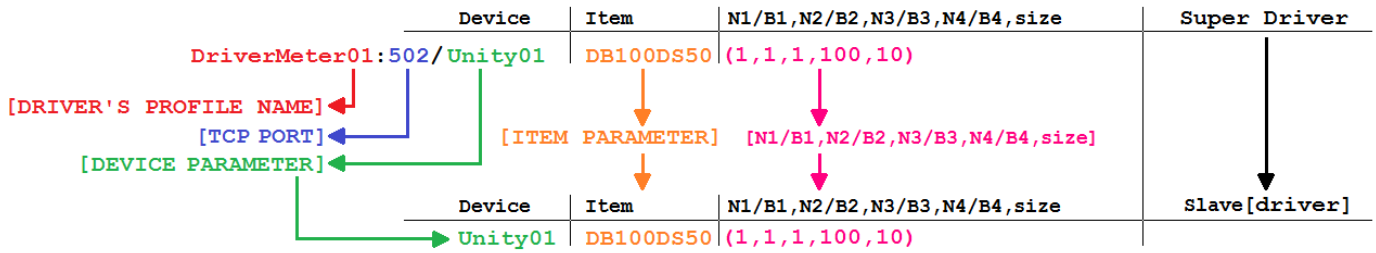
# Tag Reference

This section contains information about the configuration of *Device*, *Item*, and *[N/B]* parameters of each Tag of this Driver. Tags are classified according the categories **I/O Tags** and **Special Tags**.

# I/O Tags

An **I/O Tag** is a Tag executed exclusively by a Driver instance internal to a **Slave**, by passing parameters via that **Slave**. Elipse SuperDriver Driver creates a virtual Tag that is passed for reading or writing by a **Slave**.

The next figure outlines how that transfer of parameters of a Tag declared in Elipse SuperDriver Driver is performed, so that a virtual Tag can be read by a Driver from a **Slave**.




**Transferring parameters from a Tag**

Help documentation for each Driver used as **Slave** must be checked to configure *N/B* parameters, Block sizes, and possibly *Device* and *Item* parameters, if used.

- **Device field**: To specify a **Slave** to which a Tag belongs, use the syntax **[NAME OF A DRIVER PROFILE]:[TCP PORT]**. If the **Slave** Driver uses any *Device* parameter, use the syntax **[NAME OF A DRIVER PROFILE]:[TCP PORT]/[DEVICE PARAMETER]**, according to its help documentation
- **Item field**: If the **Slave** Driver uses any *Item* parameter, fill in this item according to its help documentation
- **N/B and size parameters**: Fill in according to its help documentation

Field values for each Element and for each PLC Tag also have their interpretation described on help documentation.

# Special Tags

A **Special Tag** is a Tag executed by only reading content stored by a **Slave** and that can perform some function. These Tags, when read, do not depend on loading, starting, or connecting to an I/O Driver of a **Slave**, therefore they have a better performance, and shorter scan times relative to I/O Tags may be required. The types of special Tags are the following:

- **IOKit**
- **IndividualScan**
- **CurrentMainIP**
- **ErrorHistory**

## IOKit

To configure a Tag from **IOKit** library, use the syntax **[NAME OF A DRIVER PROFILE]:[TCP PORT]/[IOKIT TAG]** in the **Device** field, according to the next example. The other fields are ignored.

```
Profile02:502/IO.PhysicalLayerStatus [PLC Tag]
Profile02:502/IO.IOKitEvent [Block Tag with four Elements]
```

Please check topic **Documentation of I/O Interfaces** for information about the Tags from **IOKit** library.

## IndividualScan

This Tag, when **In Advise** mode, indicates to a **Slave** subordinated to Elipse SuperDriver Driver that it must perform a Tag reading, respecting its individually configured scan periods.

To configure an **IndividualScan** Tag, use the syntax **[NAME OF A DRIVER PROFILE]:[TCP PORT]/IndividualScan** in the **Device** field. The *[N]* parameters are ignored. The **Value** property of this Tag shows a number value, incremented at each reading performed, according to the next example.

```
Profile01:502/IndividualScan
```

## CurrentMainIP

This Tag indicates the active IP address of a connection. To configure it, use the syntax **[NAME OF A DRIVER PROFILE]:[TCP PORT]/CurrentMainIP** in the **Device** field. The other fields are ignored. The **Value** property of this Tag displays a String with the IP address or alias of the active server, according to the next example.

```
Profile01:502/CurrentMainIP
```

## ErrorHistory

This Tag returns a list of errors occurred during communication with the Driver of a **Slave**. When there are no errors, it returns an empty list. This Tag must be a Block Tag with up to 2 (two) Elements. To configure this Tag, use the syntax **[NAME OF A DRIVER PROFILE]:[TCP PORT]/ErrorHistory** in the **Device** field. The other fields are ignored, according to the next example.

```
Profile01:502/ErrorHistory
```

The **Value** property of Element 1 (one) contains a numerical error code. The interpretation of this code is on the next **table**.

The **Value** property of Element 2 (two) contains a **String** with the meaning of this error. Each Driver has its own way to display the meaning of an error. Therefore, if users want to keep a pattern or even change the language, they must perform a correspondence between the table of error codes and the application.

This Tag can only be used by the following Drivers:

- Enron Modbus version **1.2**
- Modicon Modbus version **2.8**
- Elster-Instromet 999 version **1.0**

**Error codes**

| CODE | CATEGORY | DESCRIPTION |
|---|---|---|
| **1** | TIMEOUT | A Driver did not receive an answer from a device during the configured time-out |
| **2** | INVALID FRAME RECEIVED | A Driver received an invalid frame as an answer, such as a checking error, or an address field different from the request |
| **3** | INVALID ARGUMENT | A user application transferred Tags with invalid parameters to a Driver |
| **4** | CONNECTION ERROR | Error related to the physical layer, such as a failure in TCP/IP connection or an invalid serial port |
| **5** | OUT OF MEMORY | The computer in which an application is executing is out of memory, preventing allocation of resources for the operations of a Driver |
| **6** | INTERNAL ERROR | This error is returned in unexpected cases, that is, situations or contexts detected by a Driver that should not happen and may be considered as failures. This category of error category is a rare one and must be sent to **Elipse Software**'s technical support |
| **7** | PROTOCOL EXCEPTION FRAME RECEIVED | A valid response frame was received from a device, but informing an error condition defined by protocol |

# Statistics and Scan Blocking Tags

During its execution, Elipse SuperDriver Driver accumulates counting for statistics that can be accessed by an application using specific Tags for Collecting Statistics. All these Tags are read-only, except the **$BlockScans** Tag, which blocks all scans of all **Slaves** from Elipse SuperDriver Driver.

For a Tag to be recognized, its name must be in the **Device** field and the **Item** field must be blank.

- **$ScansActive**: Returns the number of scans in execution right now
- **$ScansCompleted**: Returns the number of scans completed by a Driver
- **$IndividualScansActive**: Returns the number of individual scans in execution right now
- **$IndividualScansCompleted**: Returns the number of individual scans completed by a Driver
- **$GeneralScansActive**: Returns the number of general scans in execution right now

- **$GeneralScansCompleted**: Returns the number of general scans completed by a Driver
- **$BlockScans**: Allows blocking a scan execution. Possible values for this Tag are **0**: Blocks all scans or **1**: Releases all scans. This Tag also allows writings

# Configuration Parameters

### Write-Only

| | |
|---|---|
| **B1** | -1 (minus one) |
| **B2** | 0 (zero) |
| **B3** | 0 (zero) |
| **B4** | 3 (three) |

There is a possibility of changing parameters configured on the **SuperDriver** tab at run time, with this Driver in **Offline** mode. All offline configurations use a writing of a Block Tag with 2 (two) Elements. The next topics show all parameters that can be configured this way. For additional information on the usage of this Tag, please check topic **Documentation of I/O Interfaces**.

## GeneralTagScan

Use the next Elements to change the general scan time period of all Tags in **Advise** mode:

- **Element 1**: "SUPERDRIVER.GeneralTagScan"
- **Element 2**: Scan time period, in milliseconds

## CommunicationRetries

Use the next Elements to change the number of communication retries:

- **Element 1**: "SUPERDRIVER.CommunicationRetries"
- **Element 2**: Number of retries. The maximum allowed value is 10

# Documentation of I/O Interfaces

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

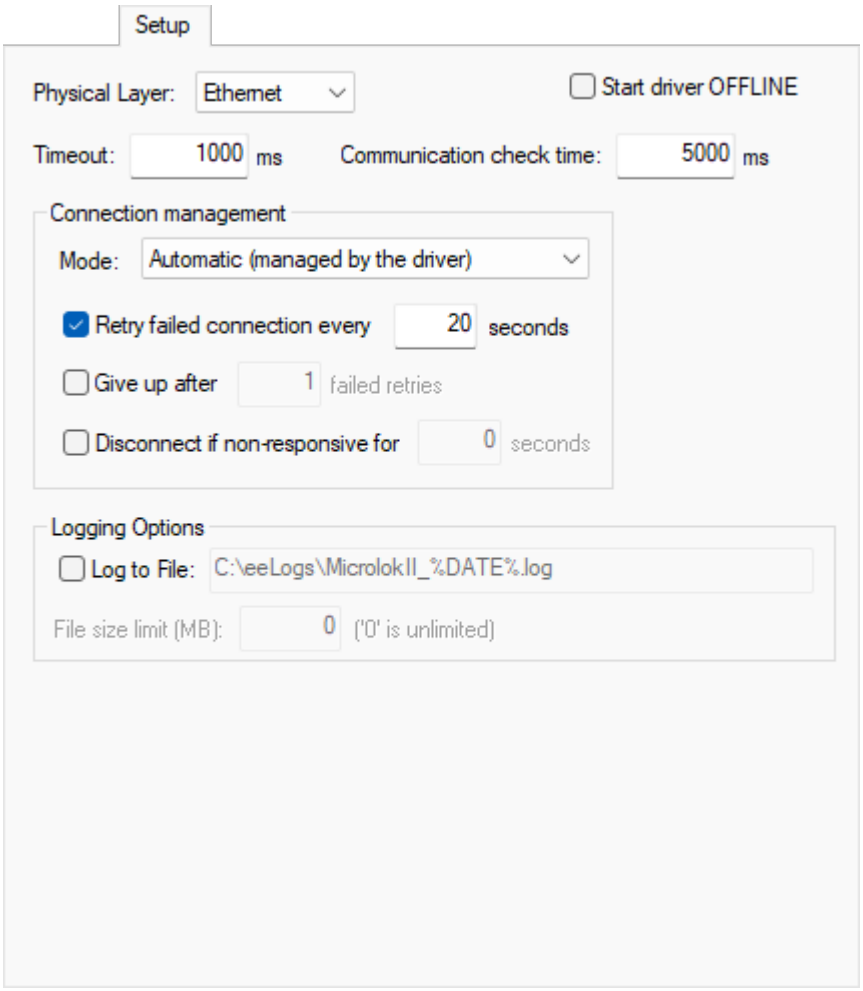
# Configuration Dialog Box

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**

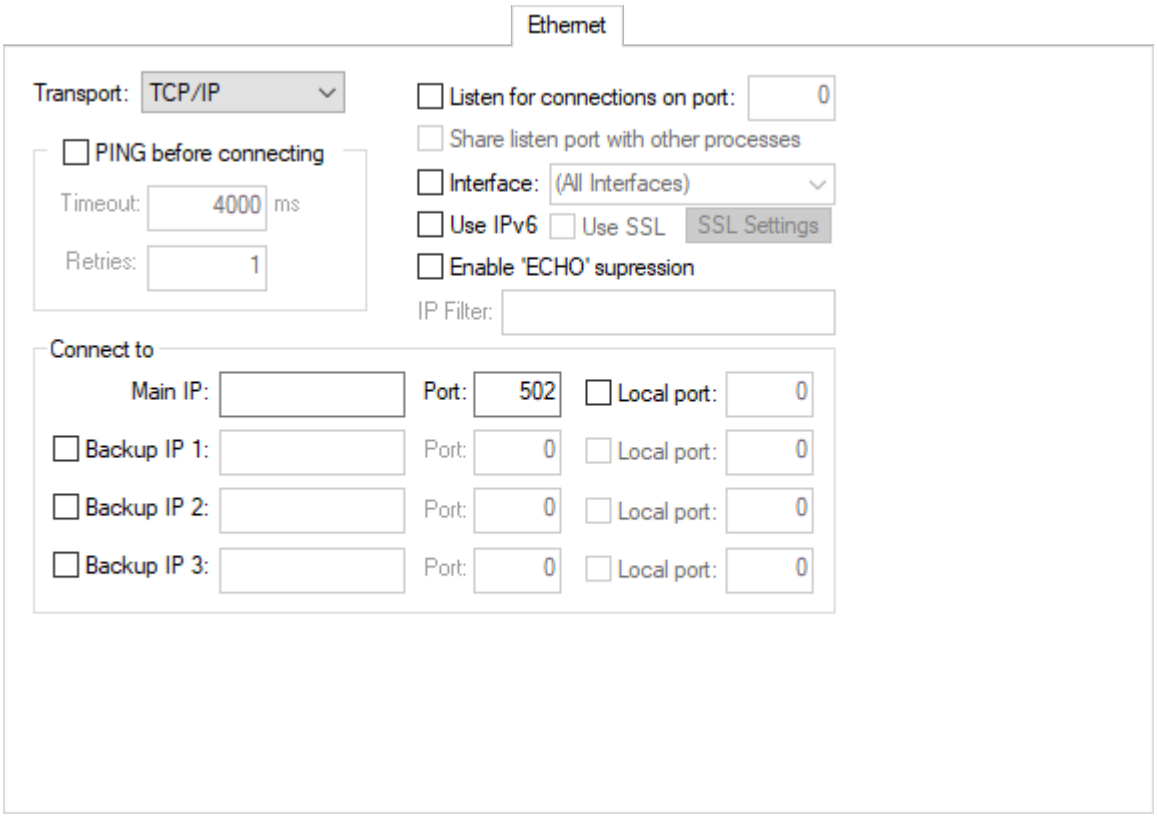
| OPTION | DESCRIPTION |
|---|---|
| **Mode** | Selects a management mode of a connection. Selecting the **Automatic** option allows a Driver to manage the connection automatically, as specified in the next options. Selecting the **Manual** option allows an application to fully manage a connection |
| **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:<br>• **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<br>• **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's Revision History

| VERSION | DATE | AUTHOR | COMMENTS |
|---|---|---|---|
| **2.1.10** | 08/06/2026 | M. Ludwig | • Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38047*). |
| **2.1.7** | 02/13/2017 | M. Ludwig | • Fixed a failure when finishing in stress conditions (*Case 21663*).<br>• Fixed a problem that triggered more than one collecting simultaneously on the same Profile or TCP/IP port, which may cause a GPF (*Case 21751*). |
| **2.1.4** | 10/17/2016 | M. Ludwig | • Fixed a lack of loading configurations in Driver Profiles for **Slaves** (*Case 21664*).<br>• Driver ported to **IOKit** library version **2.0.72** (*Case 21634*). |
| **2.1.2** | 03/17/2015 | M. Ludwig | • Implemented a withdrawal of communication retries if a device is detected as in **Offline** mode (*Case 18356*).<br>• Implemented an overwriting of Ethernet configurations to avoid a triggering of an |

| VERSION | DATE | AUTHOR | COMMENTS |
|---|---|---|---|
| | | | unwanted automatic redundancy in **General Scan** mode (*Case 18389*). |
| **2.1.1** | 11/14/2014 | M. Ludwig | • Implemented a maximum limit of concurrent communications (*Case 15645*).<br>• Changed the limit of **Slaves** to 1000. When reaching this limit, this Driver does not create more **Slaves** (*Case 17564*). |
| **2.0.5** | 03/21/2014 | M. Ludwig | • Added a limitation for **Slave** connections (*Case 15976*). |
| **2.0.4** | 11/21/2013 | F. Englert<br>M. Ludwig | • Fixed the possibility of a failure when closing this Driver (*Case 14855*).<br>• Implemented a second call for writings after executing a reading process (*Case 15176*). |
| **2.0.3** | 09/27/2013 | F. Englert | • Added Tags for collecting statistics (*Case 14954*). |
| **2.0.2** | 09/23/2013 | F. Englert<br>M. Ludwig | • Fixed memory leaks (Windows heaps) during execution (*Case 14918*).<br>• Driver ported to **IOKit** library version **2.0** (*Case 13888*).<br>• Implemented the generation of logs from sub-Drivers in different files (*Case 13890*).<br>• Elipse SuperDriver Driver now only loads Drivers supporting multiple instances on the same process (*Case 13995*). |
| **1.2.2** | 03/05/2013 | M. Ludwig | • Changed the waiting behavior when finishing (*Case 13139*).<br>• Fixed the update of parameters at run time (*Case 13358*).<br>• Fixed a lack of resource releasing when writing I/O Tags (*Case 13484*).<br>• Fixed an indefinite growth of resources usage when the **ErrorHistory** Tag is not read (*Case 13477*). |

| VERSION | DATE | AUTHOR | COMMENTS |
|---|---|---|---|
| | | | • Fixed an incorrect loading of IP addresses configured on the properties window (*Case 13476*). |
| **1.1.1** | 05/31/2012 | C. Mello | • Added a service for writing Tags for Driver Profiles with Tags in **Advise** mode (*Case 12521*). |
| **1.0.1** | 09/22/2011 | M. Ludwig | • Initial version of this Driver (*Case 11581*). |

**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)