

Voestalpine UniAC2 Driver

File Name	UniAC2.dll
Manufacturer	Voestalpine Railway Systems GmbH
Devices	UniAC[2] Axle Counting System
Protocol	Proprietary
Version	1.0.1
Last Update	12/28/2023
Platform	Win32
Dependencies	IOKit v2.0 or later and .NET Framework 4.0 or later
Superblock Readings	No
Level	0

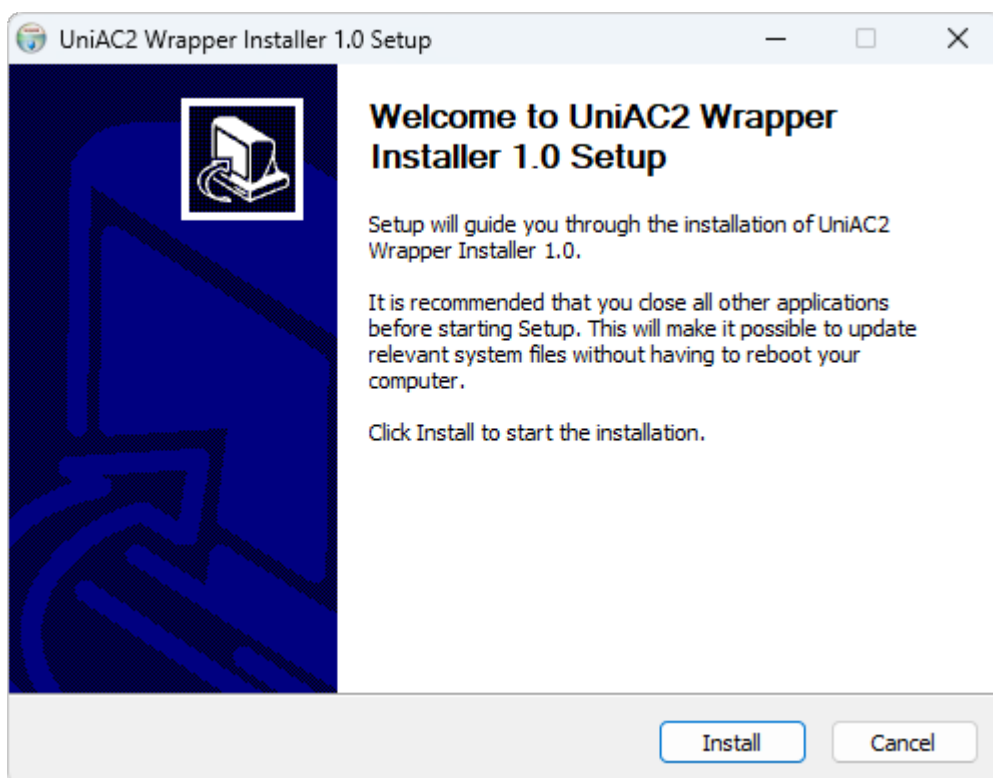
Introduction

This Driver communicates with the UniAC[2] Axle Counting System equipment through the **Voestalpine.UniAC2.ResetApi** API provided by Voestalpine Railway Systems GmbH.

Driver Configuration

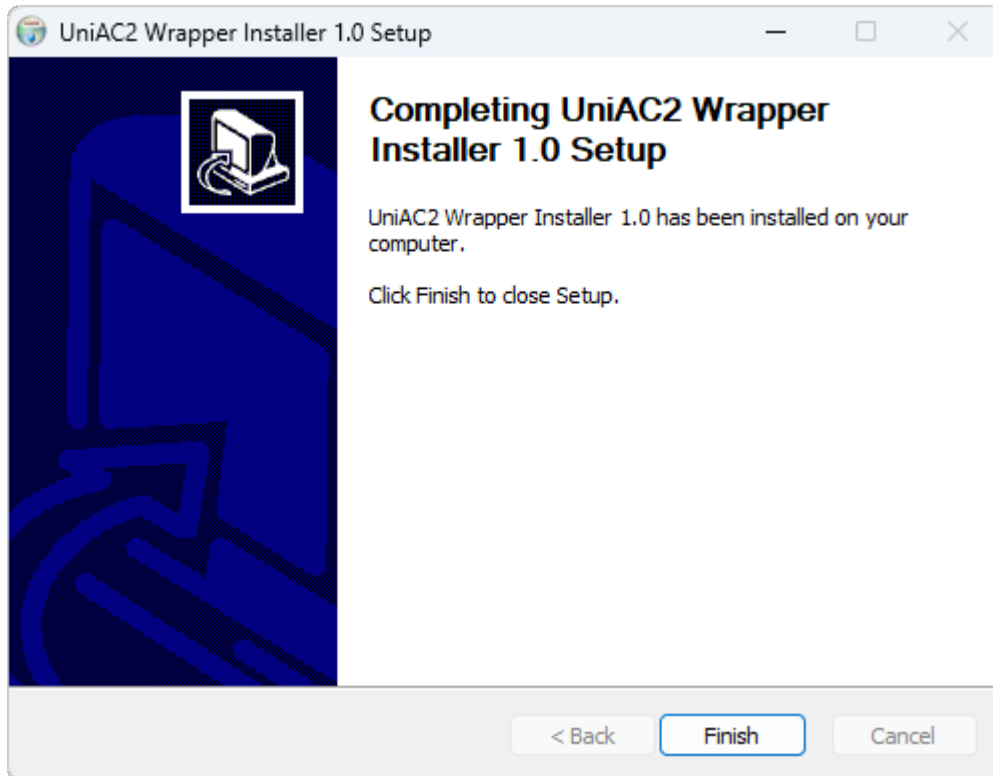
To ensure the full functionality of this Driver, execute **UniAC2 Wrapper Installer** as administrator, distributed with this Driver, according to the next steps.

1. On the initial screen, click **Install**.



Installer's initial screen

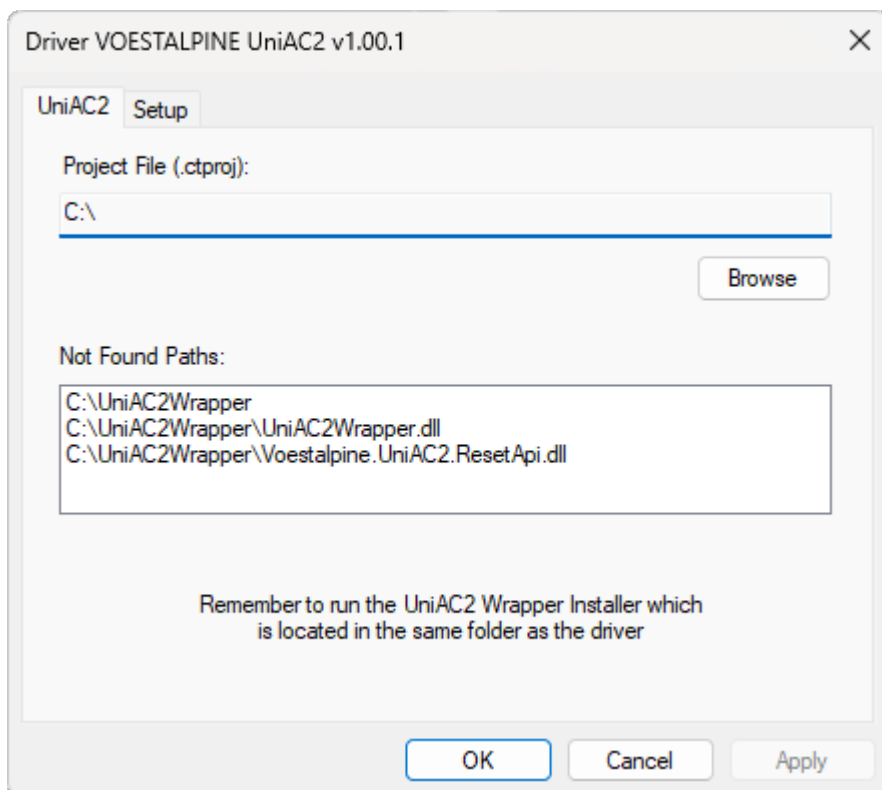
- On the next screen, click **Finish**.



Installer's final screen

Configuring Properties

The **UniAC2** tab contains specific settings for this Driver.



UniAC2 tab

Click **Browse** to select a project file, with a .ctproj extension. The **Not Found Paths** list contains folders or files not found by this Driver. If this list is not empty, please execute **UniAC2 Wrapper Installer** to automatically solve any pending issues.

Tag Reference

This section contains information about the configuration of this Driver's [N/B] Tags. Optionally, **Elipse E3** or **Elipse Power** users can import this Driver's Tags using Tag Browser.

Reading Tags

PLC or Block Tags for reading in this Driver are described on the next table.

Reading Tags of Voestalpine UniAC2 Driver

NAME	TYPE	N1/B1	N2/B2	N3/B3	N4/B4	ELEMENTS	RETURN	DESCRIPTION
GeneralState	PLC	0 (zero)	-	-	-	-	Connection State enumeration	Global communication status
Integrators	Block	1 (one)	-	-	-	3 (three)	List of IxC structures	List of integrators
Sections	Block	2 (two)	0 (zero)	-	-	5 (five)	List of Section structures	List of Sections ordered by index
			1 (one)					List of Sections ordered by identifier
Section	Block	3 (three)	Identifier	-	-	5 (five)	Section structure	Section with a specific identifier present in the <i>B2</i> parameter
ResetStatus	PLC	4 (four)	-	-	-	-	ResetExecutionResult enumeration	Last status of a reset event, SendReset or Reset . Default value is Null

Writing Tags

PLC or Block Tags for writing in this Driver are described on the next table.

Writing Tags of Voestalpine UniAC2 Driver

NAME	TYPE	N1/B1	N2/B2	N3/B3	N4/B4	DESCRIPTION
SendReset	PLC	5 (five)	Identifier	-	-	Resets the Section with the identifier in the <i>N2</i> parameter

NAME	TYPE	N1/B1	N2/B2	N3/B3	N4/B4	DESCRIPTION
Reset	PLC	6 (six)	Index	-	-	Resets the Section with the index in the N2 parameter

Enumerations and Structures Reference

This section contains information about enumerations and structures returned by the reading Tags of this Driver.

Enumerations

Values of ConnectionState enumeration

VALUE	NAME	DESCRIPTION
0 (zero)	Unknown	Unknown initial connection status
10	NoConnected	Not connected
20	XConnected	At least one of the configured UniAC[2] integrator modules is not connected
30	AllConnected	All UniAC[2] integrator modules are connected
100	InternalError	Internal error

Values of ResetStatus enumeration

VALUE	NAME	DESCRIPTION
-1 (minus one)	Unknown	Unknown initial status or no valid information available
0 (zero)	WaitForRequest	Waiting for the next reset request. This is the default status of a Section
1 (one)	WaitForReset	Pre-reset properly received and a reset is required. It can occur when a reset procedure PreReset + Reset is configured in a UniAC[2] system
2 (two)	WaitForCot	Proper trigger sequence received and a train passage is required. It can occur when any reset procedure requiring a CoT procedure is configured in a UniAC[2] system
3 (three)	NoPrereset	Received a reset request, but a pre-reset is required first. The reset request is discarded, which can occur when a reset procedure PreReset + Reset is configured in a UniAC[2] system
4 (four)	UnexpectedTrigger	Unexpected trigger received, such as a reset is expected but a pre-reset is received instead

VALUE	NAME	DESCRIPTION
5 (five)	TimeoutReset	A long time waiting for a reset request after receiving a pre-reset request. It can occur when a reset procedure PreReset + Reset is configured in a UniAC[2] system
6 (six)	PulseTimeError	Duration of hardware pulse is too long or too short. It can occur if a pulse reset is issued
7 (seven)	TimeoutCot	A long time waiting for a train passage after a trigger
8 (eight)	TimeoutBlockMode	A long time waiting for a trigger from a remote AXM module. It can occur when the Block mode is used in the configuration of a UniAC[2] system
9 (nine)	CotError	Error during a train passage. It can occur when a CoT procedure is configured in a UniAC[2] system
10	SensorOCC	A sensor on the track Section is occupied and the request is discarded
11	SensorError	A sensor on the track Section signals an error and the request is discarded
12	RmtError	Reset rejected because the remote AXM module signals an error and the request is discarded
13	CommError	Reset rejected because a communication error occurred and the request is discarded
14	CardError	Reset rejected because a fatal error occurred and the request is discarded
15	WaitForBlockMode	Waiting for resetting from the remote side in Block mode. It can occur when the Block mode is used in a UniAC[2] system
16	StatMrReject	A Master Reset request is rejected by the Slave Section. It can occur when a Master Reset request is used in a UniAC[2] system

Values of OccState enumeration

VALUE	NAME	DESCRIPTION
0 (zero)	Unknown	Unknown initial status or no valid information about the Section is available
1 (one)	OCC	Occupied
2 (two)	OccAndWaitForCot	Occupied. A train passage is required to change the status to FREE

VALUE	NAME	DESCRIPTION
3 (three)	FREE	Not occupied
100	InternalError	Internal error
101	InvalidIndex	Invalid index or identifier

Values of ResetExecutionResult enumeration

VALUE	NAME	DESCRIPTION
0 (zero)	ResetAccepted	Reset command accepted for execution
1 (one)	SectionDoesNotExist	The provided Section identifier does not exist in the current system configuration
2 (two)	CommunicationIssues	Command cannot be executed because currently there is a communication problem with the integrator module responsible for a certain Section
100	InternalError	Internal error
101	InvalidIndex	Invalid index or identifier

Structures

Values of Section structure

ELEMENT	NAME	DESCRIPTION
0	AxleCount	Number of axles on the track Section. This element can be empty
1	Identifier	Section identifier
2	SectionName	Name of a Section
3	ResetStatus	Reset status, indicated in the ResetStatus enumeration
4	State	Status of a Section, indicated in the OccState enumeration

Values of IxC structure

ELEMENT	NAME	DESCRIPTION
0	CommunicationState	Status of the connection with a UniAC[2] integrator module, indicated in the ConnectionState enumeration
1	IntegratorName	Virtual name of a UniAC[2] communication channel
2	IpAddress	IP address of a UniAC[2] integrator module


Documentation of I/O Interfaces

This section contains the documentation of I/O Interfaces referring to the **UniAC2** Driver.

Driver Configuration

I/O Interface configuration is performed on a Driver's configuration dialog box. To access the configuration of this dialog box in **E3** (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 **E3** version 2.0 or later, click **Configure driver**  on a Driver's toolbar. In **Eclipse 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 Driver for each serial port.

Configuration Dialog Box

The I/O Interfaces dialog box 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 each Driver, on the configuration dialog box.

Setup Tab

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

- **General configurations:** Configurations of 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

Physical Layer: Serial Start driver OFFLINE

Timeout: 1000 ms

Connection management

Mode: Automatic (managed by the driver)

Retry failed connection every 20 seconds

Give up after 1 failed retries

Disconnect if non-responsive for 0 seconds

Logging Options

Log to File: C:\eeLogs\Modbus_%.DATE%.log

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 reception's buffer
Start driver OFFLINE	Select this option so that the 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. Please check topic Driver Statuses for more details
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, the Driver keeps retrying

OPTION	DESCRIPTION
	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, the 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	<p>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.</p> <p>If the %PROCESS% macro is used in the log file name, it is replaced by the ID of the current process. This option is particularly useful when using several instances of the same Driver in E3, thus allowing each instance to generate a separate log file. For example, when configuring this option as c:\e3logs\drivers\sim_%PROCESS%.log, a file named c:\e3logs\drivers\sim_00000FDA.log is generated for process 0FDAh.</p> <p>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 as c:\e3logs\drivers\sim_%DATE%.log, a file named c:\e3logs\drivers\sim_2005_12_31.log is generated in 12/31/2005 and a file named c:\e3logs\drivers\sim_2006_01_01.log is generated in 01/01/2006</p>

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.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 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:** Event message, 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. Its possible values are the following:

- **0:** Physical layer stopped, that is, the 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, the Driver is in **Online** mode but the physical layer is not connected. If the **Connection management** option is configured as **Automatic**, the physical layer can be connecting,

disconnecting, or waiting for a reconnection attempt. If the **Connection management** option is configured as **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 the device is connected, only 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 the 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 **Eclipse SCADA**, users must use a Block Tag. Every parameter to configure uses two Block Elements. For example, if users want to configure three parameters, then the size of the Block must be 6 (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 writing 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 **E3**, the ability to create arrays at run time allows using an I/O Tag as well as a Block Tag. Users can use Driver's **Write** method to send all parameters to the 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 Driver's log or use the *writeStatus* parameter of the **WriteEx** method to find out the exact cause of the 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 on the **IO.PhysicalLayerStatus** Tag

In the next example, using **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 can fail when configuring a Driver to **Online** mode, that is, writing the value 1 (one). In this case, the 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

- 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 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, where a Driver manages the connection or **1**: Manual mode, where an application manages the connection.

IO.GiveUpEnable

☑ When configured to True, defines a maximum number of reconnection attempts. If all reconnection attempts fail, the Driver enters the **Offline** mode. When configured to False, the 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 reconnection is lost. If this one fails, a 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 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

9 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

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

IO.Type

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

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

Statistical Configuration

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

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

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

IO.Stats.Partial.BytesRecv

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

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

IO.Stats.Partial.BytesSent

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

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

IO.Stats.Partial.TimeConnectedSeconds

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

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

IO.Stats.Partial.TimeDisconnectedSeconds

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

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

IO.Stats.Total.BytesRecv

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

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

IO.Stats.Total.BytesSent

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

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

IO.Stats.Total.ConnectionCount

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

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

IO.Stats.Total.TimeConnectedSeconds

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

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

IO.Stats.Total.TimeDisconnectedSeconds

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

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

Properties

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

Driver Revision History

VERSION	DATE	AUTHOR	COMMENTS
1.0.1	12/28/2023	A. Fetzner	<ul style="list-style-type: none"> Initial version of this Driver (Case 33594).

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, 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)