



# ibaPDA-Interface-EtherNet/IP

## Data Interface for EtherNet/IP

Manual  
Issue 3.0

Measurement Systems for Industry and Energy  
[www.iba-ag.com](http://www.iba-ag.com)

---

## Manufacturer

iba AG  
Koenigswarterstrasse 44  
90762 Fuerth  
Germany

## Contacts

Main office +49 911 97282-0  
Fax +49 911 97282-33  
Support +49 911 97282-14  
Engineering +49 911 97282-13  
E-mail [iba@iba-ag.com](mailto:iba@iba-ag.com)  
Web [www.iba-ag.com](http://www.iba-ag.com)

Unless explicitly stated to the contrary, it is not permitted to pass on or copy this document, nor to make use of its contents or disclose its contents. Infringements are liable for compensation.

© iba AG 2023, All rights reserved.

The content of this publication has been checked for compliance with the described hardware and software. Nevertheless, discrepancies cannot be ruled out, and we do not provide guarantee for complete conformity. However, the information furnished in this publication is updated regularly. Required corrections are contained in the following regulations or can be downloaded on the Internet.

The current version is available for download on our web site [www.iba-ag.com](http://www.iba-ag.com).

Version	Date	Revision	Author	Version SW
3.0	08-2023	New version ibaPDA v8	RM	8.0.0

Windows® is a brand and registered trademark of Microsoft Corporation. Other product and company names mentioned in this manual can be labels or registered trademarks of the corresponding owners.

## Contents

<b>1</b>	<b>About this Manual .....</b>	<b>5</b>
1.1	Target group and previous knowledge .....	5
1.2	Notations .....	5
1.3	Used symbols.....	6
<b>2</b>	<b>System requirements .....</b>	<b>7</b>
<b>3</b>	<b>EtherNet/IP interface .....</b>	<b>8</b>
3.1	Introduction.....	8
3.2	EtherNet/IP communications .....	8
3.3	Types of ibaPDA modules .....	10
3.4	Configuration and engineering PLC side .....	12
3.4.1	Configuration for I/O Module with RSLogix 5000 .....	12
3.4.2	Configuration for Produced Tag module with RSLogix5000.....	16
3.4.3	Configuration for I/O Module with Schneider Electric Unity Pro XL.....	17
3.5	Configuration and engineering ibaPDA.....	28
3.5.1	General interface settings.....	28
3.5.2	Adding a module.....	29
3.5.3	General module settings.....	30
3.5.4	Signal configuration .....	34
3.6	Configuration of ibaPDA output modules.....	36
3.6.1	General module settings ibaPDA output modules.....	36
3.6.2	Signal configuration .....	37
3.7	Setting up multiple IP addresses for ibaPDA.....	37
<b>4</b>	<b>Troubleshooting and diagnostics .....</b>	<b>39</b>
4.1	Ethernet switch features important for Ethernet/IP .....	39
4.2	Conflict with other Ethernet/IP related programs.....	42
4.3	License .....	43
4.4	Connection problems linked to licenses .....	44
4.5	Connection table .....	45
4.6	Connection diagnostics with PING.....	47
4.7	Log files.....	48

4.8 Diagnostic modules ..... 49

**5 Support and contact..... 54**

# 1 About this Manual

This document describes the function and application of the software interface

*ibaPDA-Interface-EtherNet/IP*.

This documentation is a supplement to the *ibaPDA* manual. Information about all the other characteristics and functions of *ibaPDA* can be found in the *ibaPDA* manual or in the online help.

## 1.1 Target group and previous knowledge

This manual is aimed at qualified professionals who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as professional if he/she is capable of assessing safety and recognizing possible consequences and risks on the basis of his/her specialist training, knowledge and experience and knowledge of the standard regulations.

## 1.2 Notations

In this manual, the following notations are used:

Action	Notation
Menu command	Menu <i>Logic diagram</i>
Calling the menu command	<i>Step 1 – Step 2 – Step 3 – Step x</i> Example: Select the menu <i>Logic diagram – Add – New function block</i> .
Keys	<Key name> Example: <Alt>; <F1>
Press the keys simultaneously	<Key name> + <Key name> Example: <Alt> + <Ctrl>
Buttons	<Key name> Example: <OK>; <Cancel>
Filenames, paths	<i>Filename, Path</i> Example: <i>Test.docx</i>

## 1.3 Used symbols

If safety instructions or other notes are used in this manual, they mean:

---

### Danger!



**The non-observance of this safety information may result in an imminent risk of death or severe injury:**

- Observe the specified measures.

---

### Warning!



**The non-observance of this safety information may result in a potential risk of death or severe injury!**

- Observe the specified measures.

---

### Caution!



**The non-observance of this safety information may result in a potential risk of injury or material damage!**

- Observe the specified measures

---

### Note



A note specifies special requirements or actions to be observed.

---

### Tip



Tip or example as a helpful note or insider tip to make the work a little bit easier.

---

### Other documentation



Reference to additional documentation or further reading.

## 2 System requirements

The following system requirements apply for using the EtherNet/IP interface:

- *ibaPDA* v8.0.0 or more recent
- *ibaPDA* base license + license for *ibaPDA-Interface-Ethernet/IP*. The interface license allows you to use 64 connections.
- For more than 64 connections you need additional *one-step-up-Interface-Ethernet/IP* licenses for each additional 64 connections.
- If you still have an older *ibaPDA-Interface-EtherNet/IP* license (prior to June 2015) which has no support for Produced Tags or I/O Scanner and you want to use Produced Tags or the I/O Scanner module, then you should upgrade to an additional license Add-on-EtherNet/IP-Produced-Tags.

For further requirements for the used computer hardware and the supported operating systems, refer to the *ibaPDA* documentation.

### Note



It is recommended carrying out the TCP/IP communication on a separate network segment to exclude a mutual influence by other network components.

### Note



When operated on a virtual machine, a dongle with a valid license must be plugged on the host for each virtual machine. The USB ports used are assigned explicitly to the respective virtual machines.

### License information

Order no.	Product name	Description
31.001.005	ibaPDA-Interface-EtherNet/IP	Extension license for an ibaPDA system adding the data interface: Number of connections: 64
31.101.005	one-step-up-Interface-EtherNet/IP	Extension license for 64 further EtherNet/IP-connections, a maximum of 3 extension licenses is permissible
31.111.005	Add-on-EtherNet/IP-Produced-Tags	Upgrade license for using Produced Tags; applies to older licenses without Produced Tags support only.

Table 1: Available EtherNet/IP-licenses

## 3 EtherNet/IP interface

### 3.1 Introduction

iba has implemented a driver able to read the Ethernet/IP protocol over TCP/IP and UDP. This driver works close together with *ibaPDA* software. Depending on the selected module or communication mode respectively, *ibaPDA* acts as a device passively expecting connections by a scanner (I/O Module) or as a scanner actively establishing the connections (Produced Tag, I/O Scanner).

*ibaPDA* can also send data back to the EtherNet/IP controller like output signals (I/O Module and I/O Scanner only).

The following controllers have been tested successfully:

Rockwell Automation controllers:

(Used module types in brackets)

- CompactLogix (I/O Module, Produced Tag)
- FlexLogix (I/O Module, Produced Tag)
- ControlLogix (I/O Module, Produced Tag)
- SoftLogix 5800 (I/O Module, Produced Tag)
- MicroLogix (I/O Module, Produced Tag)

Other controllers:

- Schneider Electric M580 ePAC (I/O Module)
- Keyence CB-EP100 (I/O Scanner)
- WAGO 750-352 (I/O Scanner)
- Omron C2JM (Produced Tag)

### 3.2 EtherNet/IP communications

Ethernet Industrial Protocol (Ethernet/IP) is an open industrial networking standard that supports both real-time I/O messaging and message exchange.

The EtherNet/IP-network offers a complete range of control, configuration and data acquisition services. Ethernet/IP uses TCP/IP for general messaging/information exchange services and UDP/IP for I/O messaging services for control applications. This combination of well-accepted standards provides the functionality required to support both information data exchange as well as control applications.

EtherNet/IP defines two primary communication types: „explicit" and „implicit".

CIP Message Type	CIP Communication Relationship	Transport protocol	Communication Type	Typical Use
Explicit	Connected or Unconnected	TCP/IP	Request/reply transactions	Non time-critical information data
Implicit	Connected	UDP/IP	I/O data transfers	Real-time I/O data

Table 2: EtherNet/IP communication types

**Note**

CIP (Common Industrial Protocol) is an industrial protocol for industrial automation applications. CIP encompasses a comprehensive suite of messages and services for the collection of manufacturing automation applications – control, safety, synchronization, motion, configuration and information.

**Explicit Messaging** in general has a request/reply (or client/server) nature. This type of communication is used for non-real-time data, normally for information. For Ethernet/IP: Explicit Messaging uses TCP.

**Implicit Messaging** is also often referred to as “I/O” and is time-critical in nature. Typically this type of communication is used for real-time data exchange, where speed and low latency are important. Implicit messages include very little information about their meaning, so the transmission is more efficient, but less flexible than explicit. The interpretation of the transmitted data is fast. With Implicit Messaging you establish an association (a “CIP connection”) between two devices and produce the Implicit Messages according to a predetermined trigger mechanism, typically at a specified packet rate. The devices both know and agree on the data formats they will use (i.e., the format is “implied”). For EtherNet/IP: Implicit Messaging uses UDP and can be multicast or unicast.

**Note**

*ibaPDA* supports both multicast and unicast messages on its module types "I/O Module" and "I/O Scanner". On "Produced Tag" module type only unicast is supported.

However, iba recommends using unicast messages in order to avoid multicast flooding in the network. The use of a separate network is also strongly recommended.

### 3.3 Types of ibaPDA modules

*ibaPDA* can use different methods to connect to a controller or a device. These methods are represented by 3 different types of modules.

Modulertype EtherNet/IP...	Scanner	Device	Data input for <i>ibaPDA</i>	Data output for <i>ibaPDA</i>
I/O Module	PLC	PDA	Output assembly instance of device (=PDA) identified by its number	Input assembly instance of device (=PDA) identified by its number
Produced Tag	PDA	EIP device or controller, e.g. ControlLogix, Omron, ...	A tag identified by its name	None
I/O Scanner	PDA	EIP-device, e.g. Keyence CB-EP100	Input assembly instance of device (=CB-EP100) identified by its number	Output assembly instance of device (=CB-EP100) identified by its number.

Table 3: Communication methods of the 3 module types

#### General principle

The scanner initiates the connection by sending a forward open. In the forward open the scanner needs to specify which data it wants to exchange (assembly instance or tag) and how it wants to exchange the data (unicast/multicast, cycle time, timeouts, data sizes, ...).

#### I/O Module

When using the I/O Module type, *ibaPDA* acts like an I/O adapter or device. The PLC controller is a scanner. Only implicit messaging is used. *ibaPDA* passively waits for a connection established by the PLC and the data being sent. Therefore, no IP address is needed in the module configuration.

#### Produced Tag

When using the Produced Tag type, *ibaPDA* acts like a scanner. The PLC controller is an I/O adapter or device. Explicit messaging is used by *ibaPDA* to some vendor specific objects for fetching information about availability and configuration of Produced Tags (works for Rockwell PLCs only). As a scanner *ibaPDA* actively establishes the connection, which is performed by explicit messaging. Therefore, an IP-address (of the controller), the Produced Tag and size are required in the module configuration. The data from the PLC or device are then sent by implicit messaging.

Yet, *ibaPDA* can only read tag information from Rockwell PLCs and not from PLCs of other vendors. But it is possible to enter manually the tag name and then configure the signals so that they match the tag structure.

The module type Produced Tag supports CIP-routing, i.e. a connection even through different bus systems (EtherNet/IP, ControlNet, DeviceNet) if the target CPU cannot be connected directly by *ibaPDA*.

---

**Note**

When the acquisition is started then *ibaPDA* will try to connect to the Produced Tag on the CPU. When this does not succeed then *ibaPDA* will in fact give an error in the validation form, but the acquisition will still start anyway. *ibaPDA* will periodically try to reconnect to the Produced Tag as long as the acquisition is running. When the acquisition is stopped *ibaPDA* will disconnect from the Produced Tag.

---

**I/O Scanner**

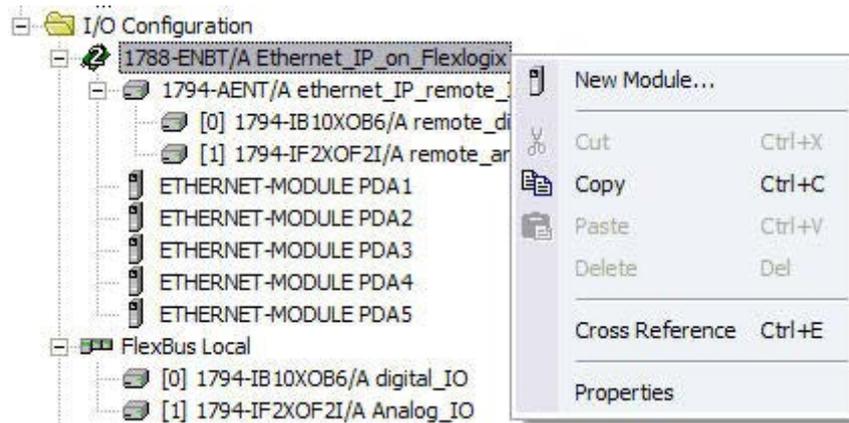
When using the I/O Scanner type, *ibaPDA* acts like a scanner and establishes the connection to a device. Such devices are usually remote I/O units (e.g. from WAGO) or measurement devices (e.g. from Keyence). An IP address (of the device), the assembly instances and size are required in the module configuration. The data from the device are then sent by implicit messaging.

## 3.4 Configuration and engineering PLC side

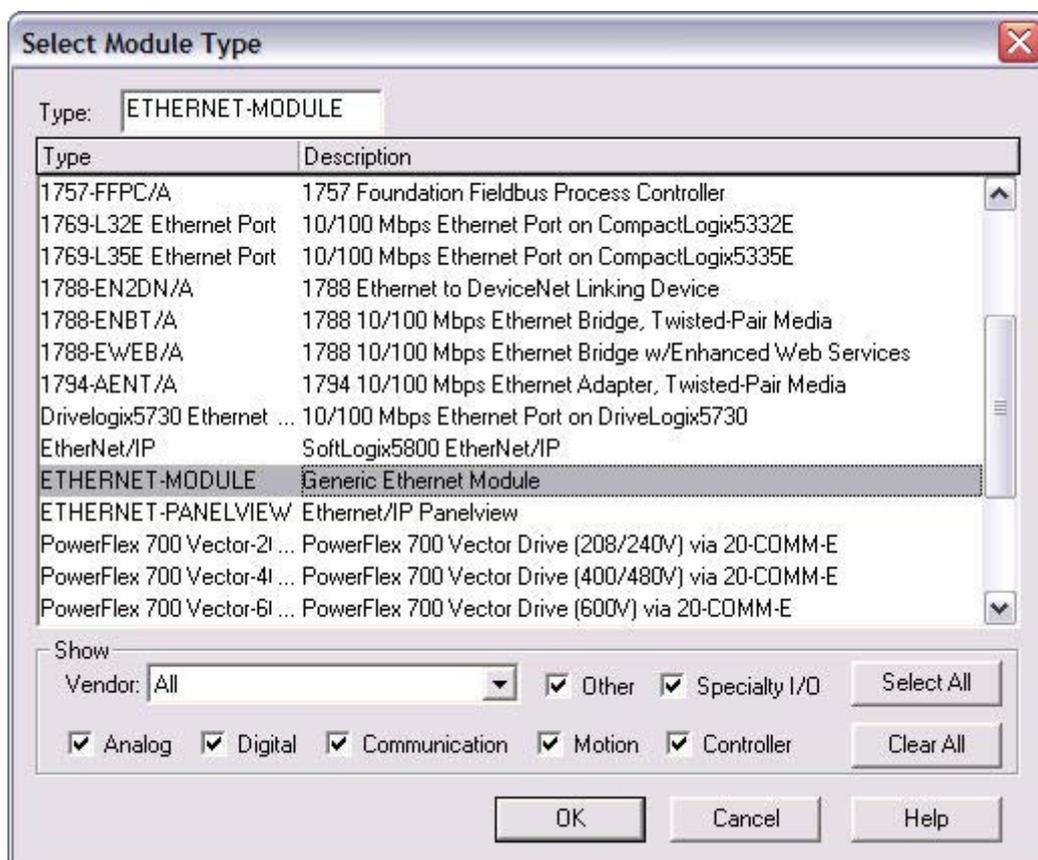
### 3.4.1 Configuration for I/O Module with RSLogix 5000

(Only Allen-Bradley/Rockwell devices)

1. In RSLogix 5000 add a new "ETHERNET-MODULE" to the EtherNet/IP- interface card (z. B. 1788-ENBT).
2. Right click the Ethernet/IP interface card. The context menu is shown.



3. Click on *New Module...*
4. In the module type list, you have to choose the "ETHERNET-MODULE" with the description "Generic Ethernet Module".



5. After you've created the new module, you will see the *Module Properties* dialog box, where you have to configure the following settings:

In the *Module Properties* dialog box, the following settings have to be configured:

- Name of the module (the array, which is later used to copy data to *ibaPDA*, will be called by this name)
- Comm(unication) Format (see below)
- IP Address or Host Name of the *ibaPDA*-System (IP Address should be used)
- Assembly instance numbers and sizes (the instance numbers must be identical for all three assemblies). Size of configuration should be zero, this data can not be used with *ibaPDA*.

The assembly instance numbers can be 1...64.

It is possible to accumulate up to 4 licenses resulting in a maximum of 256 connections. In this case the assembly instance can be up to 256.

#### Note



If more than one Generic Ethernet Module is required with connection to *ibaPDA* please observe to enter different IP addresses for each module. Each Generic Ethernet Module corresponds to one TCP/IP connection which corresponds to one module in *ibaPDA*. Therefore, multiple IP addresses have to be assigned to the network adapter of *ibaPDA*.

See chapter [↗ Configuration and engineering ibaPDA](#), page 28 for setup information.

The following data types are possible for use with *ibaPDA*:

- SINT (8 bit integer)
- INT (16 bit integer)

- DINT (32 bit integer)
- REAL (32 bit real)

Only data types without the suffix „with status“ are allowed.

Meaning of assemblies and size definitions:

- Size values are defined as multiple of the data type. If you use “Data – DINT” (32 Bit) and as size 4, it is possible to transfer four 32-Bit integers
- Input: If you only want to transfer data from the PLC to *ibaPDA* for data acquisition the input size should be one to minimize network traffic. If you also want to send data from *ibaPDA* back to the PLC the input size defines the length of the data *ibaPDA* can send. The maximum value in RSLogix is 500 bytes. If you use another data type as SINT, it is 500 divided by the size of type in bytes.
- Output: Data transferred to *ibaPDA* (maximum value: 496 Byte). If you use another data type as SINT, maximum size is 496 divided by the size of type in bytes.
- Configuration: Defines, how many configuration data bytes can be transferred within the “ForwardOpen”-Command. *ibaPDA* does not support configuration via this command. Enter 0 as the size.

---

#### Note



- The assembly instances for Input, Output and Configuration **must be identical**.
- The assembly instance numbers usually must be between 1 and 64.  
It is possible to accumulate up to 4 licenses resulting in a maximum of 256 connections. In this case the assembly instance can be up to 256.
- The assembly instance number is the same as the EtherNet/IP connection number in *ibaPDA*. The number must be unique for the *ibaPDA*.

Do not configure two connections with identical assemblies to connect with the same *ibaPDA* host, even from different controllers!

---

#### Note



The size of inputs AND outputs is always defined in the PLC configuration (e.g. in RSLogix or the Unity Pro for Schneider PLCs).

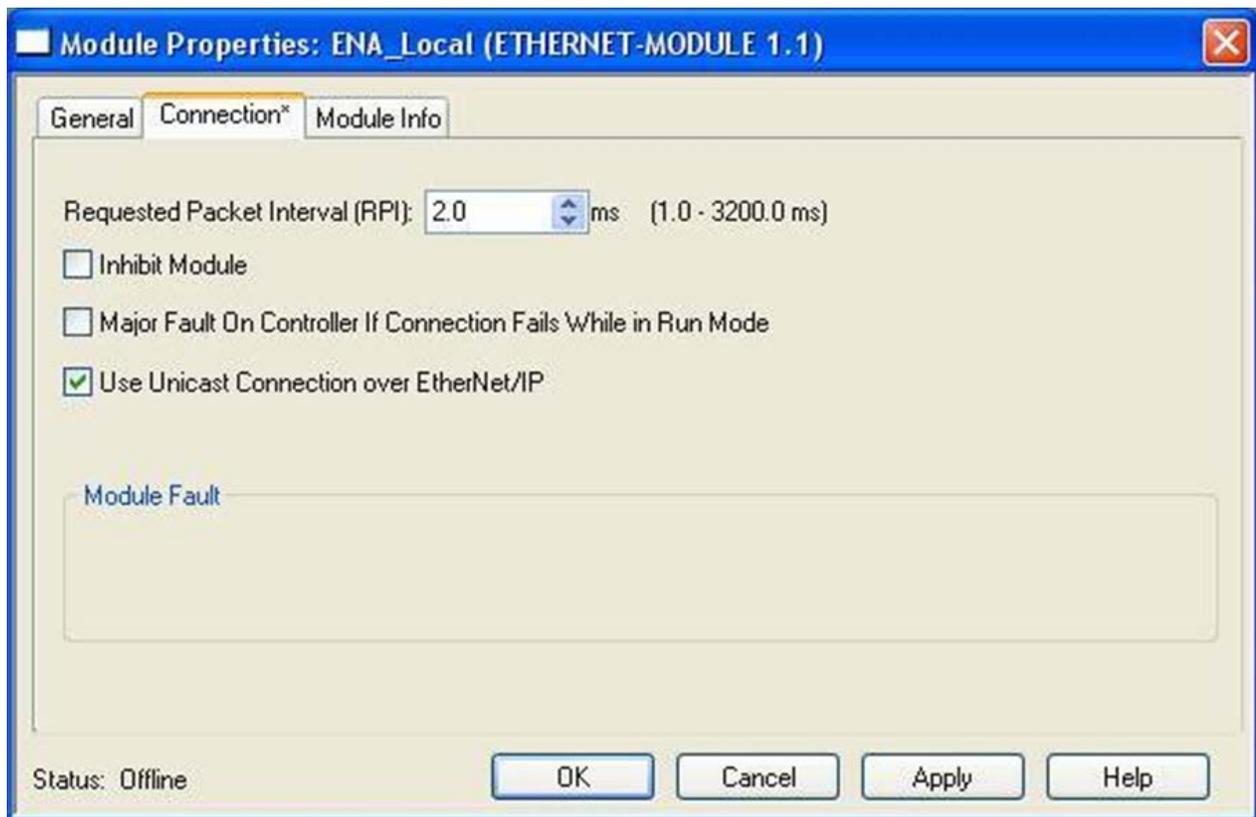
*ibaPDA* checks the size reported by the ForwardOpen information and warns the user if the signal address offset + size of (Datatype) in *ibaPDA* is larger than the size defined on the PLC side.

The validation is aborted with an error in case the address offset + size of (Datatype) is larger than 1024 bytes for the inputs and 1004 for the outputs of *ibaPDA*.

Clicking <Next> will bring you to the definition of the time interval. *ibaPDA* supports values between 1 ms and 1000 ms. Depending on the PLC and the programs on this device, you can not use very small intervals (e.g. an Allen-Bradley FlexLogic doesn't work stable with a value less than 3 ms).

*ibaPDA* supports unicast reply connections over Ethernet/IP. Hence, you may enable the option *Use Unicast Connection over Ethernet/IP*.

Configuration of communication with unicast is a little bit easier because preventive measures against multicast side effects. IGMP Snooping and IGMP Query are not necessary here.



After the new module is added, it is possible to copy data from the PLC within programs and tasks to the new array (as named like the module). This array has the following three members:

- Input: Contains all data to be transferred from *ibaPDA* to the PLC.
- Output: Contains all data that will be sent via Ethernet/IP to *ibaPDA*.
- Config: Configuration values send with the "ForwardOpen" command. Not supported by *ibaPDA*.  
The Config-Array will also exist, if the ETHERNET-MODULE was defined with a configuration size of zero.
- Each of these sub-arrays has a member "Data", which contains the values. Each task in the PLC has write and read access to this option "Data". But it only makes sense to write the "output" data.

In order to copy data to *ibaPDA*, i. e. to the Output-Array, you can use one of the following commands (ladder, function blocks or structured text commands):

- **MOV** – Copies one tag (or one element of an array or structure) to a destination (here: one element of the Output-Array). The source and the target have to have the same data type. Otherwise the PLC would change the data into the data type of the target or another error occurs. You can use this option, if you'll transfer only data of one type within one Ethernet/IP connection to *ibaPDA*.
- **COP** – Copies one tag (or one element of an array or structure) to a destination (here: one element of the Output-Array) with a defined length. You can specify the length. The source and the target could have different data types. Nothing is rejected by the PLC. If you define the ETHERNET-MODULE as data type "SINT" (8 bit) you can copy your data from the PLC as you like (you can create your own "structure"). With this option it is possible to transfer different data types within the same connection to *ibaPDA*.

### Other documentation



Please see the RSLogix-Dokumentation for further information.

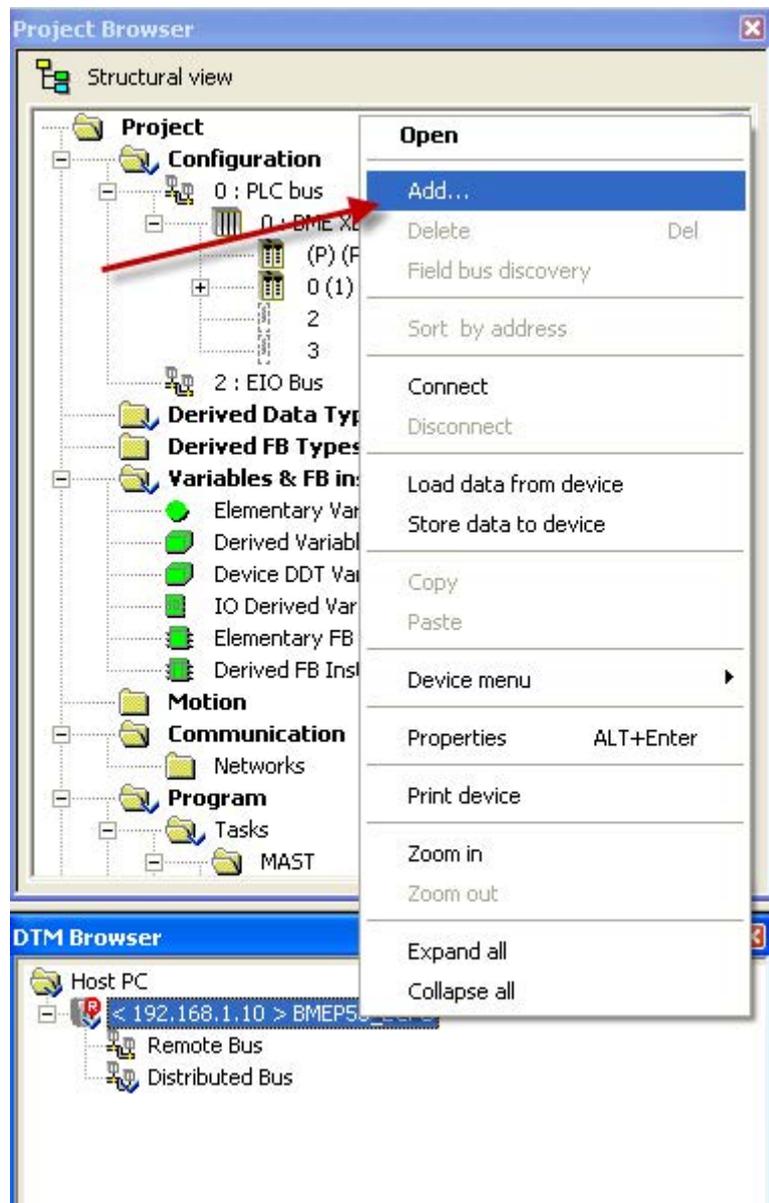
### 3.4.2 Configuration for Produced Tag module with RSLogix5000

In RSLogix5000 and Studio5000 you can configure a controller tag as a produced tag. A produced tag allows other systems to connect to it and periodically receive the data of the tag. In the tag properties you can set the type to *Produced*. By default a produced tag allows 1 consumer but you can increase it. By default it allows unicast connections. For produced tags *ibaPDA* only supports unicast connections. So, make sure that Unicast is enabled. A produced tag normally has a user-defined data type. The maximum size of the data is 500 bytes.

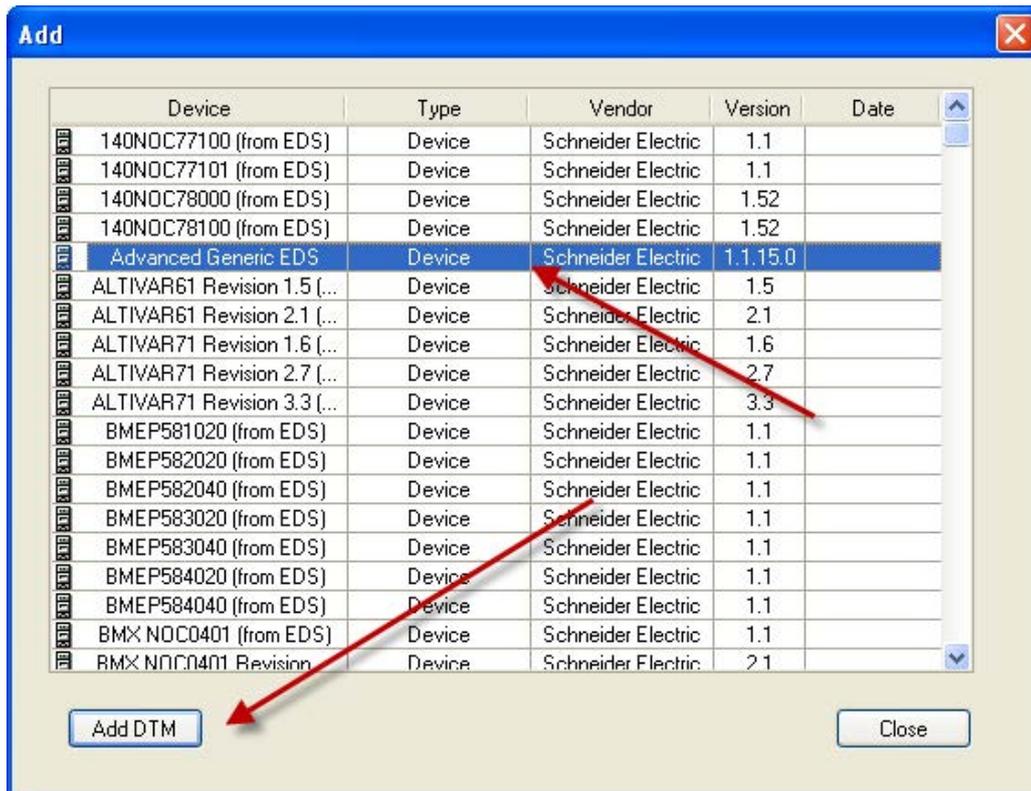
The screenshot displays the RSLogix 5000 interface. The 'Tag Properties - ProducedTag' dialog is the primary focus, showing the 'Type' dropdown set to 'Produced'. An arrow points from this dropdown to the 'Produced Tag Connection' dialog, which has 'Max Consumers' set to 1. Another arrow points from the 'Advanced...' button in the 'Produced Tag Connection' dialog to the 'Advanced Options' dialog. In the 'Advanced Options' dialog, the 'Unicast Connection Options' section is visible, with the checkbox 'Allow Unicast Consumer Connections' checked. The 'Properties' window on the right side of the interface shows the 'Produced Connection' settings, including 'Max Consumers' set to 2 and 'Unicast Connections' checked. The status bar at the bottom indicates 'RSLogix 5000 project saved successfully. Complete - 0 error(s), 0 warning(s)'.

### 3.4.3 Configuration for I/O Module with Schneider Electric Unity Pro XL

1. Open the Device Type Manager (DTM) Browser in Unity Pro and right click the controller, then click *Add*:



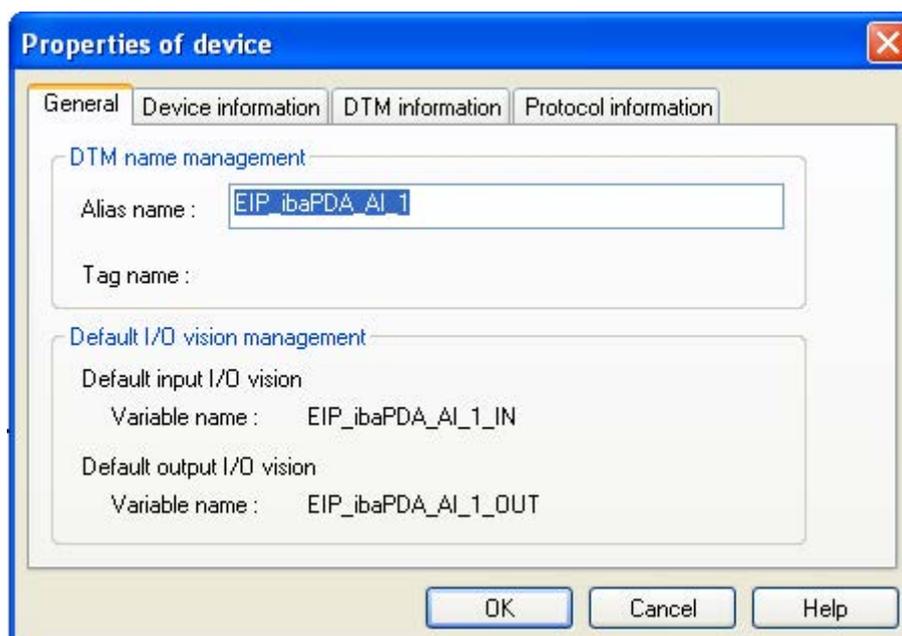
2. Select the “Advanced Generic EDS” device and click <Add DTM>.



3. The properties dialog appears:

Give the device an *Alias name* as indicated below.

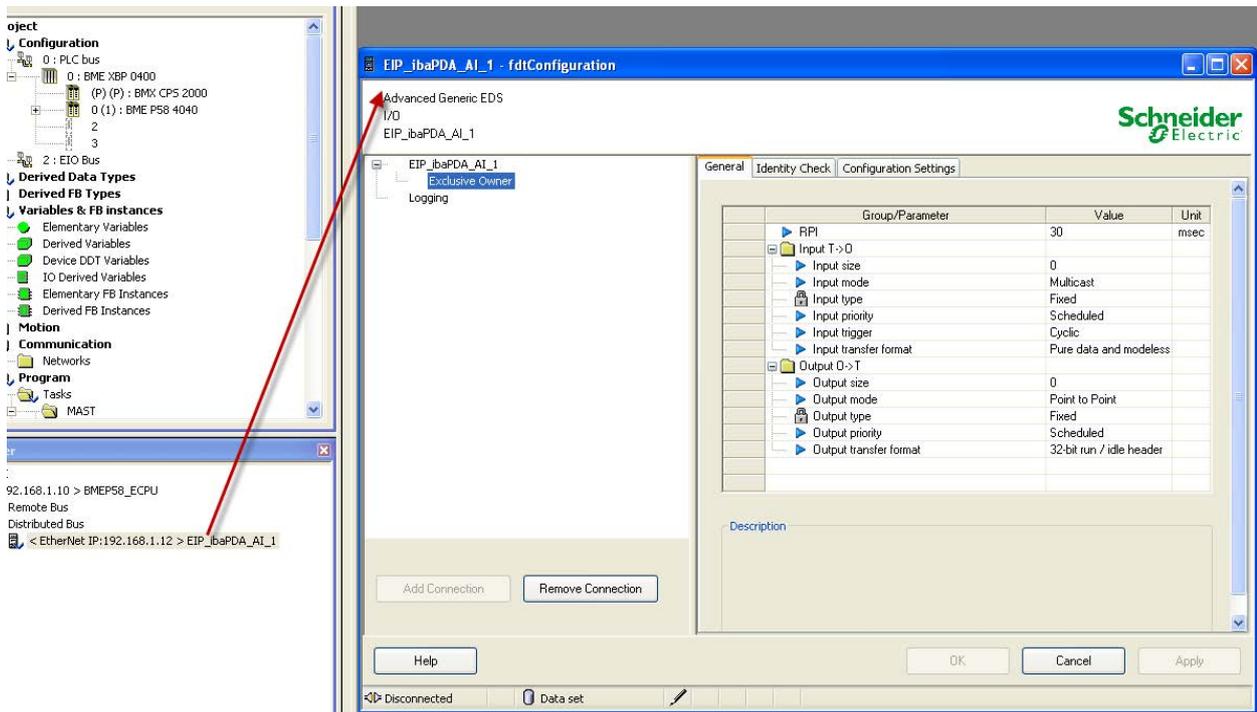
This name is used for the generation of I/O variable names. Choose a meaningful name that describes the device. The example below shows an Ethernet/IP module for sending data to *ibaPDA* with **Assembly Instance 1**.



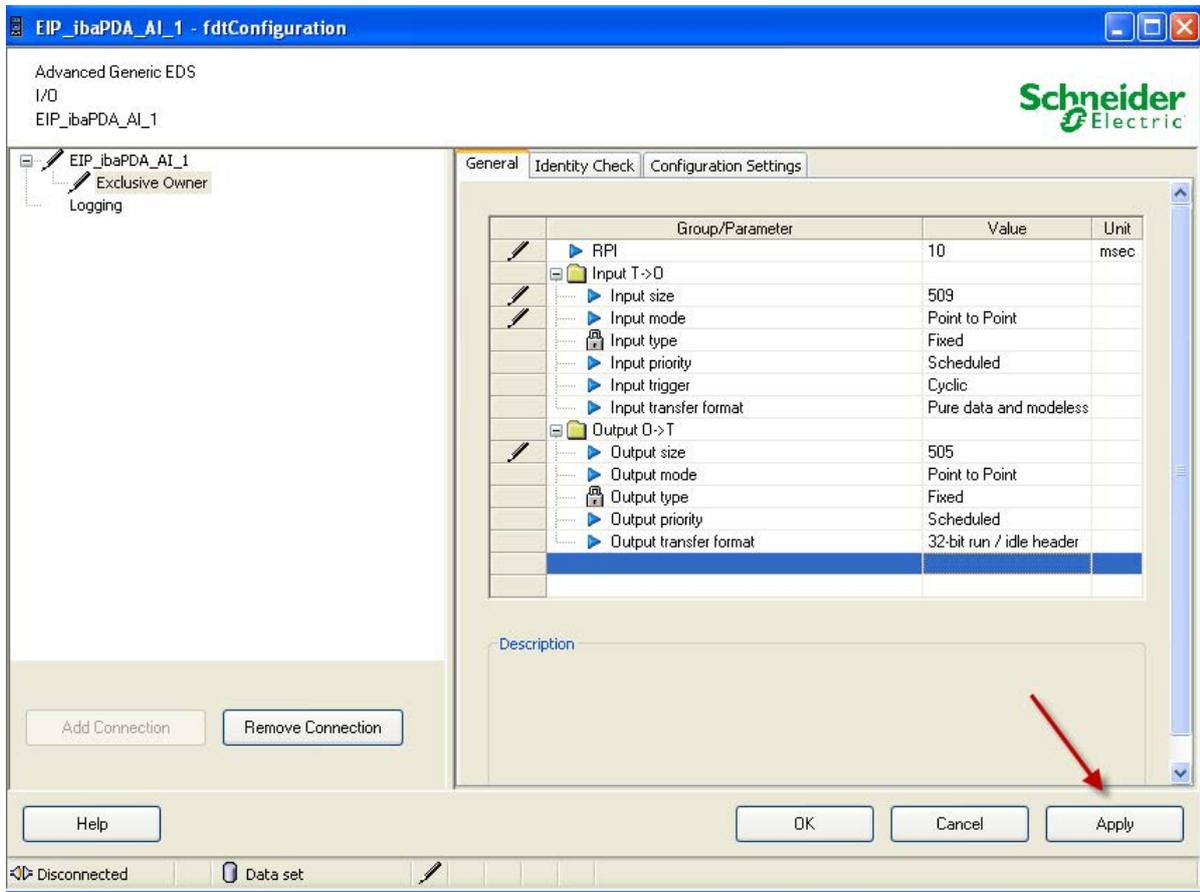
Click <OK> to add the device to the DTM Browser.

4. Double click the newly added device in the DTM Browser:

The Configuration dialog opens. Select the "Exclusive Owner" item as indicated below.



5. In the *General* tab, edit and configure the following settings:



- RPI:

Requested Packet Interval in msec. *ibaPDA* supports values between 1 msec and 1000 msec. Depending upon the PLC type and the programs running on this device, you might not use very small intervals (e.g. the lower limit on the M580 PLC is 2 msec).

- Input T-> O:

Input size: The number of bytes reserved for data from *ibaPDA*. The size should be minimal 4 byte to minimize network traffic if no input from *ibaPDA* is required. The maximum value is 509 bytes.

Input mode: *ibaPDA* supports unicast reply connections over Ethernet/IP. Hence, you may enable the option "Point to Point". Configuration of communication with unicast is a little bit easier because preventive measures against multicast side effects, such as IGMP Snooping and IGMP Query are not necessary.

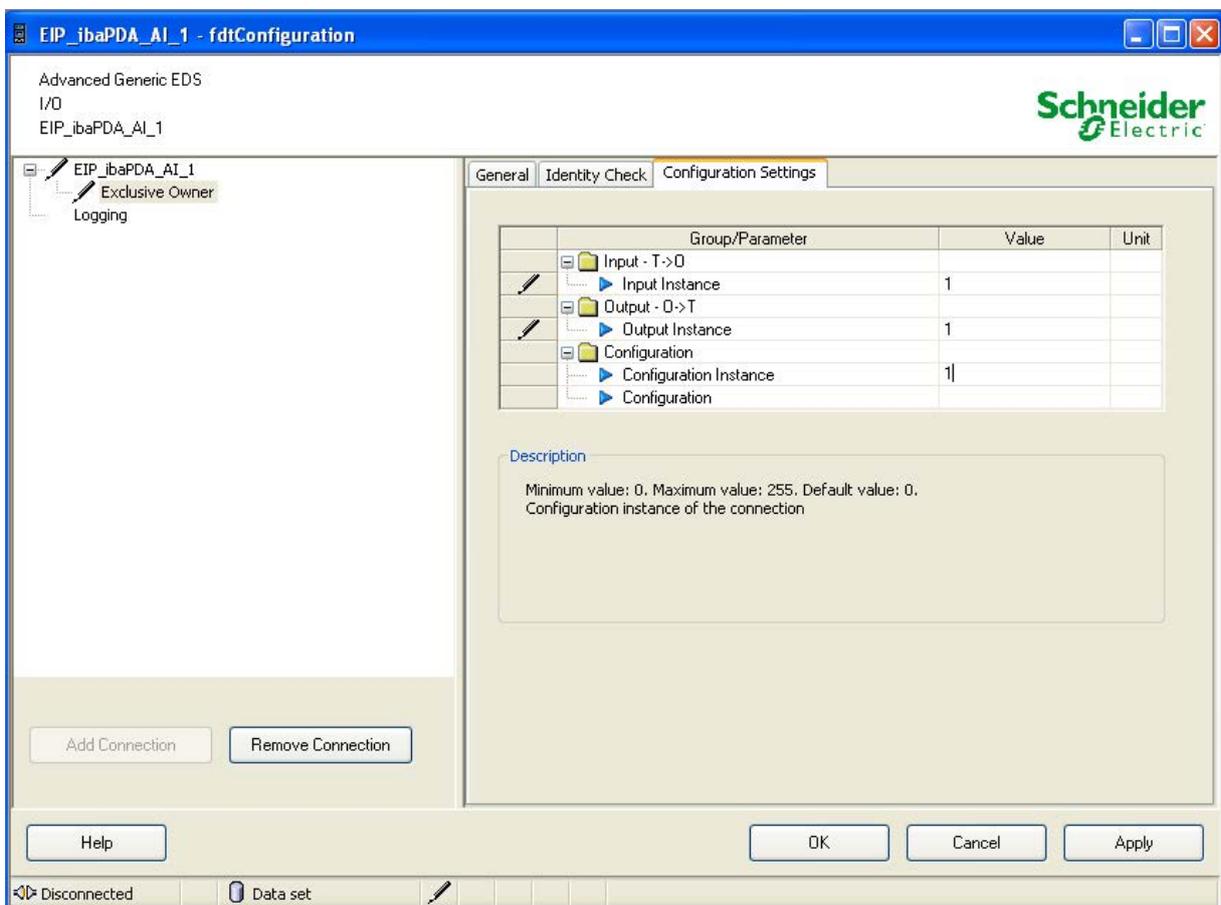
- Output O-> T:

Output size: The number of bytes reserved for data to *ibaPDA*. The maximum value is 505 bytes.

Output mode: *ibaPDA* supports unicast reply connections over Ethernet/IP. Hence, you may enable the option "Point to Point".

6. Click <Apply> to activate the settings.

7. In the *Configuration Settings* tab, configure the following settings:



- Input Instance:  
The device specific assembly number associated with input (T -> O) transmissions.
- Output Instance:  
The device specific assembly number associated with output (O -> T) transmissions.
- Configuration Instance:  
The device specific assembly number associated with device configuration settings.

---

**Note**

- The assembly instances for Input, Output and Configuration **must be identical**.
- The assembly instance numbers usually must be between 1 and 64.  
It is possible to accumulate up to 4 licenses resulting in a maximum of 256 connections. In this case the assembly instance can be up to 256.
- The assembly instance number is the same as the EtherNet/IP connection number in *ibaPDA*. This number must be unique for *ibaPDA*.

Do not configure two connections with identical assemblies to connect with the same *ibaPDA* host, even from different controllers!

---

**Note**

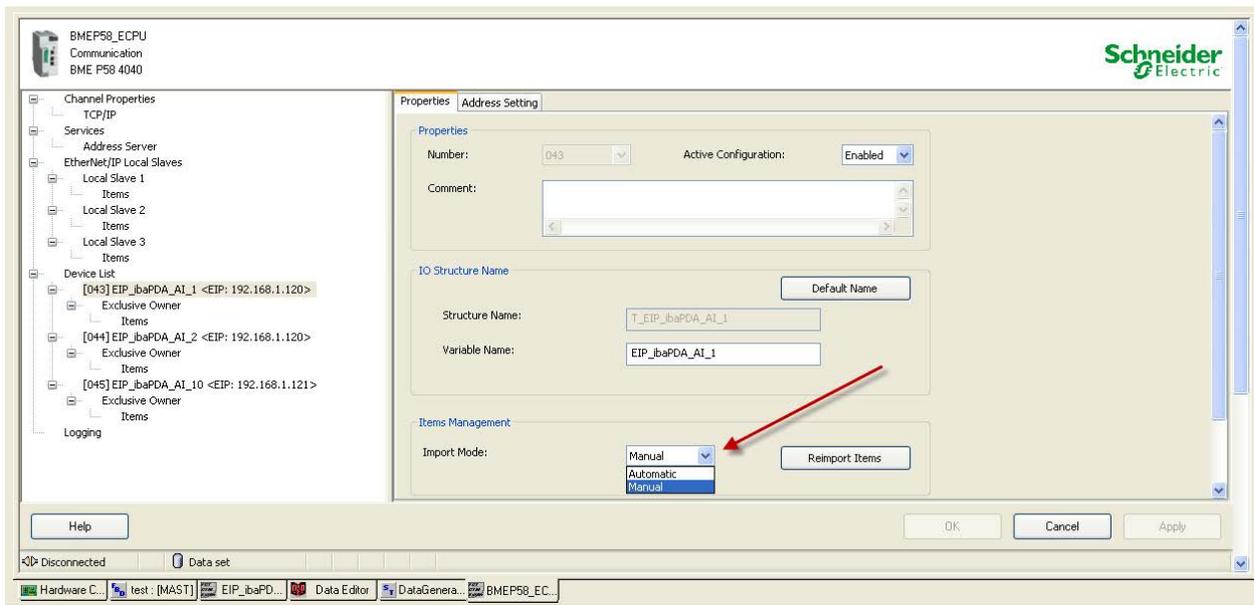
The size of inputs AND outputs is always defined in the PLC configuration (e.g. in RSLogix or the Unity Pro for Schneider PLCs).

*ibaPDA* checks the size reported by the ForwardOpen information and warns the user if the signal address offset + size of (Datatype) in *ibaPDA* is larger than the size defined on the PLC side.

The validation is aborted with an error in case the address offset + size of (Datatype) is larger than 1024 bytes for the inputs and 1004 for the outputs of *ibaPDA*.

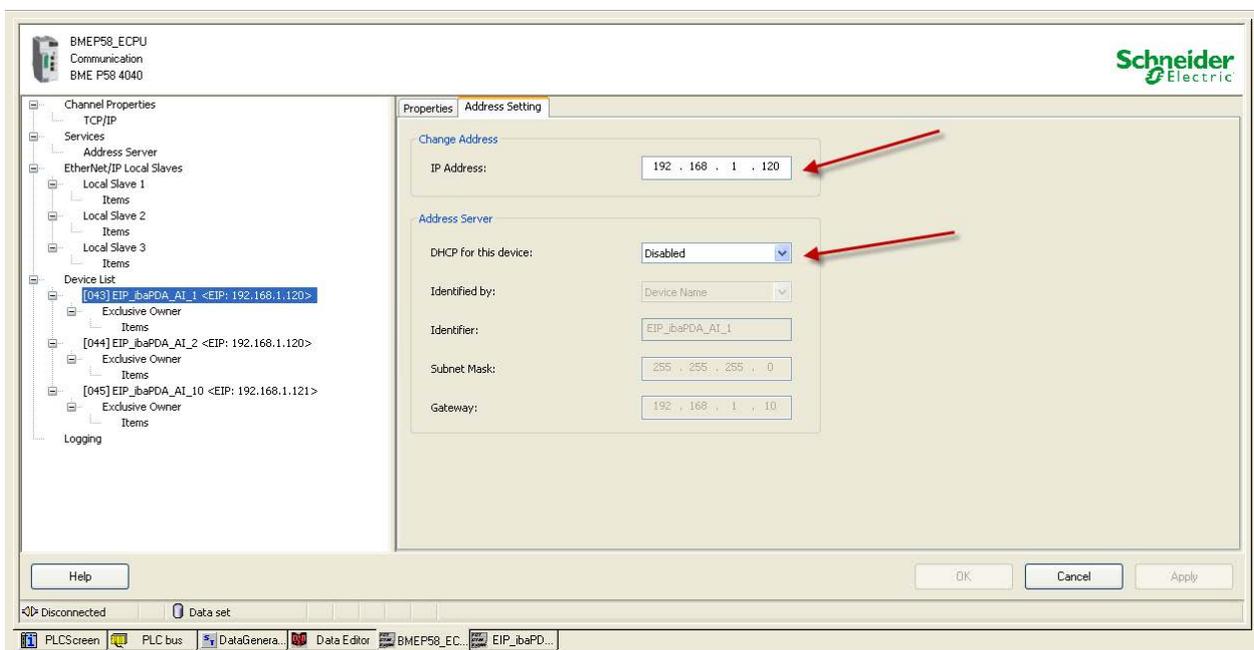
- 
8. Click <Apply> to activate the settings.
  9. Double click on the CPU in the DTM Browser.

The following dialog is shown.

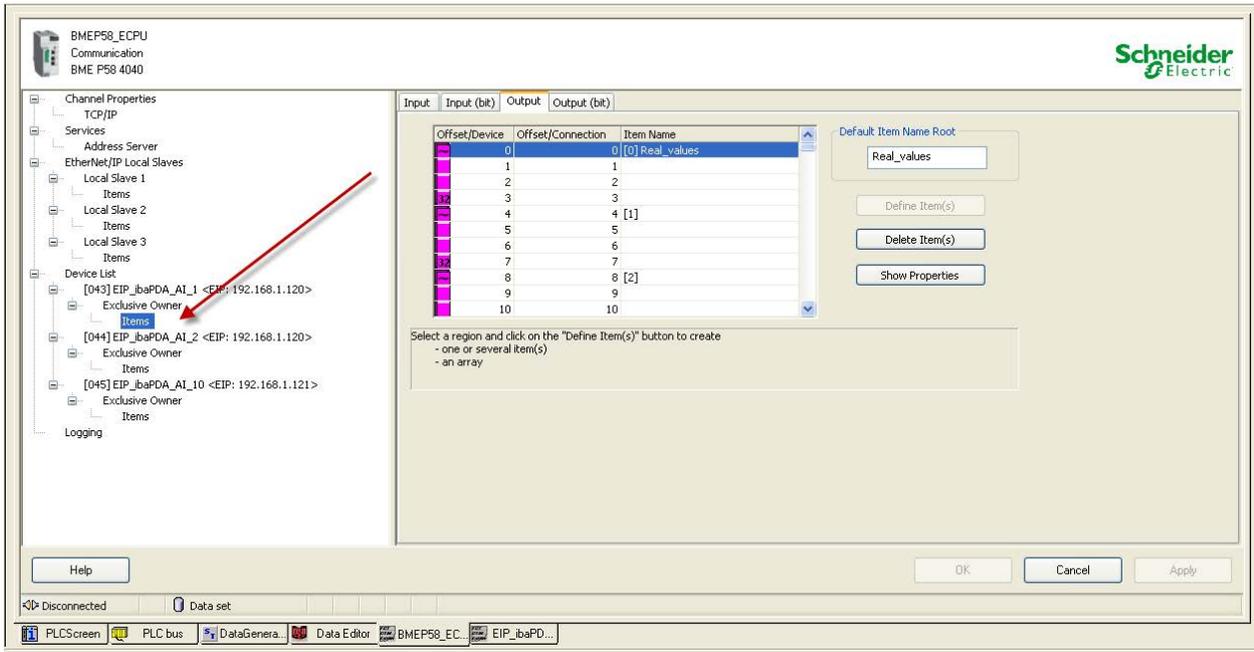


For each connection:

- Set the *Import mode* to “Manual” in the *Properties* tab as indicated in the picture above. I/O items are added when the device DTM is first added to Unity Pro. Thereafter, all I/O item edits are made manually in the device editor.
- In the *Address Setting* tab set the IP address to the IP address of the *ibaPDA* server. Also set the *DHCP for the device* to “Disabled”.



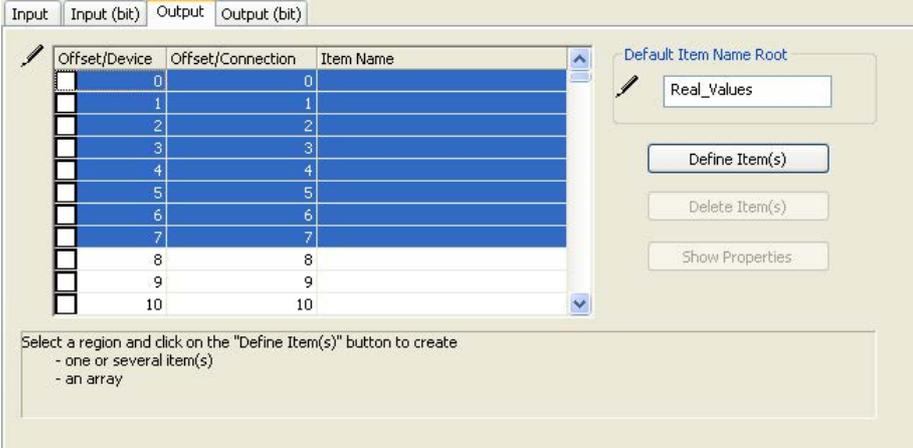
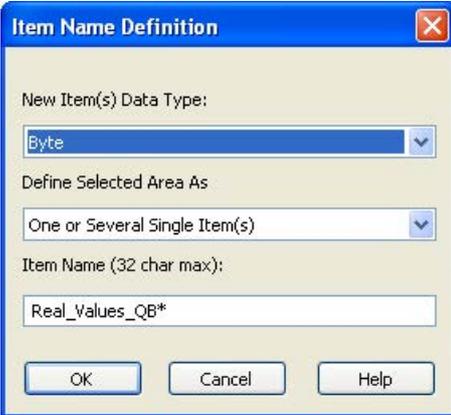
10. Select the items "Exclusive Owner".

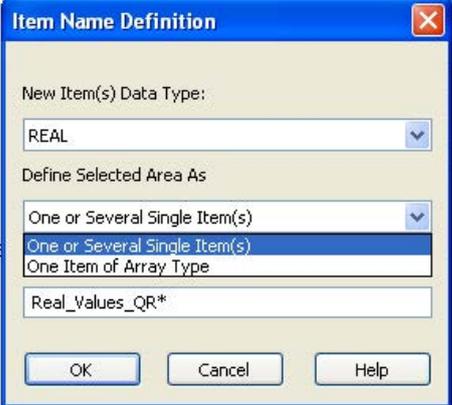
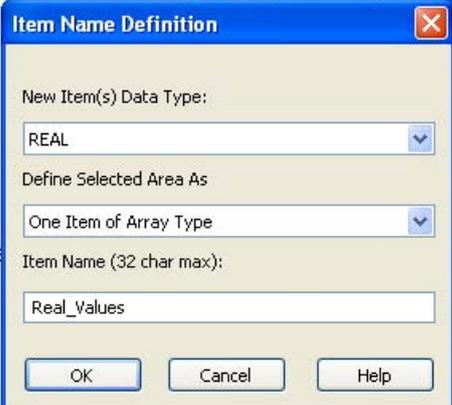
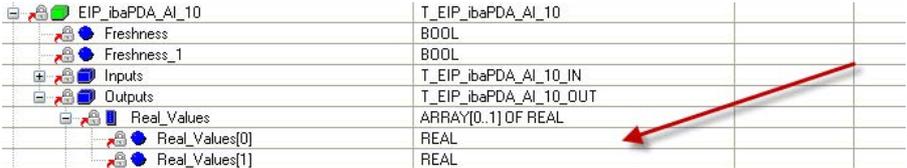


Here you can configure the I/O items:

- You can configure input and output items in groups of 1 or more single bits, 8-bit bytes, 16-bit words, 32-bit DWORDs, or 32-bit IEEE floating point values. The number of items you create depends on the data type and size of each item.
- Select the *Input* or *Output* tab depending on where you want to generate I/O items.

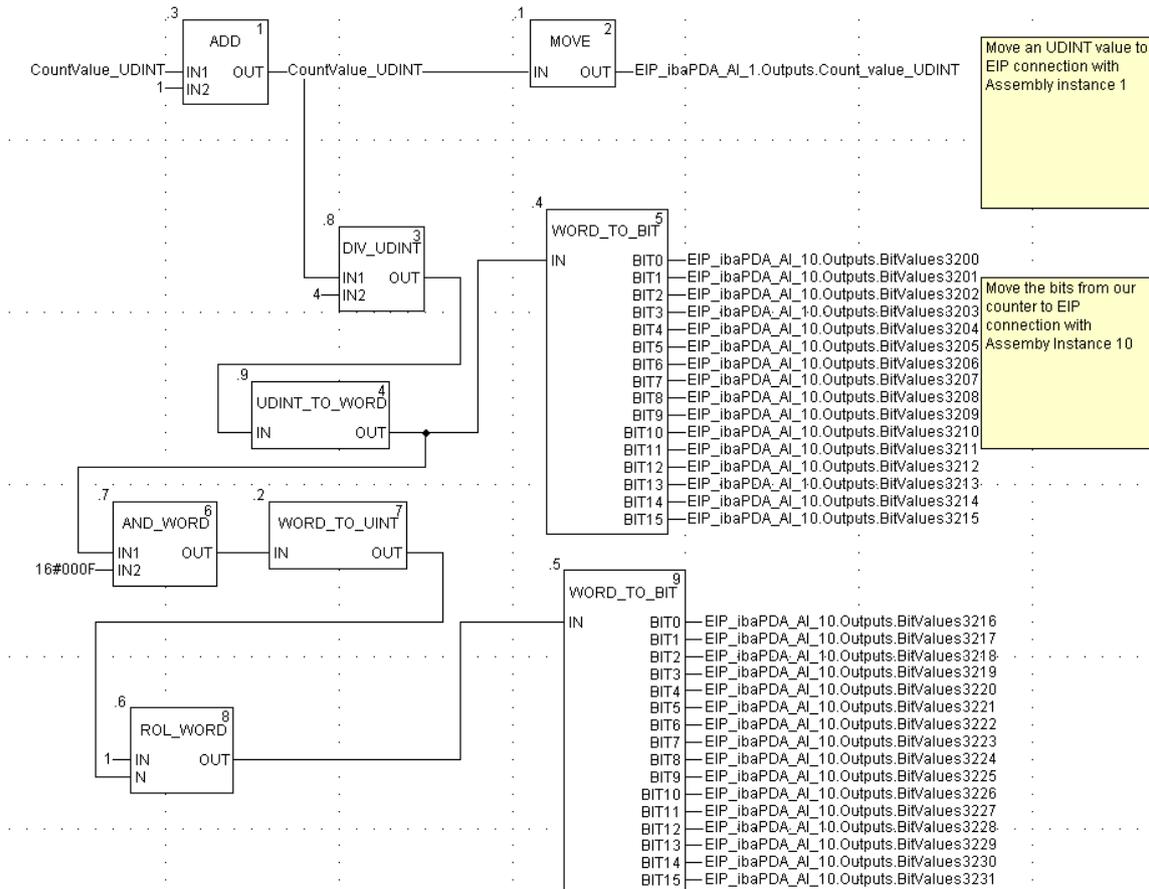
The following example shows the necessary steps to declare output items for sending data to *ibaPDA*:

Step	Action
1	<p>Click the <i>Output</i> tab to open the following page:</p>  <p>Note: In this example, each row represents a byte. Depending on the data type of the item you want to create, select multiple rows.</p> <p>Starting at the beginning of the table, select the first n rows: If you want to create e.g. 2 REAL values, select exactly 8 rows as shown above.</p>
2	<p>In the <i>Default Item Name Root</i> input box, type the item root name: e.g. “Real_Values”.</p>
3	<p>Click the &lt;Define Item(s)&gt; button.</p> <p>Result: The “<i>Item Name Definition</i>” dialog opens:</p> 

Step	Action																																				
4	<p>Select "REAL" as the <i>New Item(s) Data Type</i>.</p>  <p>Since we selected a multiple of 4 bytes, we have the choice now to handle these items as single items or as an array. Here these values are used as an array. The name of the item is specified with "Real_Values".</p>  <p>Click &lt;OK&gt; and the result is: 2 new items are created:</p> <table border="1" data-bbox="568 1339 1174 1648"> <thead> <tr> <th>Offset/Device</th> <th>Offset/Connection</th> <th>Item Name</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>[0] Real_Values</td> </tr> <tr> <td>1</td> <td>1</td> <td></td> </tr> <tr> <td>2</td> <td>2</td> <td></td> </tr> <tr> <td>32</td> <td>3</td> <td></td> </tr> <tr> <td>4</td> <td>4</td> <td>[1]</td> </tr> <tr> <td>5</td> <td>5</td> <td></td> </tr> <tr> <td>6</td> <td>6</td> <td></td> </tr> <tr> <td>32</td> <td>7</td> <td></td> </tr> <tr> <td>8</td> <td>8</td> <td></td> </tr> <tr> <td>9</td> <td>9</td> <td></td> </tr> <tr> <td>10</td> <td>10</td> <td></td> </tr> </tbody> </table>	Offset/Device	Offset/Connection	Item Name	0	0	[0] Real_Values	1	1		2	2		32	3		4	4	[1]	5	5		6	6		32	7		8	8		9	9		10	10	
Offset/Device	Offset/Connection	Item Name																																			
0	0	[0] Real_Values																																			
1	1																																				
2	2																																				
32	3																																				
4	4	[1]																																			
5	5																																				
6	6																																				
32	7																																				
8	8																																				
9	9																																				
10	10																																				
5	Click <OK> to close the items window.																																				
6	<p>Select <b>Build-&gt;Analyze</b> to save your edits and update the Device DDT variables.</p>  <table border="1" data-bbox="419 1760 1326 1928"> <thead> <tr> <th>Device</th> <th>Item Name</th> <th>Data Type</th> </tr> </thead> <tbody> <tr> <td>EIP_ibaPDA_AI_10</td> <td>T_EIP_ibaPDA_AI_10</td> <td></td> </tr> <tr> <td>Freshness</td> <td>BOOL</td> <td></td> </tr> <tr> <td>Freshness_1</td> <td>BOOL</td> <td></td> </tr> <tr> <td>Inputs</td> <td>T_EIP_ibaPDA_AI_10_IN</td> <td></td> </tr> <tr> <td>Outputs</td> <td>T_EIP_ibaPDA_AI_10_OUT</td> <td></td> </tr> <tr> <td>Real_Values</td> <td>ARRAY[0..1] OF REAL</td> <td></td> </tr> <tr> <td>Real_Values[0]</td> <td>REAL</td> <td></td> </tr> <tr> <td>Real_Values[1]</td> <td>REAL</td> <td></td> </tr> </tbody> </table>	Device	Item Name	Data Type	EIP_ibaPDA_AI_10	T_EIP_ibaPDA_AI_10		Freshness	BOOL		Freshness_1	BOOL		Inputs	T_EIP_ibaPDA_AI_10_IN		Outputs	T_EIP_ibaPDA_AI_10_OUT		Real_Values	ARRAY[0..1] OF REAL		Real_Values[0]	REAL		Real_Values[1]	REAL										
Device	Item Name	Data Type																																			
EIP_ibaPDA_AI_10	T_EIP_ibaPDA_AI_10																																				
Freshness	BOOL																																				
Freshness_1	BOOL																																				
Inputs	T_EIP_ibaPDA_AI_10_IN																																				
Outputs	T_EIP_ibaPDA_AI_10_OUT																																				
Real_Values	ARRAY[0..1] OF REAL																																				
Real_Values[0]	REAL																																				
Real_Values[1]	REAL																																				
	Repeat step 1-6 in the example to define all I/O items for output and input variables and input and output bits.																																				

Use the above generated I/O items in your program to send data to *ibaPDA*.

The examples below show code samples in FBD and Structured Text:



```

FOR i:= 0 TO 99 BY 1 DO

  i_real := udint_to_real(i);

  (* generate some Ramp signals on EIP connection with assembly ID 1 *)
  (*-----*)
  testRealValuesRampSignal[i]:= testRealValuesRampSignal[i]+ (i_real*100.0);
  (* limit the ramps by 100000000*)

  if (testRealValuesRampSignal[i] > 10000000.0)then
    testRealValuesRampSignal[i] := 0.0;
  end_if;

  EIP_ibaPDA_AI_1.Outputs.Real_values[i]:= testRealValuesRampSignal[i];

  (* generate some Sine wave signals on EIP connection with assembly ID 2*)
  (*-----*)
  testRealValuesSineSignal[i] := i_real * SIN ( 2.0 * pi_constant * (i_real / 10.0) * elapsed_time_in_sec_for_calc);
  EIP_ibaPDA_AI_2.Outputs.Real_values[i]:= testRealValuesSineSignal[i];

  (* generate some Cosine wave signals on EIP connection with assembly ID 10 *)
  (*-----*)
  testRealValuesCosineSignal[i] := i_real * COS (2.0 * pi_constant * (i_real / 10.0) * elapsed_time_in_sec_for_calc);
  EIP_ibaPDA_AI_10.Outputs.Real_values[i]:= testRealValuesCosineSignal[i];

END_FOR ;
    
```

**Other documentation**

Please see the Schneider Electric Unity Pro XL documentation for further information.

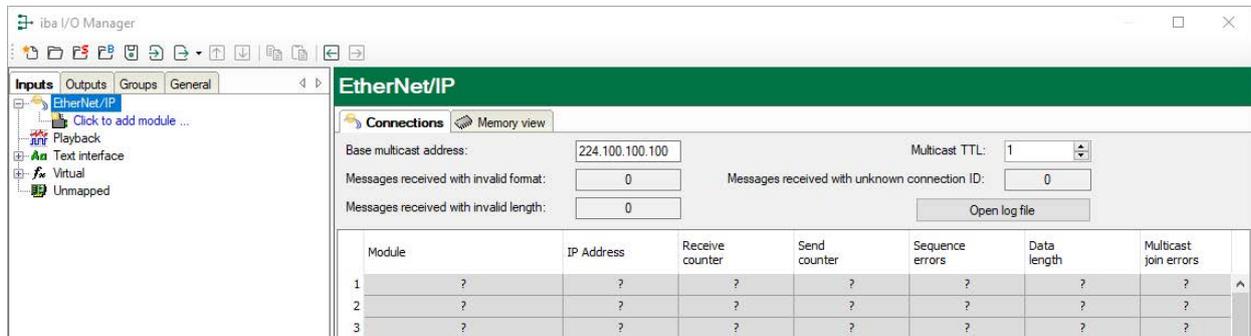
---

## 3.5 Configuration and engineering ibaPDA

Subsequently, the engineering for *ibaPDA* is described. If all system requirements are met, *ibaPDA* offers the *EtherNet/IP* interface in the interface tree of the I/O Manager.

### 3.5.1 General interface settings

The interface provides the following functions and configuration options.



#### Base multicast address

This setting applies to multicast communication only. The base multicast address is used as a destination address *ibaPDA* will send its response to. The last part of the address is replaced by the assembly instance number automatically. Generally, you can keep the default address issued by *ibaPDA*. This does not apply for the case if the address is already used or if changes in the router or firewall configuration are necessary.

#### Multicast TTL

The parameter TTL (Time-to-Live) is currently set to 1 by default. Each router between PLC and *ibaPDA* decrements the TTL value by 1 when a multicast package arrives. A router discards a package when the TTL value reaches 0 (zero). You only need to set a value greater than 1 when the PLC is behind one or more routers.

You can easily generate modules for these connections. Right click on the EtherNet/IP interface icon in the tree view and choose the context menu "Autodetect". Alternately you can generate these modules manually without an existing connection. Click on the item in the tree on the blue line *Click to add module...*

The number of the connection is equal to the configured assembly instance for the ForwardOpen. If you move the mouse cursor over an Ethernet IP connection in the tree view, you will see some additional information, such as connection length and IP addresses, in a tool tip window.

#### Message counters

The counters for messages with invalid format/invalid length/unknown connection ID are for display only.

#### <Open log file>

If connections to controllers have been established, all connection specific actions are recorded in a text file. Using this button, you can open and check this file. In the file system on the hard disk, you find the log files of the *ibaPDA* server (...\[ProgramData\iba\ibaPDA\Log](#)).

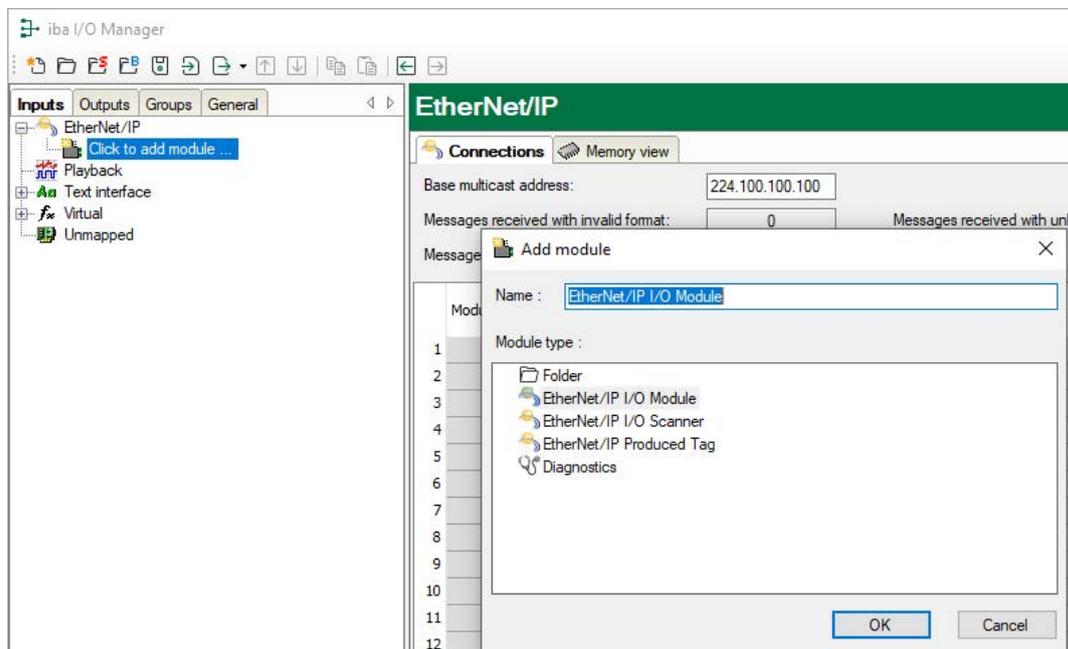
The file name of the current log file is [InterfaceLog.txt](#); the name of the archived log files is [InterfaceLog\\_yyyy\\_mm\\_dd\\_hh\\_mm\\_ss.txt](#).

For further information, please see [↗ Connection table](#), page 45

### 3.5.2 Adding a module

#### Procedure

1. Click on the blue command *Click to add module...* located under each data interface in the *Inputs* or *Outputs* tab.
2. Select the desired module type in the dialog box and assign a name via the input field if required.
3. Confirm the selection with <OK>.



#### Module types

You can add the following module types to the interface:

EtherNet/IP I/O Module

EtherNet/IP I/O Scanner

EtherNet/IP I/O Produced Tag

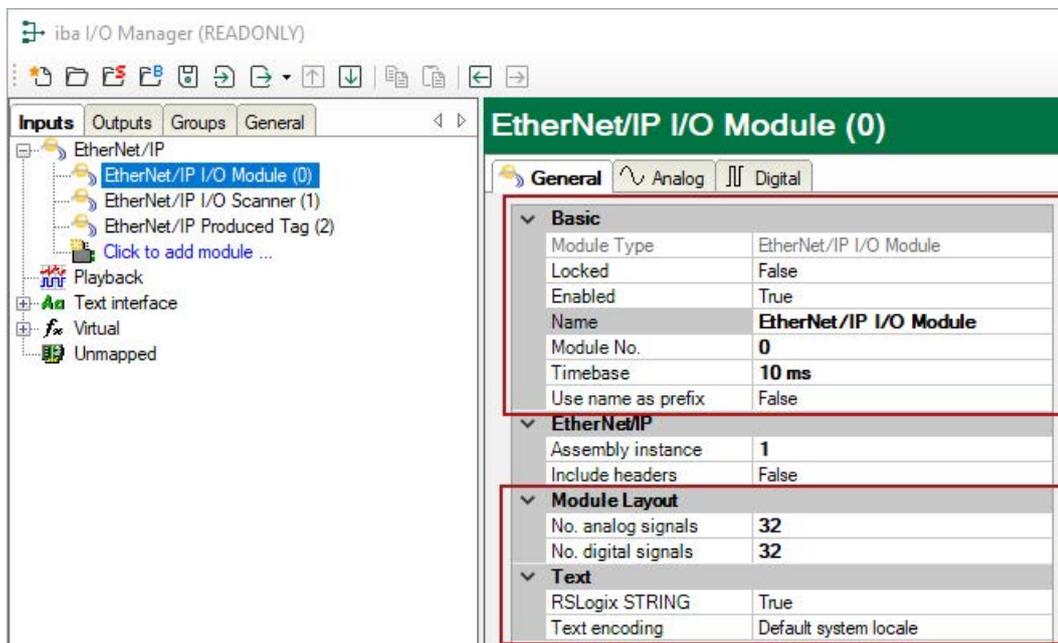
For information about the EtherNet/IP module types, see [↗ Types of ibaPDA modules](#), page 10

For information about the module type Diagnostic modules, see [↗ Diagnostic modules](#), page 49

### 3.5.3 General module settings

To configure a module, select it in the tree structure.

All modules have the following setting options.



The red framed settings are the same for all 3 types.

#### Basic settings

##### Module Type (information only)

Indicates the type of the current module.

##### Locked

You can lock a module to avoid unintentional or unauthorized changing of the module settings.

##### Enabled

Enable the module to record signals.

##### Name

You can enter a name for the module here.

##### Module No.

This internal reference number of the module determines the order of the modules in the signal tree of *ibaPDA* client and *ibaAnalyzer*.

##### Timebase

All signals of the module are sampled on this timebase.

##### Use name as prefix

This option puts the module name in front of the signal names.

## Module Layout

### No. of analog signals/digital signals

Define the number of configurable analog and digital signals in the signal tables. The default value is 32 for each. The maximum value is 1000. The signal tables are adjusted accordingly.

### Text

#### RSLogix STRING

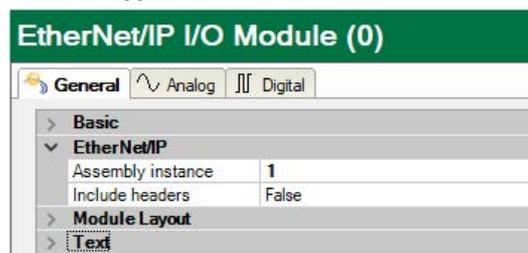
Enable this option ("True") if signals with a STRING data type refer to RSLogix strings.

#### Text encoding

You can select the type of text coding or the code page here for a correct interpretation and display of the received text data. Available for selection are:

- System locale (default): Coding according to the Windows system settings
- Western European (1252): 8-bit coding, including ASCII and Western European umlauts, special characters
- UTF-8: Unicode, for universal characters, including ASCII, Asian languages, etc.

### EtherNet/IP – I/O Module module type



#### Assembly instance

Must be set to the assembly instance number as defined in the PLC for this connection.

#### Note



There are some restrictions for the assembly instance number with regard to the EtherNet/IP interface licenses.

The assembly instance must be within the range...

1 to 64 for the first license (basic license *ibaPDA-Interface-EtherNet/IP*),

65 to 128 for the second (1. *one-step-up-Interface-EtherNet/IP-license*),

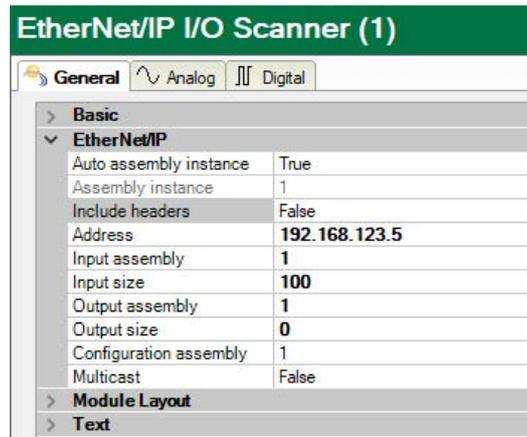
129 to 192 for the third (2. *one-step-up-Interface-EtherNet/IP-license*),

193 to 255 for the fourth (3. *one-step-up-Interface-EtherNet/IP-license*).

#### Include headers

If this option "True" is enabled, not only "pure" data will be measured. *ibaPDA* has also access to the header information of the UDP-CIP-Packet. This makes only sense for debugging purposes.

## EtherNet/IP – I/O Scanner module type



### Auto assembly instance

Set this to "True" to let *ibaPDA* automatically assign a free assembly instance to this module. Set it to "False" if you want to manually assign an assembly instance. In this case you have to make sure that the assembly instance isn't used by another module.

### Assembly instance

Not applicable for this module (read only)

### Include headers

If enabled, not only the „pure“ data will be measured, also the header information of the UDP-CIP-packet will be accessible by *ibaPDA*. This makes only sense for debugging purposes.

### Address

Enter here the IP address of the device, this module should connect to.

### Input assembly

Enter the number of the device's input assembly instance you want to read data from.

### Input size

Enter the size (in bytes) of the complete input assembly instance.

### Output assembly

Enter the number of the device's output assembly instance you want to write data to.

### Output size

Enter the size (in bytes) of the complete output assembly instance.

### Configuration assembly

Enter the number of the device's configuration assembly instance. If you do not know it, then leave it at 1.

### Multicast

If you set this option to "True", the input data will be received via multicast instead of unicast.

**Note**

In case an I/O adapter or device has inputs and outputs but you only want to get inputs, then you should enter a special assembly number into the *Output assembly* field. This assembly number – usually referred to as "Input only assembly" - should be mentioned in the vendor's documentation. For WAGO. e.g., it is "198". For *Output size* then enter "0".

**EtherNet/IP – Produced Tag module type**

EtherNet/IP	
Auto assembly instance	True
Assembly instance	1
Include headers	False
Address	192.168.123.5
Use routing	False
Slot no.	0
Produced tag	
Data size	100

**Auto assembly instance**

Set this to "True" to let *ibaPDA* automatically assign a free assembly instance to this module. Set it to "False" if you want to manually assign an assembly instance. In this case you have to make sure that the assembly instance isn't used by another module.

**Assembly instance**

Not applicable for this module (read only).

**Include headers**

If this option "True" is enabled, not only the „pure“ data will be measured, also the header information of the UDP-CIP-packet will be accessible by *ibaPDA*. This makes only sense for debugging purposes.

**Address**

Enter here the IP-address of the CPU this module should connect to.

**Use routing**

Enable this option if the target CPU cannot be reached directly by *ibaPDA*. The access can be established through different bus systems. If the target CPU can be connected directly with *ibaPDA*, disable this option (=False).

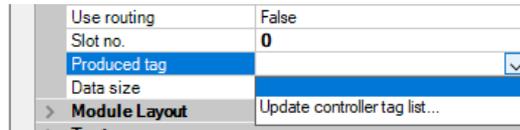
**Slot no./Path**

If routing is disabled, enter the slot no. of the CPU, which should be connected with *ibaPDA*. If routing is enabled, enter the connection path, which describes how to reach the CPU. It consists of different segments separated by a comma. Each segment consists of 2 parts also separated by a comma: the port and the destination address on the port. The port can be a Backplane, A, B or a number. The destination address can be a backplane slot, a DH+ address, ControlNet address or an IP address.

Example: Backplane,1,A,192.168.200.154,Backplane,0

### Produced tag

After you have entered address and slot number you may load the controller tags from the CPU by selecting *Update controller tag list...* from the produced tag drop-down list.



If you do so, *ibaPDA* will connect to the CPU and read out the controller tags with their structures and fill in the dropdown list. You can then select a tag. After you have selected a tag the *Data size* property will be filled in automatically with the size of the tag.

You are also asked if *ibaPDA* should update the analog and digital signals to correspond with the structure of the tag. If you select *Yes*, then the number of analog and digital signals is changed.

### Data size

Data size of the selected tag. It will be filled in automatically after the tag has been selected.

## 3.5.4 Signal configuration

Basically, the principle of signal configuration is the same for all modules.

In the tabs *Analog* and *Digital* of each module in the I/O Manager, you can assign name, comment and if required unit and scaling factor to the signals and enable or disable them. It is essential to enter address and data type.

Only in the case of Produced Tag the analog and digital signals to be measured are filled in automatically if you select *Yes* when being asked for updating the analog and digital signals after the controller tag list has been updated, see [General module settings](#), page 30

### Analog and digital tab

Name	Unit	Gain	Offset	Address	DataType	Active	Actual	+
0 TestStructArrayTag[0].First		1	0	0	FLOAT	<input checked="" type="checkbox"/>	0	
1 TestStructArrayTag[0].Second		1	0	4	FLOAT	<input checked="" type="checkbox"/>	0	
2 TestStructArrayTag[0].Third		1	0	8	INT	<input checked="" type="checkbox"/>	0	
3 TestStructArrayTag[0].FourthNonAligned		1	0	12	DINT	<input checked="" type="checkbox"/>	0	
4 TestStructArrayTag[0].dint_tussen_bits		1	0	20	DINT	<input checked="" type="checkbox"/>	0	
5 TestStructArrayTag[1].First		1	0	28	FLOAT	<input checked="" type="checkbox"/>	0	
6 TestStructArrayTag[1].Second		1	0	32	FLOAT	<input checked="" type="checkbox"/>	0	
7 TestStructArrayTag[1].Third		1	0	36	INT	<input checked="" type="checkbox"/>	0	
8 TestStructArrayTag[1].FourthNonAligned		1	0	40	DINT	<input checked="" type="checkbox"/>	0	
9 TestStructArrayTag[1].dint_tussen_bits		1	0	48	DINT	<input checked="" type="checkbox"/>	0	

### Address

The address space is depending on the data type. Hence, an adjustment of address entries may be necessary after change of data types.

The digital signals are addressed via the *Address* and *Bit no.* (0 – 31) columns.

**Data Type (analog signals only)**

In the fields of this column you can select the data type of each signal. Just click in the corresponding field and select the data type from the drop-down list.

The following data types are available:

ibaPDA Data Type	ControlLogix Data Type	Unity PRO Data Type
BYTE / SINT (8 bit)	SINT	Byte
INT / WORD (16 bit)	INT	INT / WORD / UINT
DINT / DWORD (32 bit)	DINT	DINT / DWORD / UDINT
FLOAT (32 bit)	REAL	REAL
DOUBLE <sup>1)</sup> (64 bit)	n.a	n.a
STRING (32 Bit)		

<sup>1)</sup> There are other PLC systems on the market, e.g. OMRON, which support a DOUBLE-compatible datatype like LREAL or LWORD.

**Note**

In case of a produced tag module the signal names, addresses and data types can be loaded automatically from the CPU.

**Bit no. (digital signals only)**

With the digital signals you have the possibility to get 32 single bits out of a double integer. After entering the address, the "Bit no." is automatically increased by 1, from 0 to 31, then increase of address by 4.

## 3.6 Configuration of ibaPDA output modules

If all system requirements are met, *ibaPDA* offers the *EtherNet/IP* interface in the signal tree of the *Outputs* tab. There is no need to add the interface manually.

Add the output modules in the same way as input modules.

Output signals are available for EtherNet/IP module types "I/O Module" and "I/O Scanner" only.

### 3.6.1 General module settings ibaPDA output modules

If you want to configure an output module, mark the module in the tree structure of the *Outputs* tab.

The screenshot shows the 'iba I/O Manager' application. On the left, a tree view under the 'Outputs' tab shows the hierarchy: EtherNet/IP > EtherNet/IP I/O Module (0) > EtherNet/IP I/O Scanner (1). The right pane displays the configuration for 'EtherNet/IP I/O Scanner (1)' with the 'General' tab selected. The configuration is organized into sections: Basic, EtherNet/IP, Module Layout, and Text.

Basic	
Module Type	EtherNet/IP I/O Scanner
Locked	False
Enabled	True
Name	<b>EtherNet/IP I/O Scanner</b>
Module No.	<b>1</b>
Calculation timebase	<b>10 ms</b>
Minimum output timebase	50 ms
Use name as prefix	False

EtherNet/IP	
Auto assembly instance	True
Assembly instance	1
Include headers	False
Address	<b>192.168.123.5</b>
Input assembly	<b>1</b>
Input size	<b>100</b>
Output assembly	<b>1</b>
Output size	<b>0</b>
Configuration assembly	1
Multicast	False

Module Layout	
No. analog output signals	<b>32</b>
No. digital output signals	<b>32</b>

Text	
RSLogix STRING	True
Text encoding	Default system locale

**Name**  
The name of the module.

The parameters are almost identical to those of the input module, see ↗ *General module settings*, page 30

Consider the following differences when it comes to the settings of the input modules:

#### Calculation timebase

Timebase (in ms) used for the calculation of the output values.

The calculation timebase is not the same as the output timebase with which the values are output!

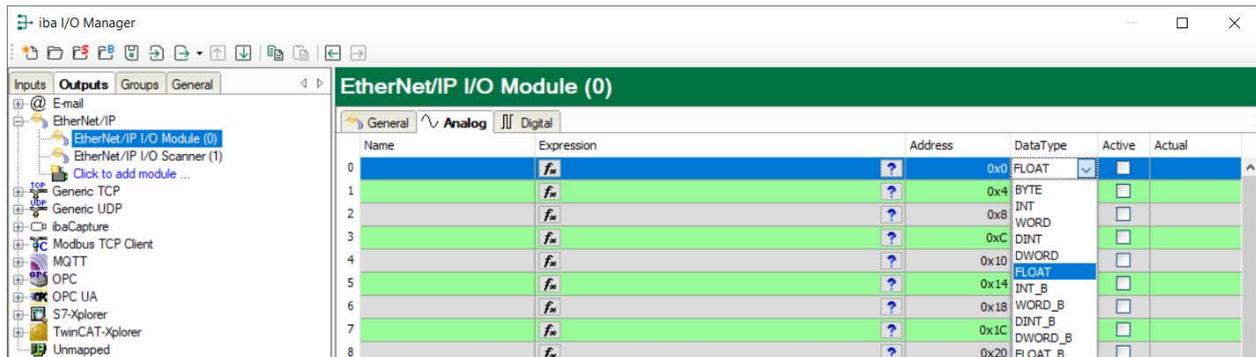
### Minimum output timebase

Timebase with which the outputs can be updated as quickly as possible.

The value is acquired automatically by the system based on the current I/O configuration and is only displayed here. The output timebase results from the smallest common multiple of all module timebases or is at least 50 ms.

### 3.6.2 Signal configuration

You can define the number of analog and digital outputs. For each output signal you have to define its value via an expression, its address and its data type.



Name	Expression	Address	DataType	Active	Actual
0	fx	0x0	FLOAT	<input type="checkbox"/>	
1	fx	0x4	BYTE	<input type="checkbox"/>	
2	fx	0x8	INT	<input type="checkbox"/>	
3	fx	0xC	WORD	<input type="checkbox"/>	
4	fx	0x10	DINT	<input type="checkbox"/>	
5	fx	0x14	DWORD	<input type="checkbox"/>	
6	fx	0x18	INT_B	<input type="checkbox"/>	
7	fx	0x1C	WORD_B	<input type="checkbox"/>	
8	fx	0x20	DWORD_B	<input type="checkbox"/>	

#### Note



*ibaPDA* processes output signals with lower priority compared to input signals in an update cycle not faster than 50 ms, depending on the I/O configuration.

## 3.7 Setting up multiple IP addresses for ibaPDA

#### Note



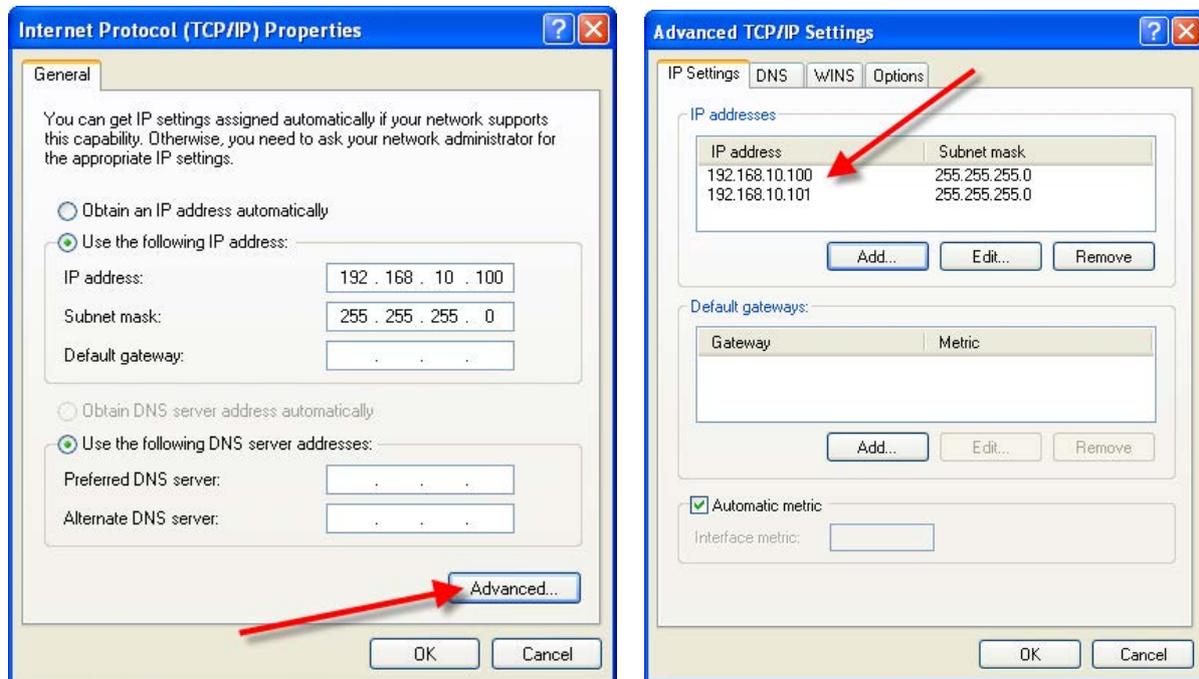
Only relevant for connection with Rockwell PLCs.

*ibaPDA* works as a server which is listening for clients requesting to connect.

*ibaPDA* has been configured to listen always on the configured TCP/IP port for any Ethernet/IP connection request. So the Rockwell/AB PLC will be the master (initiator) of communication. It means that the PLC will establish the link and *ibaPDA* will be, as we call, a passive node. The status of every Ethernet/IP connection will be displayed on the overview in *ibaPDA* I/O Manager.

It is important to pay attention that IP addresses are only allocated once. Usually there can be more than one link from the same target IP address (PLC). Since the Generic Ethernet Module is not able to make multiple connections with the same destination IP address, the network interface card (NIC) in the *ibaPDA* system has to be configured for multiple IP addresses on the same NIC. From the point of view of each individual PLC, a different IP address has to be specified for each Generic Ethernet Module.

This method involves assigning multiple IP addresses to a single NIC. This is accomplished through the Windows® Network control panel. After you have opened the properties dialog of the LAN connection which is used for Ethernet/IP, select (highlight) Internet Protocol (TCP/IP) and then click on the <Properties> button. Next click on the button <Advanced>. You will see a dialog box as depicted below. From here, you can add further IP addresses to any Network-Interface-Card (NIC) in your system.



Each Ethernet/IP connection should be considered as an *ibaPDA* module and it needs a unique module number ID which, in fact, is automatically given by *ibaPDA*. This ID is automatically allocated by the *ibaPDA*. The reference for the module/connection assignment is the assembly instance. The assembly instance has been entered in the configuration dialog of the Generic Ethernet Module in RSLogix.

The same assembly instance number must be entered in the *ibaPDA* I/O Manager, in the *Assembly instance* field on the *General* tab of the module in question.

## 4 Troubleshooting and diagnostics

### 4.1 Ethernet switch features important for Ethernet/IP

The proper selection of switches to be used in real-time (I/O) Ethernet/IP networks is critical. There are several features that are very important and can provide the appropriate infrastructure for your application. The following features need to be considered:

#### Required:

- Full-duplex capability on all ports
- IGMP Snooping
- Port Mirroring

---

#### Note



Choosing the right switch is particularly very important if the Ethernet/IP controller uses IP multicast messages. So iba recommends using unicast messages if supported by the controller.

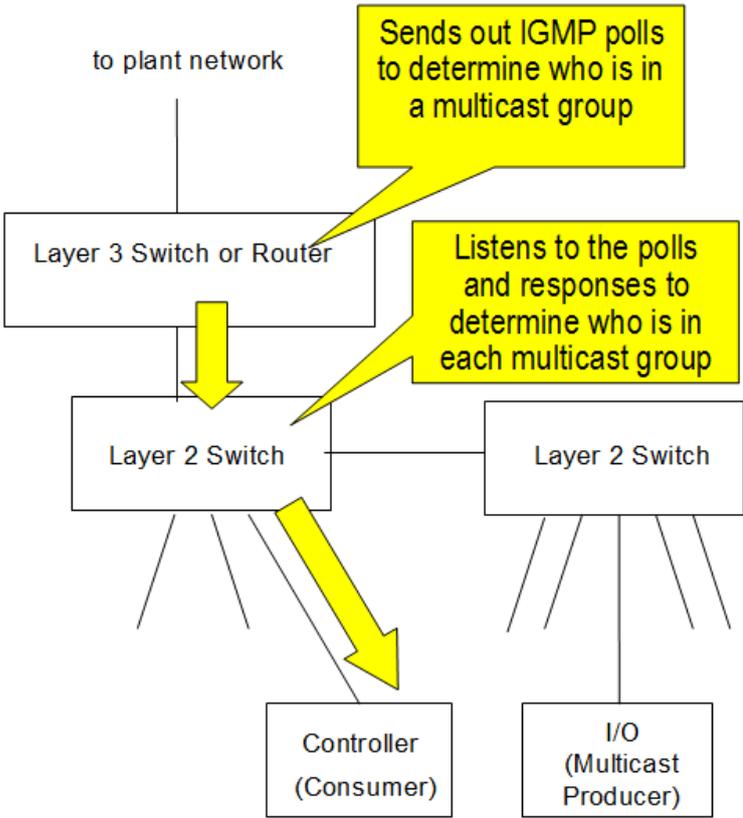
---

- Full-duplex capability on all ports

Full duplex capability eliminates collisions on the wire due to the separate transmit and receive channels for each device. Combined with the speed of switches available today, delays related to collisions or traffic in the switch can be made negligible. The end result is that you can achieve a high degree of determinism with an Ethernet/IP network and it works well for I/O control.

- IGMP Snooping

IGMP snooping limits the flooding of multicast traffic by dynamically configuring switch ports so that multicast traffic is forwarded only to ports associated with a particular IP multicast group.

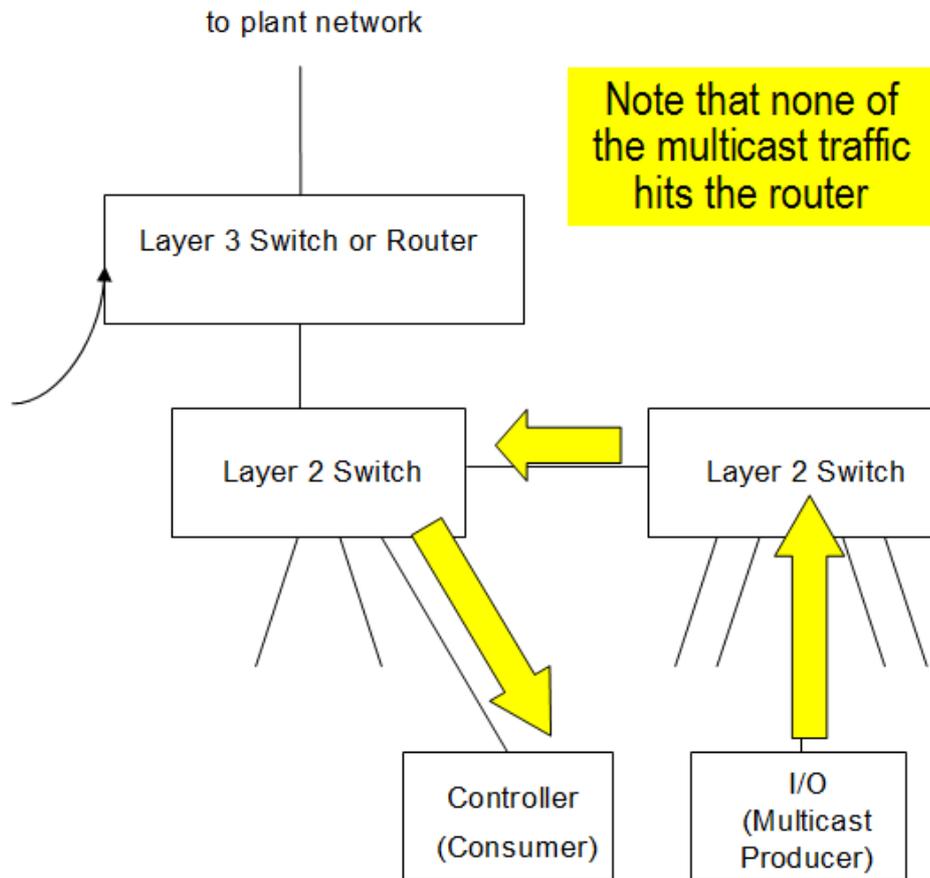


A layer 2 switch that supports IGMP snooping needs a router (which can be a layer 3 switch) to send IGMP requests from whose it learns which devices are members of the multicast group (IGMP Querier).

**Note**



Some industrial layer 2 switches support IGMP snooping without the requirement for a router or layer 3 switch to be present to send out the IGMP polls.



■ Port Mirroring:

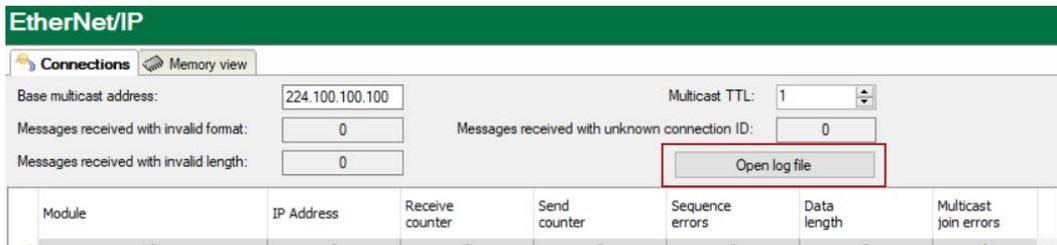
Port Mirroring refers to the ability to direct a duplicate of the frames being transmitted on one port to another port. Therefore a traffic analyzer can connect to a switch and monitor the traffic on a specific port. Without Port Mirroring, an analyzer is not able to recognize frames on other ports.

Traffic analyzers (like the freeware Wireshark) are used extensively by people who support Ethernet networks. Therefore, it is essential that a switch is selected that supports Port Mirroring so that a traffic analyzer will function correctly on the network.

## 4.2 Conflict with other Ethernet/IP related programs

A common problem of the *ibaPDA* server systems which use the EtherNet/IP-interface is the conflict with other programs which also use the EtherNet/IP Port 44818.

If another service like Rockwell RSLinx Service, starts before the *ibaPDA* server then this device listens on Port 44818 before *ibaPDA* service has the possibility to do this. To check this fact, open the EtherNet/IP-specific log file in the I/O manager of the *ibaPDA*, as shown below:



Following error message will appear in the log file:

```
7/23/2014 5:06:49 PM.227 [14: EthernetIP listening TCP thread] : **** ER-
ROR **** : Creating Ethernet/IP listening socket : Only one usage of each
socket address (protocol/network address/port) is normally permitted
```

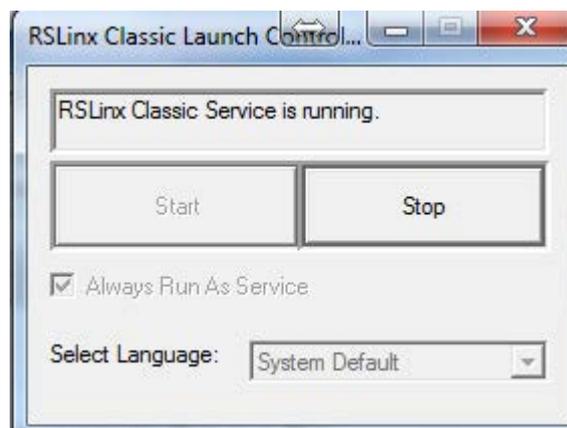
### Note



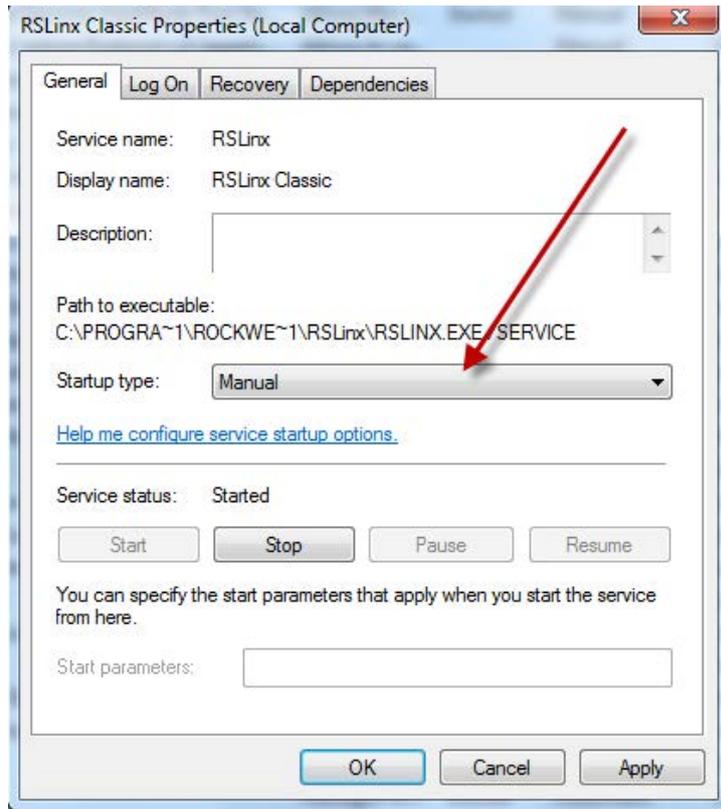
A way to root cause this is by using the **Netstat -b -a** command in a dos window and looking for the 44818 port :

```
TCP 0.0.0.0:44818 note-car1o7:0 LISTENING
[RSLINX.EXE]
```

If you see this error message and RSLinx is installed in the *ibaPDA* server, uninstall RSLinx or stop the RSLinx service via the launch console:



If RSLinx is really needed on the *ibaPDA* server, another solution is to change the startup type of the RSLinx Service to "Manual".

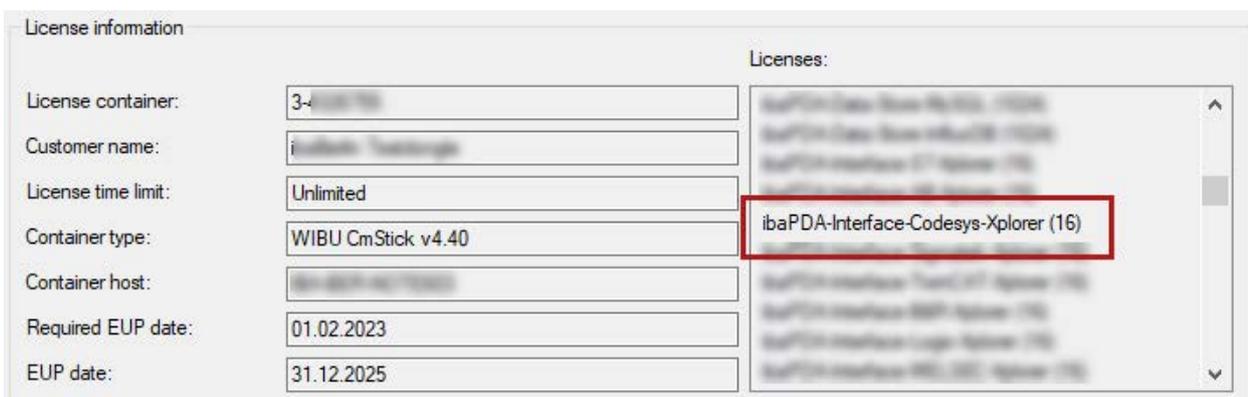


This will prevent RSLinx being started before *ibaPDA* server starts at boot time.

### 4.3 License

If the interface is not displayed in the signal tree, you can either check in *ibaPDA* in the I/O Manager under *General – Settings* or in the *ibaPDA* service status application whether your license for this interface has been properly recognized. The number of licensed connections is shown in brackets.

The figure below shows the license for the *Codesys Xplorer* interface as an example.



## 4.4 Connection problems linked to licenses

### Note



There are some restrictions for the assembly instance number with regard to the EtherNet/IP interface licenses.

The assembly instance must be within the range...

1 to 64 for the first license (basic license *ibaPDA-Interface-EtherNet/IP*),

65 bis 128 für die zweite (1. *one-step-up-Interface-EtherNet/IP-license*),

129 bis 192 for the third (2. *one-step-up-Interface-EtherNet/IP-license*),

193 bis 255 for the fourth (3. *one-step-up-Interface-EtherNet/IP license*).

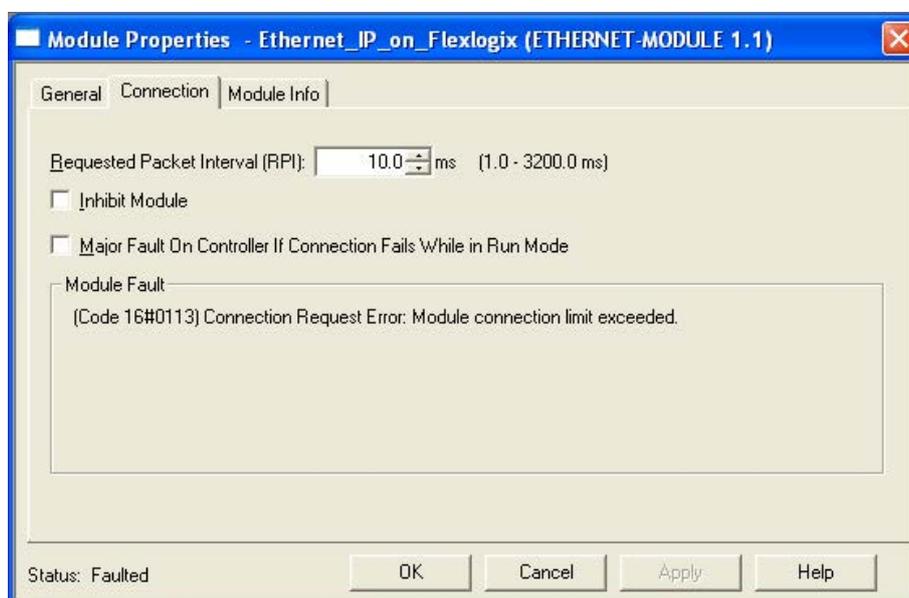
When a PLC tries to connect to an assembly instance higher than the number of licensed connections then you will see messages like these in the Ethernet/IP log file:

```
2013-08-28 13:43:13.990 [14: EthernetIP listening TCP thread] : **** ER-
ROR **** : Forward_Open: Connection not established because driver did
not allow connection (The parameter is incorrect.(0x00000057))
```

```
2013-08-28 13:43:13.990 [14: EthernetIP listening TCP thread] : **** ER-
ROR **** : Forward_Open: Connection not established, Connection instance
could not be created or configured
```

In RSLogix 5000, for example, you will see a module fault if assembly instance number does not comply with *ibaPDA* license

"Module connection limit exceeded."



## 4.5 Connection table

Connection table The connection table on EtherNet/IP interface node provides some more information.

Module	IP Address	Receive counter	Send counter	Sequence errors	Data length	Multicast join errors
1 Produced tag (0)	?	?	?	?	?	?
2 Other produced tag (4)	192.168.123.5	20631	20652	0	120	0
3 ?	?	?	?	?	?	?
4 ?	?	?	?	?	?	?
5 ?	?	?	?	?	?	?
6 ?	?	?	?	?	?	?
7 ?	?	?	?	?	?	?
8 ?	?	?	?	?	?	?
9 ?	?	?	?	?	?	?
10 ?	?	?	?	?	?	?
11 ?	192.168.123.5	3694179	3694178	0	520	0
12 ?	?	?	?	?	?	?
13 ?	?	?	?	?	?	?
14 ?	?	?	?	?	?	?
15 ?	?	?	?	?	?	?

Each row represents one connection which corresponds to one module or assembly instance respectively.

The row number corresponds to the assembly instance within *ibaPDA*.

For EtherNet/IP I/O modules this corresponds to the assembly instance configured in the generic Ethernet module of the PLC's I/O configuration.

For EtherNet/IP Produced Tag and I/O Scanner modules this is generated automatically.

The *Module* column shows with which module this connection corresponds.

The *IP Address* column shows the IP address of the connected PLC.

Beside the columns for message counters of received and sent messages and sequence error counter there is the *Data length* column. The *Data length* column shows the size of the UDP message without UDP header. The UDP data contains an EtherNet/IP header that is 20 bytes long for produced tag connections and 24 bytes long for I/O connections.

**Note**



If you double-click on a row, you will get to the corresponding offset in the *Memory view* tab.

Additional information is provided by the background color of the table rows:

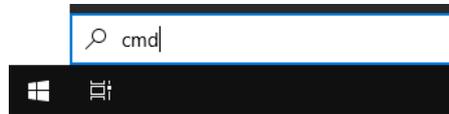
Color	Meaning
Green	Module for this assembly instance is defined and the connection is OK.
Orange	The connection is OK, however there is no module defined for this assembly instance.  Note:  This can only happen when there are generic Ethernet modules defined in the I/O configuration of the PLC. If you do <i>Autodetect</i> on the EtherNet/IP interface then I/O modules will be added for the orange connections.
Red	Module for this assembly instance is defined but the connection to the PLC has failed.
Gray	No connection and no module defined for this assembly instance

Table 4: Color code for the background colors of the connection table

## 4.6 Connection diagnostics with PING

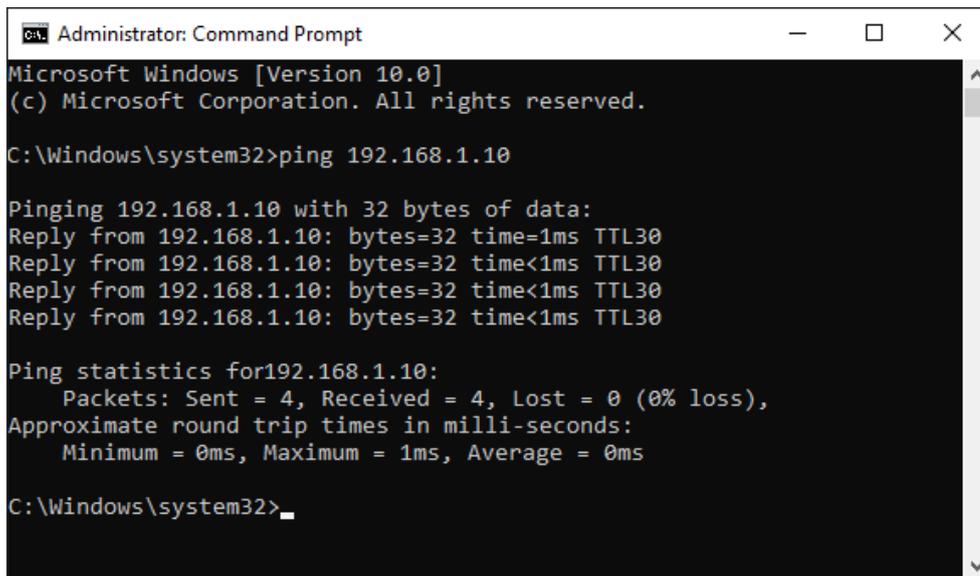
PING is a system command with which you can check if a certain communication partner can be reached in an IP network.

1. Open a Windows command prompt.



2. Enter the command "ping" followed by the IP address of the communication partner and press <ENTER>.

→ With an existing connection you receive several replies.

A screenshot of a Windows Command Prompt window titled 'Administrator: Command Prompt'. The window shows the following text:

```
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. All rights reserved.

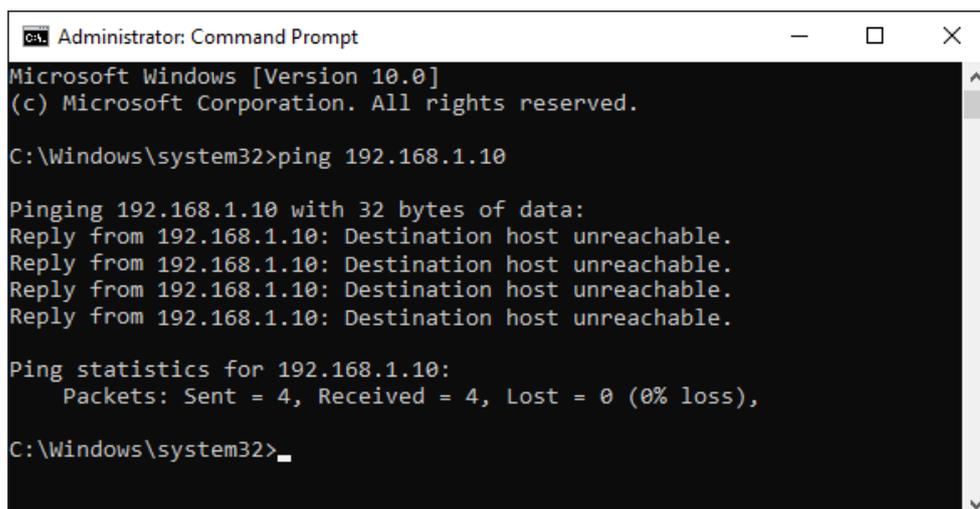
C:\Windows\system32>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time=1ms TTL30
Reply from 192.168.1.10: bytes=32 time<1ms TTL30
Reply from 192.168.1.10: bytes=32 time<1ms TTL30
Reply from 192.168.1.10: bytes=32 time<1ms TTL30

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Windows\system32>
```

→ With no existing connection you receive error messages.

A screenshot of a Windows Command Prompt window titled 'Administrator: Command Prompt'. The window shows the following text:

```
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: Destination host unreachable.

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Windows\system32>
```

## 4.7 Log files

If connections to target platforms or clients have been established, all connection-specific actions are logged in a text file. You can open this (current) file and, e.g., scan it for indications of possible connection problems.

You can open the log file via the button <Open log file>. The button is available in the I/O Manager:

- for many interfaces in the respective interface overview
- for integrated servers (e.g. OPC UA server) in the *Diagnostics* tab.

In the file system on the hard drive, you can find the log files of the *ibaPDA* server (...\[ProgramData\iba\ibaPDA\Log](#)). The file names of the log files include the name or abbreviation of the interface type.

Files named [interface.txt](#) are always the current log files. Files named [Interface\\_yyyy\\_mm\\_dd\\_hh\\_mm\\_ss.txt](#) are archived log files.

Examples:

- [ethernetipLog.txt](#) (log of EtherNet/IP connections)
- [AbEthLog.txt](#) (log of Allen-Bradley Ethernet connections)
- [OpcUAServerLog.txt](#) (log of OPC UA server connections)

## 4.8 Diagnostic modules

Diagnostic modules are available for most Ethernet based interfaces and Xplorer interfaces. Using a diagnostic module, information from the diagnostic displays (e.g. diagnostic tabs and connection tables of an interface) can be acquired as signals.

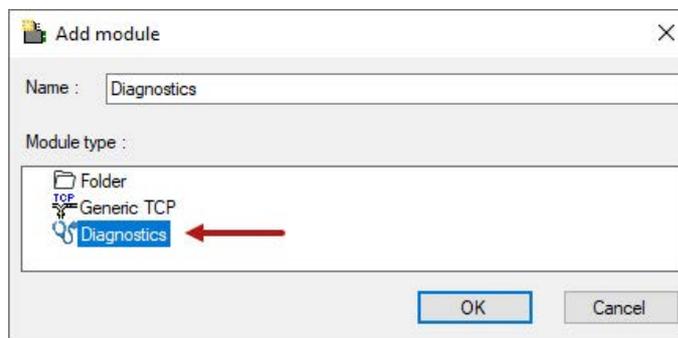
A diagnostic module is always assigned to a data acquisition module of the same interface and supplies its connection information. By using a diagnostic module you can record and analyze the diagnostic information continuously in the *ibaPDA* system.

Diagnostic modules do not consume any license connections, since they do not establish their own connection, but refer to another module.

Example for the use of diagnostic modules:

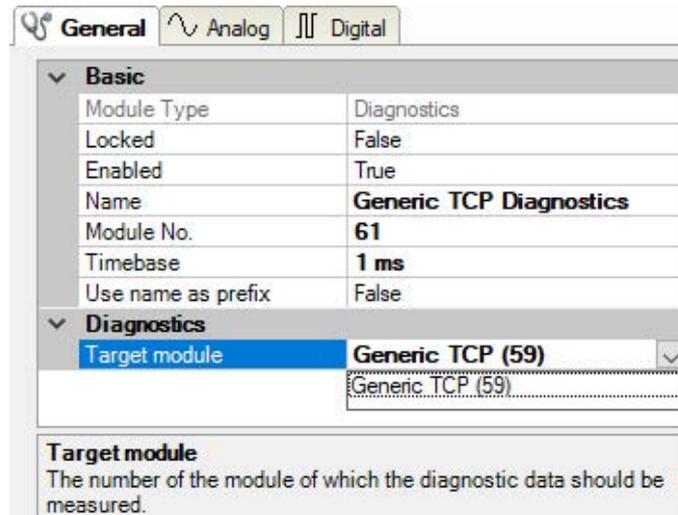
- A notification can be generated, whenever the error counter of a communication connection exceeds a certain value or the connection gets lost.
- In case of a disturbance, the current response times in the telegram traffic may be documented in an incident report.
- The connection status can be visualized in *ibaQPanel*.
- You can forward diagnostic information via the SNMP server integrated in *ibaPDA* or via OPC DA/UA server to superordinate monitoring systems like network management tools.

In case the diagnostic module is available for an interface, a "Diagnostics" module type is shown in the "Add module" dialog (example: Generic TCP).



### Module settings diagnostic module

For a diagnostic module, you can make the following settings (example: Generic TCP):



The basic settings of a diagnostic module equal those of other modules.

There is only one setting which is specific for the diagnostic module: the target module.

By selecting the target module, you assign the diagnostic module to the module on which you want to acquire information about the connection. You can select the supported modules of this interface in the drop down list of the setting. You can assign exactly one data acquisition module to each diagnostic module. When having selected a module, the available diagnostic signals are immediately added to the *Analog* and *Digital* tabs. It depends on the type of interface, which signals exactly are added. The following example lists the analog values of a diagnostic module for a Generic TCP module.

General Analog Digital						
Name	Unit	Gain	Offset	Active	Actual	
0 IP address (part 1)			1	0	<input checked="" type="checkbox"/>	
1 IP address (part 2)			1	0	<input checked="" type="checkbox"/>	
2 IP address (part 3)			1	0	<input checked="" type="checkbox"/>	
3 IP address (part 4)			1	0	<input checked="" type="checkbox"/>	
4 Port			1	0	<input checked="" type="checkbox"/>	
5 Message counter			1	0	<input checked="" type="checkbox"/>	
6 Incomplete errors			1	0	<input checked="" type="checkbox"/>	
7 Packet size (actual)	bytes		1	0	<input checked="" type="checkbox"/>	
8 Packet size (max)	bytes		1	0	<input checked="" type="checkbox"/>	
9 Time between data (actual)	ms		1	0	<input checked="" type="checkbox"/>	
10 Time between data (min)	ms		1	0	<input checked="" type="checkbox"/>	

For example, the IP (v4) address of a Generic TCP module (see fig. above) will always be split into 4 parts derived from the dot-decimal notation, for better reading. Also other values are being determined, as there are port number, counters for telegrams and errors, data sizes and telegram cycle times. The following example lists the digital values of a diagnostic module for a Generic TCP module.

General Analog Digital			
Name	Active	Actual	
0 Active connection mode	<input checked="" type="checkbox"/>		
1 Invalid packet	<input checked="" type="checkbox"/>		
2 Connecting	<input checked="" type="checkbox"/>		
3 Connected	<input checked="" type="checkbox"/>		

## Diagnostic signals

Depending on the interface type, the following signals are available:

Signal name	Description
Active	Only relevant for redundant connections. Active means that the connection is used to measure data, i.e. for redundant standby connections the value is 0. For normal/non-redundant connections, the value is always 1.
Buffer file size (actual/avg/max)	Size of the file for buffering statements
Buffer memory size (actual/avg/max)	Size of the memory used by buffered statements
Buffered statements	Number of unprocessed statements in the buffer
Buffered statements lost	Number of buffered but unprocessed and lost statements
Connected	Connection is established
Connected (in)	A valid data connection for the reception (in) is available
Connected (out)	A valid data connection for sending (out) is available
Connecting	Connection being established
Connection attempts (in)	Number of attempts to establish the receive connection (in)
Connection attempts (out)	Number of attempts to establish the send connection (out)
Connection ID O->T	ID of the connection for output data (from the target system to <i>ibaPDA</i> ). Corresponds to the assembly instance number
Connection ID T->O	ID of the connection for input data (from <i>ibaPDA</i> to target system). Corresponds to the assembly instance number
Connection phase (in)	Status of the ibaNet-E data connection for reception (in)
Connection phase (out)	Status of the ibaNet-E data connection for sending (out)
Connections established (in)	Number of currently valid data connections for reception (in)
Connections established (out)	Number of currently valid data connections for sending (out)
Data length	Length of the data message in bytes
Data length O->T	Size of the output message in byte
Data length T->O	Size of the input message in byte
Destination IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to <i>ibaPDA</i> )
Destination IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from <i>ibaPDA</i> to target system)
Disconnects (in)	Number of currently interrupted data connections for reception (in)
Disconnects (out)	Number of currently interrupted data connections for sending (out)
Error counter	Communication error counter
Exchange ID	ID of the data exchange
Incomplete errors	Number of incomplete messages

Signal name	Description
Incorrect message type	Number of received messages with wrong message type
Input data length	Length of data messages with input signals in bytes ( <i>ibaPDA</i> receives)
Invalid packet	Invalid data packet detected
IP address (part 1-4)	4 octets of the IP address of the target system
Keepalive counter	Number of KeepAlive messages received by the OPC UA Server
Lost images	Number of lost images (in) that were not received even after a retransmission
Lost Profiles	Number of incomplete/incorrect profiles
Message counter	Number of messages received
Messages per cycle	Number of messages in the cycle of the update time
Messages received since configuration	Number of received data telegrams (in) since start of acquisition
Messages received since connection start	Number of received data telegrams (in) since the start of the last connection setup. Reset with each connection loss.
Messages sent since configuration	Number of sent data telegrams (out) since start of acquisition
Messages sent since connection start	Number of sent data telegrams (out) since the start of the last connection setup. Reset with each connection loss.
Multicast join error	Number of multicast login errors
Number of request commands	Counter for request messages from <i>ibaPDA</i> to the PLC/CPU
Output data length	Length of the data messages with output signals in bytes ( <i>ibaPDA</i> sends)
Packet size (actual)	Size of the currently received message
Packet size (max)	Size of the largest received message
Ping time (actual)	Response time for a ping telegram
Port	Port number for communication
Producer ID (part 1-4)	Producer ID as 4 byte unsigned integer
Profile Count	Number of completely recorded profiles
Read counter	Number of read accesses/data requests
Receive counter	Number of messages received
Response time (actual/average/max/min)	Response time is the time between measured value request from <i>ibaPDA</i> and response from the PLC or reception of the data.  Actual: current value  Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Retransmission requests	Number of data messages requested again if lost or delayed

Signal name	Description
Rows (last)	Number of resulting rows by the last SQL query (within the configured range of result rows)
Rows (maximum)	Maximum number of resulting rows by any SQL query since the last start of acquisition (possible maximum equals the configured number of result rows)
Send counter	Number of send messages
Sequence errors	Number of sequence errors
Source IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to <i>ibaPDA</i> )
Source IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from <i>ibaPDA</i> to target system)
Statements processed	Number of executed statements since last start of acquisition
Synchronization	Device is synchronized for isochronous acquisition
Time between data (actual/ max/min)	Time between two correctly received messages Actual: between the last two messages Max/min: statistical values since start of acquisition or reset of counters
Time offset (actual)	Measured time difference of synchronicity between <i>ibaPDA</i> and the <i>ibaNet-E</i> device
Topics Defined	Number of defined topics
Topics Updated	Number of updated topics
Unknown sensor	Number of unknown sensors
Update time (actual/average/ configured/max/min)	Specifies the update time in which the data is to be retrieved from the PLC, the CPU or from the server (configured). Default is equal to the parameter "Timebase". During the measurement the real actual update time (actual) can be higher than the set value, if the PLC needs more time to transfer the data. How fast the data is really updated, you can check in the connection table. The minimum achievable update time is influenced by the number of signals. The more signals are acquired, the greater the update time becomes. Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Write counter	Number of successful write accesses
Write lost counter	Number of failed write accesses

## 5 Support and contact

### Support

Phone: +49 911 97282-14  
Fax: +49 911 97282-33  
Email: support@iba-ag.com

---

#### Note



If you need support for software products, please state the number of the license container. For hardware products, please have the serial number of the device ready.

---

### Contact

#### Headquarters

iba AG  
Koenigswarterstrasse 44  
90762 Fuerth  
Germany

Phone: +49 911 97282-0  
Fax: +49 911 97282-33  
Email: iba@iba-ag.com

#### Mailing address

iba AG  
Postbox 1828  
D-90708 Fuerth, Germany

#### Delivery address

iba AG  
Gebhardtstrasse 10  
90762 Fuerth, Germany

#### Regional and Worldwide

For contact data of your regional iba office or representative please refer to our web site

**[www.iba-ag.com](http://www.iba-ag.com)**.