Novus FieldLogger Driver

File Name	FieldLogger.dll
Manufacturer	Novus Produtos Eletrônicos
Devices	Field Logger
Protocol	Modbus RTU (class 0)
Version	3.0.5
Last Update	04/02/2026
Platform	Win32
Dependencies	FieldLoggerNG.dll
Superblock Readings	No
Level	31297

Introduction

The Novus FieldLogger Driver must be used to perform communication between **Elipse Software** applications and Field Logger devices by Novus Produtos Eletrônicos.

The I/O interface between this Driver and Novus Produtos Eletrônicos devices is performed using FieldLoggerNG.dll's API, which must be properly installed on the system.

Driver Configuration

Use this Driver's **[P]** parameters to define a COM port for RS-232 Serial communication, in addition to other details of communication, such as stop bits, parity, size of character, and baud rate.

Configuration of Driver's [P] Parameters

The *P1* parameter defines the port used for serial communication RS-232. Possible values are **1**: COM1 (default), **2**: COM2, **3**: COM3, **4**: COM4, or **n**: COM*n*.

The *P2* parameter defines stop bits, parity, size of character, and baud rate, in bps. Possible values are the following:

- **00000**: 1 (one) stop bit
- **10000**: 2 (two) stop bits
- **00000**: No parity
- **01000**: Odd parity
- **02000**: Even parity
- **00000**: Character with 8 (eight) bits
- **00100**: Character with 7 (seven) bits
- **00000**: 110 bps
- **00001**: 300 bps

- **00002**: 600 bps
- **00003**: 1200 bps
- **00004**: 2400 bps
- **00005**: 4800 bps
- **00006**: 9600 bps
- **00007**: 19200 bps
- **00008**: 38400 bps
- **00009**: 56000 bps
- **00010**: 57600 bps
- **00011**: 115200 bps

To configure this parameter, sum the values of stop bits, parity, character size, and baud rate. For example, the value **01006** configures **1 (one) stop bit**, **odd parity**, **character with 8 (eight) bits**, and a baud rate of **9600 bps**.

The *P3* and *P4* parameters are not used and must be left in 0 (zero).

Configuration of Driver's Properties

The **FieldLogger** tab contains specific configurations for this Driver. The available options on this tab are described on the next table.

Available options on the FieldLogger tab

OPTION	DESCRIPTION
Default Data Output Folder	Default path or folder to receive downloads from a device
Default Device Serial Number	Default serial number of the current device
Slave ID or Gateway address	Identifier address of a remote device
Select the Field Logger NG interface	Type of I/O interface. Possible values are **Serial RS232** or **Ethernet TCP/IP**
Target IP Address	IP address if the I/O interface is an **Ethernet TCP/IP**-type
Port	TCP/IP port number if the I/O interface is an **Ethernet TCP/IP**-type
Access Password	Password to access a device, if necessary
Connect retry number	Number of retries so that an interface can establish a connection to a device

Configuration of Driver's Properties in Offline Mode

The settings of this Driver's properties can also be accessed at run time if this Driver starts in **Offline** mode using the **Strings** described on the next table.

Configurations at run time

OPTION	FORMAT	DESCRIPTION
FieldLogger.DefaultOutputFolder	Text	Defines a default path or folder to receive downloads from a device
FieldLogger.DefaultSerialNumber	Number	Defines a default serial number for the current device
FieldLogger.ConnectSlaveAddress	Number	Defines an identifier address of a remote device. Possible values range between 1 (one) and 255
FieldLogger.SelectInterface	Number	Selects the type of I/O interface. Possible values are **0**: Serial RS-232 interface or **1**: Ethernet TCP/IP interface
FieldLogger.MbTcpIpOctet1	Number	Defines the first octet of an IP address, if the I/O interface is configured as **Ethernet TCP/IP**. For an IP address equal to **192.168.161.147**, this option represents the value **192**
FieldLogger.MbTcpIpOctet2	Number	Defines the second octet of an IP address, if the I/O interface is configured as **Ethernet TCP/IP**. For an IP address equal to **192.168.161.147**, this option represents the value **168**
FieldLogger.MbTcpIpOctet3	Number	Defines the third octet of an IP address, if the I/O interface is configured as **Ethernet TCP/IP**. For an IP address equal to **192.168.161.147**, this option represents the value **161**
FieldLogger.MbTcpIpOctet4	Number	Defines the fourth octet of an IP address, if the I/O interface is configured as **Ethernet TCP/IP**. For an IP address equal to **192.168.161.147**, this option represents the value **147**
FieldLogger.MbTcpPort	Number	Defines the number of a TCP/IP port, if the I/O interface is configured as **Ethernet TCP/IP**
FieldLogger.AccessPassword	Text	Defines a password for accessing a device, if necessary
FieldLogger.ConnectRetryNumber	Number	Defines a number of attempts so that an interface establishes a connection to a device

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

Tag Reference

All Tags of this Driver are configured with number values using the **[N/B]** parameters.

Field Logger NG Tag

Use a PLC or Block Tag to collect events from a device using the FieldLoggerNG.dll library installed by **FieldLogger Configurator** tool by Novus Produtos Eletrônicos. Collecting mass memory using that library works with the following parameters:

- A path containing data downloads from the memory of a device
- A serial number of a device
- A starting and ending period of mass memory collecting

The first step is to define a path to use as a database for downloads and mass memory collecting. This path can be defined on the **properties window** of this Driver or by writing to the **Output Folder** Tag.

Once users define this database path, use the **Data Download Tags** to transfer memory data from a device to that database path.

After downloading memory data from one or more devices to that database path, users can perform a mass memory collecting by selecting data based on the serial number of a device and a starting and ending period.

The serial number of a device can be defined on the **properties window** of this Driver or by writing to the **Device's Serial Number** Tag.

Collecting mass memory starts by writing to the **Start Download** Tag informing a starting and ending period and, optionally, a list of user-selected channels.

Tags for Databases

Group of Tags to define a path to use as a database for downloads and mass memory collecting, and also to define the serial number of a device to retrieve data.

Output Folder

Reading or Writing

N1	3010
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns or defines the current path for download files, with the extension .fl, and request for collecting mass memory.

NOTE

This Tag is used as a general parameter to define the current path for downloading and collecting mass memory. If not specified, then it considers the path defined on the **configuration window** of this Driver.

Device's Serial Number

Reading and Writing

N1	3011
N2	Query mode for a serial number (read-only). Please check the next text for more information
N3	0 (zero)
N4	0 (zero)

This Tag returns or defines the serial number of a source device for collecting mass memory.

When reading this Tag, the *N2* parameter selects a query mode for checking a serial number. Possible values are **0**: Checks the current user-defined serial number of an application, **1**: Checks the serial number of a device (requires a connection), or **2**: Checks the serial number of a device and automatically defines this serial number as the current serial number of an application (requires a connection).

NOTE

This Tag is used as a general parameter to define the current device of an application to perform a mass memory collecting. If not specified, then it considers the serial number defined on the **configuration window** of this Driver.

Tags for Downloading Data

Group of Tags to download memory data from a device to the path defined by the **Output Folder** Tag.

Download All Memory

Write-Only (Requires a Connection)

N1	Type of memory. Possible values are **3020**: Flash memory or **3021**: SD memory card
N2	0 (zero)
N3	0 (zero)
N4	Mode for clearing memory data. Possible values are **0**: Does not clear memory data or **1**: Clears memory data

This Tag downloads all data from the flash memory of a device or downloads all data from the SD memory card of a device to the directory defined by the **Output Folder** Tag and to the device defined by the **Device's Serial Number** Tag.

Download Partial Memory

Write-Only (Requires a Connection)

B1	3022
B2	0 (zero)
B3	0 (zero)
B4	Mode for clearing memory data. Possible values are **0**: Does not clear memory data or **1**: Clears memory data

This Tag downloads a specific period of data from the SD memory card of a device to the directory defined by the **Output Folder** Tag and to the device defined by the **Device's Serial Number** Tag. The Elements of this Block Tag are the following:

- **Element 1**: Starting date in **Date and Time** format
- **Element 2**: Ending date in **Date and Time** format

Donwload From Folder

Write-Only

N1	3023
N2	0 (zero)
N3	0 (zero)
N4	Mode for clearing memory data. Possible values are **0**: Does not clear memory data or **1**: Clears memory data

This Tag copies data from another source database to the directory defined by the **Output Folder** Tag and to the device defined by the **Device's Serial Number** Tag. This Tag must be in **Text** format, specifying the path of the source database from where data is copied.

Percentage of Download Progress

Read-Only

N1	3024
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns the percentage of progress of data download.

Tags for Querying Channels

Group of Tags to query the types of available channels on the database after downloading memory data from a device.

Query for Available Channels

Write-Only

B1	3031
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

This Tag receives a list of channels with available data for the informed period. The Elements of this Block Tag are the following:

- **Element 1**: Starting date in **Date and Time** format
- **Element 2**: Ending date in **Date and Time** format

NOTE
After finishing a query of all available channels, data is available for an application using the **Get Channel Tags** Tag.

Get Channel Tags

Read-Only

N1	3032
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag receives the result of a query for all available channels performed by the **Query for Available Channels** Tag. The list of values of this Tag is the following:

- **Value 1**: Name of the first channel in **Text** format
- **Value 2**: Name of the second channel in **Text** format
- **Value 3**: Name of the third channel in **Text** format
- **Value N**: Name of the nth channel in **Text** format

NOTE

Use the **OnRead** event of this Tag to receive all these values. To monitor the amount of data remaining on that list, use the **List Count of Channel Tags** Tag.

List Count of Channel Tags

Read-Only

N1	3033
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns the amount of data remaining on the list of data generated by the **Query for Available Channels** Tag.

Tags for Data Collecting

Group of Tags to collect data stored on the current database, according to a user-defined date interval.

Date and Time Interval

Read-Only

B1	3030 (obsolete, use the **Query for Date and Time Interval** and **Get Date and Time Interval** Tags)
B2	Mode for querying an interval. Please check the next text for more information
B3	0 (zero)
B4	0 (zero)

This Tag queries the available period for a mass memory collecting of the registered channels to the directory defined by the **Output Folder** Tag and to the device defined by **Device's Serial Number** Tag. When reading this Tag, the *B2* parameter selects a query mode for the available period. Possible values are **0**: Queries the most recent period available in an application or **1**: Queries the available period in a device (requires a connection). The Elements of this Block Tag are the following:

- **Element 1**: Starting date in **Date and Time** format
- **Element 2**: Ending date in **Date and Time** format

Query for Date and Time Interval

Write-Only

B1	3037
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

This Tag queries the available interval for a mass memory collecting of the registered channels to the directory defined by the **Output Folder** Tag and to the device defined by the **Device's Serial Number** Tag.

NOTE

After writing any value to start a query for the available interval, this data is available to an application by reading the **Get Date and Time Interval** Tag.

Get Date and Time Interval

Read-Only

B1	3038
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

This Tag receives the result of a query for the available period for a mass memory collecting performed by the **Query for Date and Time Interval** Tag. The Elements of this Block Tag are the following:

- **Element 1**: Starting date in **Date and Time** format
- **Element 2**: Ending date in **Date and Time** format

Start Download

Write-Only

B1	3034
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

This Tag starts collecting mass memory from all channels registered in the directory defined by the **Output Folder** Tag and for the device defined by the **Device's Serial Number** Tag. The Elements of this Block Tag are the following:

- **Element 1**: Starting date in **Date and Time** format
- **Element 2**: Ending date in **Date and Time** format
- **Element 3**: Name of the first channel in **Text** format, optional
- **Element 4**: Name of the second channel in **Text** format, optional
- **Element 5**: Name of the third channel in **Text** format, optional
- **Element n**: Name of the nth channel in **Text** format, optional

To perform a mass memory collecting from a user-defined group of channels, add *n* Elements, in **Text** format, to select the available channels in the **Query for Available Channels** Tag. If no channel is specified, all existing channels in that period are collected.

NOTE

After finishing a mass memory collecting, this data is available for an application by reading the **Get Data** Tag.

Get Data

Read-Only

B1	3035
B2	0 (zero)
B3	0 (zero)
B4	0 (zero)

This Tag receives all data from the last mass memory collecting performed by the **Start Download** Tag. The Elements of this Block Tag are the following:

- **Element 1**: Name of a channel in **Text** format
- **Element 2**: Value of a channel in 32-bit **Float** format

NOTE

The timestamp of a mass memory is linked to the timestamp of this Tag. Use the **OnRead** event of this Tag to receive all these values. To monitor the amount of data still on that list, use the **List Count of Data** Tag.

List Count of Data

Read-Only

N1	3036
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns the amount of data still remaining on the list of data generated by the **Start Download** Tag.

Library Version

Read-Only

N1	3001
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns the version of the FieldLoggerNG.dll library in **Text** format.

Connect Device

Write-Only

N1	3002
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag establishes a physical connection to a device, according to the physical layer defined in the **Select the Field Logger NG interface** option on the properties window of this Driver, **Serial RS-232** or **Ethernet TCP/IP**.

NOTE

Some commands from the FieldLoggerNG.dll library require a physical connection to a device to execute successfully. When trying to execute a command that requires a connection, and if this Tag is not user-written, this Driver automatically establishes a connection and then disconnects immediately after completing the execution of this command. If users establish a connection using this Tag, it remains active until users perform a disconnection by writing to the **Disconnect Device** Tag.

Disconnect Device

Write-Only

N1	3003
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag performs a physical disconnection from a device, previously established using the **Connect Device** Tag.

Device's Tag

Read-Only (Requires a Connection)

N1	3004
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns a description of the Tag configured for a device, in **Text** format.

Device's Firmware Version

Read-Only (Requires a Connection)

N1	3005
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag returns the firmware version of a device, in **Text** format.

Is SD Card Available

Read-Only (Requires a Connection)

N1	3006
N2	0 (zero)
N3	0 (zero)
N4	0 (zero)

This Tag check for the presence of an SD memory card in a device. Possible values are **0**: SD memory card is not available or **1**: SD memory card is available.

Documentation of I/O Interfaces

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

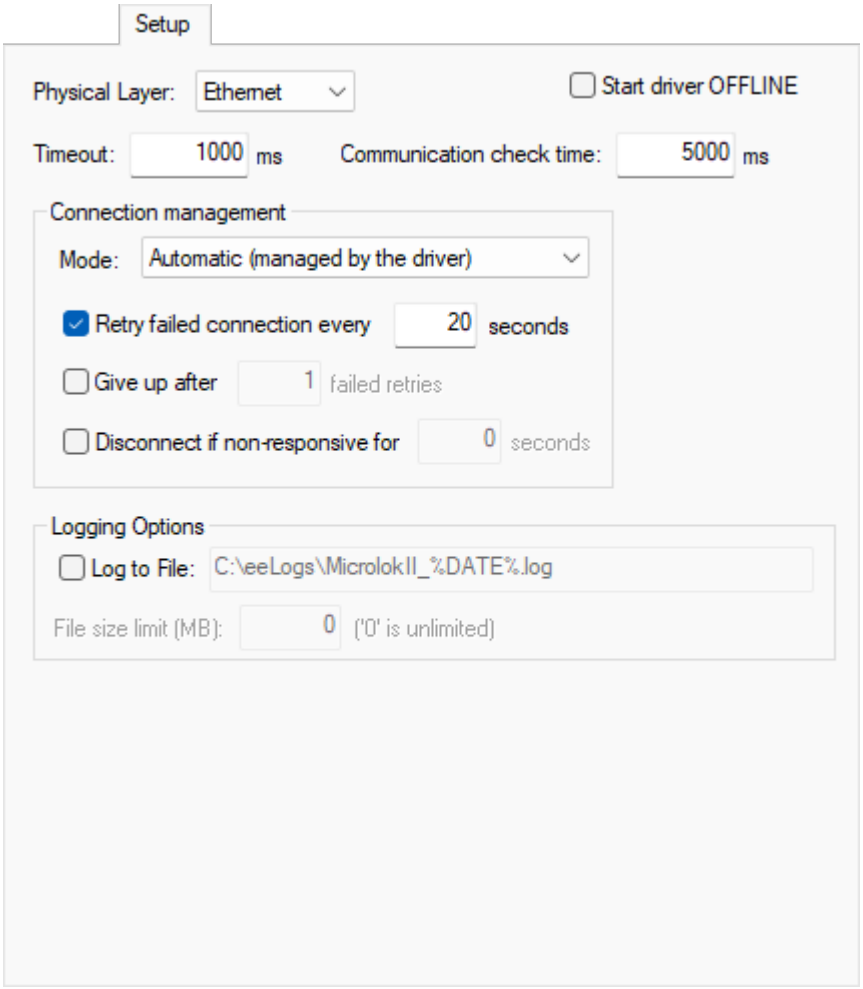
The dialog box of I/O Interfaces allows configuring the I/O connection used by a Driver. This dialog box contains the **Setup**, **Serial**, **Ethernet**, **Modem**, and **RAS** tabs, described on the next topics. If a Driver does not implement a specific I/O connection, its corresponding tab is not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

General Configurations

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

I/O Tags

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

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

IO.CommunicationStatus

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

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

IO.IOKitEvent

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

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

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

NOTE

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

IO.PhysicalLayerStatus

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

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

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

IO.SetConfigurationParameters

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

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

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

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

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

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

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

A variation of the previous example uses a bidimensional array.

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

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

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

IO.WorkOnline

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

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

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

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

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

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

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

IMPORTANT

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

Properties

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

IO.ConnectionMode

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

IO.GiveUpEnable

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

IO.GiveUpTries

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

IO.InactivityEnable

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

IO.InactivityPeriodSec

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

IO.RecoverEnable

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

IO.RecoverPeriodSec

Delay time between two connection attempts, in seconds.

NOTE

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

IO.StartOffline

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

NOTE

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

IO.TimeoutMs

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

IO.Type

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

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

Statistical Configuration

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

I/O Tags

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Driver's Revision History

VERSION	DATE	AUTHOR	COMMENTS
3.0.5	04/02/2026	M. Ludwig	• Driver updated to **IOKit** library version **3.0** and Visual Studio 2022 (*Case 38000*).
		C. Mello	• Adjustments for collecting data with a volume that exceeds 65,000 items (*Case 34560*).
3.0.4	02/13/2017	C. Mello	• Provided version **1.8** of FieldLoggerNG.dll's API for use with this Driver (*Case 22190*).
3.0.3	02/13/2017	C. Mello	• Final adjustments to source code and documentation.
3.0.2	02/06/2017	C. Mello	• Added new baud rate options for RS-232 Serial communication (*Case 21973*).
3.0.1	06/23/2016	C. Mello	• Adjustments for migrating to **IOKit** library version **2.0** (*Case 20279*). • Adjustments for updating to FieldLoggerNG.dll library version **1.8** (*Case 20276*). • Fixed a possible memory leak for some internal processes of this Driver (*Case 20958*). • Implemented the **Query for Date and Time interval** and **Get Date and Time Interval** Tags (*Case 20961*). • Removed the **Field Logger Collect** Tag from the old FieldLoggerCollect.dll library. • Removed the **Modbus** Tag. • Removed the **Field Logger** Tag.
2.1.1	12/14/2011	C. Mello	• Added commands for collecting mass memory via FieldLoggerNG.dll library (*Case 12230*).
2.0.0	10/06/2008	C. Mello	• Added commands for collecting mass memory using the FieldLoggerCollect.dll library (*Case 8892*).

VERSION	DATE	AUTHOR	COMMENTS
			• Added support for reading and writing via **Modbus RTU** protocol (*Case 8892*). • Modified the *P2* parameter of this Driver. • Modified the *B1* parameter of the Block Tag used in this Driver in version **1.0**.
1.0.1	07/16/2008	C. Mello	• Updated the protection identifier (*Case 9665*).
1.0.0	10/21/1999	R. Farina	• Initial version of this Driver.

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

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

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

www.elipse.com.br

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

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

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

Microsoft Partner
Gold Independent Software Vendor (ISV)