



ibaPDA

I/O Manager

Manual Part 2
Issue 8.14

Measurement Systems for Industry and Energy
www.iba-ag.com

Manufacturer

iba AG
Gebhardtstrasse 10-20
90762 Fuerth
Germany

Contacts

Main office	+49 911 97282-0
Support	+49 911 97282-14
Engineering	+49 911 97282-13
E-mail	iba@iba-ag.com
Web	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 2026, 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 and can be found in the iba help center docs.iba-ag.com.

Version	Date	Revision	Author	Version SW
8.14	06-2026	Module overview: new column Samples/s; OPC RM UA server: refresh rate of the tags		8.14.0

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

1	About this documentation	9
1.1	Target group and previous knowledge	9
1.2	Notations	9
1.3	Used symbols.....	10
1.4	Documentation structure	11
2	Introduction	12
3	I/O Manager toolbar	13
4	General settings and information	16
4.1	Settings	16
4.2	System information	19
4.3	Signal names.....	21
4.4	Unit table	23
4.4.1	Unit systems	26
4.4.2	Units	27
4.4.3	Abbreviations.....	29
4.4.4	Using the unit system in the signal table	31
4.4.5	Changing the unit system	33
4.4.6	Indicating the active unit system	34
4.4.7	Unit table in the data file	35
4.5	Address books	36
4.6	Certificates.....	38
4.6.1	Certificates.....	38
4.6.2	Central certificate store	39
4.6.3	Manage certificates	41
4.6.3.1	Generate a new certificate	42
4.6.3.2	Add certificate	43
4.6.3.3	Export certificates.....	43
4.6.4	Use certificates	43
4.6.5	Save and protect certificates	44
4.6.6	Check information about a certificate	45

4.7	Time synchronization.....	45
4.7.1	DCF77	46
4.7.1.1	DCF77 Configuration.....	47
4.7.2	IEC 1131.....	49
4.7.3	PTPv2	50
4.7.4	ibaClock	53
4.7.5	NTP via Windows.....	54
4.8	Module overview.....	55
4.9	Knowhow protection	59
4.9.1	Application range of know-how protection	59
4.9.2	Creating a protection scheme.....	60
4.9.3	Application of a protection scheme.....	62
4.9.4	Removing the protection	62
4.9.5	Importing and exporting protected elements	63
4.9.6	Unlocking protected items	63
4.10	Watchdog	64
4.10.1	Information about the watchdog telegram structure, ASCII format.....	65
4.10.2	Information about the watchdog telegram structure, binary format.....	66
4.11	External configuration	67
4.12	Stop prevention	68
4.13	Boards.....	69
4.14	Interfaces	71
4.15	Interrupt Info	74
4.16	Multistation	77
4.17	OPC server	78
4.18	OPC UA Server	79
4.18.1	General information	79
4.19	SNMP server	80
4.20	IEC61850 server.....	81
4.21	Signal usage display	81

5	Groups and vector signals	82
5.1	Adding groups.....	82
5.2	Renaming a group.....	83
5.3	Deleting groups.....	83
5.4	Move groups.....	83
5.5	Assigning signals to a group.....	83
5.6	Deleting signals from a group.....	84
5.7	Remove inactive signals from a group.....	84
5.8	Creating vector signals.....	84
5.9	Export groups	85
6	Text signals and text processing.....	86
6.1	Using the text signals.....	86
6.2	Properties of the text signals.....	87
6.3	Using text signals	94
6.3.1	General	94
6.3.2	Configuration	95
6.3.3	Creating text signals.....	97
6.3.4	Delete the text signals	98
6.3.5	Defining and assigning text signals	98
6.4	Text interface	99
6.5	Text splitter module	100
6.5.1	Add text splitter module.....	101
6.5.2	Module settings	101
6.5.2.1	Fixed width split mode	103
6.5.2.2	Delimited split mode	104
6.5.2.3	Split mode JSON	106
6.5.3	Signal configuration	107
6.6	Text signals via TCP/IP and UDP.....	108
6.6.1	Add module	108
6.6.2	Module settings.....	109
6.6.3	Signal configuration	110

6.7	Text signals via serial (COM) interface	111
6.7.1	Add module	111
6.7.2	Module settings	111
6.7.3	Signal configuration	112
6.8	Text signals via text file (file text)	113
6.8.1	Add module	113
6.8.2	Module settings	113
6.8.3	Signal configuration	115
6.9	Custom text (text creator)	116
6.9.1	Add module	116
6.9.2	Module settings	116
6.9.3	Text creator (Plaintext)	118
6.9.4	Text creation XML/JSON	121
6.9.4.1	Data structure tree	122
6.9.4.2	Signal tree	126
6.9.4.3	Synchronize arrays	128
6.9.5	Signal configuration	129
6.10	Text via ibaQPanel text input	130
6.10.1	Add module	130
6.10.2	Module settings	130
6.10.3	Signal configuration	131
6.11	Text shift register	132
6.11.1	Add module	132
6.11.2	Module settings	132
6.11.3	Signal configuration	134
6.11.4	Application example for text shift register	135
6.12	Text signals via OPC / OPC UA	137
6.13	Text signals via other interfaces	137
7	Outputs	138
7.1	Create outputs	140
7.2	Adding output modules	140
7.3	Setting up output modules	141

7.3.1	General module settings.....	141
7.3.2	OPC output module	142
7.3.2.1	General tab	142
7.3.2.2	Analog and Digital tab	143
7.3.3	Email	143
7.3.3.1	Basic settings (accounts and fields)	144
7.3.3.2	Email module settings	145
7.3.4	ibaNet750-BM, -BM-D output modules	149
7.3.4.1	General tab	149
7.3.4.2	Analog and Digital tab	150
7.3.5	ibaL2B... output modules	151
7.3.5.1	General tab	151
7.3.5.2	Analog and Digital tab	151
7.3.6	Reflective Memory output module	152
7.3.6.1	General tab	152
7.3.6.2	Analog and Digital tab	152
7.3.7	EtherNet/IP I/O output modules	152
7.3.8	Generic TCP output module	153
7.3.8.1	General tab	153
7.3.8.2	Analog and Digital tab	154
7.3.9	Generic UDP output module	155
7.3.9.1	General tab	155
7.3.9.2	Analog and Digital tab	155
7.3.10	Modbus-TCP Client output module	156
7.3.11	ABB-Xplorer output module	157
7.3.12	OMRON-Xplorer output module	158
7.3.13	S7-Xplorer output module	159
7.3.14	TwinCAT-Xplorer output module	159
7.3.15	ibaCapture output module	161
7.3.15.1	General tab	161
7.3.15.2	Video server tab	161
7.3.15.3	Analog and Digital tab	162
7.3.16	ibaVision output module	163
7.3.17	FOB alarm output module	163

7.3.17.1	General tab	163
7.3.17.2	Analog and Digital tab	163
7.3.18	Outputs to the iba modular system (ibaPADU-S).....	164
7.3.19	Outputs to iba bus modules (ibaBM-...)	165
7.3.20	Outputs to iba system interconnections (ibaLink-...)	165
7.3.21	OPC UA client output module.....	166
7.3.22	MQTT output module	168
7.3.23	Output module for SQL commands to SQL databases.....	172
7.3.24	ibaNet-E output modules	173
7.3.25	ibaM-DAQ outputs.....	174
8	The I/O configuration file	175
8.1	Saving an I/O configuration	175
8.2	Loading an existing I/O configuration	176
8.3	Loading a backup configuration file.....	177
8.4	Exporting an I/O configuration to a text file	177
8.5	Importing a configuration text file	180
9	Support and contact.....	181

1 About this documentation

This documentation describes the function and application of the software *ibaPDA*.

1.1 Target group and previous knowledge

This documentation 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 they are capable of assessing safety and recognizing possible consequences and risks on the basis of their 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</i> , <i>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.

1.4 Documentation structure

This documentation fully describes the functionality of the *ibaPDA* system. It is designed both as a tutorial as well as a reference document. The sections and chapters essentially follow the procedure for configuring the system.

In addition to this documentation, you can examine the version history in the main menu, *Help – Version history* (file [versions.htm](#)) for the latest information about the installed version of the program. This file not only lists the bugs that have been eliminated, but also refers to extensions of the system in note form.

In addition, special "NewFeatures..." documentation comes with any software update that includes significant new features, which provides a more detailed description of the new features.

The state of the software to which the respective part of this documentation refers is listed in the revision table on page 2.

The *ibaPDA* system documentation (PDF and printed version) is divided into seven separate parts. Each part has its own section and page numbering beginning at 1, and is updated independently.

Part 1	Introduction and installation	General notes, license policy and add-ons Installation and program start User interface, system architecture, client server User management, printing
Part 2	I/O Manager	Basic information about the I/O Manager, general settings Groups and vector signals, text signals, outputs, configuration files
Part 3	Data interfaces and modules	Interfaces for the measurement data acquisition Standard interfaces, ibaFOB, Ethernet-based interfaces and more. For interfaces for which there are separate manuals, these are referred to.
Part 4	Expression builder	All functions for calculating virtual signals
Part 5	Data storage	Types of data recording, recording profiles, signal selection
Part 6	Data visualization	All display modes for live data, their operation and settings
Part 7	Appendix	Various additions, error listings, etc.

2 Introduction

"I/O" stands for "Input/Output". The I/O Manager is the main dialog for all configuration settings related to signals and interfaces.

All data sources and signals must first be defined in the I/O Manager. How the signals are then eventually used (display, recording, etc.) is subsequently determined in the other dialogs.

All setting options are clearly displayed in the I/O Manager. When configuring the input and output modules, the user is actively supported through context-sensitive selection menus. This helps to avoid configuration errors. Each change in configurations is checked by the system before being applied.

Note



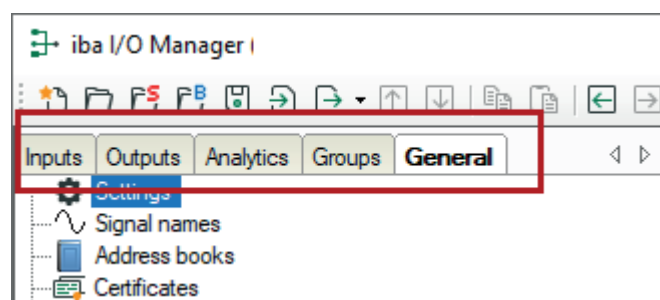
Changing the configuration settings in the I/O Manager always causes acquisition to automatically stop and restart!

The I/O Manager can be accessed as follows:

- Via the *Configure - I/O Manager* menu
- Via the button

You will find four main sections in the I/O Manager where adjustments can make other settings:

- Inputs
- Outputs
- Analytics
- Groups
- General



Due to the extent and scope, the description of the data interfaces, which are an integral part of the I/O Manager, is addressed separately in part 3 of the manual.

3 I/O Manager toolbar

Toolbar

Symbol	Description
	New configuration; then choose whether an empty configuration (reset to factory settings without auto-detect) or a new configuration with auto-detecting the connected modules should be created. If you click on this button while the acquisition is running, an additional option appears offering a new configuration with auto-detect after automatic stop of acquisition.
	Opens an existing I/O configuration file (XML-file .io). Afterwards, if applicable, the possibility to keep or delete initial values, if the configuration contains corresponding modules.
	Opens an existing I/O configuration file (.io or .zip) in simulation mode. The loaded configuration is not aligned with the currently existing hardware or licenses and cannot be applied. The simulation mode only provides for display of any I/O configuration.
	Opens access to the automatically saved backups of the I/O configuration in the path %ProgramData%iba\ibaPDA\Backup on the ibaPDA-Server computer. In a dialog window you can select and load the desired backup. Furthermore, you can use the dialog to delete a backup.
	Saves the current, full I/O configuration in an XML-file. You can select or enter the path and filename in the following dialog. The file extension .io is added automatically.
	Importing an I/O configuration, which is available in the form of a text file. The I/O configurations of an older ibaPDA system (< V6) as well as modules, groups or profiles that were exported from another ibaPDA system or configured using other tools, e.g., MS Excel, can also be imported. Signal addresses can be imported as decimal or hexadecimal (prefix "0x") values. You cannot import a complete I/O configuration by using the import function.
	Button with drop-down list to export to a text file one out of the following: ■ the current I/O configuration, i.e. all modules, groups and profiles ■ all or selected modules ■ all groups ■ all InSpectra-profiles
 	These buttons are disabled, unless you can move selected modules within the interface. Move the module by one position up or down in the signal tree of the interface.
	This Button is only enabled if a signal table is open. Pressing this Button will copy the open signal table to the Windows clipboard in order to paste it into another signal table or e.g. Microsoft Excel/Word.

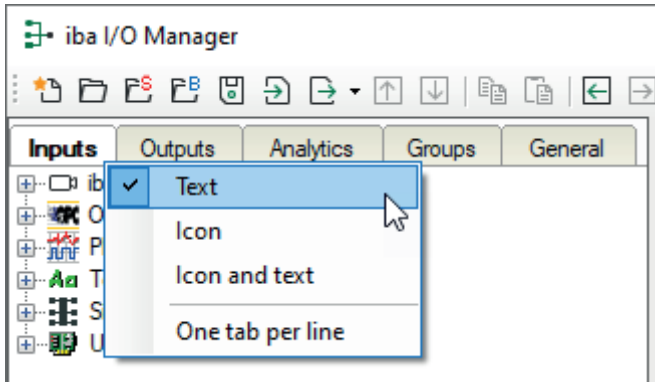
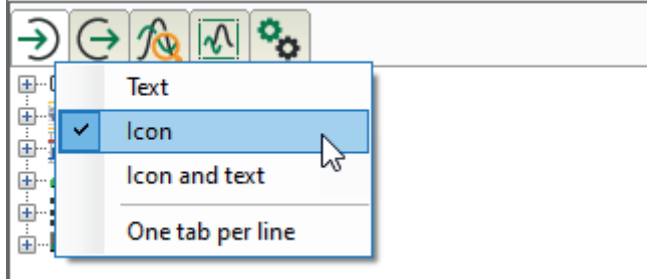
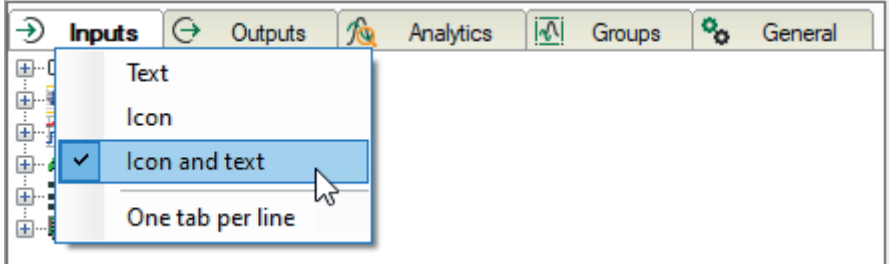
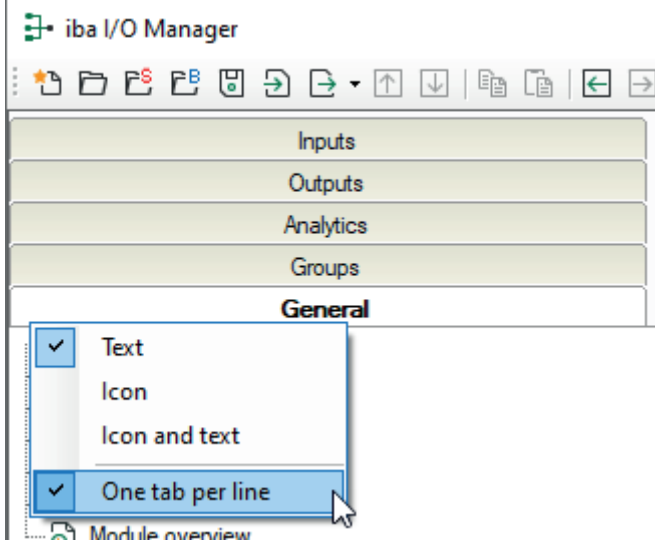
Symbol	Description
	This Button is only enabled if a signal table is open. Clicking this Button will paste the contents of the Windows clipboard, provided it is a compatible table, created in I/O Manager or e.g. Microsoft Excel/Word.
 	Go forward/backward buttons: Navigate to the previous or following step in the I/O Manager.

Tabs

Tabs	Icon	Description
Inputs		Here, you can configure data sources (interfaces), modules and signals.
Outputs		Here, you can configure output signals, e-mails and SQL commands for interfaces and modules, which support output of data.
Analytics		Here, you'll find internal interfaces and modules which are rather used for analytic processing of the input and other signals. Included interfaces and add-ons are: ■ Virtual ■ InCycle ■ InSpectra ■ Snapshots Before introducing the <i>Analytics</i> tab, these interfaces were placed in the <i>Inputs</i> tab. When updating from older <i>ibaPDA</i> versions to v8.4.0 or higher these interfaces will be moved automatically to the <i>Analytics</i> tab.
Groups		Here, you can define signal groups, assign signals to groups and define multidimensional signals (vectors).
General		Here, you can carry out general settings for interfaces and modules, such as address books and certificate management, time synchronization set-up or visibility of interfaces.

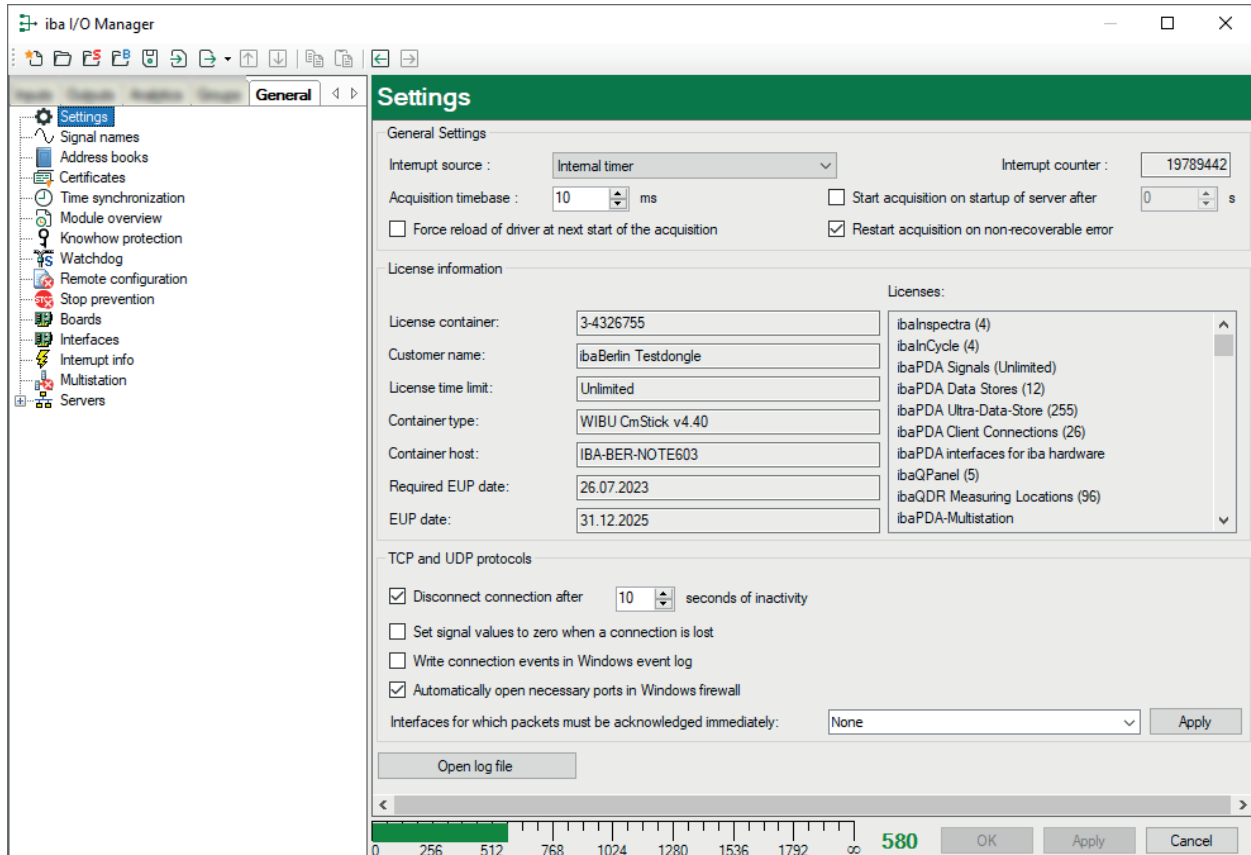
You can set up the representation and arrangement of the tabs according to your needs. The arrangement of the tabs is responsive to the available size of the window. The program tries to show the complete text in the tabs. If there is not enough space, the tabs will be arranged one above the other.

Concerning the contents of the tabs you may choose between text, icon or text + icon. Use the context menu on any tab to make your choice.

Text (default)	 The screenshot shows the Iba I/O Manager toolbar with a dropdown menu open. The 'Text' option is selected, indicated by a checkmark. Other options visible are 'Icon', 'Icon and text', and 'One tab per line'.
Icons	 The screenshot shows the Iba I/O Manager toolbar with a dropdown menu open. The 'Icon' option is selected, indicated by a checkmark. Other options visible are 'Text', 'Icon and text', and 'One tab per line'.
Icons and text	 The screenshot shows the Iba I/O Manager toolbar with a dropdown menu open. The 'Icon and text' option is selected, indicated by a checkmark. Other options visible are 'Text', 'Icon', and 'One tab per line'.
One tab per line	 The screenshot shows the Iba I/O Manager toolbar with a dropdown menu open. The 'One tab per line' option is selected, indicated by a checkmark. Other options visible are 'Text', 'Icon', and 'Icon and text'.

4 General settings and information

This section describes the information provided in the *General* tab.



4.1 Settings

General settings

Interrupt source

The interrupt source determines the entire timing of the data acquisition in *ibaPDA*. From this drop-down list, select one card which should act as the interrupt source on the PCI-/PCIe-bus.

The iba cards are fitted with a very precise and stable timer chip. If iba cards are installed in the computer they are listed in the drop-down list for the interrupt source. If cards of the *ibaFOB-D* family are used they are preferred as interrupt source compared to older cards.

If no iba cards are used, the less stable Windows timer will be used instead.

Interrupt counter

The interrupt counter is incrementing with a rate of 1000 / sec when the drivers and *ibaPDA* server are started and the application time base is set to 1 ms.

Acquisition timebase

Enter the timebase here.

The acquisition timebase is the superior clock for the entire system and the fastest period for sampling data in running acquisition. Slower sampling can be configured for the individual modules by the module's timebase setting.

Excluded are data which are acquired over fiber optic connections with ibaNet 32Mbit Flex protocol, data from *ibaInSpectra* modules and *ibaInCycle* modules as well as virtual modules. In these cases timebases faster than the acquisition timebase can be configured.

Faster sampling induces higher load and data volume. Therefore, it is always a reasonable approach to set the sampling on an appropriate level, i.e. as low as required for the intended application.

The configured module timebases cannot be chosen arbitrarily and independently from each other in order to ensure the good performance of the system even with high amounts of data. An important quantity is the “driver period” which results from the module timebases. It is determined by the least common multiple of all module timebases.

The longest driver period should not exceed 1000 ms. The ratio of smallest module timebase and driver period must be limited. Bad or inadmissible configurations will be indicated during the validation on start of the acquisition by warning or error messages.

You will find an overview of all modules including their configured time bases in *Module overview* (see [➔ Module overview, page 55](#)).

As the name implies, the acquisition timebase refers only to data acquisition (sampling) not recording. The timebase for the data recording which is used to write the measured values into the data file is configured in the data storage profiles.

Start acquisition on startup of server after ... s

If you check this option, the acquisition will be started automatically after a system restart of the server (equivalent of the GO button). You have to select this option if you need a full automatic startup sequence of the system, e.g. after a power failure. A prerequisite is, of course, that the *ibaPDA* server service starts automatically when the system is booting.

In the field next to it, you can set a delay time in seconds. A delay of >0 may be useful if other components, drivers or interfaces are involved, which are needed to correctly start the acquisition and that need some time to be available (e.g. remote OPC server).

Force reload of driver at next start of the acquisition

If you check this option, the drivers will be reloaded and restarted with the next change of the I/O configuration. The selection of this option will not be retained. The option will be disabled automatically after the driver has been loaded and started. You have to enable it again for the next repetition.

Selecting this option is recommended if *ibaPDA* does not experience any interrupts. The system always checks the interrupts when the acquisition starts. If no interrupts are detected, an error message is shown.

Restart acquisition on non-recoverable error

If you check this option, data acquisition will be stopped automatically in the event of a non-recoverable error.

The automatic restart is the attempt to fix the error. At the same time, faulty modules may eventually be disabled. For most of the applications it is vital, that the data acquisition with *ibaPDA* is running continuously. It is therefore more beneficial to have a running data acquisition, even with reduced signal count, than a stopped data acquisition.

License information

In the *License information* area you'll find important information about your current software license.

License container

The license container number is important for all service queries and upgrades. Please provide the license container number to our support team. It is linked to your license container, i.e. either MARX dongle, WIBU dongle or WIBU soft license and stored in our database.

Customer name

In this field, you will find the name of the customer for which this license has been approved. For projects, which were initially completed via system integrators, registration of the end user should have been requested from iba either from the outset or, at the latest, after the warranty period has expired. This makes it easier to assign the license in the case of later extensions or support cases.

License time limit

This field shows the validity time of the licenses. Depending on the container type there is either a remaining validity time given in days or hours or there is an expiration date.

Container type

This field indicates whether a MARX dongle, a WIBU dongle (WIBU CmStick) or a WIBU soft license (WIBU CmActLicense) is used.

Container host

This field shows the name of the computer where the license container is attached, which supplies the licenses obtained by the application.

Required EUP date and EUP date

The EUP date specifies the date until which the period of free software updates is valid. After the update period has expired you still can continue running the program as usual but only with the features which were already available until then.

For more information, see Part 1, *Update policy*.

Licenses

This window displays all the available basic and additional licenses, such as clients, data storages, interfaces, plug-ins etc.

If multiple licenses are applicable, e.g. clients, Xplorer interfaces, data storages etc., the number of respective licenses is given in brackets.

Active signals

A bar on the scale and the value behind the scale show the number of currently active signals. The end value of the scale indicates the maximum permissible number of signals according to the license.

TCP and UDP protocols

Disconnect connection after... seconds without activity

Enabling of this option will activate a timeout for TCP/IP connections. If no messages are received over the TCP/IP connections within the specified time (to be entered in the entry field next to the checkbox), the corresponding connection will be disconnected. Disconnecting unused connections saves resources.

Set signal values to zero when a connection is lost

Enabling this option sets all measured signals of a connection to zero if the connection is interrupted. Otherwise the signal values would show the value at the time of the connection interruption.

Write connection events in Windows event log

Use this option to enter selected events from communication field in the Windows event log so that they are visible in the Windows event display "System". This refers to Ethernet-based interfaces which are incapable to generate their own log, e.g. because they are implemented in the driver. Most of the Ethernet-based interfaces which are supported by *ibaPDA* generate their own log files and write them to a dedicated subfolder in the server program path on the hard disk. You'll find more information on this in the manuals of the interfaces.

Automatically open necessary ports in Windows firewall

The installer of *ibaPDA* only adds the application exceptions for *ibaPDA* client, server and S7-Xplorer proxy to the Windows firewall. Exceptions for network protocols which are handled by the *ibaPDA* driver can be added automatically or manually. If you enable this option (default) the required port exceptions are added automatically. If you disable this option then you can manually add the port exceptions to the firewall by clicking the button <Allow ports through firewall> on each interface that requires a listen port.

Interfaces for which packages must be confirmed immediately...

For TCP-based interfaces that are to receive data faster than 200 ms, a so-called "fast acknowledge" must occur. For this purpose, the parameter "TcpAckFrequency" must be changed in the Windows registry. So that the user does not need to edit the registry, this option can be used to select the network adapter(s) for which a quick response is required. The changes in the registry are then made automatically.

<Open log file> button

Via this button, you open the log file of the *ibaPDA* server in the standard text editor of the (Windows) system. All system-related events and processes from the perspective of the server acquire logged this log.

If you create a support file in the support case using the *Save information for iba support...* function in the *Help* menu then this log file is included.

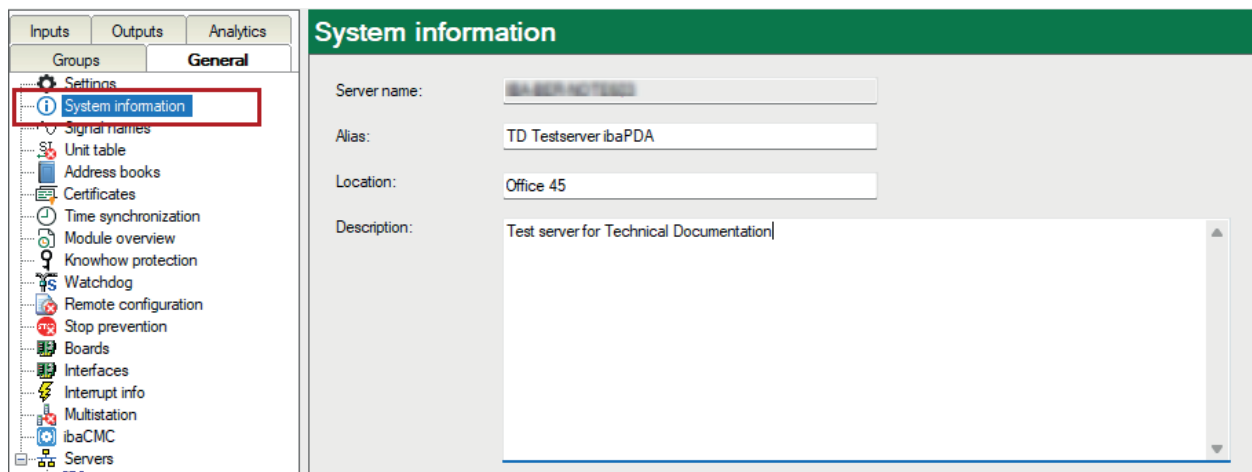
4.2 System information

The node *System information* offers some information about the *ibaPDA* server computer. The display of the server name derives from the Windows computer name.

You can add more information:

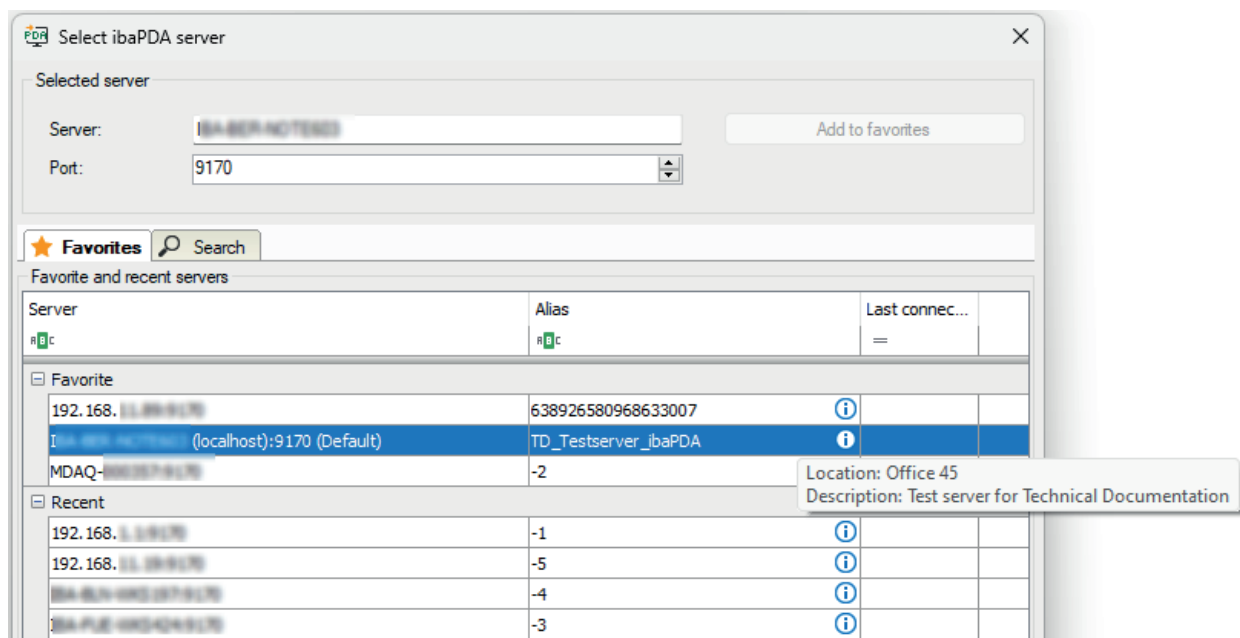
- Alias
- Location
- Description

The information is stored in the I/O configuration file.



Alias

If needed, you can assign an alias to the computer, which is, for example, more intelligible than the host name. This alias will also be available on client side in the *Select server* dialogs in the *Search* and *Favorites* tabs and hence, helps to identify the desired server.



The other information *Location* and *Description* is displayed in the tooltip.

Location

Enter a clear denomination for the location where the computer is installed, e.g. describing text or a standardized location designation (+ ...).

Description

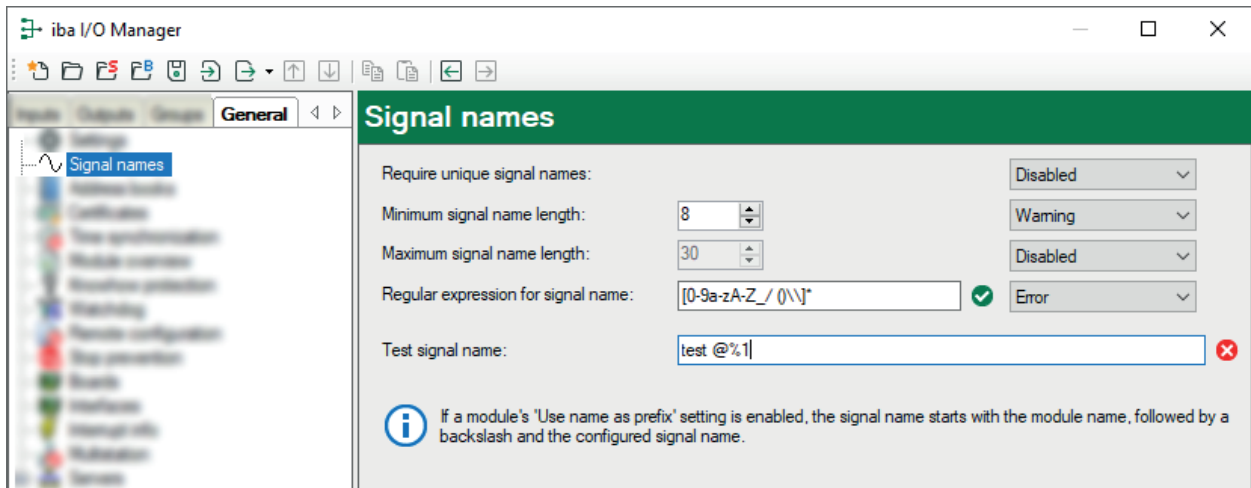
Enter more describing information here, if needed.

Note



If you transfer an I/O configuration file to another computer in another location, make sure to update the system information in the configuration to avoid confusion.

4.3 Signal names



The creation of signal names can be checked against four rules, which can be enabled or disabled individually:

- The signal name must be unique through the entire I/O configuration
A signal name must not be created twice in the I/O configuration.
- Guarantee a minimum signal name length
The signal name must at least consist of the number of characters as specified.
- Limit the maximum signal name length
The number of characters in the signal name must be equal or less than the specified number.
- The signal name must comply with a regular expression
The signal name must only consist of characters, which are permitted by the specified regular expression. (Default: `[0-9a-zA-Z_/\(\)\ \]*`)

For each rule you can use the selection button on the right side to determine whether the rule is disabled or whether it should generate a warning or an error in case of violation.

You can enter any name into the field *Test signal name* in order to check if the rules work properly. As long as the test signal name does not follow the rules, a red circle with a white cross appears on the right end of the field.

The compliance with the enabled rules is already checked while you work in the I/O Manager. As soon as a signal name does not comply with the rules a warning or error icon indicates this in the signal table. The following examples refer to the settings in the picture above.

General Analog Digital		
	Name	Expression
0	sdjtn_(0)der	f_{∞} [3:0]
1	dfri_123	f_{∞} [6:0]
2	✖ sdkko.fd	f_{∞} [3:1]
3	✖ [7:0]	f_{∞} [7:0]
4	⚠ bcde_1	f_{∞} [7:0]
5	123	f_{∞}
*	⚠ The name of [12:4]: bcde_1 has 6 characters, this is less than the configured minimum of 8.	

In case more than one rule has been broken, all reasons are mentioned in the tooltip on mouse-over.

General Analog Digital		
	Name	Expression
0	sdjtn_(0)der	f_{∞} [3:0]
1	dfri_123	f_{∞} [6:0]
2	✖ sdkko.fd	f_{∞} [3:1]
3	✖ [7:0]	f_{∞} [7:0]
4	⚠	
5	1	
*		f_{∞}

✖ The name of [12:3]: [7:0] has 5 characters, this is less than the configured minimum of 8.
 ✖ The name of [12:3]: [7:0] does not match the regular expression [0-9a-zA-Z_/\ \\\]*.

Furthermore, the compliance with the rules is checked during the validation of the I/O configuration, e. g. at start of acquisition. The corresponding warning or error messages will be displayed and in case of an error the I/O configuration will be invalid.

Applying new I/O configuration (27.01.2023 12:22:46)

- Checking license...
- Validating I/O configuration...
 - data and tracking (3)
 - copy thickness channel for different QDR profiles (4)
 - Generic TCP 1 (6)
 - Generic TCP 2 (8)
 - Generic TCP 3 (9)
 - Generic TCP 4 (10)
 - Generic TCP 5 (11)
 - SignalNames (12)
 - ✖ The name of [12:2]: sdkko.fd does not match the regular expression [0-9a-zA-Z_/\ \\\]*.
 - ⚠ The name of [12:3]: [7:0] has 5 characters, this is less than the configured minimum of 8.
 - ✖ The name of [12:3]: [7:0] does not match the regular expression [0-9a-zA-Z_/\ \\\]*.
 - ⚠ The name of [12:4]: bcde_1 has 6 characters, this is less than the configured minimum of 8.
- ✖ I/O configuration is NOT valid

4.4 Unit table

The node Unit table offers the possibility to configure different unit systems in order to be able to switch the display of numerical values to a preferred unit system, if needed.

For example, a German manufacturer can configure the *ibaPDA* system in accordance with the metric SI unit system. Later, another user in the anglosaxon region can switch to the British or American unit system (Imperial units or U.S. Customary units) in the *ibaPDA* client and will find the values and units in his or her preferred system.

Note



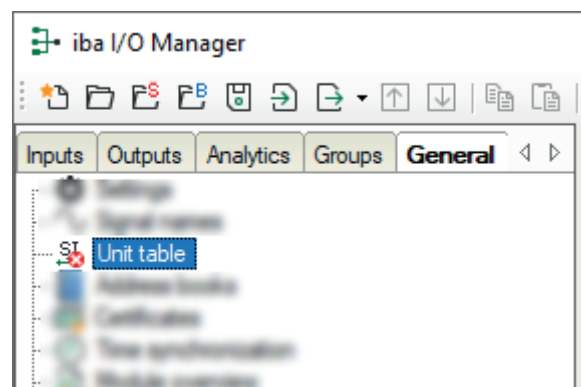
Numerical values in data files or in *ibaHD-Server* are always stored with the original unit according to the I/O configuration. Usually this is the primary unit system. The conversion into other units is exclusively done for visualization purposes.

Switching between unit systems has the following effects:

- The views of *ibaPDA* client will show the values of all *ibaPDA* and *ibaHD* signals converted to the new unit system.
- The signal units in the legend will be switched.
- All views in *ibaQPanel*, incl. the offline trendgraph and HD Time periods table, will show the values in the new unit system.

Configuring and enabling the unit table

Select the node *Unit table* in the *General* tab of the I/O Manager.



If no unit table is available yet, you'll be asked whether you want to load the standard table. This is a table with the most common units in the metric and imperial systems. You can extend or change the table if needed.

If you reject loading the standard table, you'll get an empty table which you can fill by yourself.

Description of the unit table

Unit table

☒ Enabled

Name:

Check signal unit is part of unit table:

Category	Name	Description	Active	Metric Abbreviation	Imperial Abbreviation	Scaling factor	Scaling offset
Acceleration							
Acceleration			☒	m/s ²	ft/s ²	3,0487804878	0
Acceleration	g		☒	m/s ²	g	0,1019716213	0
Area							
Area			☒	mm ²	in ²	0,0015500031	0
Area			☒	m ²	ft ²	10,7639104167	0
Area			☒	km ²	mi ²	0,3861021585	0
Electrical							
Electrical			☒	V	V	1	0
Electrical			☒	A	A	1	0
Electrical			☒	Ω	Ω	1	0
Electrical			☒	kV	kV	1	0
Electrical			☒	kA	kA	1	0
Energy							
Energy			☒	J	J	1	0
Energy			☒	kJ	kJ	1	0
Energy			☒	kWh	kWh	1	0
Energy			☒	VA	VA	1	0
Energy			☒	VA _r	VA _r	1	0
Flow rate							
Flow rate			☒	L/s	cfm	2,1188800033	0
Flow rate			☒	m ³ /s	cfs	35,3146667215	0
Force							
Force			☒	N	lbf	0,2248089431	0
Length							
Length			☒	mm	mil	39,3700787402	0

The table consists of unit systems and units. One of the unit systems is the primary unit system (standard table: metric). For the other unit system(s) there are additional columns with scaling factor and scaling offset containing the values for the conversion. Optionally, you can open test columns to verify the conversion with sample values.

There is one row for each unit.

Each unit has a category, a name, a description, an abbreviation (unit symbol) and scaling information. For a clear view you can group the units by categories. See chapter [Units, page 27](#) for more information about units.

Above the table there are 3 settings.

Enabled

If you don't enable the unit table, then you cannot switch the unit system in the views of *ibaPDA* client. Default setting: disabled.

Name

Enter a name for the unit table here. The name will be used when you export the table or save it as a file. The name will also be used when saving the table in the data file.

Check signal unit is part of unit table

Select what should happen if a used signal unit is not part of the unit table:

- Disabled: Signals and their units are not checked.
- Warning: A warning message will be issued in the signal table and at validation of the I/O configuration. However, the I/O configuration will be accepted and the acquisition starts.
- Error: An error message will be issued in the signal table and at validation of the I/O configuration. The I/O configuration is not valid and the acquisition does not start until the error is fixed.

Symbol buttons

Above and next to the table, there are some buttons with the following features:

	Create a new unit table
	Load a unit table from file (*.ut); replace current unit table
	Save unit table in file (*.ut)
	Add new unit system
	Delete unit system
	Import unit table (unit system); adds the imported unit system to the existing unit table.
	Export unit table (unit system)
	Show/hide test columns
	Group/ungroup units by category
	Add unit/row
	Delete selected unit(s)/row(s)

Create a new unit table

Click on the corresponding icon in the tool bar and choose whether you want to load the standard table or an empty table.

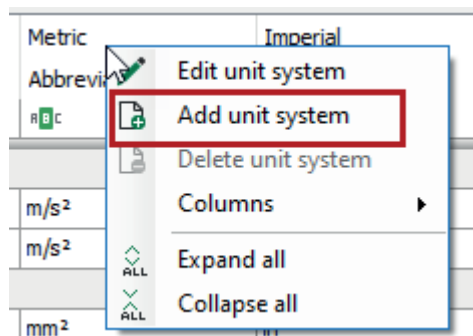
A table based on the metric unit system as primary system will be created automatically. Then add the destination unit system of your choice or enter the units first.

4.4.1 Unit systems

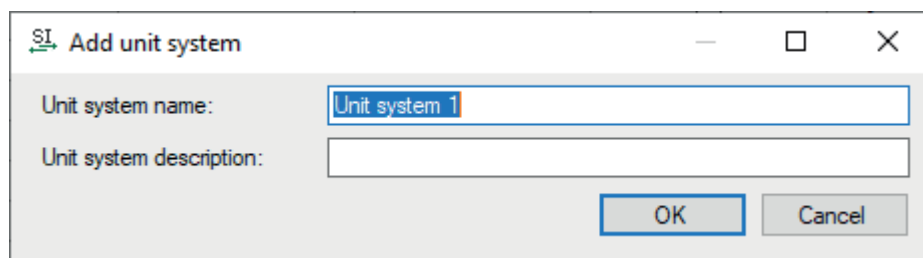
You can add, edit and delete unit systems.

Add unit system

Click on the corresponding icon in the toolbar or make a right mouse click on a unit system column header and choose *Add unit system*.



A dialog opens, where you can enter a name and a description. Enter at least a unique name. It is not allowed to have multiple unit systems with the same name in the unit table. If you enter a description, it will be shown as tooltip when hovering the column header of a unit system in the table with the mouse.



Edit unit system

Make a right mouse click on the unit system column header and choose *Edit unit system*. A dialog opens, where you can edit the name and the description.

Delete unit system

Make a right mouse click on the column header of the unit system you want to delete and choose *Delete unit system*. Alternatively, you can click on the corresponding icon in the toolbar and select the unit system(s) to be deleted in the dialog.

Import/export unit systems

Use the corresponding icons in the toolbar for these functions.

You can import one or more unit systems from a file (*.ut).

In the import dialog you can select the unit system(s) to be imported. The primary unit system is always selected because it is the reference for the conversion. Existing units will be overwritten with new values during import. New units will be added.

You also can export one or more unit systems and save them in a file (*.ut).

In the export dialog you can select the unit system(s) to be exported. The primary unit system is always selected.

4.4.2 Units

This chapter provides information about the configuration of units in the unit table.

Properties

Each unit (row) has the following properties (columns):

Category

The category can be used to group the units. Physical entities are used as categories, such as temperature, velocity, force. Click in the table cell to select the appropriate category. The list of categories is predefined.

Name

This is the name of the unit. This field can be left empty as long as no other unit has the same abbreviation. See also ↗ *Abbreviations, page 29*.

Description

Here, you can enter additional information or a description regarding the unit. This information is only visible in this table.

Active

This property determines whether the unit is taken into account or not. If you uncheck the checkbox the unit will be ignored.

Per unit system a unit has the following properties:

Abbreviations

This is the unit symbol which stands next to the signal value if the unit table is enabled. This field cannot be empty. See also ↗ *Abbreviations, page 29*.

Scaling factor (not available for primary unit system)

This is the factor that a signal value in the primary unit system should be multiplied with to get the correct value in the destination unit system.

Scaling offset (not available for primary unit system)

This is an additional offset which should be added to or subtracted from the calculated value in the destination unit system.

See for example the temperatures in the standard unit table.

The scaling always refers to the primary unit system (metric in the standard table). Therefore, the columns *Scaling factor* and *Scaling offset* are not available for the primary unit system.

Testing the unit conversion

Click on the corresponding icon in the tool bar to show or hide additional test columns. You can enter a value in the test column of one unit system and *ibaPDA* will calculate immediately the values for all other unit systems (works in both directions).

Grouping units

Click on the corresponding icon in the tool bar to group the units by categories. The groups are built by category names in ascending alphabetical order.

Adding and deleting units

In order to add units to or delete units from the table use the symbol buttons on the right side of the table or the context menu inside the table. The delete command refers to the selected unit(s) in the table. Multiselection is possible by using <Ctrl> or <Shift> keys.

Showing and hiding columns

Open the context menu by a right mouse click on the header of the table. Select here which columns are to be displayed. *Reset* restores the default column selection and column arrangement. You can change the column arrangement in the table by drag & drop.

Furthermore, there is a filter row in the table header, which can be used for filtering the table by content.

You will find more detailed information about the filter functions in Part 7 of the *ibaPDA* manual, in chapter *Advanced filter in tables*.

4.4.3 Abbreviations

Basically, the abbreviation of a unit must be unique in order to identify the unit and convert the values correctly. Empty abbreviations are not allowed. An error symbol in the table indicates an invalid abbreviation. The tooltip provides more information when hovering the mouse about.

However, there are use cases where the same abbreviation is needed multiple times, e.g. when converting kilometers (km) in miles (mi) and also kilometers (km) in nautic miles (nmi).

Length			☒	cm	in	0,3937007874	0
Length			☒	m	ft	3,280839895	0
Length			☒	km	mi	0,6213711922	0
Length			☒	km	nmi	0,5399568035	0
Length	in		☒	µm	in	3,93701E-05	0

In such cases just add a name to the unit. Later, this name will be shown together with the abbreviation, when you select the unit in the signal table.

Length			☒	m	ft	3,280839895	0
Length			☒	km	mi	0,6213711922	0
Length	nautic	nautical mile	☒	km	nmi	0,5399568035	0
Length	in		☒	µm	in	3,93701E-05	0

In this example, the name “nautic” is added to the abbreviation. A tooltip on the abbreviations indicates the display name of the unit.

mi	ft
km	mi
km	nmi
µm	in
km (nautic)	
W	W

For a better understanding you may enter additional information in the *Description* column.

Another use case of multiple usage of the same abbreviation is, that a unit has the same abbreviation in different unit systems.

For example, you can use the abbreviation “s” for second in all unit systems.

If this unit is identical over all unit systems, i.e. the scaling factor is 1 and the scaling offset is 0, then the unit abbreviation is displayed without any supplement.

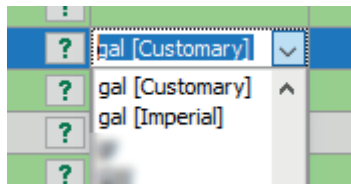
If the unit abbreviation is the same but the scaling factor is not 1, then the unit system is indicated after the abbreviation.

This is the case, for instance, when the unit systems Imperial Units and Customary Units are configured. In both unit systems there are identically named units such as gallon (gal) or fluid ounce (fl.oz.).

If you hover the mouse about the abbreviation, the tooltip will show the corresponding unit system.

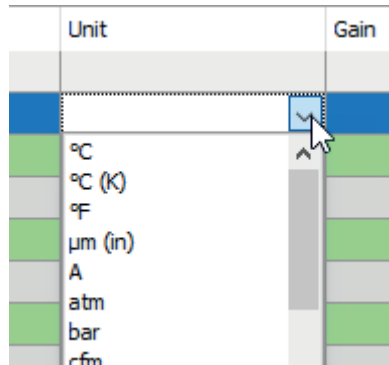
Imperial			Customary		
Abbreviation	Scaling factor	Scaling offset	Abbreviation	Scaling factor	Scaling offset
gal	=	=	gal	=	=
gal	0,2199692483	0	gal	0,2641720524	0

In the same way, the unit will be offered for selection in the signal tables.



4.4.4 Using the unit system in the signal table

If you have enabled the unit table (see [↗ Configuring and enabling the unit table, page 23](#)), the units are available for selection in the signal tables in the *Unit* column. You can either select a unit from the unit table or you can enter another unit into the field.



Units in the signal table, which are also available in the unit table are displayed without any markup.

Depending on the setting for checking the units, those units which are not part of the unit table will be marked with a warning or error symbol.

General ~ Analog ⏏ Digital				
Name	Unit	Gain	Offset	Active
18 114 Temp. before F1	°C		1	0 ☒
19 115 M3 center deviation	mm		1	0 ☒
20 116 ThicKn. dev. beh. F7	⚠ μm		1	0 ☒
21 117 Delta-width beh. F7	mm		1	0 ☒
22 118 Strip width beh. F7	mm		1	0 ☒

In this example, the unit μm is not part of the unit table.

The tooltip on the warning or error symbol provides more information.

		1	0	☒	Unit check with warning
⚠ μm		1	0	☒	
mm		1	0	☒	
mm	⚠ The unit of [3:20]: μm is not part of the unit table.		0	☒	

		1	0	☒	Unit check with error
❌ μm		1	0	☒	
mm		1	0	☒	
mm	❌ The unit of [3:20]: μm is not part of the unit table.		0	☒	

If the unit check is disabled there will be no indication.

You can use units even if they are not part of the unit table. However, those units will not be considered when changing the unit system and hence the values will not be converted.

Measured values in different units

When configuring the signals, you can mix units from different unit systems. Therefore, all units which are part of the unit table are available in the selection list and not only those units from the primary system.

If, for instance, two temperature gauges transmit their measured values like one in °C and the other in °F, you can enter the different units in the signal table. The conversion of the values into the unit of the active unit system for display is done automatically where needed.

Note

The signal tables of the I/O Manager, always show the signals with their configured units. This means that the actual values shown in the modules or diagnostic modules are always the received values but not the converted values.

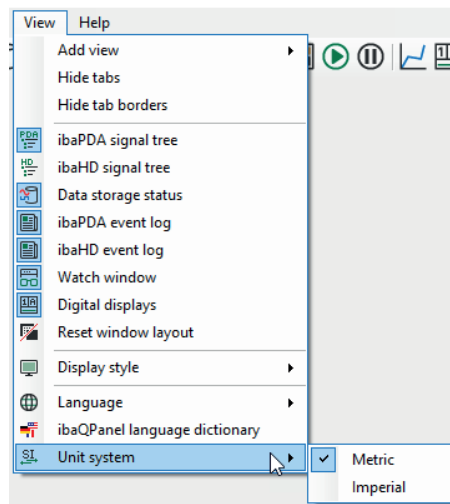
4.4.5 Changing the unit system

The metric unit system is selected, when the unit table is enabled for the first time. After changing the unit system, the new unit system will be stored as active unit system in the registry.

There are different methods to change the unit system.

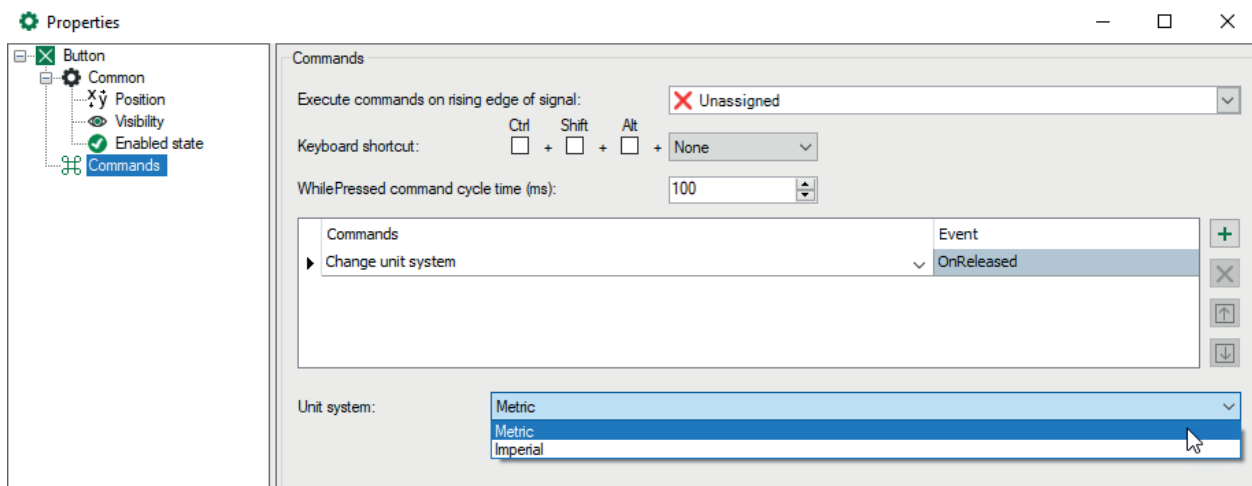
Switching in the View menu

► In *ibaPDA* client go to menu *View - Unit system* and select the desired system.



Switching by button in ibaQPanel

The button in *ibaQPanel* has a new command *Change unit system*.



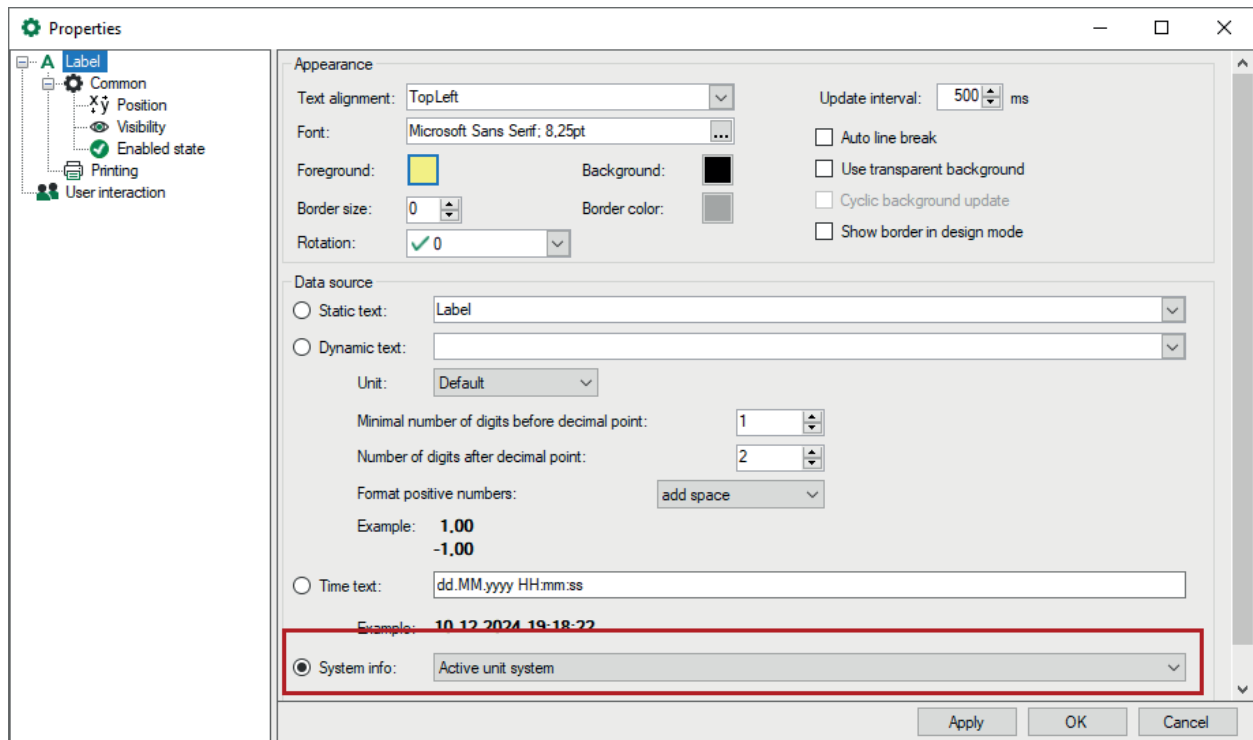
In the settings of the command you can define the unit system to be activated when clicking the button.

4.4.6 Indicating the active unit system

ibaPDA offers several features for users being informed about the currently active unit system.

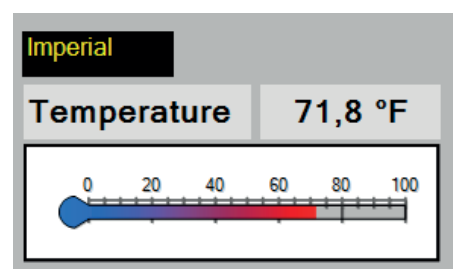
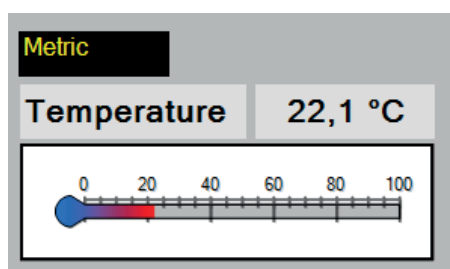
Indication in ibaQPanel

In the properties of the *Label* control you can select *Active unit system* for *System info* in the *Data source* area.



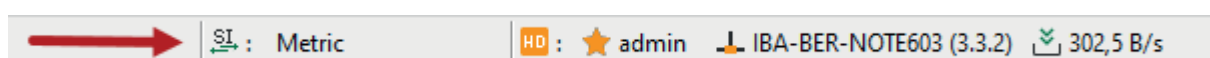
If you place this label at a highly visible position in the QPanel, then the actual unit system is always noticeable.

The following example shows such a label in the upper left corner.



Indication in the ibaPDA client status bar

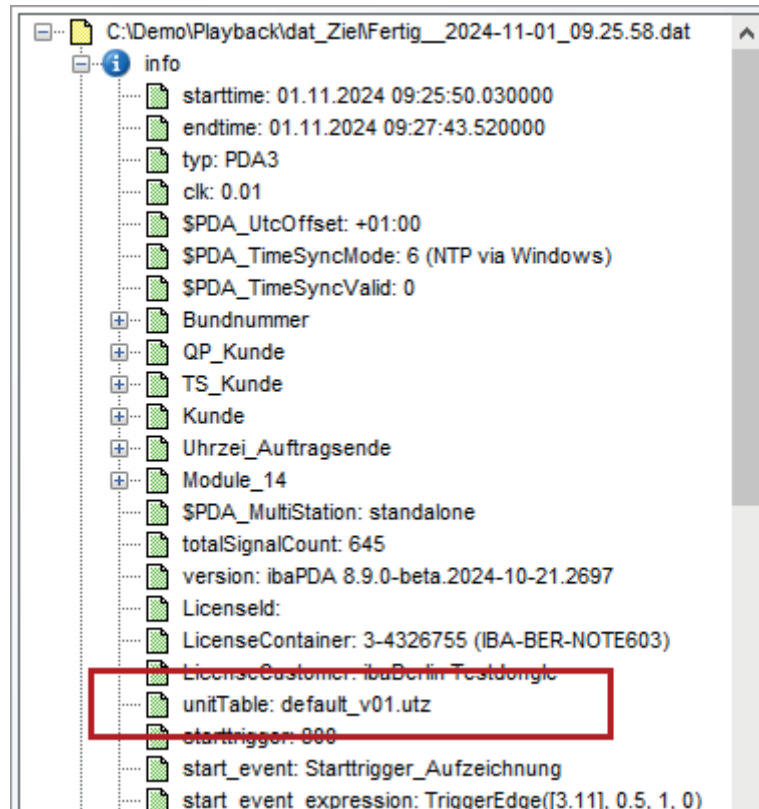
If the unit table is enabled, the actual unit system is indicated in the status bar of the *ibaPDA* client.



4.4.7 Unit table in the data file

If the unit table has been enabled, it will be stored automatically in the data file.

If a unit table is stored in the data file, then the info-field “unitTable” is available in the info-branch of the data file. The following picture shows the view in *ibaAnalyzer*.

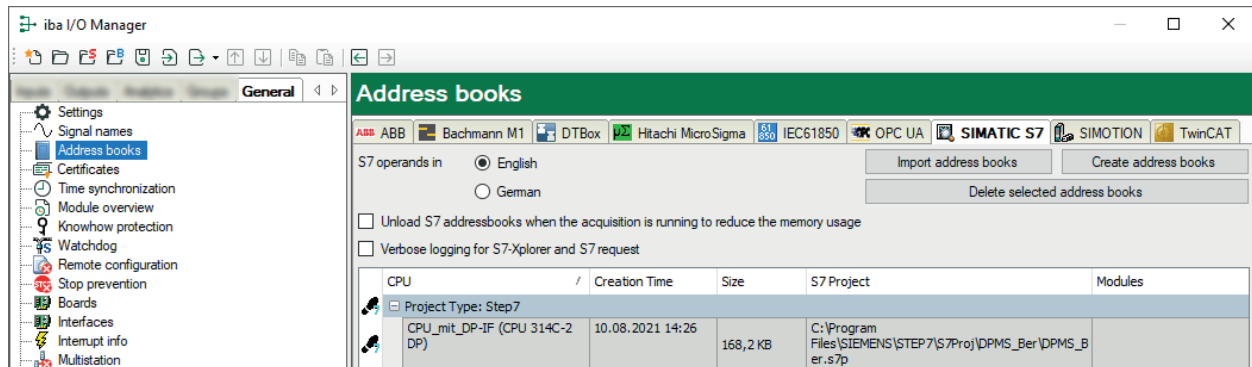


If no unit table is embedded in the data file, the info-field is not available.

Next to the name of the info-field you'll find the name of the unit table as it was configured in the I/O Manager. The filename extension of the unit table is *.ut^z, because it is stored as a compressed file in the data file.

4.5 Address books

The *Address books* node is used to configure and manage address books, which are needed for address book-based connections to other systems, such as some Xplorer or SIMATIC interfaces.



The respective address book tabs are only visible if appropriate interfaces and/or licenses are active on the *ibaPDA* system.

The basic structure is the same for all tabs.

In the upper part there are some settings that depend on the connection type as well as the Button for importing, generating and deleting address books.

Below is a table listing all of the address books available on the system, with reference to the PLC, the creation date and – if available – the project. The last column lists the modules that have been configured for the specific CPU.

Address book tabs are available for the following systems:

Name	Tab visible ...
ABB	... with license <i>ibaPDA-Interface-ABB-Xplorer</i>
B&R	... with license <i>ibaPDA-Interface-C&Rodesys-Xplorer</i>
Bachmann M1	... with license <i>ibaPDA-Interface-Bachmann-Xplorer</i> , <i>ibaPDA-Request-Bachmann-M1</i>
Codesys	... with license <i>ibaPDA-Interface-Codesys-Xplorer</i>
DTBox	... with license <i>ibaPDA-Request-DTBox-xxxx</i>
Hitachi MicroSigma	... with license <i>ibaPDA-Interface-Hitachi-MicroSigma</i>
IEC61850	... with license <i>ibaPDA-Interface-IEC61850-Client</i>
Logix	... with license <i>ibaPDA-Interface-Logix-Xplorer</i>
OPC UA	... with license <i>ibaPDA-Interface-OPC-UA-Client</i>
SIMATIC S7	... with licenses <i>ibaPDA-Interface-S7-Xplorer</i> , <i>ibaPDA-Request-S7</i> or <i>ibaPDA-Request-S7-UDP</i> ... if an S7 module (all types) is included in the I/O configuration.
SIMOTION	... with license <i>ibaPDA-Interface-SIMOTION-Xplorer</i>
TwinCAT	... with license <i>ibaPDA-Interface-TwinCAT-Xplorer</i> , <i>ibaPDA-Request-TwinCAT</i>

Table 1: Systems with address book support

Simatic S7 example**S7 operands in English/German**

Here you can choose the language in which the S7 operands will later be available when browsing through the signal tables.

<Create address books>

This button opens the "S7 address book generator" dialog. You can select the source directory of an S7 project for creating the S7 address book. This can be a local or network drive.

<Import address books>

Import address books which are already available as ZIP files.

<Delete selected address books>

Delete address books from the *ibaPDA* server's directory.

Unload S7 address book when the acquisition is running to reduce the memory usage

By enabling this option, the address book is outsourced to the hard disk during the acquisition in order to free up the main memory for the acquisition.

4.6 Certificates

The *Certificates* node gives you access to the central certificate store. This node is available in both the I/O manager and the data store index. From both locations, you can access the same pool of certificates and can manage the relevant certificates.

4.6.1 Certificates

For secure and encrypted TLS/SSL communication between a client and a server, so-called certificates are used because they enable secure authentication.

Certificates used by iba programs can be administered in a central certificate store.

Before a client can connect to a server, an application certificate must first be configured. Certificates can be provided from both the server and client side. Communication can only take place if each partner trusts the partner certificate.

Certificates can either be exchanged spontaneously when a connection is established or registered as trusted in advance. If a previously unknown certificate is offered when a connection is established, the user must manually accept or reject the certificate. Accepted certificates are automatically entered into the table in the *certificate store* and marked as trusted. If the certificate is rejected, then no communication will take place.

You can also register certificates and then mark them as "not trusted". Communication with partners with such certificates is then always denied. Once certificates have been registered, i.e., entered in the table in the certificate store, the user will no longer be notified or prompted when communication is established – regardless of whether the certificates are marked as "trusted" or "not trusted".

Note



Some interfaces in *ibaPDA*, such as the e-mail output, use Windows certificates. Other features, such as OPC UA server or MQTT data stores use certificates from the central certificate store of *ibaPDA*.

4.6.2 Central certificate store

All registered certificates are listed in a table. The following figure shows the certificate store in the I/O Manager of *ibaPDA* as an example.

Certificates				
Name	Properties	Issued By	Expiration Date	Used By
OPC UA Server SHA-256	 	OPC UA Server SHA-256	20.12.2031 12:29:10	OPC UA server
DigiCert Global Root G2		DigiCert Global Root G2	15.01.2038 13:00:00	
ibaPDA@Station123		ibaPDA@Station123	07.02.2032 10:26:08	
ibaPDA@PCOneMqtt	 	ibaPDA@PCOneMqtt	11.02.2032 10:16:01	

Each row refers to one certificate.

The columns *Name*, *Properties*, *Expiration Date* and *Used By* are displayed by default.

If needed, you can add or remove other columns via the context menu.

Certificates				
Name	Properties	Issued By	Expiration Date	Used By
OPC UA Server SHA-256	 	OPC UA Server SHA-256	20.12.2031 12:29:10	OPC UA server
DigiCert Global Root G2		DigiCert Global Root G2	15.01.2038 13:00:00	
ibaPDA@Station123		ibaPDA@Station123	07.02.2032 10:26:08	
ibaPDA@PCOneMqtt	 	ibaPDA@PCOneMqtt	11.02.2032 10:16:01	

The *Name* column holds the name of a certificate. Different certificates may have the same name, thus it is not unique. Only the finger print of a certificate is unique.

The symbols in the *Properties* column have the following meaning:

Symbol	Meaning
	The certificate is trusted as long as it has not expired.
	This certificate is not trusted.
	A private key for this certificate is available.
	This certificate can also be used for user authentication.
	This certificate is invalid. If the certificate is invalid because it expired, the expiration date is highlighted in red color.

Table 2: Symbols for certificate properties

The *Used By* column shows by which application/function the certificate is used. In the example shown in the figure above, the certificate *OPC UA Server SHA-256* is used by the OPC UA server in *ibaPDA*. This means that this certificate was selected during configuration of the OPC UA server.

Note

One characteristic of *ibaPDA* is that certificates from different areas are managed in the certificate store: from the I/O Manager and from the data storage configuration.

For that matter the *Used By* column field has a link function. Via a double-click on a filled field you jump to the respective dialog.

If the entry belongs to the other manager, the link does not work. In the example above, a double-click on the “OPC UA-Server” entry would open the configuration dialog of the OPC UA server, provided it’s done in the I/O Manager. If you had opened the certificate store in the data storage configuration, the link to the OPC UA server does not work.

Vice versa, jumping to an “MQTT data store” would only work if the certificate store was opened in the data storage configuration.

The column *Expiration date* shows the date which marks the end of the validity of the certificate. Beyond this date the certificate cannot be used anymore. You have to renew the certificate or replace it by another, yet valid certificate. A red highlighted date indicates an expired certificate.

4.6.3 Manage certificates

The central certificate store is used to manage the certificates. Here you can add, create and delete certificates.

In the certificate store toolbar you will find a number of buttons with the following functions:

Button	Function
	This button opens a dialog that allows you to load an existing certificate file. Various file formats are supported (.der, .cer, .crt, .cert, .pem, .pfx, .p12). If you have a certificate with an unknown file extension, expand the file filter to "*.*)" and try to open the file anyway. This works in most cases.
	This button opens a dialog that lets you create a new certificate file.
	This button lets you export a certificate to a file to register it for Windows or another application, e.g. on an OPC UA client. Multiple file formats are supported here as well.
	Use this button to remove the selected certificate from the table.
	Use this button to open a window with all detailed information about the selected certificate.
	Use this button to designate the selected certificate as "trusted".
	Use this button to designate the selected certificate as "not trusted". However, the certificate will still remain in the certificate store table. However, certificates that are not trusted are not available in the selection list for use in the corresponding configuration dialog.
	With this button you can define whether a certificate can also be used for user authentication for OPC UA.

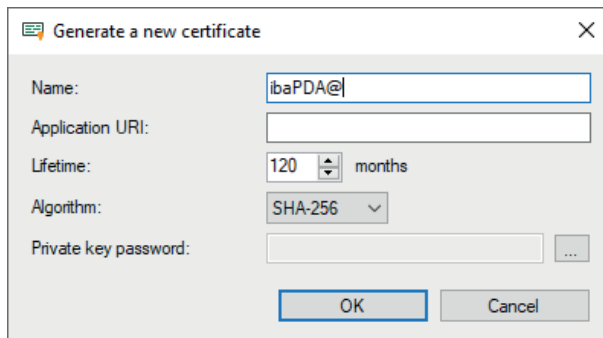
Table 3: Buttons in the toolbar for certificate management

The commands always refer to the certificate selected in the table, which is indicated by an arrow on the left at the start of the line.

4.6.3.1 Generate a new certificate

If no certificates are available to load, it is necessary to generate one.

1. Click the button  and the following dialog box will open:



2. Enter a name of your choice for the certificate.
3. If required, enter an Application URI.
The URI (Uniform Resource Identifier) is a global unique identifier for the application. If you do not fill in this field, a standard URI will be generated, provided that the OPC UA client verifies an Application URI. This standard URI consists of the machine name and the name of the application:
`urn:machinename:applicationName.`
4. Define the desired validity period (lifetime) of the certificate.
5. Select the desired hash algorithm for the encryption.
You have the choice between the algorithms SHA-256, SHA-384 and SHA-512.
Make sure that the other communication partners support the selected algorithm too.
6. Define a password for the private key. If no password has been entered, the <OK> button remains inactive. To assign the password, click the <...> button and enter the password twice and confirm with <OK>. There are no special requirements for the password. Keep the password in a safe place so that the self-generated certificate can be exported and used for Windows or other applications.
7. Close the dialog with <OK>.

The new certificate is now entered into the list held by the certificate store and immediately assigned the properties "trusted" + private key.

You can now also export the certificate and register it with the communication partner, e.g., an OPC UA client. Afterwards, the client can then connect to *ibaPDA* (OPC UA-Server).

4.6.3.2 Add certificate

1. In the certificate store toolbar, click the button  .
A dialog will open that lets you navigate to the desired certificate file and open it.
Different file formats are supported (.der, .cer, .crt, .cert, .pem, .pfx, .p12).
If you have a certificate with an unknown file extension, expand the file filter to "*.*)" and try to open the file anyway. This works in most cases.
2. When the certificate is loaded, it appears in the certificate store list.
3. If you have not already done so, trust the certificate.

Certificates can sometimes be added without manual import.

Thus, during the first connection attempt by an OPC UA client to the OPC UA server (*ibaPDA*), the application certificate of the OPC UA client is automatically added to the certificate list and initially rejected.

Once you have selected the OPC UA client certificate in the list and confirmed it as trusted with the button  the OPC UA client can subsequently connect automatically.

Use the button  to reject a certificate at any time or to classify it as not trusted.

4.6.3.3 Export certificates

All certificates in the certificate store can be exported individually as a file and subsequently used for Windows or other applications. An exported certificate can also be re-imported into.

To export a certificate, first select the desired certificate in the table and then click the button  in the toolbar for the certificate store.

If you wish to export a certificate without a private key, a dialog that lets you save the file opens immediately.

If the certificate to be exported has a private key, there are some options.

First, you will be asked if the existing private key should be exported as well. If you answer "no", the certificate will be saved immediately, just like a certificate without a key.

If you answer "yes", then you must enter the correct password afterwards. The correct password is the password used when importing or generating the certificate. If the password is correct, the certificate can be saved as a PFX-file. This file is password protected and contains the certificate and the private key.

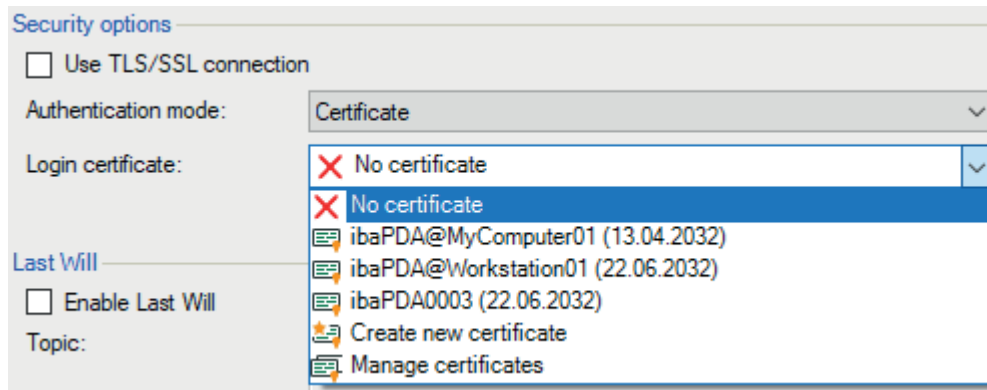
If the password is incorrect, the certificate will not be exported.

Under certain circumstances, a certificate with a private key may be stored, however the key is not password protected. In this case, the certificate can only be exported without a private key. You will then be notified accordingly.

4.6.4 Use certificates

At the points where certificates are applied, you will find a drop-down list offering the available certificates for selection.

There are the following options:



- No certificate: No certificate is used. As a rule, this leads to an invalid configuration.
- Available certificates: All certificates are displayed that are contained in the central certificate store, are valid, and are suitable for use at this point.
- Create a new certificate: The dialog for creation of a certificate opens. If the operation is successful, the new certificate is also selected immediately. If not, "No certificate" is selected.
- Manage certificates: Calling up the central certificate store.

Note



The selected certificate is saved in the registry file of the computer. In case of a new configuration, the same certificate will be selected unless another certificate is actively selected.

Other documentation



For more information on the use and functions of the certificates, please refer to the descriptions and manuals of the relevant applications, such as *ibaPDA*, div. *ibaPDA* interfaces, *ibaHD-Server*, *ibaDatCoordinator* etc.

4.6.5 Save and protect certificates

The certificates are stored in the `settings.xml` file, which is located in the folder `c:\ProgramData\iba\Name of application\Certificates`. This file is automatically encrypted.

There are a number of measures whereby certificates with private keys can be used to protect your identity or that of your organization. Specifically, these are measures that make their simple export and reuse in Windows or other applications more difficult.

- Certificates are always stored in encrypted form.
- For certificates with a private key, the input of a password is required...
 - when a new certificate is generated
 - when a certificate with a private key is exported
 - when a certificate with a private key is imported
- Certificates with a private key can only be exported if there is also a password for the key. If there is no password or the password is unknown, the certificate can no longer be exported. Therefore, keep the passwords in a safe place.
- The password for a private key cannot be changed.
- It is not necessary to enter a password to use a certificate. The `settings.xml` file can be copied from one installation to another to transfer the certificates there. Password entry is not required for this either.

Should the private key fall into the wrong hands, many types of misuse are possible. Therefore, make sure that the passwords are kept safe.

4.6.6 Check information about a certificate

If you want to get more detailed information about a certificate then proceed as follows:

1. Go to the certificate overview in the certificate store.
 2. Select the certificate in question by clicking on the corresponding row.
 3. In the certificate store toolbar above the table, click on the info-button (i).
- A window opens showing all relevant information, which are available for this certificate.

4.7 Time synchronization

ibaPDA offers several methods of time synchronization. The purpose of time synchronization is to set the time of the *ibaPDA* system in accordance with a central time server. In this case, *ibaPDA* is a "time slave". In the case of PTP time synchronization, *ibaPDA* can itself be a time server for other systems and be configured as a "time master".

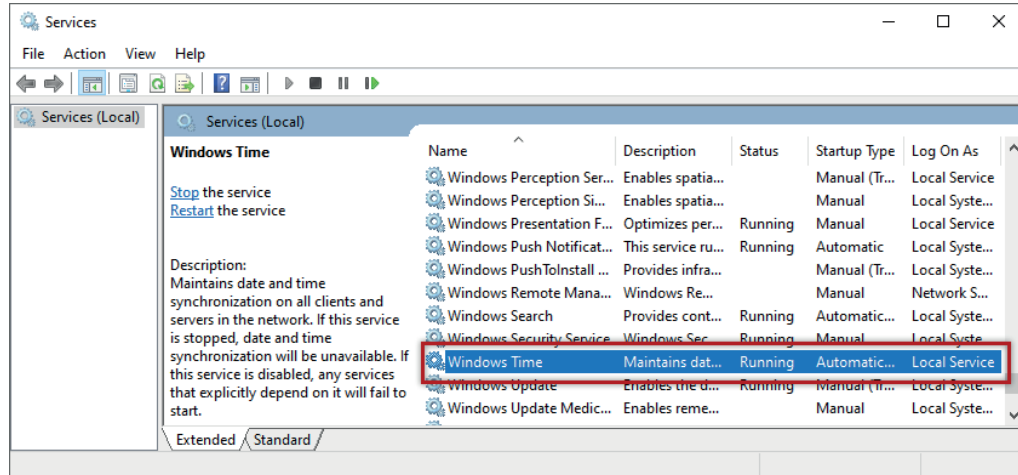
Note



You should use the "Use system time" option for data storage when time synchronization is active. Otherwise, the start time of the data file will not be accurate.

Note

To avoid conflicts between the Windows time management and the *ibaPDA* time synchronization, you should disable the local service *Windows time source* on the computer as soon as you enable one of the *ibaPDA* time synchronization modes.



There are the following ways to synchronize time:

- DCF77
- IEC 1131
- PTPv2
- *ibaClock*
- NTP via Windows

Depending on the use of special interfaces other methods for time synchronization may be offered.

4.7.1 DCF77

With this method, you can configure an external time synchronization of the system by means of the DCF77 signal. The system time of the *ibaPDA* computer is set. Other time sources may also work provided they generate a signal in compliance with the DCF77 convention.

What is DCF77?

The acronym DCF77 is the label for a coded time signal which is transmitted on the standard frequency of 77.5 kHz on long wave. It can be received within a range of 2,000 km (1,243 miles) around Frankfurt/Main, Germany.

DCF77 derives from the international frequency list where D stands for Deutschland (Germany), C for long wave, F for "near Frankfurt" and 77 for the carrier frequency.

The time transmitted is the official time of the Federal Republic of Germany and thus the Central European time or the Central European daylight saving time.

The time base currently consists of three cesium atomic clocks, operated by the National Metrology Institute of Germany (PTB).

There are many devices on the market for receiving and converting the signal.

DCF77 signal and time information

The time information is transmitted as a digital signal on the normal carrier frequency of 77.5 kHz. The digital signal is formed by a negative modulation of the signal (decreasing the amplitude of the carrier frequency down to 25 %) in a one-second cycle.

The modulation starts at the beginning of each second from second 0 to 58 of one minute. In the 59th second, no lowering takes place and the following second mark indicates the beginning of the next minute. The length of the amplitude lowering at the beginning of a second provides the binary value of the digital signal. A 100 ms lowering stands for the value "0" (FALSE); 200 ms for "1" (TRUE).

Thus, with every second, part of the time information is transmitted so that it takes at least one minute for all of the time information to be transmitted.

The time and date are encoded by 58 digits, i.e., 58 seconds.

In addition to the pure time and date information, more information is included, such as operating information (seconds 1 to 14) as well as the transmitter type used, time zone, daylight saving time and switching seconds (second marks 15 to 19).

Connection to ibaPDA

The integrated DCF77 driver decodes the time information from a digital input signal that must be generated in compliance with the DCF77 standard.

If the *ibaPDA* system is installed in the range of the original DCF77 signal, there are several devices on the market that convert the DCF77 radio signal into a discrete digital signal. ASCII strings containing the time information cannot be processed by *ibaPDA*.

In order to be able to transmit the DCF77 time to *ibaPDA*, the discrete pulse outlet of the DCF77 receiver should be connected to a digital input on an *ibaPADU* or *ibaNet* device.

In the configuration dialog of the I/O Manager of *ibaPDA*, you then just have to select this input as the DCF77 signal.

Devices which have already been used with *ibaPDA*:

- Meinberg COM52HS DCF77 receiver for DIN-rail mounting
- Meinberg RU226: DCF77 receiver in compact housing

Since using a simple digital input, it is possible to synchronize the *ibaPDA* system time with external time sources, such as clocks.

ibaPDA can thus be synchronized even in regions where the DCF77-signal is not available.

4.7.1.1 DCF77 Configuration

The configuration dialog offers the possibility to configure two DCF77 signals. In this way, a redundant time management can be realized.

Time slave

Time master

Time synchronization mode:

DCF77

Primary DCF77 signal:

×

Inactive

▼

Polarity:

Active low

▼

Time signal is:

Local time

▼

Status:

Last received bit:

Last received time:

Secondary DCF77 signal:

×

Inactive

▼

Polarity:

Active low

▼

Time signal is:

Local time

▼

Status:

Last received bit:

Last received time:

Difference between system time and received time:

System clock frequency adjustment:

Primary DCF77 signal

In this field, select the digital input signal which is used for transmitting the decoded DCF77 signal. Click on the little arrow button in the field. A signal tree showing all available modules and signals opens. Select the appropriate signal by mouse click.

Polarity

In this field, select the correct polarity of the pulse. Depending on the device type you are using, the control pulse may be logical "1" (TRUE) or logical "0" (FALSE). Please refer to the manual of the DCF77 receiver and set the correct polarity.

Time signal is

Open this selection field in order to set the time mode of the DCF77 signal. Available for selection are:

- Local time
- UTC time (Universal Time Coordinated)
- UTC time with DST compensation

Comment: DST stands for daylight saving time. In this mode, *ibaPDA* will convert the time from UTC to local time just like in the normal UTC time mode. If the daylight saving time bit is set in the telegram, *ibaPDA* will subtract 1 hour to compensate the daylight saving time. In all other modes, the daylight saving time bit is ignored.

Status

Clear text messages regarding the DCF77 connection are displayed here.

Status	Meaning
Inactive	DCF77 synchronization is disabled.
Invalid	An invalid sequence of bits was received (parity error).
Valid	<i>ibaPDA</i> is synchronized with DCF77, a sequence was received.
Sequence received	A complete sequence (bits 0 to 59) was received.
Signal received	A minute pulse was received.
Signal lost	No minute pulse was received.

Table 4: Status of a DCF77 connection

Last received bit

This line displays the bit no. and the value of the last received bit. Bit no. 0 ... 59, value: 0 (FALSE) or 1 (TRUE). With the DCF77 decoding scheme, it is possible to get the meaning of the bit. For service purposes only.

Last received time

This line displays the last received time as clear text in the format dd/mm/yy hh:mm:ss.ms. This shows the time of the last synchronization of the system.

Difference between system time and received time

The received time is cyclically compared with the system time whereas deviation is determined. Depending on whether the difference is rather positive or rather negative, a corresponding frequency adjustment of the internal system clock is calculated.

System clock frequency adjustment

The value of the frequency adjustment has been determined from the difference between the system time and the received time. Thus, a system clock which is too fast or too slow is corrected.

4.7.2 IEC 1131

What is IEC 1131 time?

In compliance with the standard IEC 1131 (EN 61131), time can be transmitted by a DWORD value. This time represents the number of seconds that have passed since 1st January 1970, 00:00.

A second DWORD with a microsecond counter that goes from 0 to 999999 is required by *ibaPDA* for higher accuracy. The counter is reset at the start of every second.

The time source can be any system, time server or PLC capable of generating such a time value.

IEC 1131 time settings

IEC 1131 time signal

Select the analog signal carrying the time value from the drop-down list.

Microsecond signal

Select the analog signal carrying the microsecond value from the drop-down list.

Time signal is local time/UTC time

Check the appropriate option according to the time format generated by the time source. If "Local time" was selected, *ibaPDA* will accept that time without any conversions. If "UTC" is selected, *ibaPDA* will convert the time to the local time according to the time zone settings of the computer.

Status

The status field shows the current status of the time synchronization. The time synchronization only works when the acquisition is running.

Difference between system time and received time

The received time is cyclically compared with the system time whereas deviation is determined. Depending on whether the difference is rather positive or rather negative, a corresponding frequency adjustment of the internal system clock is calculated.

System clock frequency adjustment

The value of the frequency adjustment has been determined from the difference between the system time and the received time. Thus, a system clock which is too fast or too slow is corrected.

4.7.3 PTPv2

What does PTP mean?

The Precision Time Protocol (PTP) is a communications protocol for synchronizing the system time of different systems (PC, PLC and other devices) in a network. The protocol is defined in accordance with Standard IEEE 1588.

In a typical PTP topology, there is one time master and one or more time slaves that synchronize to the master. The highest synchronization accuracy of software-based PTP applications, as in *ibaPDA*, is 5 ms.

ibaPDA can be configured as slave. When an *ibaPDA* system is configured as a time slave, it synchronizes to a time master in the network, provided one is available.

ibaPDA supports the standard PTPv2 according to IEEE 1588-2008 (or "IEEE 1588-V2"). With the new process, only the multicast address 224.0.1.129 is used in all domains to distribute the time in the network. The domain number can have the value 0 to 255 and is part of the time telegram.

PTP settings, time slave

Network interface

Here, choose the network interface of the *ibaPDA* computer to be used for time synchronization.

PTP domain

If *ibaPDA* is configured as a slave, set the same PTP domain on which the time master is also sending.

If *ibaPDA* is configured as a master, you can set any PTP domain between 0 and 255.

Delay mode

The accuracy of the PTP protocol is based on the continuous computation of delays which can occur at any time when transmitting information in a network and taking these delays into account for the time synchronization of the time slaves. You can choose between two methods which are available for this purpose.

- **End-to-End:** This method determines the delay between time master and time slave over the entire way through the network. Therefore, the slave sends a "Delay request" message to the master, which in return responds with a "Delay response" message. This is used to calculate the transit time of the message through the network. The method is suitable for any network even if network components such as switches, hubs or routers are not necessarily PTP-compliant. However, the achievable accuracy of the end-to-end method in a network is lower than the accuracy of the peer-to-peer method in a network of the same topology and

with "PTP-transparent clocks" Moreover, the end-to-end method generates a higher network load, particularly if multiple slaves are in the network.

- **Peer-to-Peer:** This method calculates the delay between the ports of adjacent devices by means of so called "Peer-delay-request/-response" messages. All network devices along the way between time master and time slave determine the delay on the section towards the devices they are directly connected with and pass on this information to the next device. The total delay between master and slave is calculated by so called PTP-transparent clocks (switch, hub or router) by writing the received delay value - supplemented with the delay which may be generated by the device itself - into the correction field of a PTP event message (e.g. sync telegram). The peer-to-peer method applies only to networks which are fully PTP-compliant. It generates less network load and it is less sensitive regarding asymmetries or changes in the network compared to the end-to-end method.

Status

The status field shows the current status of the time synchronization. The time synchronization only works when the acquisition is running.

Current master

If *ibaPDA* is configured as a slave, the IP address of the time master is shown here for information purposes.

Difference between system time and received time

The received time is cyclically compared with the system time whereas deviation is determined. Depending on whether the difference is rather positive or rather negative, a corresponding frequency adjustment of the internal system clock is calculated.

System clock frequency adjustment

The value of the frequency adjustment has been determined from the difference between the system time and the received time. Thus, a system clock which is too fast or too slow is corrected.

Message counters

This overview shows different message counters. There is one counter per telegram type. If there is a master-slave connection, the counters usually count up: *sync*, *announce*, *delay request* and *delay response*.

4.7.4 ibaClock

What is ibaClock?

ibaClock is a compact device for the time synchronization of several connected *ibaPDA* systems.

ibaClock sets the time base for all connected *ibaPDA* systems as the central master clock and ensures an exact and timely synchronized acquisition of the data. An accuracy of better than 150 ns can be achieved.

Additionally, *ibaClock* can synchronize more members by using other time protocols:

- PTP
- NTP
- DCF77

The highly accurate time is received via an active GPS antenna.

Alternatively, IRIG-B or PTP can be used as external time source.

In order to be able to use *ibaClock*, an *ibaFOB-... D* card must be inserted into the computer.

You also need to first configure the device in the I/O Manager under the *ibaFOB-... D* interface.

ibaClock settings

Other documentation



A detailed description of how to set up the *ibaClock* can be found in the device manual.

Information is displayed in the *Time slave* tab of the *time synchronization* node about the device, the clock and the system time synchronization. You cannot adjust any settings here. The information is identical to that in the *Diagnostics* tab of the *ibaClock* module under the *ibaFOB-... D* card.

<Write firmware> button

With this button, it is possible to perform firmware updates. Select the update file, `clock_v[xx.yy.zzz].iba` in the browser and start the update with <OK>.

Note



This process may take several minutes and must not be interrupted.

Only perform a firmware update after consulting the iba support team.

<Reset to factory defaults> button

Click this button to reset all settings to the factory settings after confirming the following prompt with <Yes>.

4.7.5 NTP via Windows

With this setting, *ibaPDA* simply uses the Windows internal time management via NTP.

The fields in this view are for display purposes only. You do not need to configure anything here.

The status information is only displayed when the acquisition is running.

Time slave	Time master
Time synchronization mode:	NTP via Windows
Status:	Receiving valid time: 23.06.2022 15:45:05.622
Windows Time service:	Running
Last sync time:	23.06.2022 15:07:08.179
Source:	iba-fue-dc02.iba-ag.local

The information is also written to the time synchronization info fields in the data file.

Via the functions *TimeSinceLastSync* and *TimeSyncStatus* in the expression builder, you can read out the interval to the last synchronization and the status, and can additionally display or record this.

Note



An internal *ibaPDA* and an internal Windows time synchronization should not be active in parallel. In such cases, a warning is issued during the validation of the I/O configuration, prompting the user to disable one of the timing modes.

4.8 Module overview

The *Module overview* node offers an overview of the time base settings of the modules and their signals. Here, it's easy to see the effect that changes of module timebases have on the output time and client update time. If you change the module timebases, the resulting values are accordingly updated here immediately.

Module overview

Acquisition timebase:

Minimum output time:

Interrupt cycle time:

Client update time:

Direction:

☒ Hide disabled modules

Interface:

☐ Hide modules with a fixed timebase

	Module			Timebase	Active signals		Configured signals	
	Number	Name	Type		Analog	Digital	Analog	Digital
	=	RIC	RIC	=	=	=	=	=
	0	Hydr. Adjustment	Playback	20 ms	32	1	32	1
	1	Shear / RSF / S1-S6	Playback	20 ms	32	12	32	30
	2	Stands 1-7 a roll forces	Playback	20 ms	32	0	32	0
	3	IBA-Logic	Playback	20 ms	24	20	32	28
	4	Shear	Playback	20 ms	1	0	1	0
	5	Virtual	Virtual	10 ms	0	0	0	0
	6	Shift register	Shift register	10 ms	0	8	0	8
	7	CoilNo	Text creator	10 ms	1	0	32	0

System time values (only display)

- *Acquisition time base*: Central acquisition time base, as it is set in the Register *General, Settings* node.
- *Interrupt cycle time*: Interrupt cycle of the acquisition
- *Minimum output time*: The fastest cycle in which signals that are configured as outputs can be updated and output. The shortest achievable cycle time of the outputs equals the smallest common multiple of all module time bases, but is at least 50 ms.
- *Client update time*: Cycle in which the appearance of the live data is updated on the connected clients. This value too, depends on the smallest common multiple of all module time bases. The minimum here is 200 ms.

Table settings

Hide disabled modules

If this option is enabled, modules which have been disabled in their module settings will not be shown in the table.

Hide modules with set time base

Some virtual modules, e.g. 16 and 32-bit decoders, are permanently bound to the central acquisition time base. If this option is enabled, the affected modules will not be shown in the table.

Direction

Choose which modules should be shown in the table: Inputs or Outputs.

Interface

If you want to show only modules of a certain interface, then you can select the interface here. Multiselection is not possible.

You can setup more filters in the columns.

Table contents

All configured modules are listed in the table. Make a double-click on a module to get to the settings of the respective module. There you may change the module settings.

In the column *Timebase* you can change the value right away. You'll see the effects of the change directly on the right side. Additionally, you'll get a note if a change of the output time is the consequence.

The *Active signals* column shows how many analog and digital signals of the respective module are active.

The *Configured signals* column shows how many analog and digital signals are configured in the respective module. Configured signals may also be inactive. Inactive signals do not account for the total number of used and licensed signals (bar graph on the bottom of the I/O Manager).

The *Samples/s* column is hidden by default. You can show the column by opening the context menu of the table header and selecting *Column/Band Chooser*. The values in this column show the number of samples per second acquired by each module. The values result from the number of active signals of a module divided by the timebase, given in seconds.

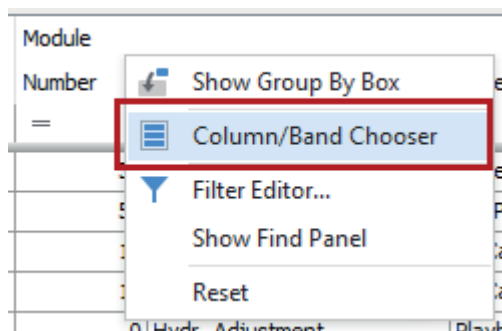
Customize table layout

You can adjust the layout of the table according to your needs by showing or hiding columns or column groups ("bands"). The column headers have two lines, where the upper line represents the bands and the lower line represents the columns.

Module					Active signals		Configured signals	
Number	Name	Type	Timebase	Samples/s	Analog	Digital	Analog	Digital
=	RIC	RIC	=	=	=	=	=	=

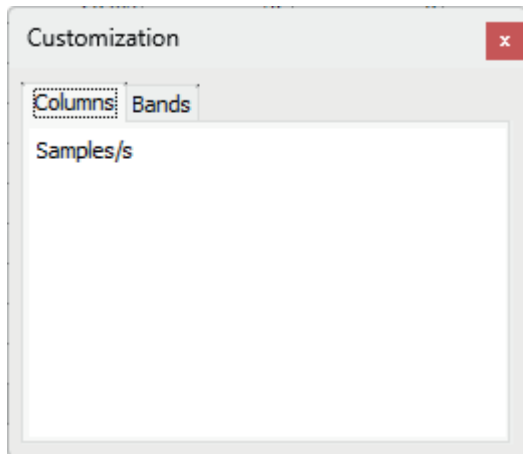
To customize the table layout follow these steps:

1. Open the context menu on the band or column headers. The context menus differ but both have the *Column/Band Chooser* command.



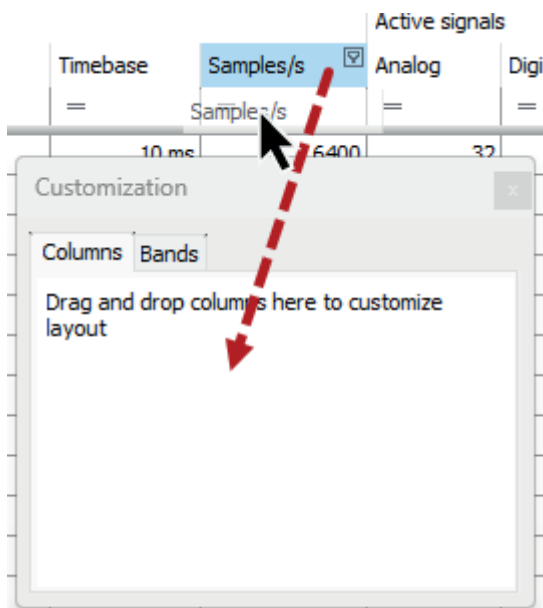
2. Select Column/Band Chooser.

The *Customization* dialog opens.



The *Columns* and *Bands* tabs contain the columns and bands which are currently hidden.

3. To show the column or band again just doubleclick on their name.
4. In order to hide columns or bands from the table, drag the column or band in question into the customization window.



Filter the table

You may filter the table entries in different ways:

- Via the filter row:
Select a search option via the <ABC> button or <=>. Then enter the search string or a value.
- Via the hidden filter button in the header. The button appears when moving the mouse over the respective column header.
Open the filter window with the filter button. Similar to Microsoft Excel you can now filter the entries in this column.
- Via the search row:
Via a right-click in the header you can open the search row by choosing <Show find panel> in the context menu. Enter here the search string or value. The entire table will be browsed for your search string or value and filtered.

For detailed information about the filter functions, please refer to part 7 of the manual, chapter *Advanced filters in tables*.

4.9 Knowhow protection

The *Knowhow protection* node offers mechanisms for protecting intellectual property associated with certain calculations and/or settings in *ibaPDA*, which are considered as user knowhow worth protecting.

4.9.1 Application range of know-how protection

In principle, the know-how protection can be applied to all module types which use profiles, such as

- InSpectra and InCycle modules
- Computation module
- Lookup table
- Parameter set
- Process condition

The following protective functions are realized:

- Protection against change
The protected elements cannot be changed without entering a password.
- Read protection
The configuration of the protected elements is not displayed without entering a password.
- License protection
The protected elements are only executed on systems that run with a previously registered dongle or registered soft license. Several license numbers can be registered.

The protection is realized via so-called protection schemes that, once defined, can always be applied again.

Knowhow protection

Protection schemes

ProtScheme01

Id: 6874370e-2999-47 Version: 1

Name: ProtScheme01

Author: iba AG

Copyright: This is a copyright text, which can be very long

Allow execution only on systems with a dongle having one of the following license numbers:

V987654
V654321

Change scheme

Apply changes

Cancel changes

Change password

Import scheme

Export scheme

Add connected dongle

Protect Unprotect

Protectable elements

Select...	Name	Scheme name	Scheme id	Scheme version
Type: InSpectra Auto-Adapting profiles				
☐	InSpectraLearnProfile1	-	-	-
☐	InSpectraLearnProfile2	-	-	-
Type: InSpectra Expert profiles				
☒	InSpectraProfile1	ProtScheme01	6874370e-2999-47	1
☐	InSpectraProfile2	-	-	-

The basic procedure is as follows:

1. Generating a protection scheme
2. Applying a protection scheme to an element

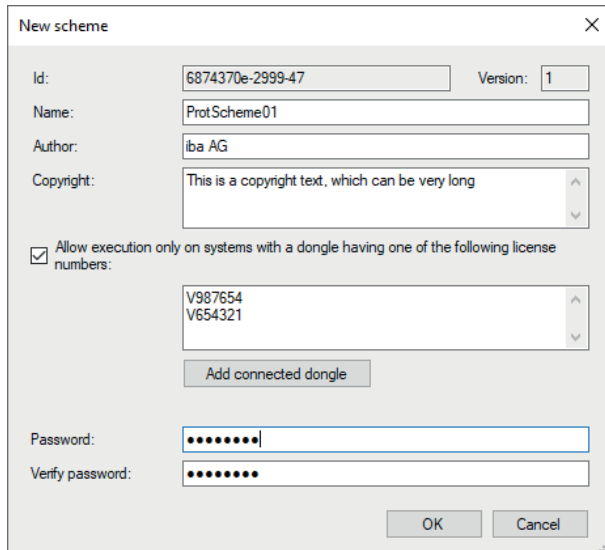
Tip



If you click on the gear icon then you get directly to the configuration dialog of the respective profile.

4.9.2 Creating a protection scheme

1. Open the I/O Manager and highlight the node *Knowhow protection* in the *General* tab.
2. Click the button  to add a new scheme.
The “New scheme” dialog opens.
The parameters *ID* and *Version* are automatically generated.



3. Now enter the other parameters and then click on <OK>.

The settings and entries to be made for a protection scheme are specifically as follows:

Author (optional)

Enter the name of the author here.

Copyright (optional)

You can enter a note text here about the copyright of the elements protected by this scheme.

Only permit execution on systems with one of the following license numbers

Enable this option if the elements protected by this scheme are only to be executed on systems with certain license numbers (license protection). Then enter all respective license numbers in the field below. You can easily enter the number of the respective connected dongle or active soft license using the <Add connected dongle> button. If you do not enable this option, there is no execution restriction of the protected elements with respect to the license number.

Password

Enter a password that consists of at least 8 characters. Spaces are not permitted. You will need the password for the following actions:

- Viewing the configuration of a protected element
- Changing the configuration of a protected element
- Changing or removing the protection scheme

Note

What to do, if you don't know the password anymore?

The password of a protection scheme is encrypted and saved in the I/O configuration. Note the password and store it in a safe place where you can find it.

If you forgot the password you won't be able to open or edit protected profiles for *ibaInSpectra* or *ibaInCycle*, for instance. Because you cannot reset the password by yourself, the only way to fix it is to save the I/O configuration of your system and send the configuration to the iba support desk. You may as well take the project file or simply generate a support file over the *Help* menu and send it to the iba support desk, conjoined with the request for resetting the protection scheme passwords.

iba can only remove the passwords but is not able to retrieve them. iba erases the passwords from the configuration and send it back to you. Then you can load this configuration into your system. Finally, you can define a new password.

4.9.3 Application of a protection scheme

An element can always only be protected by one protection scheme, but a protection scheme can be applied to several elements.

If you have elements in your *ibaPDA* configuration worth protecting, such as InSpectra profiles, then these elements are shown in the table below in the dialog

In order to protect one or more elements, first highlight the desired scheme in the list of the protection schemes (top left).

Then highlight the relevant lines at the bottom by setting a check mark in the selection box and click on the button <Protect>.

Then enter the password for the respective protection scheme and click <OK>.

4.9.4 Removing the protection

In order to remove the protection for one or more elements, highlight the corresponding lines in the table below in the dialog by setting a check mark in the selection box. Then click on the button <Unprotect>.

Then enter the password for the respective protection scheme and click <OK>.

If you want to remove the protection for several elements at the same time that are protected with different schemes, then you have to enter the passwords for all respective schemes in the password dialog.

4.9.5 Importing and exporting protected elements

When elements are protected, they can be exported and imported. The configuration of the elements is encrypted in the export files (*.protectionScheme). You therefore have an easy way of spreading protected know-how to different *ibaPDA* systems.

For an export, highlight the desired element and click <Export scheme>. Then select the desired storage path, enter a file name and close the dialog by pressing <Save>.

If you have highlighted several elements for export, then a prompt dialog appears asking whether all highlighted elements should be saved in the export file or whether you want to select more first.

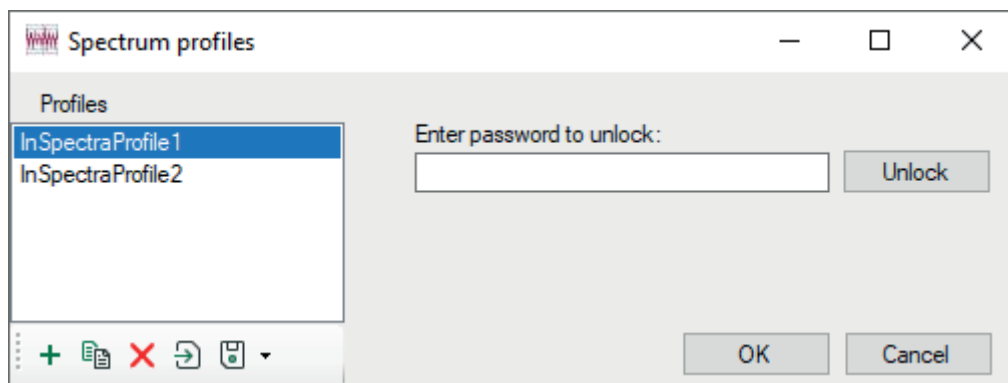
If you want to import a protected element file, click on <Import scheme>, select the desired file (*.protectionScheme) and close the dialog by pressing <Open>.

4.9.6 Unlocking protected items

If you want to access protected items in the application, then you must first unlock the respective item by entering the password for the protective rule.

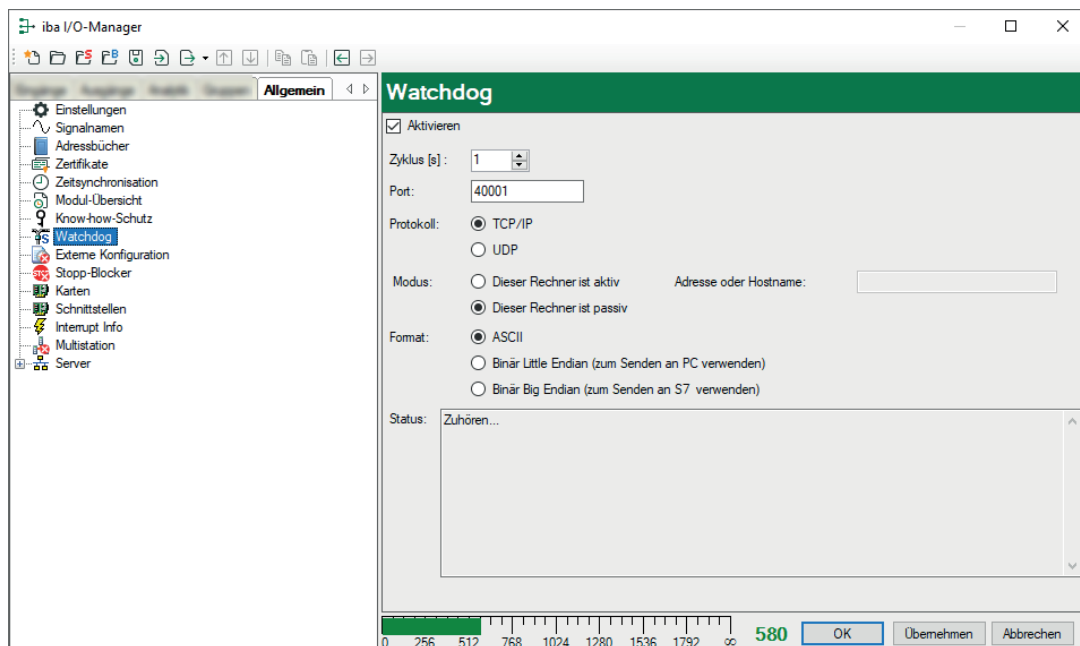
If, for example, you want to view the configuration of the protected InSpectra profiles, then you must first enter the password in the “Spectrum” dialog and click on <Unlock>.

The access remains protected until the I/O manager is closed again.



4.10 Watchdog

The watchdog function is used to monitor the status and operation of *ibaPDA* with the help of another system, e. g. a supervisory PLC.



Enable

Check this box if you want *ibaPDA* to send a watchdog telegram to one or more remote computers. This allows *ibaPDA* to be monitored by other systems. If this option is enabled, the watchdog telegrams will be sent cyclically – irrespective of a running or halted acquisition.

Cycle

Choose or enter the cycle time in seconds for the transmission cycle of the watchdog telegram.

Port

Port address for the watchdog telegram. Default port number is 40001, according to the RFC definition. Adjust if required.

The port address for this connection must be the same in the target computer.

Protocol

Select the protocol for transmitting the watchdog telegram, depending on the receiving supervisory system:

- TCP/IP
- UDP

Generally, the UDP protocol contains less overhead information.

Mode

- *This node is active*

Select this option if the *ibaPDA* PC is to actively establish a UDP or TCP/IP connection. The IP address or host name of the target computer (host) in the network to which *ibaPDA* is to establish the connection must be entered in the *IP address or host name* field. In active mode, only one target computer is allowed.

■ *This node is passive*

Select this option if the TCP/IP connection initiative should be established by another or several other computers to *ibaPDA*. If *Enable* is also selected for the watchdog telegram, the sign of life is sent to all the computers establishing and maintaining a connection to *ibaPDA*.

Format

Choose between the telegram formats

- ASCII
- Binary Little Endian
- Binary Big Endian

Depending on how the watchdog telegram should be interpreted by other systems or devices, one of the formats must be selected. The ASCII format is recommended for displays or ASCII-based systems whereas the binary format is suitable for computers and PLC systems. If the target system is a SIMATIC S7 controller, choose the Big Endian format for a binary telegram.

In general, the binary format is easier to interpret because it has a fixed size and the information is written at fixed offsets within the telegram.

Status

This field displays status messages which reflect the status of the watchdog connection. If the watchdog is in passive mode, then all the connections are displayed. If in active mode, only one connection is shown.

Further status messages:

- Stopped watchdog is disabled.
- Connected to ... xyz: Connection to computer xyz active
- Listening: *ibaPDA* is waiting for another computer to establish the connection
- Error: Error message (text)

4.10.1 Information about the watchdog telegram structure, ASCII format

As described, *ibaPDA* can be configured in a way that it cyclically sends watchdog messages to one or more computers in the network. Besides the information that *ibaPDA* is "still alive", in that it is capable of sending this message, a range of other information is contained in the telegram indicating the current status of *ibaPDA*.

Time stamp

Date and time in ms resolution

PC name

Name of the *ibaPDA* computer in the network

File name (in data storage 1, 2, ... n)

Name of the current data file of the relevant data storage on the hard disk.

Status data store 1, 2, ... n

Status of measuring and data recording; possible values:

- Idle: no measurement (after start of *ibaPDA* or driver restart)
- WaitForTrigger: Measurement is running, waiting for start trigger, pre-trigger phase
- Recording: Data recording is running
- PostTrigger: Stop trigger has arrived, post-trigger phase

MB free memory for data storage 1, 2, ... n

Free memory space on the hard disk

The telegram extends for each further data storage by the last three items.

The entries of information mentioned above are separated by commas inside the message. The end of the telegram is marked by the NUL character (\0). The entries are in ASCII characters. If only one data storage is used, the last entry field is the one with free memory for data storage 1. The message length varies depending on the length of the entries, e.g. path and file names.

If no data storage is enabled, the telegram consists of: *Time stamp,PC name,PDA6,,Idle,-1*

The telegram ends with the NUL character ('\0').

Note

If the system is an *ibaQDR* system, then the telegram is enhanced with further information:

- QDR is running (QDR, Running)
- QDR is stopped (QDR, Idle)

An example of a possible telegram structure can be found in Part 7, *ASCII watchdog telegram structure*.

4.10.2 Information about the watchdog telegram structure, binary format

A list shows the C structure of the telegram in binary format.

An example of the binary code of a watchdog telegram can be found in Part 7, *Binary watchdog telegram*.

Structure of the binary watchdog telegram

Byte offset	Data type	Contents	Remark	
0	int32		Message counter is incremented after each message	Message
1				
2				
3				
4	int32		Version number (currently =1)	
5				
6				
7				
8	int32		Measurement status =1: PDA is measuring	
9				
10				
11				
12	char	7 6 5 4 3 2 1 0	Bit 0 = 1: Everything is OK Bit 1 = 1: There are disabled signals	General
13	char		Reserved	
14	char		Reserved	
15	char		Reserved	
16	int32		MSB	Capture-CAM
17				
18				
19			LSB	
20	int32		MSB	Capture-CAM
21				
22				
23			LSB	
24	int32		MSB	Capture-HMI
25				
26				
27			LSB	
28	int32		MSB	Capture-HMI
29				
30				
31			LSB	
32	short		Status QDR data store	QDR data store
33				
34				
35				
36	int32		Free disk space in MB	
37				
38				
39				
40	char		Reserved	
41	char		Reserved	
42	char		Reserved	
43	char		Reserved	
44	short		Status normal data store	1 st normal data store
45				
46				
47				
48	int32		Free disk space in MB	
49				
50				
51				
52	char		Reserved	
53	char		Reserved	
54	char		Reserved	
55	char		Reserved	
56				2 nd data store
57				
58				
59				
60				3 rd data store
61				
62				
63				
64				more data stores
65				
66				
67				

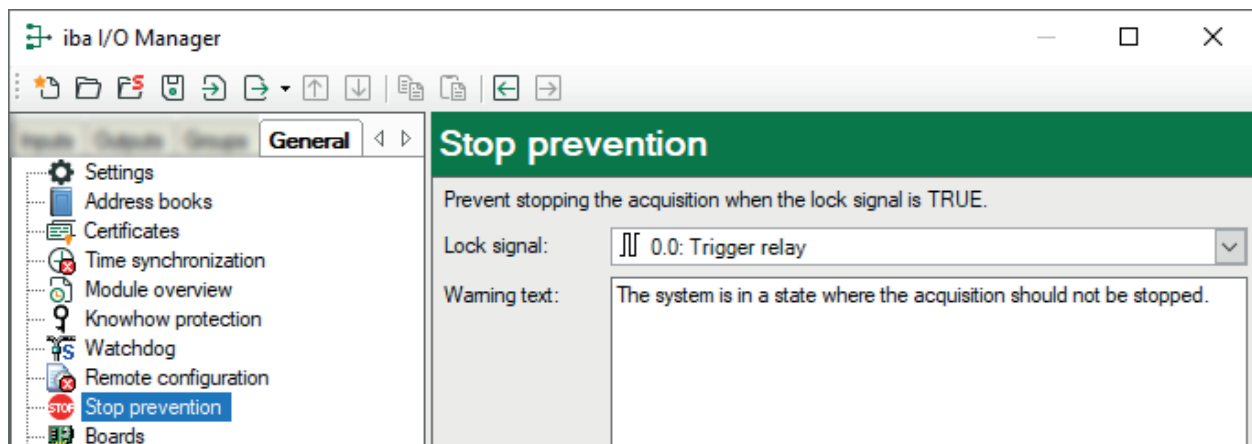
4.11 External configuration

For a detailed description of the remote configuration, see Part 1, *Remote configuration*.

4.12 Stop prevention

By enabling the “Stop prevention function you have the possibility to prevent the acquisition being stopped by accident.

You can enable the stop prevention by selecting any digital signal. If attempts to stop the acquisition occur on an *ibaPDA* client while the signal is TRUE, a message box pops up asking for confirmation. Only if you answer this question with “Yes” the acquisition will be stopped. Using this feature you can protect a running acquisition with signals from the plant, the process or by an operation.



Lock signal

Select the appropriate signal from the signal tree in the drop-down list.

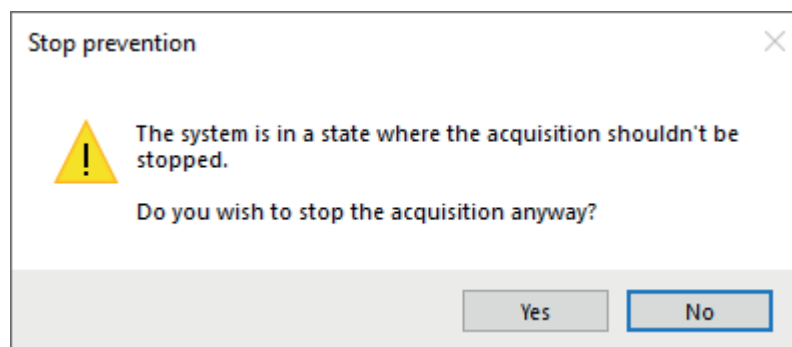
All digital signals can be used as lock signal, e. g.

- Digital input signals from sensors or PLC
- *ibaQPanel* inputs
- Virtual signals

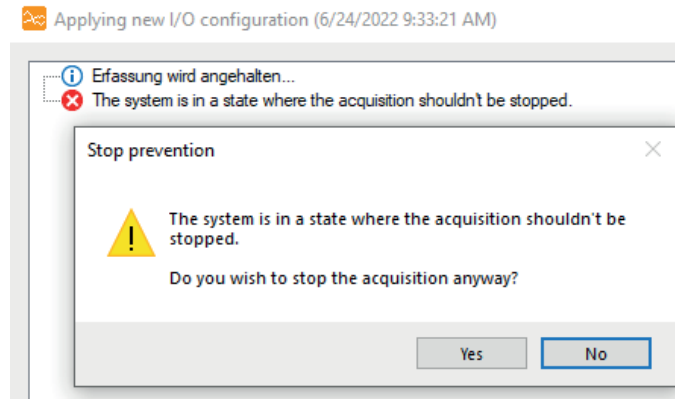
Warning text

You can overwrite or modify the standard text in this field. This helps you to include practical advice, directions or contact data in the warning message.

If the stop prevention is triggered by clicking on the < button STOP, the warning message pops up immediately after confirmation of the stop request.



If the stop prevention is triggered by a change of the I/O configuration, the warning message pops up, supplemented by an entry in the validation dialog.



4.13 Boards

Several optional settings regarding iba PC boards can be made at this stage, for example, in order to determine the system's behavior in case of communication errors. Referring to the following boards

- ibaFOB-io-D/-Dexp, ibaFOB-2io-D/-Dexp, ibaFOB-4i-D/-Dexp or ibaFOB-2i-D/-Dexp
- ibaFOB-io-ExpressCard, ibaFOB-io-USB
- ibaFOB-PlusControl, ibaFOB-H
- ibaCom-L2B-4-8 or -8-8
- ibaN-2E

ibaCom-L2B

Profibus slaves response timeout [s]

Enter the waiting time for the connection setup to the DP master here.

When the acquisition starts (Button <GO>), *ibaPDA* tries to establish a connection to the DP master. If the connection is not made within the time set here, the connection is recognized as faulty.

ibaFOB-D, ibaFOB-io-ExpressCard and ibaFOB-io-USB

These settings apply only to either fast measurements (25 kHz) with a higher data transmission rate (5 Mbit/s) or measurements with a fast data transmission rate (32 Mbit/s).

Stop acquisition when a broken link is detected

If you select this option, then all FOB connections of the corresponding card are monitored. If a connection fails, then the measurement is stopped and an error message is output in the event log, provided the report option is also enabled.

Restart acquisition when a broken iBaNet 5Mbit link is reconnected

Select this option if the system is to be put back into operation as soon as the connection has been restored. Any disabled modules will be re-enabled.

Stop acquisition when a link buffer is empty

The link buffer should never be empty. An empty link buffer indicates a malfunction between the card and computer.

Set signals to zero when a link is broken

Enabling this option sets all measured signals of a connection to zero if the connection is interrupted. Otherwise the signal values would show the value at the time of the connection interruption.

Use Message Signaled Interrupts (MSI)

MSI is disabled by default. Enabling MSI on a computer may be useful in case of irregular interrupts or poor interrupt processing.

The firmware version E2 or higher is required.

ibaFOB-D network

IP address and subnet mask

Default setting 172.29.1.100/255.255.0.0

Both during the installation of *ibaPDA* or of the *ibaFOB-D* network as well as during any validation of the I/O configuration, it is automatically checked whether the default IP address is already part of an existing subnet. If this was the case, then the IP address for the *ibaFOB-D* network would have to be changed. Use the <Check IP address> button to manually trigger this check. A corresponding message is displayed after the check is completed.

iBaNet 32 Mbit/s Flex UDP port

You can set the port number here, which is used for the UDP communication from the iBaNet 32 Mbit Flex protocol. The default port is 62012. If the communication via this port is not permitted by the firewall, you can reconfigure the firewall using the button <Allow port through firewall> so that the port is enabled.

You can always reset the port 62012 using the button <Reset port to default>.

ibaFOB-PlusControl and ibaFOB-H

These settings apply only if an ibaFOB-Plus board or ibaFOB-H board is installed.

Allow start of the acquisition when a link is broken

If one or more connections with the PLUSCONTROL system or Hitachi MACH3 HVDC control system cannot be established at the start of the acquisition, the acquisition will start anyway.

Set signals to zero when a link is broken

See above.

Update preferred modules when a new address book is detected during the start of the acquisition

When establishing the connection with a PLUSCONTROL system or MACH3 HVDC control system, *ibaPDA* reads and analyses the address book. For "preferred signals", "preferred modules" will be created automatically. If a new address book is available, you usually have the possibility to choose how to handle the "preferred modules" (update, replace or no change). If you enable this option, then the existing "preferred modules" will be updated with the "preferred modules" taken from the new address book. Signal-IDs of the already existing "preferred signals" will remain unchanged. If you disable this option then no update will be made.

Restart acquisition when a new address book is detected

If you enable this option, then *ibaPDA* checks periodically every 10 s whether the address book has changed or not. If a new address book is detected, the acquisition stops and restarts. The address book will be read at restart of the acquisition. If the option above is enabled too, the "preferred modules" will be updated automatically as well.

ibaN-2E

You can select a time synchronization method here, which applies if the system runs in stand-alone mode.

4.14 Interfaces

The *Interfaces* node performs two functions:

- Assignment of the physical location of boards and their IDs
- Control of the visibility of the interfaces in the I/O tree

Physical location

Both a bus and a slot number are automatically assigned by the BIOS of the computer to the plug-in boards in a PC. Based on this information, *ibaPDA* assigns an ID to each iba board. This ID is shown in the 7-segment display of the board. Since a maximum of 8 boards can be inserted into one PC per board type (e.g. *ibaFOB-D/-Dexp* or *ibaFOB-TDC*), IDs from 0 to 7 are assigned for each board type. The sort order of the boards in the interface tree in the I/O Manager is carried out automatically in ascending order according to the ID. However, this order does not necessarily match the physical order or arrangement of the cards in the computer. So it might also occur that the inserted board in the first slot has a higher ID than the board in the second slot.

If you want the order of the boards in the I/O Manager to match their physical order in the slots then you can make this correction in the "Physical location" table.

Note that this function is only available for certain motherboards, which are supported by *ibaPDA* and used in the *ibaRackline* and *ibaDeskline* computers:

- Avalue EAX-C246P
- Protech BA-0951
- SuperMicro C2SBC-Q
- SuperMicro X10SAT
- SuperMicro X11SPL-F

The table in the "Physical location" area shows the detected boards in slots X1 ... with their current ID.

Interfaces

Physical location

☒ Assign board numbers to physical locations

☒ Add physical location to interface name

	Device	Current ID	New ID
X1	ibaFOB-PlusControl	1	0
X2	Empty		
X3	ibaFOB-2io-Dexp	0	0
X4	ibaCom-L2B-4-8	0	0
X5	ibaFOB-4io-Dexp	1	1
X6	CP 16 16	0	0
X7	ibaFOB-4io-Dexp	1	0
X8	Leer		0

Preview

Visibility

☐ Hide empty address nodes

Name	Visible
ibaNet-E	☒
X6: CP 16 16	☒
X3: ibaFOB-2io-Dexp	☒
X5: ibaFOB-4io-Dexp	☒
X7: ibaFOB-PlusControl	☒
X1: ibaFOB-PlusControl	☒
X4: ibaCom-L2B-4-8	☒
DGM200E	☒
DTBox Request	☒
DTBox Request UDP	☒
EGD	☒
E-mail	☒
EtherNet/IP	☒
GCOM	☒
Generic TCP	☒

If you want to change an ID, click in the *New ID* column and select the desired ID number.

Note that the IDs per board type must always be unique (no duplicate IDs).

Boards with altered ID are marked in orange in the table.

Pressing the <Preview> button makes the new ID number flash in the 7-segment display of the board. You can therefore check whether the assignment is correct before you confirm the new configuration with <OK>.

If you have already configured the modules and signals for a board, the modules do not migrate with the ID! You must therefore be sure that the modules are assigned to the correct boards (connectors). If not, you must manually move the modules in the I/O Manager.

ibaPDA ensures that a board's ID is retained if more boards are added.

However, if one of multiple boards of the same type is removed, the ID may be changed automatically.

If you, e.g., have installed 3 *ibaFOB-D* boards and the board with ID 1 is removed then the board with ID 2 automatically receives the ID 1 since there cannot be any gaps in the ID numbering.

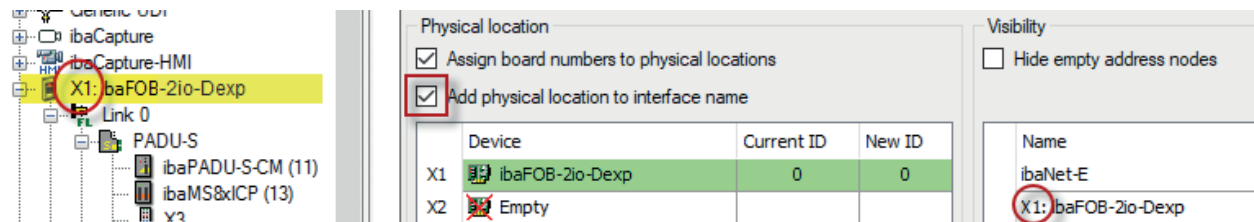
Assign board numbers to physical locations

If this option is enabled, the board IDs are assigned according to the slot order. Recognized boards are highlighted in green in the table.

If *ibaPDA* is being installed on an *ibaRackline* or *ibaDeskline* computer for the first time, then the option is enabled by default. In all other cases, it is disabled.

Add physical location to interface name

If this option and the option *Assign board numbers to physical locations* are enabled, then the card names in *ibaPDA* receive a prefix with the physical location.



The screenshot shows the I/O Manager interface. On the left, a tree view shows the hardware configuration, with 'X1: baFOB-2io-Dexp' highlighted. On the right, the 'Physical location' settings are shown, with both 'Assign board numbers to physical locations' and 'Add physical location to interface name' checked. Below these settings is a table with columns 'Device', 'Current ID', and 'New ID'. The table shows two rows: 'X1' with 'ibaFOB-2io-Dexp' (highlighted in green) and 'X2' with 'Empty'. To the right of this table is the 'Visibility' section, which includes a checkbox 'Hide empty address nodes' and a table with columns 'Name'. The 'Name' table shows two rows: 'ibaNet-E' and 'X1: baFOB-2io-Dexp' (highlighted in green).

Device	Current ID	New ID
X1 ibaFOB-2io-Dexp	0	0
X2 Empty		

Name
ibaNet-E
X1: baFOB-2io-Dexp

Visibility

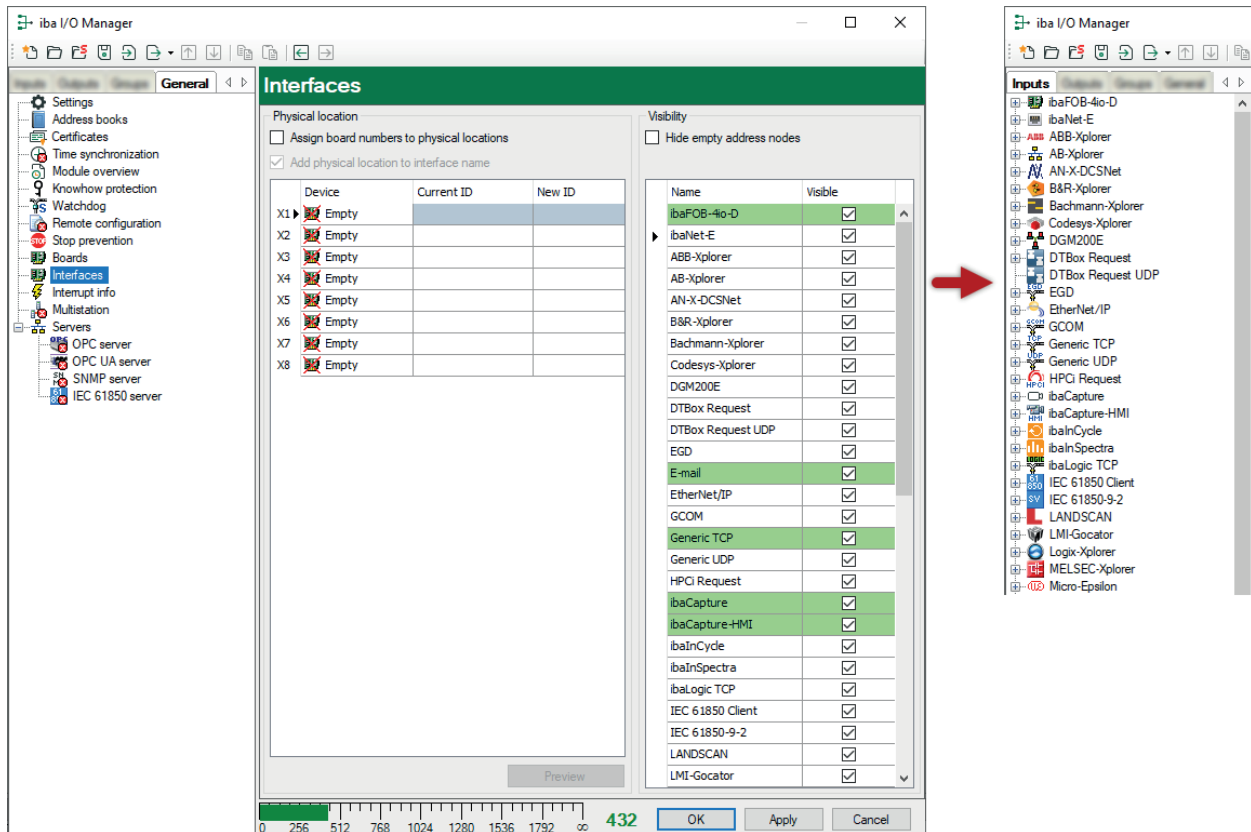
The table *Visibility* lists all the interfaces that are available either through licenses or installed cards. These interfaces can also be viewed in the interface tree.

You can hide or display the interfaces not required in the interface tree by using the checkbox in the *Visible* column.

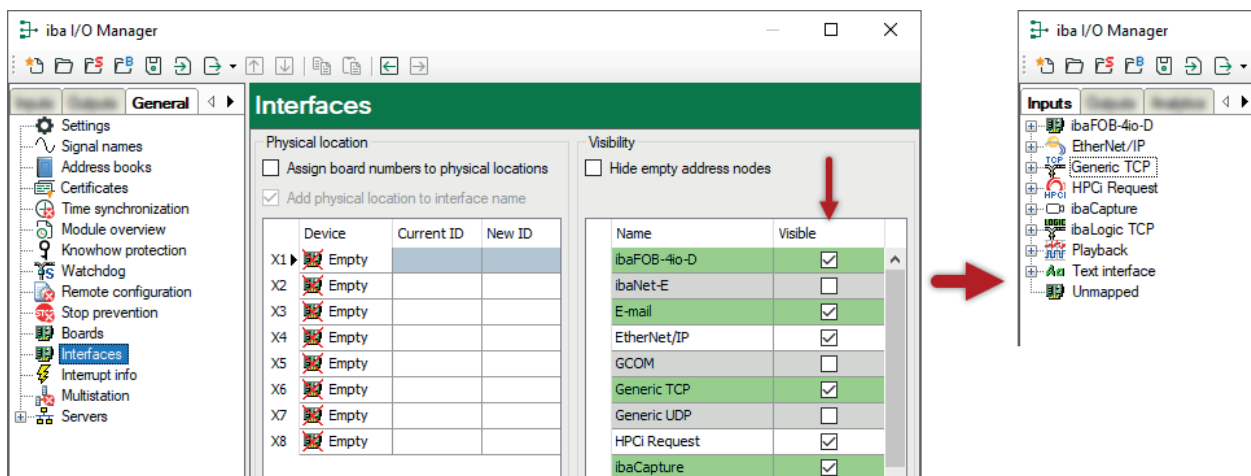
Interfaces with configured modules are highlighted in green and cannot be hidden.

If you double-click on an interface in the table, you will jump directly to this interface in the signal tree in the I/O Manager.

All interfaces visible:



Selected interfaces are visible, the others are hidden:



4.15 Interrupt Info

The display for the driver status (*ibaPDA* driver) must show the "Loaded" status on green background for proper operation. In case of a failure, the field turns red and an error message appears below the start button. If, for example, another process is already running with *ibaPDA*-drivers, the message would indicate that an instance of the program is already running. In addition, you will find a number of displays that reflect the interrupt behavior. The interrupts are good indicators of the proper function of the system.

Interrupts

The total number of interrupts acquired is displayed here, which is composed of iba's own and external interrupts. When using the internal timer this value equals the "internal time counter".

iba PCI interrupts

This is an interrupt counter for interrupts that are generated by an iba card (PCI, PCIe or ExpressCard). In proper operation, it counts upwards at a rate of 1000/s (corresponding to an interrupt cycle of 1 ms), if the acquisition time base is 1 ms.

Non-iba PCI interrupts

This counter reading will only change if also other installed PC cards generate interrupts. This could occur in case of "shared interrupts." If the iba card generating the interrupt is assigned an exclusive interrupt, the counter must not change.

Internal timer counter

The internal timer counter is active when no interrupt source (PC card) is available. In this case the Windows timer is used.

The counter usually also count upwards with 1000 / sec.

Time correction

The value for time correction will only be displayed during acquisition and the internal timer counter serves as interrupt source.

Generally, the internal timer counter has not exactly the same resolution like the basic sample time. Hence, each counter step has a deviation of a few microseconds. These microseconds are added in the time correction counter. As soon as the sum matches the timer resolution, a double counter pulse will be triggered. This method enables the internal timer to match the basic sample time in average.

Acquisition thread CPU usage

This is the CPU time the server needs to read data from the driver and process them, e.g. evaluation of virtual signals, writing to the data file, buffering for transmission to clients etc.

Interrupt buffer (fill level)

This display shows how much data is in the interrupt buffer of the driver.

The *ibaPDA* server periodically reads from this buffer. If the interrupt service routine (ISR) of the driver takes too long, the *ibaPDA* server has not enough time to read all data out of the buffer and the buffer overflows. This would cause a stop of the acquisition. In that case, either a reduction of the number of signals or an increase of the basic sample time is required in order to ensure proper operation. This would reduce the read-out time for the ISR.

In the *I/O-Manager – General* tab – *Interrupt Info* node you may increase the size of the interrupt buffer up to 1024 MB, in order to prevent fast buffer overflow. Default size of the buffer is 50 MB.

Timer resolution

This time value given in ms should approximately correspond to the measurement time base. The timer is used for the internal generation of interrupts.

Interrupt buffer size

Enter here the maximum size for memory-buffered items.

Information for an extended diagnosis is available in 3 sub-tabs.

Interrupt tab

Interrupt times

These values are only displayed when the acquisition is running.

The system measures the actual processing time of the ISR (=interrupt time) and stores the shortest (Min) and longest (Max) time. The ISR reads the requested data from the different PC cards. The ratio of interrupt time to interrupt cycle time shows the percentage of CPU time required to read the data from the cards.

The actual interrupt time should never exceed double the interrupt cycle time. If so, interrupts will get lost and the measurement is not accurate.

Interrupt cycle times

These values are only displayed when the acquisition is running.

The system measures the actual cycle time of the interrupt and stores the shortest (Min) and longest (Max) time.

Asynchronous mode tab

In this tab you will find information about the timing of the DPC routine in asynchronous mode. In fact, information is only available if cards using the asynchronous mode are installed in the system.

Basically, the system works in "synchronous" mode. Synchronous means that the data collected is copied from the card to the PC's working memory by means of the interrupt service routine (ISR). In asynchronous mode, this process takes place outside the ISR with a DPC (deferred procedure call) routine. The DPC routine has a lower priority and thus does not block the system if the processing of routine is not fast enough, e.g. due to a high volume of data. With each interrupt, it is checked whether the DPC routine has stopped. If the DPC routine is still active for an interrupt, the "DPC busy counter" value is incremented.

The switchover to asynchronous mode can be enabled for older cards, such as *ibaFOB-S* or *ibaFOB-X* (I/O Manager, *General* tab, *Boards* node). Other cards, e.g., for reflective memory, can also be used asynchronously. Newer cards, such as *ibaFOB-D*, do not require asynchronous mode as they write the data directly into the processor memory using DMA technology.

The DPC routine always copies data blocks that are grouped according to acquisition times (time base). Thus, all blocks of data, e.g., that are recorded with 10 ms, are grouped together in one entry. Each of these entries is displayed in the table in the lower part of the tab.

The "Time in DPC routine (ms)" information shows the sums of the copy times of all the entries that were copied at the same time.

The other columns in the table display the times needed to copy the data of the respective entry. The entries to be copied are listed in a waiting list. The maximum number of entries in the queue is normally the max. copy time divided by the cycle time. If the maximum number of entries in the queue is significantly higher then this means that the system could not copy the data fast enough, measured at the set cycle time / acquisition time base.

If the queue contains more than 25 entries, the measurement is stopped with an error message.

Multiprocessor tab

In this tab you will find information about the distribution of the operations or interrupts in multi-core processors. This information is used for the extended diagnostics by iba support.

4.16 Multistation

The multistation operation offers the option to synchronize data acquisition on various *ibaPDA* computers with high precision. The multistation operation is used in cases where the number of measuring signals needed exceeds the maximum capacity of the input cards of a computer or if several *ibaPDA* computers are required due to spatial conditions.

You need a special license to use the multistation function:

ibaPDA-Multistation license (order no. 30.001130)

A multistation configuration can consist of 2 to 33 *ibaPDA* systems.

Perform the configuration of the *ibaPDA* system for operation in a multi-station network with the dialogs in this branch. Here, you determine whether the *ibaPDA* system is active as master or slave in the multistation group.

The standalone mode is set at the factory or if there is no multistation license.

Other documentation

For detailed information on the configuration of the multistation system and how it works, see the separate documentation for the product *ibaPDA-Multistation-License* (order no. 30.001130).

4.17 OPC server

On this node, you can adjust the parent settings for using the OPC DA interface.

There is a separate configurations dialog for OPC UA.

OPC DA ("classic" OPC) uses the DCOM protocol for communication. Please note the following information:

Note



Due to a vulnerability in the DCOM protocol, Microsoft has released security update KB 5004442, which was activated in the final stage on March 14, 2023, and affects the connectivity of OPC components.

With the activation of the new security feature, only DCOM connections (network connections) between OPC servers and clients that support packet-based authentication are possible. Communication between OPC Classic servers and clients running on the same computer is not affected.

Detailed information about the effects of KB 5004442 DCOM security update is available on the Microsoft website (<https://support.microsoft.com/en-us/topic/kb5004442-manage-changes-for-windows-dcom-server-security-feature-bypass-cve-2021-26414-f1400b52-c141-43d2-941e-37ed901c769c>)

iba AG does not provide support for problems with OPC Classic remote network connections. Depending on the application, there is also the option of communicating via OPC UA instead, thereby avoiding communication via DCOM.

You can enable and disable the OPC server here. The disabled status is indicated by an icon (white cross on red circle) in the tree on the left.

Only the number of the maximum permissible OPC clients that can be connected to *ibaPDA* at the same time must be set at this point.

This setting serves to prevent overloading the resources of *ibaPDA* by too much OPC communication. The default setting is 64. Only increase the value when absolutely necessary.

The other fields show different actual values for OPC communication.

In addition, you have the option to suppress OPC alerts.

Note



The tags in the OPC server will be updated like outputs of *ibaPDA*. Thus, the fastest update cycle derives from the least common multiple of all module time bases or is at least 50 ms respectively.

All other settings for the OPC communication are made in the interface and modules.

Also see part 3, *OPC interface*.

Also see part 3, *Module type OPC client*.

Also see part 3, *Module Type Redundant OPC client*.

Also see part 3, *Module type OPC server*.

4.18 OPC UA Server

Other documentation



You will find an detailed description of the OPC UA server features in the manual about the product *ibaPDA-OPC-UA-Server+*.

4.18.1 General information

By default, *ibaPDA* provides the OPC UA server functionality in order to make data and information about its own status publicly available. This includes information such as

- Issue
- Active licenses
- Status of data acquisition and recording
- Connected OPC UA clients

The OPC UA server function is thus used as an alternative to the SNMP interface or the watch-dog telegram in order to inform other systems about the status of *ibaPDA* .

With the *ibaPDA-OPC-UA-Server+* extension, you can also publish all recorded or calculated signals as well as text channels via OPC UA. This allows other systems with OPC UA client functionality to access data collected by *ibaPDA* .

The signals that are published via OPC UA can be conveniently selected using the signal tree in the *ibaPDA* I/O Manager.

The signal values are updated cyclically.

Note



The tags in the OPC UA server, in the *Modules* node, are refreshed at the same rate as the *ibaPDA* outputs. Thus, the fastest update cycle derives from the least common multiple of all module time bases or is at least 50 ms respectively.

The tags in the nodes *Licensing*, *Acquisition*, *Client*, *Data stores* and *Configuration* are updated once every second.

The OPC UA clients can only read (but not write) the tags provided by *ibaPDA* as the OPC UA server.

If required, you may add so called *Writable Tags* (analog and digital) to the OPC UA server data model. By means of an *OPC UA-Server module*, which can be added on the interface node *OPC UA*, it is possible that OPC UA clients can write values into the OPC UA server of *ibaPDA*. In order to use *Writable Tags* and the *OPC UA-Server module* you will also need the license *ibaPDA-OPC-UA-Server+*.

You can imported or generate the certificates required for communication between the OPC UA server (*ibaPDA*) and an OPC UA client in *ibaPDA*.

4.19 SNMP server

ibaPDA has a built-in SNMP server.

SNMP (simple network management protocol) was designed to be able to monitor and control network elements (e.g., routers, switches, printers, computers, etc.) from a central station. By using the SNMP server, the *ibaPDA* system can be monitored by a network-monitoring tool, e. g., Paessler PRTG or Nagios. The SNMP versions, V1, V2c and V3, are supported. The authentication as well as the data transmission is secured by current encryption algorithms.

By default, the SNMP server provides a set of diagnostic signals that give feedback about the state of the system and the various data stores. With the additional license (*ibaPDA-SNMP-Server +*, order no. 30.670050), you can also make any signal acquired via *ibaPDA* available as an object in the SNMP server.

Note



The tags in the SNMP server will be updated like outputs of *ibaPDA*. Thus, the fastest update cycle derives from the least common multiple of all module time bases or is at least 50 ms respectively.

Other documentation



You will find a detailed description of the SNMP server features in the manual about the product *ibaPDA-SNMP-UA-Server+*.

4.20 IEC61850 server

ibaPDA has an integrated IEC61850 server, which can be used with a corresponding license (*ibaPDA-IEC61850-Server*, art. no. 30.670052).

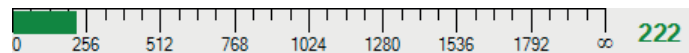
Up to 16 IEC61850 client connections can be configured. The configuration can be exported into a CID file.

Other documentation



You will find a detailed description of the IEC61850 server features in the manual about the product *ibaPDA-IEC61850 server*.

4.21 Signal usage display



The bar chart at the bottom of the I/O manager dialog indicates the current usage of signals. The scale limit corresponds to your *ibaPDA* license, i.e. the maximum number of signals. Only the signals actually used, i.e. the signals with active checkboxes in the signal tables, are taken into account.

5 Groups and vector signals

The signal group feature provides more clarity from a technological point of view.

Open the group area by clicking on the *Groups* tab in the I/O Manager.

An arbitrary number of signals from different data sources, modules or interfaces can be assigned to a group. This provides a more comprehensive view on technological relationships. However, it is your choice whether signals of the same type are to be grouped, e.g. all temperatures, all pressures etc., or to group all signals of a machine or technological device. Of course, a signal can be assigned to multiple groups.

Once you have created groups, you have the choice to get the signals displayed in the signal tree of the *ibaPDA* client, sorted by groups.

The group assignment of the signals and the group names are stored in the data file (*.dat). Hence, they are available in *ibaAnalyzer* too and you may have the signals displayed in the signal tree, sorted by groups.

A group can also be configured as a so called vector signal. Vector signals consist of a number of related signals, such as the signals of a flatness measurement system or the different zones of a temperature scanner. Vector signals are stored in the DAT file and can be used in *ibaAnalyzer* for 2D-top view or 3D representation.

In *ibaQPanel*, vector signals can be used for online 2D top view representation in the trend graph.

5.1 Adding groups

In order to add a new group, click *Click to add group* in the group tree. Alternatively, right-click in the group tree area and select *Add group* from the context menu.

You can also form a divided structure consisting of groups and subgroups. To create a subgroup, right-click on a group and select "*Add group*" from the context menu that appears.

You can give the group a name in the "Group name" field.

If you are using *ibaInSpectra* modules then you can adjust the configuration of the *ibaInSpectra* interface that groups should automatically be created for the *ibaInSpectra* modules. All signals of the *ibaInSpectra* modules are included in this group.

After the text, "Group linked to:", the interface or module name will appear as a hyperlink for certain group types.

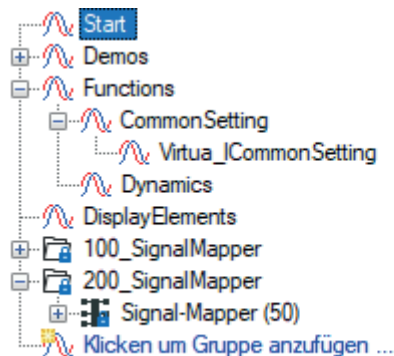
For groups of *ibaInSpectra* modules, for example, you can select a desired group and then click on the blue hyperlink. You are then directed to the module settings of the relevant *ibaInSpectra* module.

If you create folders in the interface tree of the I/O Manager, the folders are shown among the groups by default too. You can disable this function for each folder. Folders are not shown among the groups unless they have a content.

These folders have a lock symbol because they were created outside of the *Groups* tab and cannot be deleted here. However, it is possible to drag and drop signals or other folders inside.

Note

Names of groups, vectors and folders which are on the same level or sub-level have to be unique!



The ID of a group, vector or folder consists of path and name, e. g. ID = Functions/CommonSettings.

5.2 Renaming a group

To rename a group, you have the following options:

- Right-click on group name in the tree and select *Rename* in the context menu.
- Select group in the tree and change the name in the *Group name* field
- Select the group in the tree and press the <F2> key.

5.3 Deleting groups

To delete a group, you have the following options:

- Select the group name in the tree and press the key.
- Right-click on group name in the tree and select *Remove* in the context menu.

5.4 Move groups

To move a group up or down within the group tree, use the arrow buttons in the I/O Manager. If you move a group to another group by dragging and dropping it, it will become a subgroup.

5.5 Assigning signals to a group

In the right part of the dialog you see all available data sources, modules and signals as configured in the I/O manager (Inputs) before.

You can assign signals to groups easily by drag and drop.

A signal can be assigned to multiple groups.

You may select multiple signals with the usual key-mouse-combinations <Ctrl> + mouse click or <Shift> + mouse click and assign them together to a group.

You may even assign entire modules to a group when you select the module name and drag it into the group.

By using the <Swap order> button, you can reverse the order of the signals within the vector. The group must be selected (highlighted) to do so. If you like to change the position of one or more signals, select the signal(s) in question and use the up/down arrow buttons in the group tree window. Furthermore, cameras from an *ibaCapture* module can be added to groups as well.

5.6 Deleting signals from a group

In order to delete a signal from a group, click on the corresponding signal in the group tree and press . Alternatively, make a right mouse click on the signal name and choose *Remove* from the context menu.

You may select multiple signals with the usual key-mouse-combinations <Ctrl> + mouse click or <Shift> + mouse click and delete them together in the same manner from a group.

5.7 Remove inactive signals from a group

If signals of a group have been disabled and are no longer needed in the group, they can easily be removed from the group. Disabled signals in the group tree are greyed out.

To do this, highlight the group in question. If disabled signals are included in this group then the context menu on this group shows the option to remove disabled signals.

Click on *Remove disabled signals* and the signals in question will be removed from the group.

For information about enabling/disabling signals see part 3, *Notes on working with signal tables*.

5.8 Creating vector signals

In order to create a vector signal, first add a group as described above. Select the group in the tree view and enable the *Vector* option. The group is converted into a vector signal indicated by a yellow-red 2d icon. You may add signals to the vector from the signal tree by drag and drop.

An existing group can be transformed into a vector and vice versa at any time. Enable/disable only the option *Vector*.

By using the <Swap order> button, you can reverse the order of the signals within the vector. This is useful if, among other things, the order of the signals in the I/O Manager is not suitable for the spatial view of the visualized measurement, e.g., for 2D false color representations of material profiles.

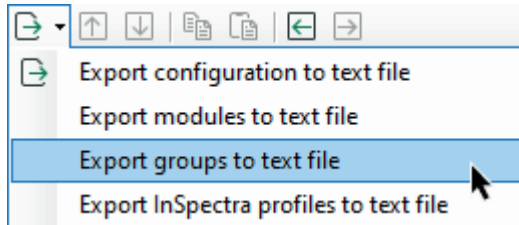
5.9 Export groups

You can export the configuration of your groups to a text file.

So you can, e.g, efficiently edit and extend your group tree by using a suitable text editor or MS Excel. You can then import the edited file again.

The text file can also be used for documentation or transferred to other *ibaPDA* systems.

1. Click the export button in the I/O Manager toolbar and select the groups to export.



2. Use the browser to select the desired directory (drive and folder) in which to store the file, and specify a prefix for the file name. The .txt extension is added by *ibaPDA* automatically.
3. Click on <Save>.

After two headers, the text file contains a line per signal with tab-delimited entries for the group name, signal ID and signal name.

Example

Configuration	Export
<pre> Stand_F7 ├── Positions │ ├── [0:24]: 024 F7 Pos. DS Entry │ ├── [0:25]: 025 F7 Pos. DS Exit │ ├── [0:26]: 026 F7 Pos. OS Entry │ └── [0:27]: 027 F7 Pos. OS Exit ├── [2:31]: 095 F7 roll force OS ├── [2:30]: 094 F7 roll force DS ├── [2:16]: 072 F7 RPM ├── [2:17]: 073 F7 current ├── [3:17]: F7 Stand loaded ├── [3:26]: Thickn. behind F7 └── [3:27]: Width behind F7 </pre>	<pre> 1 GROUPS 2 Group → SignalId → SignalName 3 Stand_F7 → [2:31] → 095 F7 roll force OS 4 Stand_F7 → [2:30] → 094 F7 roll force DS 5 Stand_F7 → [2:16] → 072 F7 RPM 6 Stand_F7 → [2:17] → 073 F7 current 7 Stand_F7 → [3:17] → F7 Stand loaded 8 Stand_F7 → [3:26] → Thickn. behind F7 9 Stand_F7 → [3:27] → Width behind F7 10 Stand_F7\Positions → [0:24] → 024 F7 Pos. DS Entry 11 Stand_F7\Positions → [0:25] → 025 F7 Pos. DS Exit 12 Stand_F7\Positions → [0:26] → 026 F7 Pos. OS Entry 13 Stand_F7\Positions → [0:27] → 027 F7 Pos. OS Exit 14 </pre>

6 Text signals and text processing

ibaPDA is able to acquire, display and record alpha-numerical data in addition to purely numerical information.

Note



With version v7.0.0 of *ibaPDA*, the processing of text information was fundamentally reorganized as regards program technology. Instead of the earlier “technostrings,” text signals are now used, which are fully integrated in the configuration of the modules, analog signals area.

There is no more “technostring” category in the I/O Manager.

6.1 Using the text signals

In addition to pure numerical measured values, processing text information also makes it possible to acquire process-relevant information, such as technological information, specifications about the product, recipes, primary data, reference values, process information or customer data. Interactive text input by the user can thus be included in the analysis of the measurement data.

Some examples:

- Save text signals in the data file

All text or text parts that were defined as text signals are available in the signal selection during the data recording configuration and can be saved in the data file or in the *ibaHD-Server*. Text signals can be shown with other measurement signals in *ibaAnalyzer* and used in the report generator or extracted to databases.

- Text signal for naming the data file

Every text signal can be used to name the data file. This must be configured under *Files* in the data storage configuration dialog.

- Dragging numerical values from text signals

Every numerical character or numerical section of a text can be directly converted into a numerical value and is then available for the numerical calculation of virtual signals.

- Linking and processing texts

Special text operations, such as *ConcatText*, *TrimText*, *ReplaceText*, etc. are available in the expression builder in order to increase the flexibility of text processing and to increase the informational content of the display and record.

- Display text signals

Any text signal can be displayed live in the *ibaPDA* client and in *ibaQPanel*.

6.2 Properties of the text signals

Text signals are analog signals of the STRING or STRING[x] type. STRING[x] is a text (string) with a maximum length of x characters.

Text signals can be transmitted to *ibaPDA* through various interfaces. Depending on the type of interface, the text signals are configured in the normal modules or in separate text modules.

A text signal can contain the entire source text or parts of it. The information of a text can be split into several sections in the I/O Manager, whereby each section shows a text signal and can be processed as individual pieces of information.

A text containing a lot of different information may be split up into any number of text signals.

With some interfaces, a text can be split directly in the interface module. With other interfaces, another module *Text splitter* must be added to split a text. The newly created sections are configured in the same module as new text signals of the data type STRING or a numerical data type.

Text signals are used wherever numerical signals are used.

The start/stop index of the characters (for fixed text length) or different delimiters (for changeable text lengths) serve as criteria for splitting the text. In many cases, the JSON format can also be used.

Text signals that only contain numbers can be interpreted as a numerical value with or without decimal marks and can be used for calculations.

A text (signal) can contain up to 10,000 characters.

Every text signal can be enabled or disabled individually.

The different types of texts are classified by their source or the interface which is used for transmission.

Some of the interfaces depend on a corresponding release in the dongle.

Text source	Note
TCP/IP text UDP Text	The source text is transmitted as a TCP/IP telegram or UDP datagram to <i>ibaPDA</i> via the computer network. You can configure this in the I/O Manager under the node <i>Text interface</i>. In the <i>TCP text or UDP text</i> module, <i>Analog</i> tab, the source text with the STRING data type and the text signals derived from the source text with the STRING data type or a different data type are configured. For numerical data types, the conversion is carried out automatically. A <i>TCP text or UDP text</i> module must be configured for each source text or telegram.

Text source	Note
Serial text	The source text is transmitted via a serial interface to <i>ibaPDA</i> (COM1...COM4). You can set the configuration in the I/O Manager under the node <i>Text interface</i>. In the <i>Serial text</i> module <i>Analog</i> tab, the source text with the STRING data type and the text signals derived from the source text with the STRING data type or a different data type are configured. For numerical data types, the conversion is carried out automatically. Depending on the source text or telegram, a module of the type <i>Serial text</i> must be configured.
File text	The source text is contained in a text file (.txt, .csv, .json) that is provided on a dedicated drive in a defined path (local or network) by an external system. You configure this in the I/O Manager under the node <i>Text interface</i>. In the <i>File text</i> module, <i>Analog</i> tab, the source text with the STRING data type and the text signals derived from the source text with the STRING data type or a different data type are configured. For numerical data types, the conversion is carried out automatically. Depending on the source text or file, a module of the type <i>File text</i> must be configured.
Text creator	The source text is created on the <i>ibaPDA</i> server in the <i>Text creator</i> module, <i>Text creation</i> tab. It can be a typed in fixed text, the value of a signal or the text of another text signal. It can also be a combination of all three forms. Furthermore, XML and JSON data structures can be used for text creation. You can set the configuration in the I/O Manager under the node <i>Virtual</i>. In the <i>Analog</i> tab of the same module, you will then configure the derived text signals with the data type STRING or a different data type. For numerical data types, the conversion is carried out automatically. Depending on the source text, a module of the type <i>Text creator</i> must be configured.
ibaQPanel text input	The text can be entered manually via a text input control of <i>ibaQPanel</i>. You can set the configuration in the I/O Manager under the node <i>Virtual</i>. A module of the type <i>ibaQPanel text input</i> can receive several text signals. In the process, every text signal corresponds to exactly one text input control in the <i>ibaQPanel</i> view. Optionally, these text signals can be broken down into sections with a <i>Text splitter</i> module and, if necessary, converted into other data types.

Text source	Note
Text shift register	Management of existing text signals in the form of a shift register. It provides the last 1 to 64 values of other text signals. The current value of a text signal is applied to the shift register with a trigger signal (rising edge). You can set the configuration in the I/O manager under the node <i>Virtual</i>. The required text signals must have been created beforehand with the different modules.
OPC / OPC UA	The source text is transmitted to <i>ibaPDA</i> via the PC network (Ethernet TCP/IP) by an OPC or OPC UA server. You can set the configuration in the I/O Manager under the node <i>OPC</i> or <i>OPC UA</i>. In the <i>OPC client</i> or <i>OPC UA client</i> module, <i>Analog</i> tab, the source text is configured with the data type STRING. To divide the text from the OPC tag onto individual text signals, a <i>Text splitter</i> module must be configured per source text under the <i>Virtual</i> node.
B&R-Xplorer	The source text is transferred via the B&R-Xplorer interface (Ethernet TCP/IP) to <i>ibaPDA</i>. The processing of symbols of the data type STRING or WSTRING is supported. The configuration can be done in the I/O Manager in a <i>B&R-Xplorer</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with a string data type. To divide the source text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
Codesys-Xplorer	The source text is transferred via the Codesys-Xplorer interface (Ethernet TCP/IP) to <i>ibaPDA</i>. The processing of symbols of the data type STRING or WSTRING is supported. This can be configured in the I/O manager in a <i>Codesys V2 or V3</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with a string data type. To divide the source text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
Logix-Xplorer	The source text is transferred via the Logix-Xplorer interface (Ethernet TCP/IP) to <i>ibaPDA</i>. The processing of symbols of the data type STRING is supported. This can be configured in the I/O manager in a <i>Logix-Xplorer</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with a string data type. To divide the source text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.

Text source	Note
S7-Xplorer	The source text is transferred via the <i>S7-Xplorer</i> interface (Ethernet TCP/IP) to <i>ibaPDA</i>. The SIMATIC S7-300, S7-400, S7-1200, S7-1500 and WinAC controllers are supported. The configuration can be done in the I/O Manager in a <i>S7-Xplorer</i> module. One or more source texts are formed in the <i>Analog</i> tab from S7 operands or symbols with the string data type. To divide the source text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
TwinCAT-Xplorer	The source text is transferred via the TwinCAT-Xplorer interface (Ethernet TCP/IP) to <i>ibaPDA</i>. The processing of symbols of the data type STRING or WSTRING is supported. The configuration can be done in the I/O Manager in a <i>TwinCAT-PLC or BC/BX controller</i>. The source text is formed in the <i>Analog</i> tab from the symbols with a string data type. To divide the source text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
ibaVision	The source text is sent from <i>ibaPDA</i> output modules, which are configured in <i>ibaVision</i>, to <i>ibaPDA</i>. The text signals are configured in the I/O Manager under the node <i>ibaCapture</i> in so-called <i>ibaVision text input</i> modules. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
EtherNet/IP	The source text is transmitted to <i>ibaPDA</i> as a TCP/IP telegram via the EtherNet/IP connection. The configuration can be done in the I/O Manager in an <i>EtherNet/IP</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
Generic-TCP	The source text is transmitted to <i>ibaPDA</i> as a TCP/IP telegram via the Generic-TCP interface. You can configure this in the I/O Manager in a <i>Generic TCP</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING[32] data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.

Text source	Note
Generic-UDP	The source text is transmitted to <i>ibaPDA</i> as a UDP datagram via the Generic-UDP interface. This can be configured in the I/O Manager in a <i>Generic unicast UDP</i>, <i>Generic multicast UDP</i>, <i>HiPAC Request</i> or <i>TwinCAT Request</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING[32] data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
Modbus TCP Server	The source text is transmitted to <i>ibaPDA</i> as a TCP/IP telegram via the Modbus TCP server interface. This can be configured in the I/O Manager in a <i>Modbus Generic</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING[32] data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
Reflective memory	The source text is transmitted via a Reflective Memory (RM) connection to <i>ibaPDA</i>. A special RM interface card is required for the <i>ibaPDA</i> computer. Perform the configuration in the I/O Manager under the node <i>Reflective Memory</i> in a <i>Reflective Memory Text</i> module. The source text and the text signals with the data type STRING derived from the source text or a different data type are configured in the <i>Analog</i> tab of the module. For numerical data types, the conversion is carried out automatically.
ScramNet+	The source text is transferred to <i>ibaPDA</i> via a ScramNet+ connection. Therefore, a special interface card SC150 (manufacturer: Curtiss Wright) must be installed in the <i>ibaPDA</i> computer. Perform the configuration in the I/O Manager under the node <i>ScramNet+</i> in a <i>ScramNet Text</i> module. The source text and the text signals with the data type STRING derived from the source text or a different data type are configured in the <i>Analog</i> tab of the module. For numerical data types, the conversion is carried out automatically.
Sisteam TCP	The source text is transmitted to <i>ibaPDA</i> as a TCP/IP telegram via the Sisteam-TCP interface. You can configure this in the I/O Manager in a <i>Sisteam TCP Generic</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING[32] data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.

Text source	Note
SIMADYN D	The source text is transmitted to <i>ibaPDA</i> via a connection with the SIMADYN-D system (CS12/13/14). This requires a special ibaFOB-SD/-SDexp interface card for the <i>ibaPDA</i> computer. Perform the configuration in the I/O Manager under the node <i>iba-FOB-SD/-SDexp</i> in a <i>Simadyn-D text</i> module. The source text and the text signals with the data type STRING derived from the source text or a different data type are configured in the <i>Analog</i> tab of the module. For numerical data types, the conversion is carried out automatically.
SIMATIC TDC	The source text is transmitted to <i>ibaPDA</i> via a connection with the SIMATIC TDC system (CP52M0 in GDM). This requires a special ibaFOB-TDC/-TDCexp interface card for the <i>ibaPDA</i> computer. Perform the configuration in the I/O Manager under the node <i>iba-FOB-TDC/-TDCexp</i> in a <i>TDC text</i> module. The source text and the text signals with the data type STRING derived from the source text or a different data type are configured in the <i>Analog</i> tab of the module.
TDC TCP/UDP	The source text is transmitted to <i>ibaPDA</i> as a TCP/IP telegram or UDP datagram via the TDC-TCP/UDP interface. This can be configured in the I/O Manager in a <i>TDC TCP/UDP Generic</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING[32] data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.
VIP TCP/UDP	The source text is transmitted to <i>ibaPDA</i> as a TCP/IP telegram or UDP datagram via the VIP-TCP/UDP interface. This can be configured in the I/O Manager in a <i>VIP TCP/UDP Generic</i> module. The source text is formed in the <i>Analog</i> tab from the symbols with the STRING[32] data type. To divide the text into individual text signals, a module <i>Text splitter</i> must be configured under the <i>Virtual</i> node.

Table 5: Sources and interfaces for text signals

Note

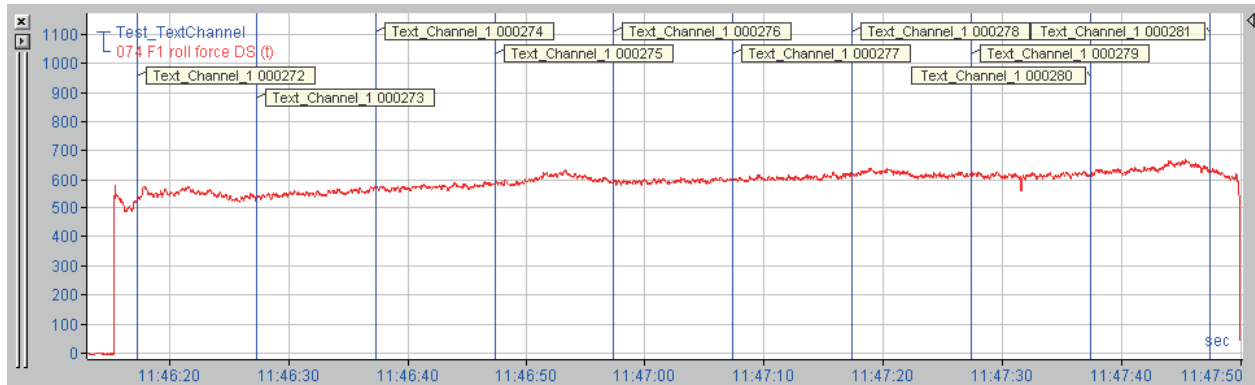
Different information structures are supported for parsing the texts:

- Fixed width: The information is always located at the same place in the text and is determined by the start and stop index of the characters.
 - Delimited: The information units within a text are separated by delimiters (e.g., semicolons, commas, tabs, any character).
 - JSON: If the text is transmitted in JSON format (e.g., a text file), the information units are interpreted as elements of a JSON object.
-

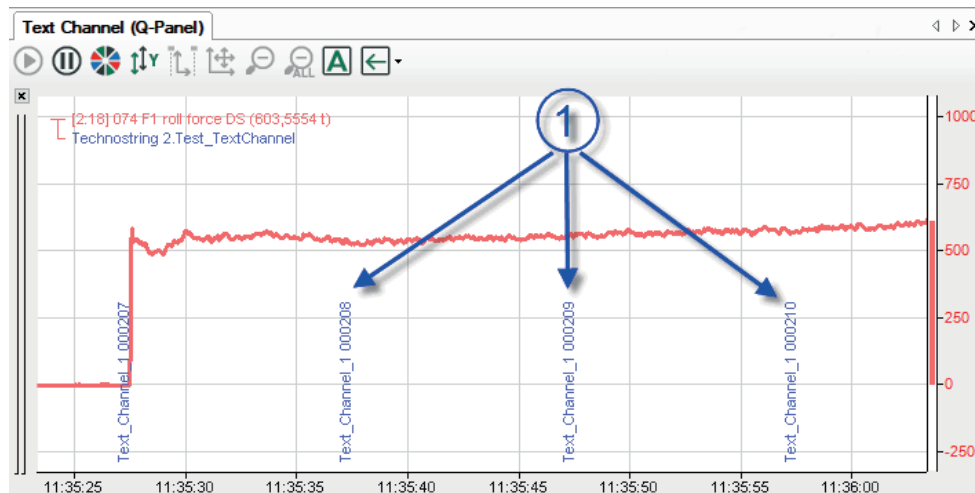
6.3 Using text signals

6.3.1 General

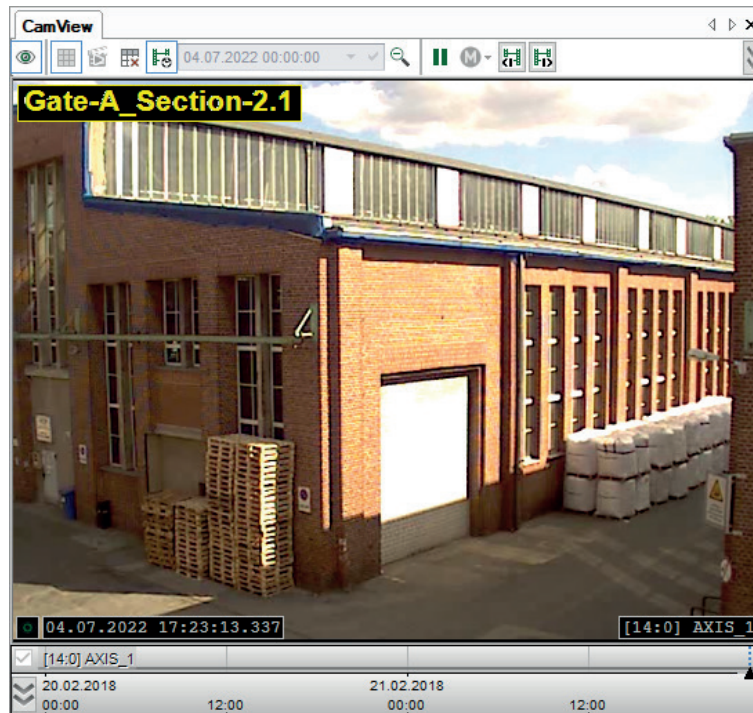
Text signals can be written in a data file like other analog signals. *ibaAnalyzer* shows the text signals as labels at the time of occurrence or their change. They can be added to the graph like usual signals.



If you use *ibaQPanel* for online display on your *ibaPDA* client, the text signals can be displayed along with the *ibaQPanel* trend graph. An example for this is shown in the figure below.



If you use an *ibaCapture* camera view either on your normal *ibaPDA* client or on an *ibaQPanel*, you can use text signals for overlay text displays. An example for this is shown in the figure below.



6.3.2 Configuration

In order to measure and record text signals together with other signals, the text signals are located in the respective module as analog signals with the STRING data type. They therefore receive a standard signal ID ([module number: signal number]).

When using a text splitter module, the created text signals receive the ID from the text separator module and not from the source module, which was used to acquire the text.

The example in the figure shows the analog signal table of the text splitter module.

General Analog Diagnostics								
	Name	Unit	Begin	End	DataType	Filter	Active	Actual
0	Text_complete			0	41 STRING[42]	☐	☒	Text with two counters: 000461 and 000046
1	Value 1		25	30	DINT	☒	☒	461
2	Value 2		36	41	FLOAT	☒	☒	46
3			3	3	STRING[1]	☐	☐	t
4			4	4	STRING[1]	☐	☐	
5			5	5	STRING[1]	☐	☐	w
6			6	6	STRING[1]	☐	☐	i

The yellow part highlights the selection. To change the selection you can edit the "Begin" and "End" columns in the signal grid. You can also change it by clicking and dragging the mouse in the preview while the "Begin" or "End" column is selected.

Text with two counters: 000461 and 000046

The "Filter" property (see configuration table in the illustration above) controls the acquisition of the text signals. If the filter is disabled, then each time a new source text is received the text signal value is written to the data file.

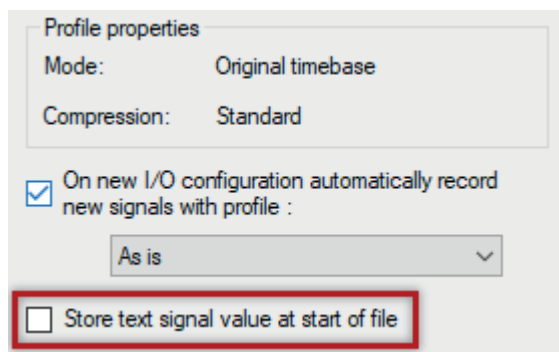
If the filter is enabled (checkmark), the text signal value is only written to the data file if it differs from the previous values.

Tip

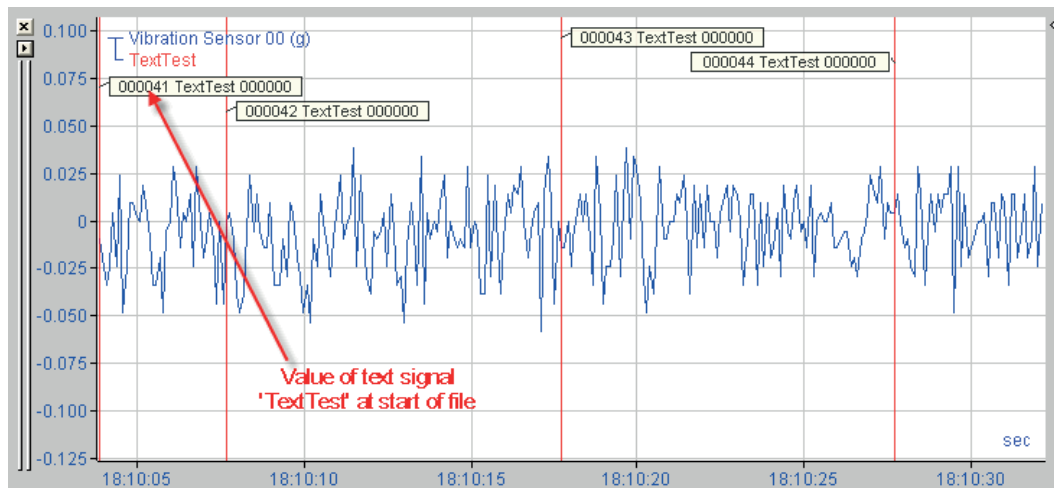


Usually, the values of the text signal are only saved in the data file when a value changes. So if the value of a text signal does not change during the recording, no text appears in the file. If the value changes during the file recording, only the text signal value from that time point onwards is known.

When a text signal already has a value at the beginning of the data storage, the option *Store text signal values at start of file* can be selected for ordinary data storage in the *Signal selection* dialog. If this option is enabled then the actual values of the assigned text signals are always stored at the beginning of each data file.



So, you can look up the value of the text signal for the whole file. The following screenshot shows the result in *ibaAnalyzer*.



6.3.3 Creating text signals

Text signals are created in the *Analog* tab of a suitable module.

Text source / type	Interface	Module	+ Text splitter module
TCP/IP text	Text interface	TCP text	No
UDP text		UDP text	No
COM (serial text)		Serial text	No
Text file		File text	No
Custom	Virtual	Text creator	No
<i>ibaQPanel</i> text input		<i>ibaQPanel</i> text input	Optional
Shift register		Text shift register	No
<i>ibaVision</i>	ibaCapture	<i>ibaVision</i> text input	Yes
OPC DA	OPC	OPC	Yes
OPC UA client	OPC UA	OPC UA Client	Yes
S7-Xplorer	S7-Xplorer	S7-Xplorer	Yes
B&R-Xplorer	B&R-Xplorer	B&R-Xplorer	Yes
Codesys-Xplorer	Codesys-Xplorer	Codesys V2 or Codesys V3	Yes
Logix-Xplorer	Logix-Xplorer	Logix-Xplorer	Yes
TwinCAT-Xplorer	TwinCAT-Xplorer	TwinCAT PLC or BC/BX controller	Yes
EtherNet/IP	EtherNet/IP	EtherNet/IP	Yes
Generic TCP	Generic TCP	Generic TCP	Yes
Generic UDP	Generic UDP	Generic unicast UDP, Generic multicast UDP, HiPAC Request or TwinCAT Request	Yes
Modbus TCP Server	Modbus TCP Server	Modbus Generic	Yes
Reflective Memory	Reflective Memory	Reflective Memory Text	No
SIMATIC TDC	ibaFOB-TD-C/-Dexp	TDC text	No
SIMADYN D	ibaFOB-SD/-Dexp	SIMADYN-D Text	No
ScramNet+	ScramNet+	ScramNet Text	No
Sisteam TCP/IP	Sisteam TCP/IP	Sisteam TCP/IP Generic	Yes
TDC TCP/UDP	TDC TCP/UDP	TDC TCP/UDP Generic	Yes
VIP TCP/UDP	VIP TCP/UDP	VIP TCP/UDP Generic	Yes

Table 6: Interfaces and module selection for text signals

Depending on the module type, first enter the text signal(s) containing the received text. Some module types, such as TCP/IP text, only allow a source text. To do this, you offer the option of directly deriving additional text signals by splitting the source text into individual text signals and, if necessary, converting them into other data types.

With other modular types, such as some Xplorer modules, several text signals can be received. However, you also need a *Text splitter* module with the source text signal as the reference signal to split the text and generate text signals for further use.

The configuration steps are described in the following chapters for each module type.

6.3.4 Delete the text signals

Delete a text signal like any other signal from the signal table.

When using a *Text splitter* module, you can completely delete the sections derived from a source text signal by removing the *Text splitter* module.

Note



By deleting the source text signal, all text signals linked with it and any *Text splitter* modules become invalid.

6.3.5 Defining and assigning text signals

You can configure text signals even if no text has been received. You will need information about the exact structure of the string.

It is much easier (and more reliable) if you can already receive a text. Then the received text is shown in the preview area of the text modules in the *Analog* tab. The prerequisite for this is that the connection to the text source is established and the acquisition is running.

- For fixed width texts, the desired sections for the derived text signals can easily be selected with the mouse.
- For texts whose informational units are separated by delimiters, the individual text parts can be detected and selected.
- For texts in the JSON format, the values of the detected objects are listed.

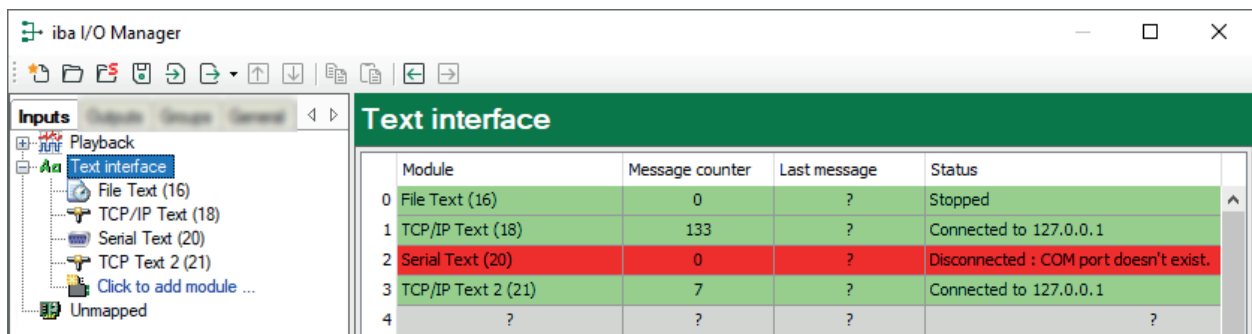
So, any number of text signals can be defined and provided with information from the source text.

6.4 Text interface

You can add the following module types under the node *Text interface* in the I/O Manager:

- TCP Text
- UDP Text
- Serial Text
- File Text

If you select the node *Text interface*, then you will see an overview of all configured modules with the associated telegram counters and status information.



Text interface			
Module	Message counter	Last message	Status
0 File Text (16)	0	?	Stopped
1 TCP/IP Text (18)	133	?	Connected to 127.0.0.1
2 Serial Text (20)	0	?	Disconnected : COM port doesn't exist.
3 TCP/IP Text 2 (21)	7	?	Connected to 127.0.0.1
4 ?	?	?	?

6.5 Text splitter module

The *Text splitter* module is required to generate text signals for some interfaces and can generally be used to generate text signals from existing text signals.

A *Text splitter* module contains a received or generated text signal as an input signal and then provides the complete received text or parts of it as text signals or signals of other analog data types.

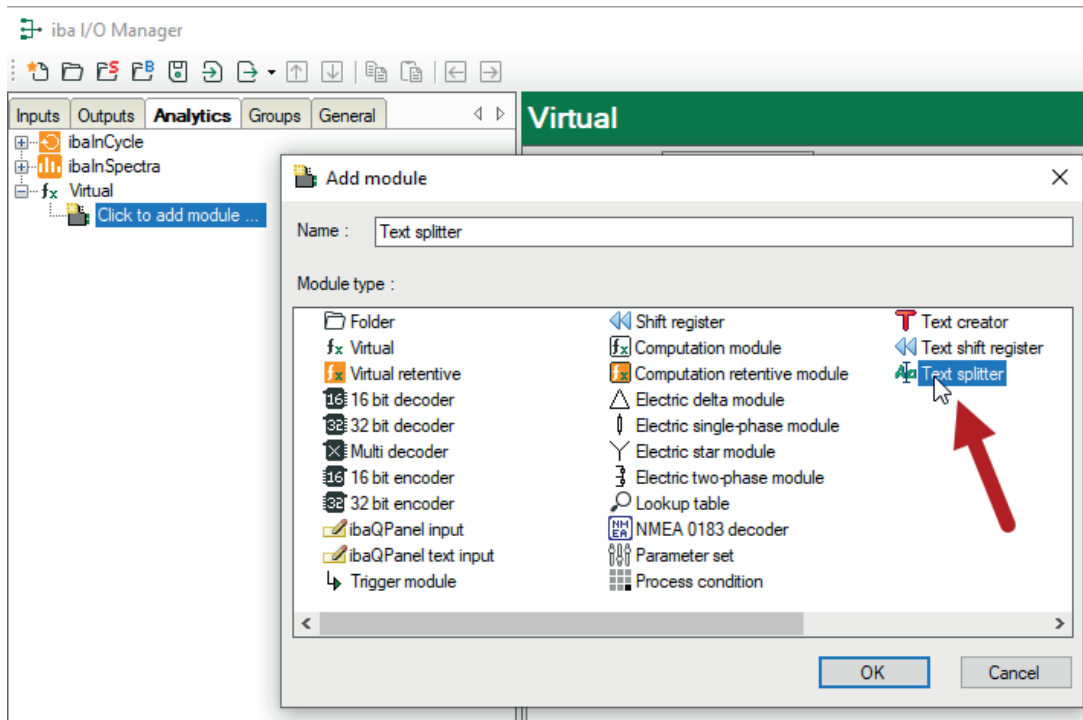
The principle of the text signal definition as carried out in the *Text splitter* module can also be found with other module types.

You need a *Text splitter* module for text signals of the following interfaces:

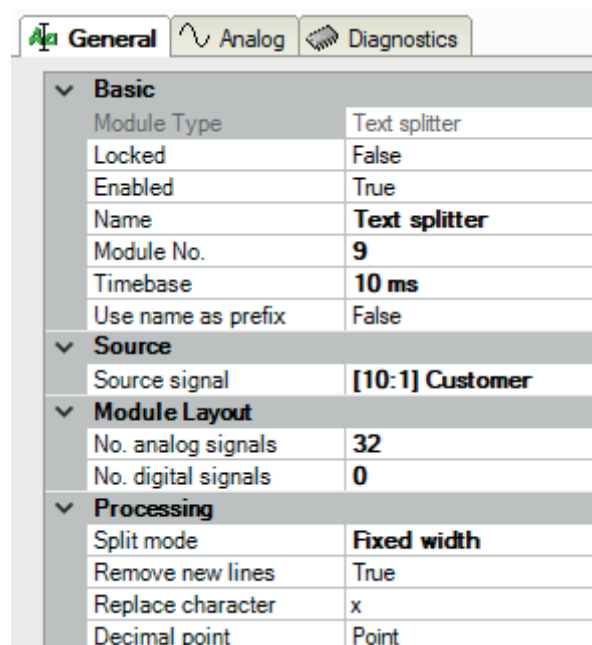
- OPC / OPC UA
- S7-Xplorer
- B&R-Xplorer
- Codesys-Xplorer
- Logix-Xplorer
- TwinCAT-Xplorer
- EtherNet/IP
- Generic TCP
- Generic UDP
- Modbus - TCP - Server
- Sisteam TCP/IP
- TDC - TCP/UDP
- VIP-TCP/UDP
- ibaVision

6.5.1 Add text splitter module

You can add a module by clicking under the *Virtual* interface.



6.5.2 Module settings



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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Source**Source signal**

Enter the signal from the interface module here, with which the text was received.

Module structure

The number of analog signals is preset to 32, the number of digital signals to zero. If required, you can change the number. Permissible range: 1 to 1000.

Processing**Split mode**

Select the suitable mode here that corresponds to the input text structure in order to always correctly read out the information contained:

- Fixed width
- Delimiter
- JSON:

The different settings and their meaning are described in the following chapters.

Remove new lines

If you activate this option (true), then all line feeds ('new lines') will be removed when parsing the input text.

Replace characters

Enter a character here that is to replace all non-printable characters in the input text. Default: x.

Decimal point

Select the decimal separator here so that numerical values with decimal points in the text are correctly interpreted: Period or comma.

6.5.2.1 Fixed width split mode

The *Fixed width* mode is best selected when the input text always has the same length and all information within the text is always at the same place, and therefore always has the same number of characters.

In this mode, you must set a start and stop index for the first and last character for each (text) signal that you want to derive from the input text. You can set this in the *Analog* or *Digital* tab of the module.

	Name	Unit	Begin	End	DataType	Filter	Active	Actual
0	Text_complete		0	41	STRING[42]	☐	☒	Text with two counters: 000033 and 000337
1	Value 1		25	30	FLOAT	☐	☒	33
2	Value 2		36	41	FLOAT	☐	☒	337
3			3	5	STRING[1]	☐	☐	s
4			4	4	STRING[1]	☐	☐	
5			5	5	STRING[1]	☐	☐	a
6			6	6	STRING[1]	☐	☐	

The yellow part highlights the selection. To change the selection you can edit the "Begin" and "End" columns in the signal grid. You can also change it by clicking and dragging the mouse in the preview while the "Begin" or "End" column is selected.

Text with two counters: 000033 and 000337

If a text has already been received, then it is shown in the preview area under the signal table.

To determine the character area, you can manually enter the start and stop values in the *Start* or *Stop* columns or set them with the spinner buttons. The index for the first character in the text is zero (0). You can also apply this method if no text has been received yet, but you already know the structure.

If an input text is already visible, then you can alternatively use the mouse: Click in the line of the desired signal with the mouse in the column *Start* or *Stop*. Then click in the preview area on the first character in the text for the desired signal, keep the mouse button depressed and drag the mouse to the last character. The selected area will be highlighted in yellow and the indexes will automatically be updated in the *Start* and *Stop* columns.

Then set the desired signal data type. This can either be a text (STRING[x] type) or a numerical data type. In the latter case, the text is parsed in order to generate a numerical value. If a decimal mark is used within the numbers in the text, ensure that the decimal mark (period or comma) is correctly set in the *General* tab.

The different colors refer to the following states in the preview area:

Color	Description
Green (1)	All characters of the text associated with a text signal are highlighted in green.
Red (2)	The red color indicates characters in the text which are assigned to multiple text signals.
Yellow (3)	The characters of a text signal selected in the table are highlighted in yellow.

Table 7: Color code in the preview area for text signals (fixed width split mode)

(1) = least dominant, (3)= most dominant color

6.5.2.2 Delimited split mode

Processing	
Split mode	Delimited
Remove new lines	True
Replace character	x
Decimal point	Point
Number of delimiters	2
Delimiter 1	Semicolon
Delimiter 2	Other: VAL
Remove empty parts	False

Select this mode if the information units in the input text are separated by defined delimiters. In this case, the length of the text does not have to be constant any more and the character positions are irrelevant. Only the sequence of the information units must always be the same, because the sections from delimiter to delimiter is counted from 0 to n .

A delimiter may consist of an individual or several characters. The input text determines what you need.

In addition, several delimiters can be defined. Enter the number of possible delimiters and then select the desired character or character combination for each delimiter. Whenever one of the delimiters appears in the text, this will be interpreted as a new section.

The most common characters are offered in a drop-down list. However, you can define any string as a delimiter ("Other...").

Example with several delimiters

Input text:

This is a text with three counters:VAL000007;000015;000079VALEndOfText

Delimiters are the character string VAL and the semicolon.

The text therefore consists of five parts (0 to 4).

The screenshot shows the 'General' tab of the software interface. The signal grid table is as follows:

	Name	Unit	Part	DataType	Filter	Active	Actual
0	Text_Intro		0	STRING	☐	☐	This is a text with three counters:
1	Section_1		1	WORD		☐	7
2	Section_2		2	WORD		☐	15
3	Section_3		3	WORD		☐	79
4	Text_End		4	STRING	☐	☐	EndOfText
5			5	STRING	☐	☐	

Below the table, a text preview area shows the input text split into five lines, with the last line highlighted in yellow:

```

This is a text with three counters :
0 0 0 0 0 7
0 0 0 0 1 5
0 0 0 0 7 9
EndOfText

```

A note below the preview states: "The yellow part highlights the selected part. To change the selected part you can edit the "Part" column in the signal grid. You can also change it by clicking on the desired row in the preview while the "Part" column is selected."

In the delimiter mode, the input text is displayed split in the preview area. Each line corresponds to one part of the input text. In the *Part* column of the signal table, you can now assign the text parts to the signals in case you do not want to use all of the recognized parts or just intend to build a different sequence.

Either enter the number of the part manually or use the spinner buttons. The section set is highlighted in yellow in the preview area.

Alternatively, click in the row of the respective signal in the *Part* column and then in the preview area on the desired text part. The part number will then be updated automatically.

6.5.2.3 Split mode JSON

Processing	
Split mode	JSON
Partial JSON behavior	Keep last values

The text is interpreted as a JSON object in JSON mode. If *ibaPDA* has received a text in the JSON format, the preview area of the *Analog* tab will show a value per line. The text received in the JSON format will be shown already broken down.

Text splitter (23)

General Analog Diagnostics

	Name	Unit	JSON path	DataType	Filter	Active	Actual
0	product.id		product.id	STRING	☐	☒	DF123452
1				STRING	☐	☐	
2				STRING	☐	☐	
3				STRING	☐	☐	
4				STRING	☐	☐	
5				STRING	☐	☐	
6				STRING	☐	☐	
7				STRING	☐	☐	
8				STRING	☐	☐	
9				STRING	☐	☐	
10				STRING	☐	☐	

The yellow part highlights the selected part. To change the selected part you can edit the "JSON path" column in the signal grid. You can also change it by clicking on the desired row in the preview while the "JSON path" column is selected.

Add all JSON values

```
{
  "product": {
    "id": "DF123452",
    "weight": "23411kg",
    "length": "2176m",
    "width": "1234mm",
    "thickness set": "2mm",
    "thickness act": {
      "avg": "2.023mm",
      "min": "1.998mm",
      "max": "2.119mm"
    }
  }
}
```

You must now set the JSON path for the desired signals in the signal table. If you want to configure individual values specifically or in a different sequence, proceed as follows:

Click in the signal table in the JSON path column and then on the line with the desired value in the preview area. The correct path will automatically be entered in the signal table. The JSON path will be adopted as the signal name. If necessary, however, you can change the signal name as you wish. Then select the correct data type, which the signal should receive.

To apply the detected values, click on the <Add all JSON values> button.

According to the JSON definition, values in quotation marks will be interpreted as text and values without quotation marks will be interpreted as numerical values. The data type will accordingly automatically be set to STRING or FLOAT.

The following figure shows JSON text signals after all values have been added. The signal names partially have been changed.

Text splitter (23)

Name	Unit	JSON path	Data Type	Filter	Active	Actual
0		product.id	STRING	☐	☒	DF123452
1		product.weight	STRING	☐	☒	23411kg
2		product.length	STRING	☐	☒	2176m
3		product.width	STRING	☐	☒	1234mm
4		product[thickness set]	STRING	☐	☒	2mm
5		product[thickness act].avg	STRING	☐	☒	2.023mm
6		product[thickness act].min	STRING	☐	☒	1.998mm
7		product[thickness act].max	STRING	☐	☒	2.119mm
8			STRING	☐	☐	
9			STRING	☐	☐	
10			STRING	☐	☐	

The yellow part highlights the selected part. To change the selected part you can edit the "JSON path" column in the signal grid. You can also change it by clicking on the desired row in the preview while the "JSON path" column is selected.

```

{
  "product": {
    "id": "DF123452",
    "weight": "23411kg",
    "length": "2176m",
    "width": "1234mm",
    "thickness set": "2mm",
    "thickness act": {
      "avg": "2.023mm",
      "min": "1.998mm",
      "max": "2.119mm"
    }
  }
}

```

0 256 512 768 1024 1280 1536 1792 ∞ 353 OK Apply Cancel

In the module settings you can also specify the behavior in case only a JSON part was received. You can choose between *Keep last values* and *Reset values*.

The setting *Reset values* sets the numeric and analog signals not contained in the received messages to zero, text signals to an empty string and digital signals to False.

6.5.3 Signal configuration

Configure the text signals in the *Analog* or *Digital* tab as described in the chapters on the split modes.

Note



The value of a digital text signal can only be 0 or 1. All characters in the input text including the 0 (zero) result in the value 0. All numerical values (single or multi-digit) that are not equal to 0 result in the value 1. Alphanumeric combinations also result in 0.

6.6 Text signals via TCP/IP and UDP

The source text is transmitted via TCP/IP telegram or UDP datagram over the computer network to *ibaPDA*. You can set the configuration in the I/O Manager under the node *Text interface*. In the *TCP Text* or *UDP Text* module's *Analog* tab, the source text with the STRING data type and the text signals derived from the source text with the STRING data type or a different data type are configured. For numerical data types, the conversion is carried out automatically.

A *TCP Text* or *UDP Text* module must be configured under the *Text interface* node for each source text or telegram.

Please note the limitation of the receive buffer when configuring the text size.

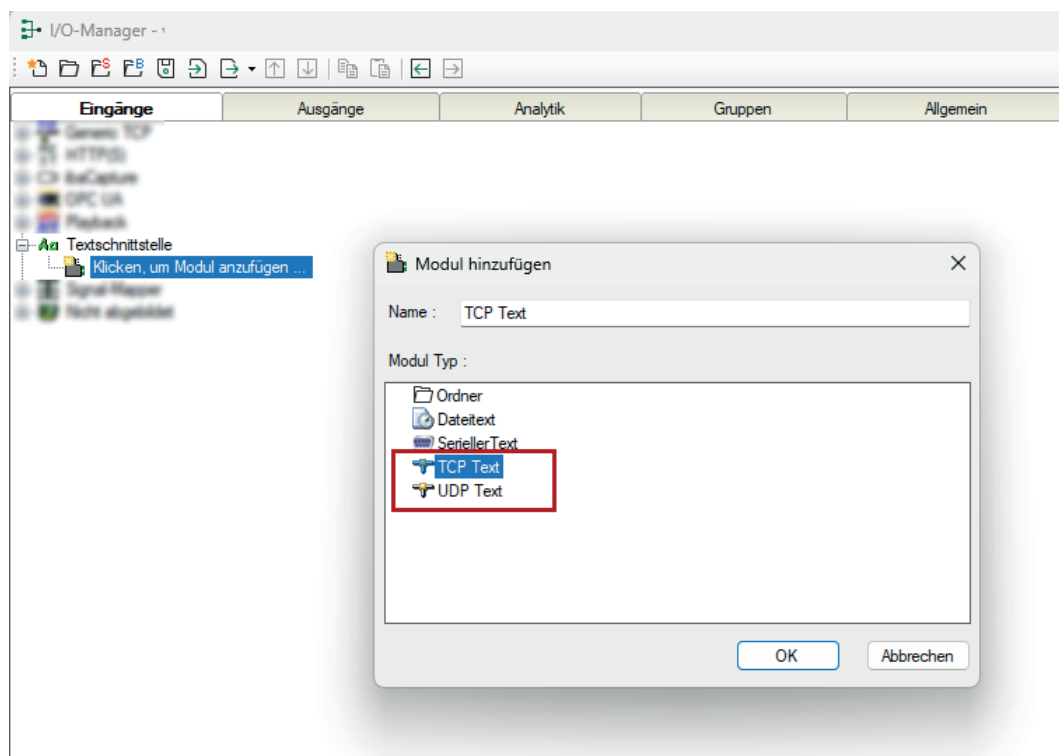
Transmission protocol/module type	Receive buffer size
TCP Text module	20,000 Byte
UDP Text module	64 kB

Thereby, the quantity of data to be processed per message is limited too.

One character needs 1 Byte or 2 Byte, depending on the character set or encoding.

6.6.1 Add module

You can add a module *TCP Text* or *UDP Text* under the node *Text interface*.



6.6.2 Module settings

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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Text encoding

You can select the type of text encoding or the code page here for a correct interpretation and display of the received text data for inputs as well as of the text data to be sent for outputs. Available for selection are, beside system locale according to the Windows system settings (default) and UTF-8 Unicode, all other encodings.

Source

Terminator

This is the agreed termination character for the text, which is expected by *ibaPDA* and which should be used by the transmitting system. The terminator could be any ASCII-character, e. g. Carriage Return or another decimal coded character. You may alter the terminator according to your needs. The default value is 13 ("carriage return").

Port

The port number must match the port number used by the transmitting system in order to establish a TCP/IP connection and receive the text telegrams.

Active connection (TCP Text only)

This option is set to "False" by default. This means that *ibaPDA* waits for a connection to be established by the source computer of the text while listening on the specified port.

Enable this option (true) if you want *ibaPDA* to establish the connection and request data from a host computer. In "Active connection = true" mode, it is necessary to enter a valid IP address or computer name. Entering an alternative address is recommended.

Module structure and processing

See the *Text splitter* module. ➔ *Module settings, page 101*

6.6.3 Signal configuration

The yellow part highlights the selection. To change the selection you can edit the "Begin" and "End" columns in the signal grid. You can also change it by clicking and dragging the mouse in the preview while the "Begin" or "End" column is selected.

Text with two counters: 000461 and 000046

In the *Analog* tab, select the desired characters, parts or JSON values for the text signals, depending on the split mode. In the figure, the *Fixed width* split mode is selected.

You can freely assign the signal name. If numbers are to be numerically interpreted in the text, simply select a corresponding data type for the respective text signal.

In the *Digital* tab, there is only one signal that displays whether there is a connection to the text source.

Tip



A good possibility to test the function of the TCP/IP text is using the [TcpIpTest.exe](#) and [TextSend.exe](#) help programs, which come along with the *ibaPDA*. This means that process computer presets can be simulated as part of a system test or commissioning even if the original systems are not yet available.

6.7 Text signals via serial (COM) interface

The source text is transmitted via a serial interface to *ibaPDA* (COM1...COM4). You can set the configuration in the I/O Manager under the node *Text interface*. In the *Serial text* module *Analog* tab, the source text with the STRING data type and the text signals derived from the source text with the STRING data type or a different data type are configured. For numerical data types, the conversion is carried out automatically.

Depending on the source text or telegram, a module of the type *Serial text* must be configured.

6.7.1 Add module

You can add a module *Serial text* under the node *Text interface*.

6.7.2 Module settings

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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Text encoding

You can select the type of text encoding or the code page here for a correct interpretation and display of the received text data for inputs as well as of the text data to be sent for outputs. Available for selection are, beside system locale according to the Windows system settings (default) and UTF-8 Unicode, all other encodings.

Source**Terminator**

This is the agreed termination character for the text, which is expected by *ibaPDA* and which should be used by the transmitting system. The terminator could be any ASCII-character, e. g. Carriage Return or another decimal coded character. You may alter the terminator according to your needs. The default value is 13 ('carriage return').

COM interface

Here, select the correct COM interface (COM1...COM4) of the server computer which is connected to the source computer.

Baud rate

Select the appropriate baud rate (600...115200) for data transmission from the selection list. Source-system-dependent.

Data bits and stop bits

Select the appropriate settings from the dropdown list. Source-system-dependent.

Parity

Select the parity (none, odd, even, mark, space) from the selection list. Source-system-dependent.

Character spacing

If you enable this option (true), then the telegram will automatically be closed 500 ms after receiving the last character. This is an alternative to the termination character or is a solution if there is no terminator.

Module structure and processing

See the text splitter module. ➤ *Module settings, page 101*

6.7.3 Signal configuration

Configure the text signals in the *Analog* tab as described for the text splitter modules, see ➤ *Text splitter module, page 100*.

In the *Digital* tab, there is only one signal that displays whether a connection is set to the text source.

6.8 Text signals via text file (file text)

The source text is in a text file (.txt, .csv, .json) that is provided by an external system on a dedicated drive in a defined path (local or network). You can set the configuration in the I/O Manager under the node *Text interface*. In the file text module, *Analog* tab, the source text with the STRING data type and the text signals derived from the source text with the STRING data type or a different data type are configured. For numerical data types, the conversion is carried out automatically.

Depending on the source text or file, a module of the type *File text* must be configured.

6.8.1 Add module

You can add a module *File text* under the node *Text interface*.

6.8.2 Module settings

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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Text encoding

You can select the type of text encoding or the code page here for a correct interpretation and display of the received text data for inputs as well as of the text data to be sent for outputs. Available for selection are, beside system locale according to the Windows system settings (default) and UTF-8 Unicode, all other encodings.

Source

Update time

Set the update time here in which *ibaPDA* is to search for and load the specified file in the specified directory.

Default: 1000 ms

Use terminator

If you disable this option (false), then the entire text in the file is used.

If you enable this option, then another line appears for specifying the termination character.

This is the agreed termination character for the text, which is expected by *ibaPDA* and which should be used by the transmitting system. The terminator could be any ASCII-character, e.g., Carriage Return or another decimal coded character. You can alter the terminator according to your needs. The default value is 13 ('carriage return').

File path

Enter a valid path and the full file name of the file containing the text in this field. If *ibaPDA* finds the relevant file, it will be immediately read. The file content is interpreted as a source text.

Wildcards are supported in file name. This allows you to use, e.g., a combination of prefix and dynamic part in the file name.

Therefore, the source system generating the text files must ensure that a text file with the source text is copied **in time** into the specified path whenever a new text should be available for *ibaPDA*.

User name, password

When the file path is located on a network drive, enter the login information here for the access right to the drive.

File mode

By selecting the file mode, you specify how to proceed with the file after it has been acquired and read.

■ Delete file

The file is read and deleted immediately. If the same text is to be processed again or another text is to be processed, a new file must be provided again. This setting is suitable if it has been ensured that a file with the correct text will always be available at the right time.

■ Keep file

The file is read and not deleted. The file is read again if its modification date changes. Thus, you can transmit new texts to *ibaPDA* by overwriting the file. This setting is suitable if it is possible or likely that there is not always an up-to-date file and the old text is the "lesser evil" compared to an empty text.

Module layout and processing

See the text splitter module. ➔ *Module settings, page 101*

6.8.3 Signal configuration

Configure the text signals in the *Analog* tab as described for the text splitter modules, see [➤ Text splitter module, page 100](#).

In the *Digital* tab, there is only one signal that displays whether a connection is set to the text source.

The following figure shows an example for text coming out of a text file in “delimited” split mode.

General Analog Digital Diagnostics								
	Name	Unit	Begin	End	DataType	Filter	Acti...	Actual
0	Housing		0	0	STRING[1]	☐	☐	TS8
1	Height		1	1	STRING[1]	☐	☐	2000
2	Width		2	2	STRING[1]	☐	☐	800
3	Depth		3	3	STRING[1]	☐	☐	800
4	Protection		4	4	STRING[1]	☐	☐	IP41
5	Color		5	5	STRING[1]	☐	☐	RAL7035
6	Plate thickness		6	6	STRING[1]	☐	☐	1,5
7	Door		7	7	STRING[1]	☐	☐	Front
8			8	8	STRING[1]	☐	☐	

The yellow part highlights the selection. To change the selection you can edit the "Begin" and "End" columns in the signal grid. You can also change it by clicking and dragging the mouse in the preview while the "Begin" or "End" column is selected.

```

TS 8
2 0 0 0
8 0 0
8 0 0
IP 4 1
RAL 7 0 3 5
1 , 5
Front

```

6.9 Custom text (text creator)

The source text is created on the *ibaPDA* server in the module *Text creator*, tab *Text creation*. It can be a typed in fixed text, the value of a signal or the text of another text signal. It can also be a combination of all three forms. Furthermore, XML and JSON data structures can be used for text creation. You can set the configuration in the I/O Manager under the node *Virtual*. In the *Analog* tab of the same module, you will then configure the derived text signals with the data type *STRING* or a different data type. For numerical data types, the conversion is carried out automatically.

Depending on the source text, a module of the type *Text creator* must be configured.

6.9.1 Add module

Select the *Analytics* tab in the I/O Manager and add a *Text creator* module under the *Virtual* node.

6.9.2 Module settings

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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Format

Mode

The mode determines the format which is used to create the source text.

You can choose between the following options:

- Plain text (default): Create a simple, plain text
- XML: Create an XML structure
- JSON: Create a JSON structure

Compact text (XML/JSON mode only)

If you disable this option (false), then the text of the output signal is formatted with indentation for better readability (as it is done in the preview). If you let this option enabled (true, default), the text is formatted without indentation.

Format Booleans as integers (XML/JSON mode only)

In the XML/JSON format, Booleans are always resolved to *true* or *false*.

Enable this option if you want to use Boolean values as numbers 0 and 1.

Note



Changing the plain text setting to either JSON or XML will remove the configured signals, derived from the plain text. A warning is displayed to prevent unintentional changes.

Module structure (Plain text only)

The number of signals is set to 32 analog signals and 0 digital signals. You can change the numbers, if required. Permissible range: 1 to 1000.

Processing

Split mode (Plain text mode only)

Select the appropriate mode, which corresponds to the structure of the source text in order to read the information correctly:

- Fixed width (description see ↗ *Fixed width split mode, page 103*)
- Delimited (description see ↗ *Delimited split mode, page 104*)
- JSON (description see ↗ *Split mode JSON, page 106*)

Note



The split mode set here applies to the formation of text signals in the *Analog* tab based on the static texts, process signals and text signals defined in the *Text creation* tab. If you select the *Delimited* split mode, for example, then a new part will be recognized whenever a corresponding delimiter appears in the created text.

Remove new lines (Plain text mode only)

If you enable this option (True), then all line feeds will be removed when parsing the source text.

Replace character (Plain text mode only)

Here, you can enter a character which should be used as replacement for non printable characters. Default: x.

Decimal separator (Plain text mode only)

Select the appropriate decimal separator in order to read numbers with decimals correctly: point or comma.

→ Continue with *Plain text* text creation: ↗ *Text creator (Plaintext)*, page 118

→ Continue with *XML/JSON* text creation: ↗ *Text creation XML/JSON*, page 121

6.9.3 Text creator (Plaintext)

In the *Text creation* tag, you construct the text for the text signals to be created in the next step. The following figure shows an example for text creation with static text, text signals and measurement signal.

Source	Format	Length
Model:		7
21:0: Housing_type		6
25:0: SN	1	5
11:10: Input Quality		32
11:11: Input Class		3
*		

Example:

Model: TS8 4202very good A

If you want to use the signal values of other signals in the new text, this must have been defined beforehand.

Above the table, you can define when the new text is to be created.

Create new text in the event of any change.../rising edge...

The choice between these two options determines whether the text is to be considered a new text as soon as its content changes or with the rising edge of a digital signal.

If you choose the first option, then a new text is automatically generated if the values of the source signals change that are part of the text. Changes that occur within the period of the set dead time are ignored.

If you intend to control the text creation with a signal, enable the second option and select a digital signal from the dropdown list in the tree structure.

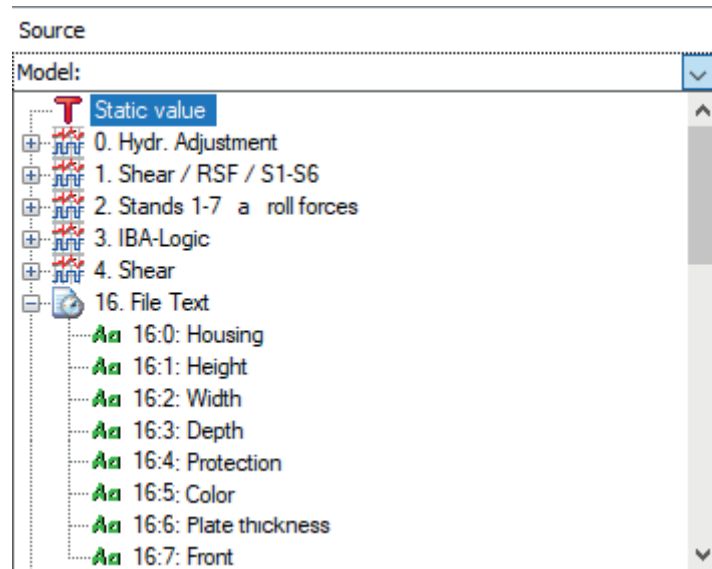
Configuration table for the text

Configure the text using the table. Each row corresponds to a source delivering alpha-numeric data. One or more sources, or rows, can be configured for a text. In total, the resulting text may not contain more than 10,000 characters.

As soon as writing starts in one row, the table automatically expands by a row.

The sequence of the sources (top-down) determines the sequence of their contents in the text (left-right). The arrow keys on the right next to the table allow you to change the arrangement of the sources. If you want to remove a source from the table, select the corresponding row and click on the button .

There are different types of sources available. You can select the source type or the source signal in the first column of the table using the dropdown list in the cell.



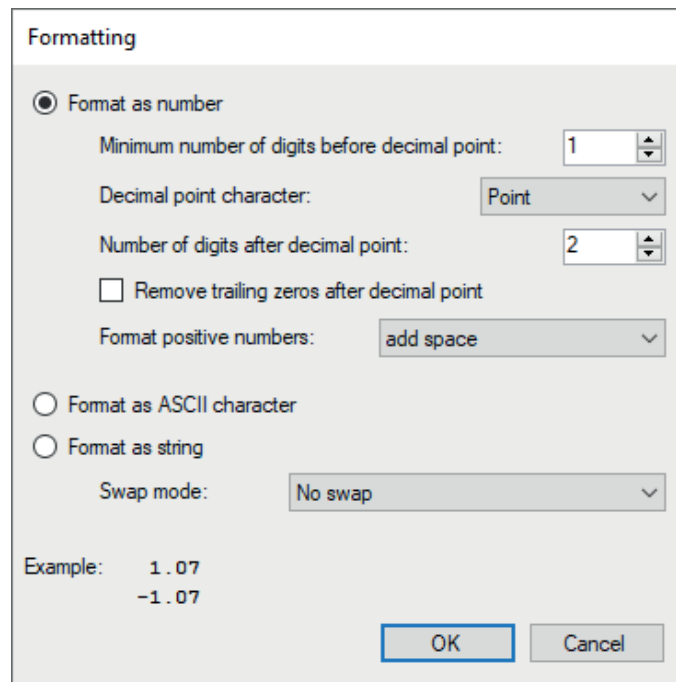
You have the following options:

Static value

Select this type if you want to enter fixed text, which does not change. Enter the text in the *Source* column. The *Format* column is not relevant. The *Length* column shows the character length of the text. It is updated automatically when you click outside the *Source* column.

Signals from different interfaces

Select any signals from the signal tree to use the values of measured or virtual signals in your text. Optionally, you can specify the format of the signal value. To do so, click the <...> browser button in the *Format* field.



The image shows a 'Formatting' dialog box with the following settings:

- Format as number** (selected with a radio button)
- Minimum number of digits before decimal point: 1
- Decimal point character: Point (dropdown menu)
- Number of digits after decimal point: 2
- ☐ Remove trailing zeros after decimal point
- Format positive numbers: add space (dropdown menu)
- Format as ASCII character** (radio button)
- Format as string** (radio button)
- Swap mode: No swap (dropdown menu)
- Example: 1.07, -1.07
- Buttons: OK, Cancel

If you like to use the value in decimal format specify the number of digits before and after the decimal point. Consider the possible size of the signal value and allow for enough digits. If negative values are permitted, decide how to deal with positive numbers; either choose from the selection list whether a space or a "+" should be added or no operation is required.

If you enable the Option *Format as ASCII character* option, the signal value will be interpreted as ASCII code and the corresponding ASCII character will be used for the text, e.g., "65" --> "A."

If you select the option *Format as string*, the raw bytes of the signal value are interpreted as characters. A floating point value is 4 bytes which are interpreted as 4 characters. Int32 is also 4 characters and Int16 is 2 characters. You can choose between different swap options. The general result is shown in the example area of the dialog window.

Text signal

Simply select already existing text signals (recognizable by the green Aa icon) if you want to re-use existing texts in the custom text. The *Format* column is not relevant. In the *Length* column, you will initially find the length of the selected text signal (number of characters). You can shorten the length if you only want to use part of the original text signal.

Once you have set up the source(s), you will see the complete text in the preview area under the table.

The preview area only shows the apposition of the text from the sources and does not take any delimiters into consideration. The latter first occurs in the next step, in the *Analog* tab, provided the split mode *Delimited* was selected in the *General* tab.

6.9.4 Text creation XML/JSON

If you select the mode XML or JSON under *Format* in the module settings, then you will get the tab *Xml text creation* or *Json text creation*.

In this tab, you can define the data structure, which is then to be used in the next step as text source.

Note



The module is able to translate XML-specific structures into valid JSON and vice versa. However, it is not recommended to define an XML-targeted structure (e.g. with XML attributes) and then use it as JSON.

The following example explains the text creation with JSON.

On top, you can define when the new text is to be created.

Create new text in the event of any change.../rising edge...

The choice between these two options determines whether the text is to be considered a new text as soon as its content changes or with the rising edge of a digital signal.

If you choose the first option, then a new text is automatically generated if the values of the source signals change that are part of the text. Changes that occur within the period of the set dead time, are ignored.

If you intend to control the text creation with a signal, enable the second option and select a digital signal from the dropdown list in the tree structure.

<Load file> button

If you already have a file (*.json or *.xml) at hand, which you want to use as a template for the data structure, you can import the file over this button.

Synchronize array items

If you enable this option, then all structural changes you make for one array item will be copied to all other items of the same array.

For more information see chapter [↗ Synchronize arrays, page 128](#)

Configuration area

The dialog is divided in 3 sections:

- Signal tree (left)

The signal tree contains all applicable signals, see [↗ Signal tree, page 126](#)

- Data structure tree (center)

In the data structure tree you define the data structure. [↗ Data structure tree, page 122](#)

- Preview pane (right)

The preview shows the result of the data structure. Whenever you change the data structure, the preview is refreshed automatically. The preview does not display signal values. Instead, it shows a combination of signal name and signal ID.

6.9.4.1 Data structure tree

Learn here, how you can set up a data structure.

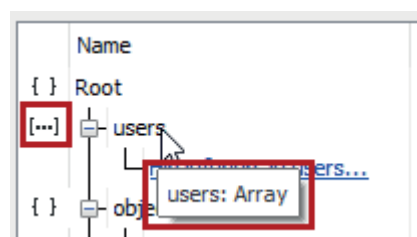
The data structure is represented by nodes. Depending on the node type more nodes can be nested infinitely in terms of number and depth.

Node types

The following node types are available.

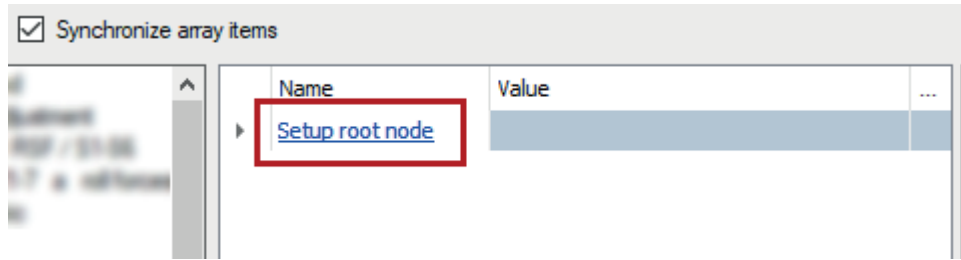
Symbol	Name
{ }	Object
[...]	Array
{X}	Property
	Attribute (XML mode only)

The node type is indicated by an icon in front of the name or as a tooltip when hovering the mouse over the node.

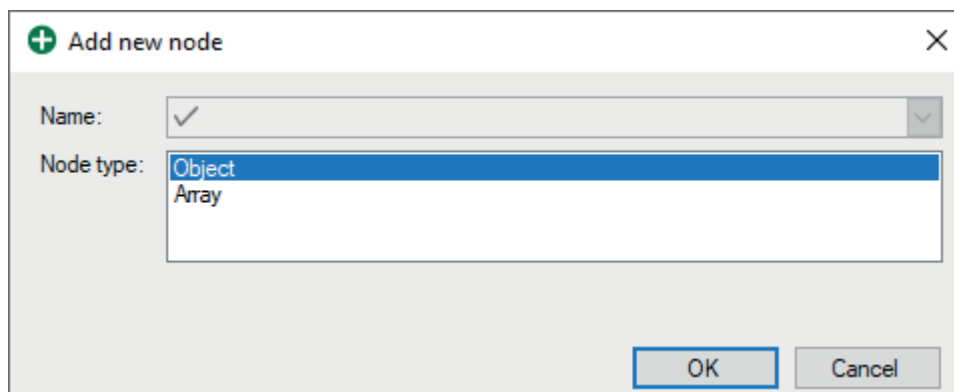


To build a data structure proceed as follows:

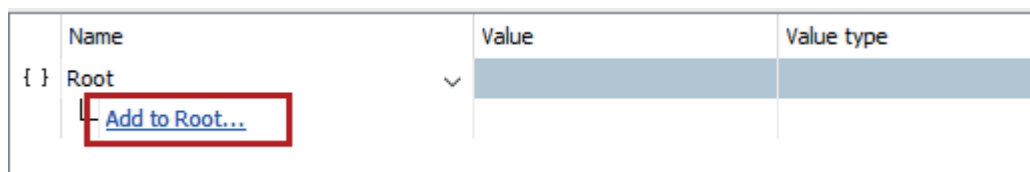
1. Before configuring a data structure the root node must be defined. Therefore, click on *Set up root node* in the *Name* column.



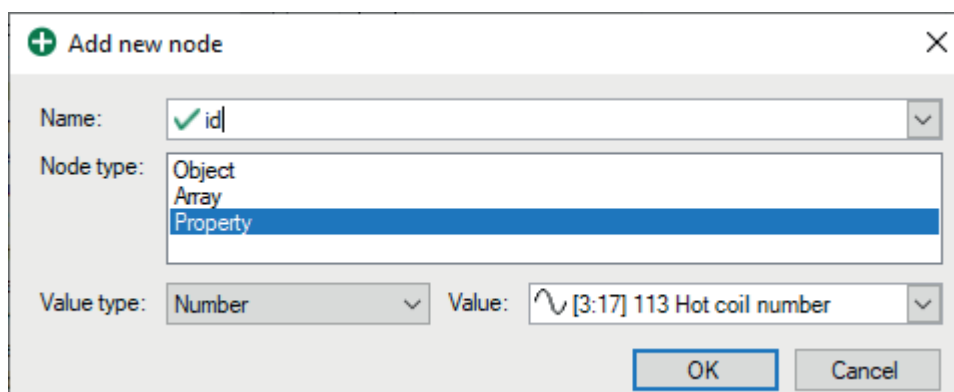
2. A dialog opens, providing the option to either add the root node as an object or as an array. In most cases, the root node is an object. Select *Object*, for example, and click <OK>. You cannot enter a name because the *Name* field is not available.



3. Click on *Add to root...* to add more nodes.



4. In the following dialog select the desired node type for each, enter or select the additional information and add the node by clicking on <OK>.



5. In this way, arrange your data structure.

Name	Value	Value type
{ } Root		
[...] users		
{ } [0]		
	Add to [0]...	
{ } [1]		
	Add to [1]...	
	Add to users...	
{ } object		
	Add to object...	
{x} property	Aa test	String
	Add to Root...	

Properties of node types

Note



You can always add nodes of the types *Object*, *Array* or *Property* to objects. In XML mode you can additionally choose the *Attribute* type.

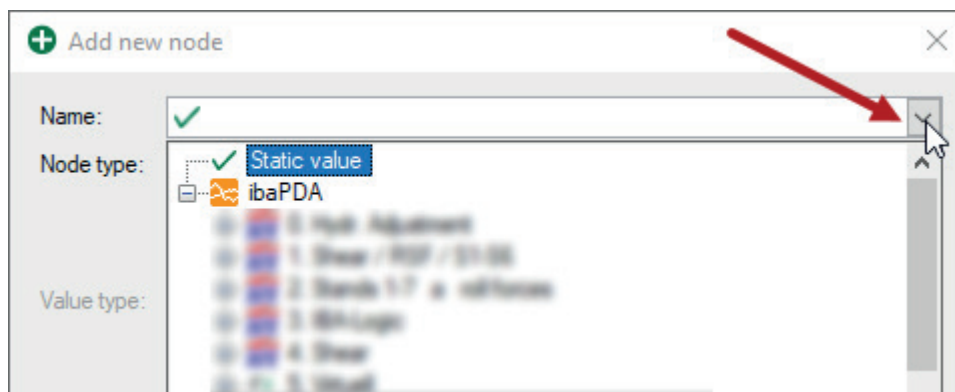
Name

The node name can be a static value or a signal.

Enter the static name manually in the field.

The node names on the same tree level must be unique. If you enter duplicate names, then an error message pops up and the entry is rejected.

If you want to use a signal, then click on the arrow icon for the drop-down list and select the desired signal from the signal tree. The signal name will be used for node name.



Note



Nodes whose parent node is an array cannot have a name. Nodes or items in an array will always be added with their value.

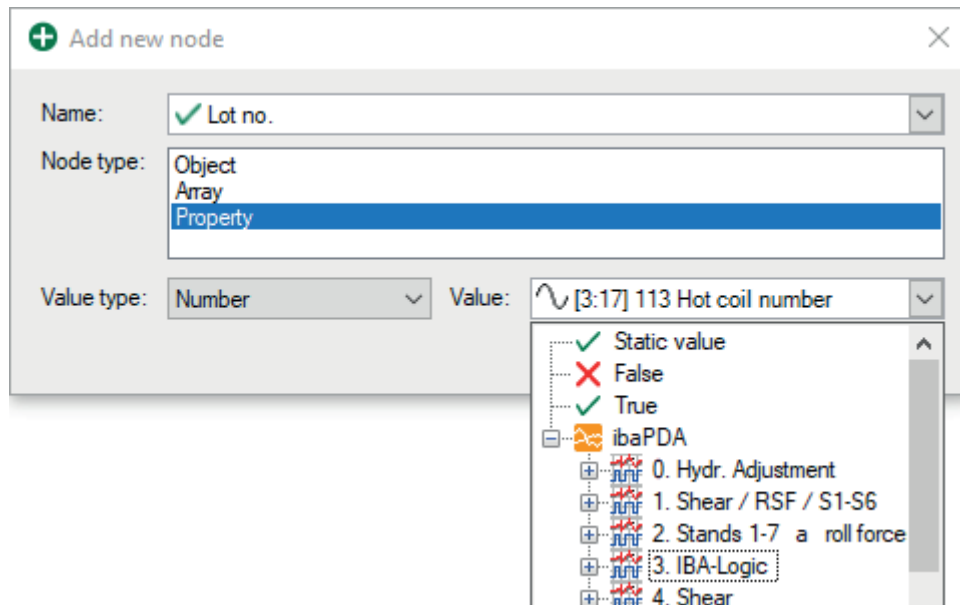
To be able to define a name for a new subordinated node, first add an object to the array and then add the new node to that object.

Value type and value

These two parameters are available only for node type *Property*. *Object* and *Array* do not offer these settings.

The following value types are available for *Property*:

Value type	Static value	Value from signal
String	Select <i>Static value</i> and enter text	Select text signal
Number	Select <i>Static value</i> and enter number	Select analog signal
Boolean	Select True or False	Select digital signal



Note



Because XML does not have the concept of data types, Booleans are always resolved to *true* or *false* in the XML mode.

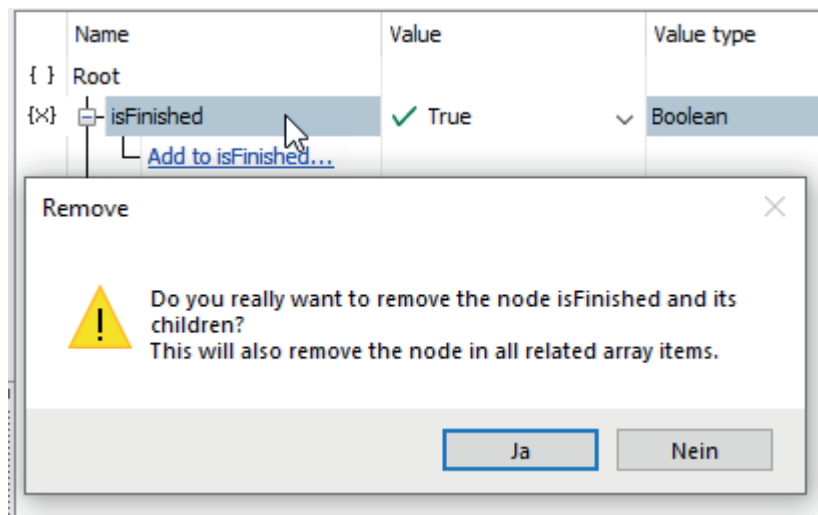
Name	Value	Value type	
{ } Root			
{x} isFinished	✓ True	Boolean	
└ Add to isFinish...			
{x} isValid	✗ False	Boolean	
└ Add to isValid...			
└ Add to Root...			

```
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <isFinished>True</isFinished>
  <isValid>False</isValid>
</Root>
```

Remove nodes

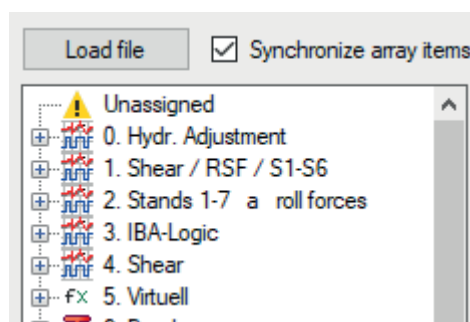
In order to delete a node from the data structure, make a right mouse click on the node in question and select *Remove [x]* in the context menu.

Alternatively, you can press the key while the node in question is selected in the data structure tree. To prevent unintentional removal a warning pops up.



6.9.4.2 Signal tree

The signal tree in the left pane of the dialog XML/JSON text creation provides all applicable signals for the data structure.



You can add one or more signals by drag & drop. Depending on where you drop the signal from the signal tree, different actions are taken. When dragging the mouse over the nodes, a tooltip shows the action to be taken when dropping the signal.

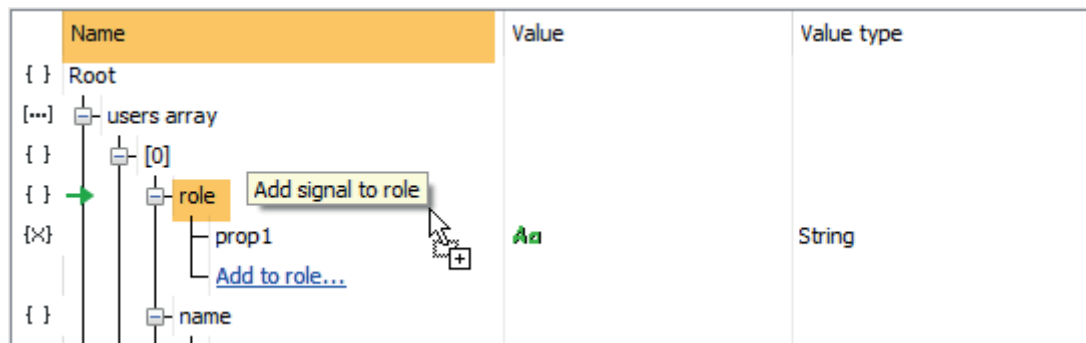
The following table shows the different actions depending on node type and column:

Drop signal on...	... in name column	... in value column
Object	Signal will be added as a new node on the same level.	Value type and value will be replaced by the signal.
Array	Signal will be added as a new node on the same level.	Value type and value will be replaced by the signal.
Property	Name, value type and value will be replaced by the signal.	Value type and value will be replaced by the signal.

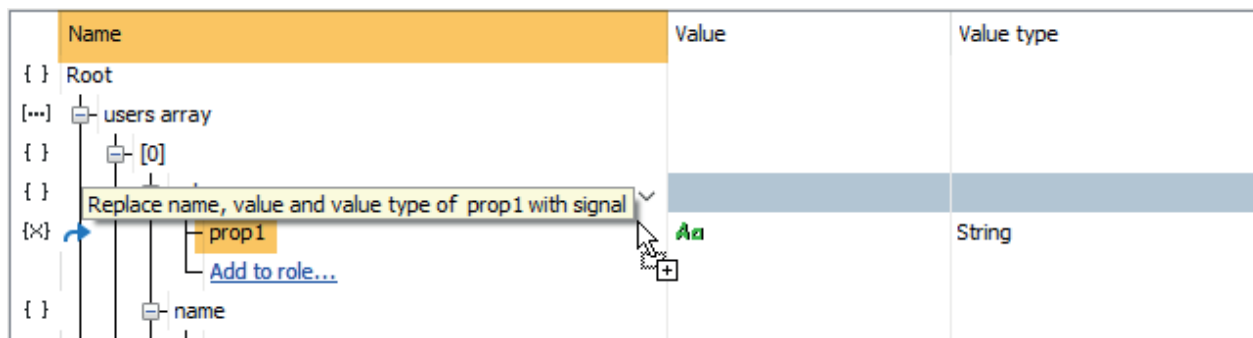
If you select multiple signals in the signal tree at once and drag them on the links *Add to...* or on an object/array, they are added.

Examples

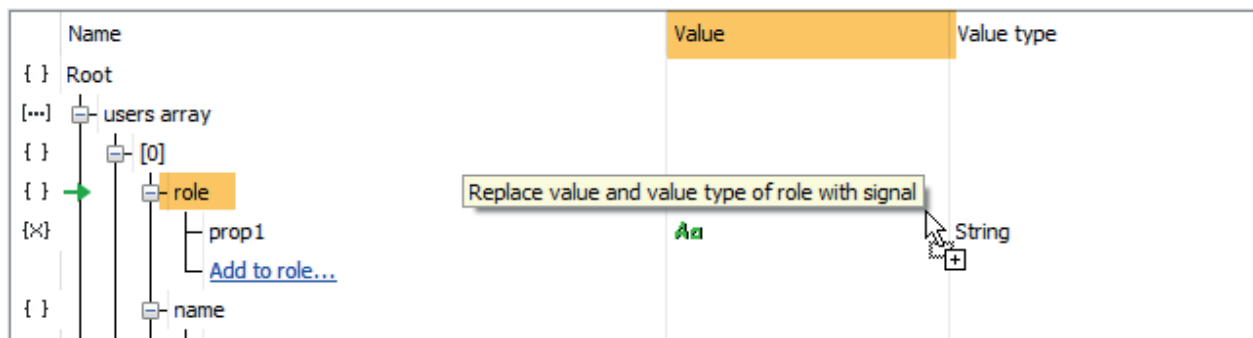
Drag signal on an object, *Name* column



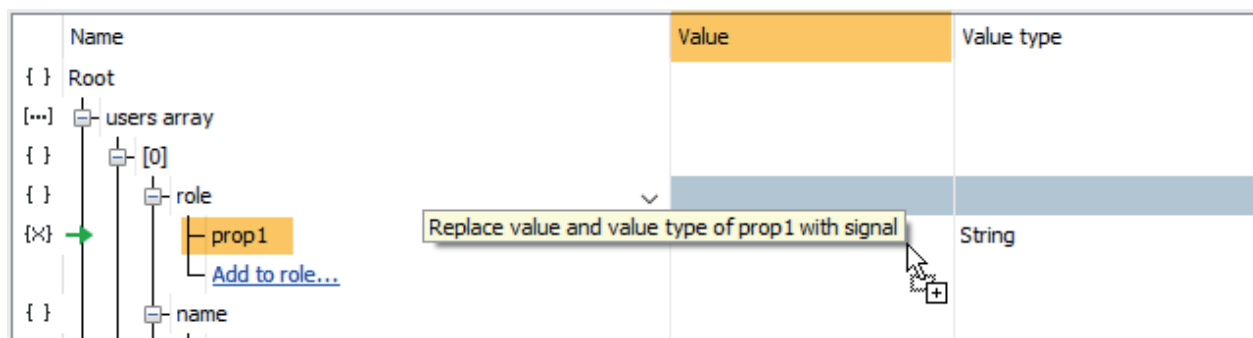
Drag signal on a property, *Name* column



Drag signal on an object, *Value* column



Drag signal on a property, *Value* column

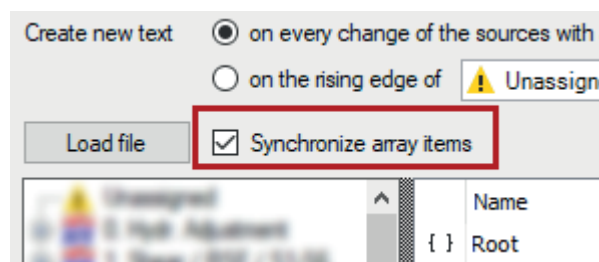


6.9.4.3 Synchronize arrays

When working with larger and more complex data structures, it takes a lot of effort to change the structures of arrays and keep them consistent.

For example, when renaming a property in the first array item, you would also have to rename it in every other array item of the same array to keep the structure the same.

For your convenience, there is the option *Synchronize arrays* in the dialog of XML/JSON text creation.



This function only applies to changes which you currently make. Differently structured arrays and items which are already existing will not be synchronized retroactively by enabling this option.

Array items which have been created by the synchronization option can be altered individually after disabling the option.

Example

The following structure shows renaming the node “text” to “comment” in the first array item (left picture). Due to the synchronization option all other corresponding array items are re-named, too (right picture).

Name	Value	Value type	Name	Value	Value type
{ } Root			{ } Root		
[...] rooms			[...] rooms		
{ } [0]			{ } [0]		
{< } room no.	~ 1	Number	{< } room no.	~ 1	Number
{< } status	Aa occupied	String	{< } status	Aa occupied	String
[...] cleaning			[...] cleaning		
{ } [0]			{ } [0]		
{< } text	Aa morning	String	{< } comment	Aa morning	String
Add to [0]...			Add to [0]...		
{ } [1]			{ } [1]		
{< } text	Aa afternoon	String	{< } comment	Aa afternoon	String
Add to [1]...			Add to [1]...		
Add to cleaning...			Add to cleaning...		
Add to [0]...			Add to [0]...		
{ } [1]			{ } [1]		
{< } room no.	~ 2	Number	{< } room no.	~ 2	Number
{< } status	Aa empty	String	{< } status	Aa empty	String
[...] cleaning			[...] cleaning		
{ } [0]			{ } [0]		
{< } text	Aa pause	String	{< } comment	Aa pause	String
Add to [0]...			Add to [0]...		
{ } [1]			{ } [1]		
{< } text	Aa maintenance	String	{< } comment	Aa maintenance	String
Add to [1]...			Add to [1]...		
Add to cleaning...			Add to cleaning...		
Add to [1]...			Add to [1]...		
Add to rooms...			Add to rooms...		
Add to Root...			Add to Root...		

6.9.5 Signal configuration

Configure the text signals in the *Analog* tab as described for the text splitter modules, see [↗ Text splitter module, page 100](#).

T General
Aa Text creation
Analog
Diagnostics

	Name	Unit	Begin	End	DataType	Filter	Acti...	Actual	
0	Full_text		0	29	STRING[30]	☐	☒	Model: TS8 4201 very good	^
1	Model		7	10	STRING[4]	☐	☒	TS8	
2	SerialNo		11	17	STRING[7]	☐	☒	4201	
3	Quality		18	28	STRING[11]	☐	☒	very good	
4	Class		29	33	STRING[5]	☐	☒	1A	
5			5	5	STRING[1]	☐	☐	:	v

The yellow part highlights the selection. To change the selection you can edit the "Begin" and "End" columns in the signal grid. You can also change it by clicking and dragging the mouse in the preview while the "Begin" or "End" column is selected.

Model: TS8 4201 very good 1A
Preview

6.10 Text via ibaQPanel text input

The text can be entered manually via a text input control of *ibaQPanel*. You can set the configuration in the I/O Manager under the node *Virtual*.

A module of the type *ibaQPanel text input* can receive several text signals. In the process, every text signal corresponds to exactly one text input control in the *ibaQPanel* view. Optionally, you can break down these text signals into sections with a *text splitter* module and, if necessary, convert them into other data types.

6.10.1 Add module

You can add a module *ibaQPanel text input* under the node *Virtual*.

6.10.2 Module settings

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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Text encoding

You can select the type of text encoding or the code page here for a correct interpretation and display of the received text data for inputs as well as of the text data to be sent for outputs. Available for selection are, beside system locale according to the Windows system settings (default) and UTF-8 Unicode, all other encodings.

ibaQPanel input

Remember last values

If you enable this option, then in the case of a new I/O configuration, the last known values are loaded instead of the default values *Analog*, *Column Default*).

Module structure

See the text splitter module. ➔ *Module settings, page 101*

6.10.3 Signal configuration

General		Analog			
	Name	Initial value	Length	Active	Actual
0	Input_Quality		32	☒	sehr gut
1	Input_Class		32	☒	A
2			32	☒	

Each line corresponds to a text signal. Each text signal can be assigned a text input control in *ibaQPanel*. In the figure, the *ibaQPanel* text input module has for instance two inputs.

If necessary, you can specify a default text, which is a value in the text signal, if no input has occurred via the *ibaQPanel* element, e.g. "Text input required."

Tip



The default text is also shown in the *ibaQPanel* text input control if you set the text mode "Input == Output" in the properties of the element.

Since the text input control does not have any length restriction for the text, you can define the number of characters in the *Length* column. Before the text input, only the first *n* characters are used as the value for the text input control.

Example for configuration of the ibaQPanel text inputs

Quality		Apply
Class		Apply

Objects in the Qpanel view for the input of quality information. Clicking on <Apply> will apply the input text as a value of the assigned text signal.

General	Input == Output	Input <> Output
Signals		
Destination signal:	A [11:0] Input_Quality	
Trigger signal:	X Unassigned	Reset signal: X Unassigned

The desired text signal from the *ibaQPanel text input* module must be selected as the target signal in the properties of the *ibaQPanel* text input element.

6.11 Text shift register

Using the *Text shift register* module, the last 1 to 64 values of other text signals can be saved. The current value of one or more text signals is applied to the shift register with a trigger signal (rising edge). You can set the configuration in the I/O Manager under the node *Virtual*.

The required text signals must have been created beforehand with the different modules.

The memory depth of the shift register ([1]... [n]) can be adjusted from 1 to 64.

The shift register module provides all tab contents indexed as new text signals (text a [1], text a [2], ... text a [n], with n = depth of the shift register)

The values in the shift register are moved with a trigger signal (rising edge of a digital signal).

All texts contained in the shift register can be used independently of each other, e. g., for display purposes.

6.11.1 Add module

You can add a module *Text shift register* under the node *Virtual*.

6.11.2 Module settings

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.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

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 module name as prefix

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

Text encoding

You can select the type of text encoding or the code page here for a correct interpretation and display of the received text data for inputs as well as of the text data to be sent for outputs. Available for selection are, beside system locale according to the Windows system settings (default) and UTF-8 Unicode, all other encodings.

Shift register**Source signals**

Select the text signal(s) here that are to be processed in the shift register.

Trigger signal

Select the digital signal with the rising edge of which the current value of the source texts is stored and the values in the tab are shifted one position.

Reset signal

Select the digital signal here with whose rising edge all values in the shift register are cleared.

Depth

Select the desired memory depth ([n]) of the shift register. The values can be adjusted from 1 to 64.

Remember last values

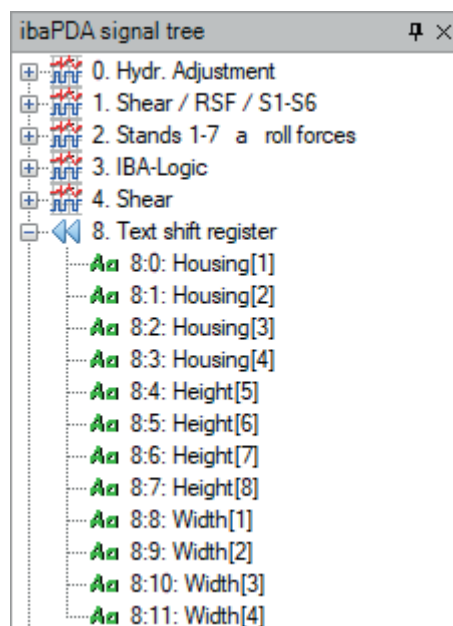
If you enable this option, then in the case of a new I/O configuration or when restarting the acquisition, the last known values are loaded instead of the default values (*Analog*, *Column Default* tab).

6.11.3 Signal configuration

General		Analog				
	Name	Unit	Initial value	Act...	Actual	+
0	Housing[1]			☒		
1	Housing[2]			☒		
2	Housing[3]			☒		
3	Housing[4]			☒		
4	Height[5]	inch		☒		
5	Height[6]	inch		☒		
6	Height[7]	inch		☒		
7	Height[8]	inch		☒		
8	Width[1]	inch		☒		
9	Width[2]	inch		☒		
10	Width[3]	inch		☒		
11	Width[4]	inch		☒		

The number of resulting text signals is determined by the number of source signals and the register depth. The figure shows the text signals of a text shift register with a depth of 4.

The text signals provided by the text shift register are available in the signal tree:



Tip



For analog signals, there is a corresponding function with the "Shift register" module in the *Virtual* node, *Hardware* area in the I/O Manager.

6.11.4 Application example for text shift register

For a facility that produces steel sheets for the automotive industry, an on-screen display is to be made with *ibaQPanel* showing the customer names of the last 5 coils produced in addition to the batch numbers. The “customer” text signal is created in a text shift register. The already existing “QP_Customer” text signal always contains the appropriate customer name for the production. This is manually entered into *ibaQPanel* but it can also be any other automatically-generated text. The trigger signal is fired each time a new coil enters the plant. Since the last 5 coils are supposed to be displayed, the register depth is set to 5.

Text shift register Custom (8)	
General	
Module Type: Text shift register	
Locked	False
Enabled	True
Name	Text shift register Custom
Module No.	8
Timebase	10 ms
Use module name as prefix	False
Shift register	
Source signals	[11:2] QP_Customer
Trigger signal	[3.11] F1 Stand loaded
Reset signal	[12.0] Reset button
Depth	5
Remember last values	False

Five digital text displays are configured in *ibaQPanel*, each of which accesses an indexed shift register text signals.

[1] = latest register, [5] = oldest register

Recent coils

Current/recent coil N	Customer
8.704208	ACME
Coil N-1	
8.704200	ACME

Properties

Text digital display

Common

- Position
- Visibility
- Enabled state
- User interaction

Data

Data signal:

Update interval:

Meter

Back color:

Title font:

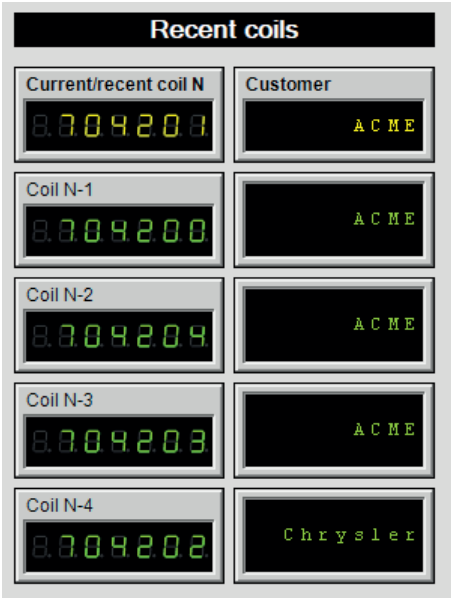
Title:

8. Text shift register Customer

- 8.0: QP_Customer[1]
- 8.1: QP_Customer[2]
- 8.2: QP_Customer[3]
- 8.3: QP_Customer[4]
- 8.4: QP_Customer[5]

9. Text splitter

The result: The customer names for the last 5 coils now appear in the right row. The most recent or current coil is at the top. The left row was implemented with digital numeric displays and analog signals in the (numeric) shift register module.



6.12 Text signals via OPC / OPC UA

The source text is transmitted to *ibaPDA* via the PC network (Ethernet TCP/IP) by an OPC or OPC UA server. You can set the configuration in the I/O Manager under the node *OPC* or *OPC UA*. In the *OPC client* or *OPC UA client* module, *Analog* tab, the source text is configured with the data *STRING*. To divide the text from the OPC tag onto individual text signals, a module *Text splitter* must be configured per source text under the *Virtual* node.

6.13 Text signals via other interfaces

Other documentation



Text signals are supported for a number of additional interfaces. For a description of the settings, please refer to the manual for the interface product.

- Reflective Memory
 - HiPAC Request
 - ScramNet+
 - SIMATIC TDC/SIMADYN D
 - EtherNet/IP
 - S7-Xplorer
 - B&R-Xplorer
 - TwinCAT-Xplorer
 - Codesys-Xplorer
 - Logix-Xplorer
 - Generic TCP
 - Generic UDP
 - TwinCAT Request
 - Modbus TCP Server
 - Sisteam TCP
 - TDC TCP/UDP
 - VIP TCP/UDP
 - S7 TCP/UDP
 - ibaVision
-

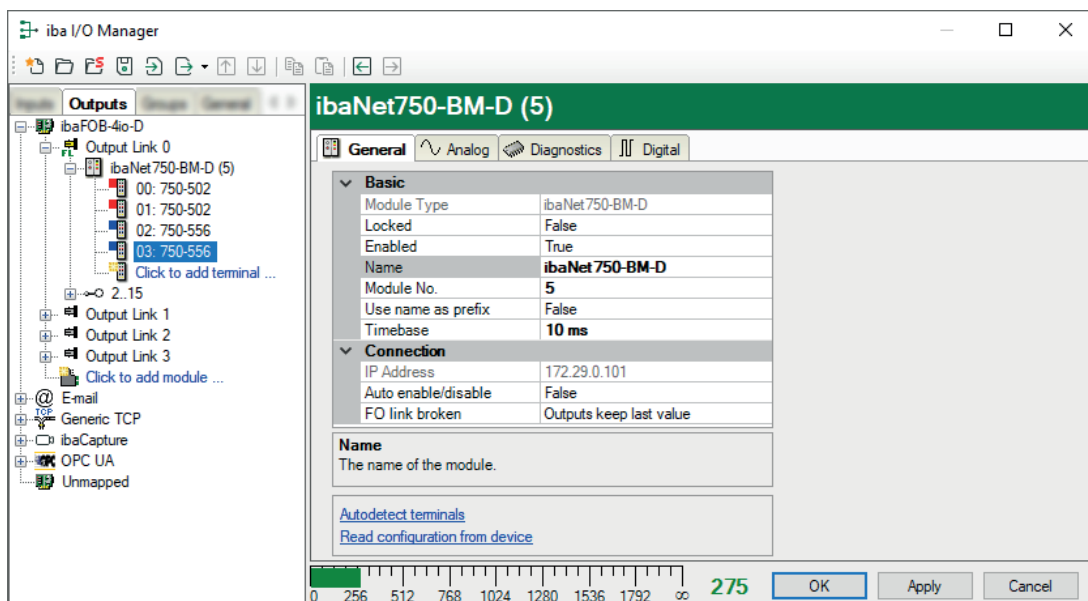
Interface	Output modules	Comment
OPC UA	OPC UA client OPC UA server	Only available with license OPC UA client
E-mail	E-mail	Standard, always available up to 256 configurable e-mails
Generic TCP	Generic TCP Output	Only available with ibaPDA-Interface-Generic-TCP license; passive mode is supported.
Generic UDP	Generic UDP Output	Only available with ibaPDA-Interface-Generic-UDP license; passive mode is supported.
EtherNet/IP	EtherNet/IP I/O Scanner EtherNet/IP I/O module	Only available with ibaPDA-Interface-EtherNet/IP
Reflective memory	Reflective Memory Output	Only available with RM board and ibaPDA-Reflective Memory Access license
Modbus TCP Client	Modbus client	Only available with ibaPDA-Interface-Modbus over TCP-Client license
ABB-Xplorer	ABB Xplorer MMS	Only available with ibaPDA-Interface-ABB-Xplorer license
OMRON-Xplorer	OMRON FINS	Only available with ibaPDA-Interface-OMRON-Xplorer license
S7-Xplorer	S7-Xplorer	Only available with ibaPDA-Interface-S7-Xplorer license
TwinCAT-Xplorer	TwinCAT PLC	Only available with ibaPDA-Interface-TwinCAT-Xplorer license
MQTT	MQTT	Only available with ibaPDA-Interface-MQTT license
SQL database	SQL command	Only available with at least one of the following licenses: ibaPDA-Interface-MySQL, -Oracle, -PostgreSQL, -SAP-HANA or -SQL-Server

Table 8: Interfaces with output function and corresponding output modules

Note



The calculation timebase is a time base specifically for the calculation of output values. You can set this time base independently of the general timebase and the timebase of the input modules. Output values are sent with a low priority and can always be delayed or displaced by the higher prioritized tasks of the measured value acquisition, since they represent the core function of *ibaPDA*. The shortest achievable cycle time of the outputs is independent of the calculation timebase and is 50 ms or equal to the smallest common multiple of all input time bases. Check the current output cycle time of the system in the I/O Manager under *General* tab, *Module overview* node.



7.1 Create outputs

The basic steps to configure alarms and output signals are:

- Installation of a suitable interface card in the *ibaPDA*-server PC and/or licensing of a suitable interface.
- Adding the appropriate output module in the I/O Manager
- Definition and activation of digital and/or analog output signals

The configuration of the output signals follows the same concept as for the measured signals.

7.2 Adding output modules

The method of adding output modules is basically the same for all types. Only the module and signal settings vary.

If the required hardware is installed (see above), follow these steps:

1. Open the I/O manager and select the *Outputs* tab.
The tree structure shows an overview of the available data interfaces with their output links.

2. By *Click to add module ...* you add a new module to the respective interface.
A dialog opens where you can name the new module. Select the appropriate module type.
3. Enter a name for the module and press <OK> to close the dialog.
4. The new module will be added to the next free link of the interface.

Of course, you can also right-click on the link of your choice in order to add a module via *Add module* in the context menu.

You can now configure the module settings and signals.

7.3 Setting up output modules

7.3.1 General module settings

The general module settings are usually defined when adding a module in the *Inputs* area in the I/O Manager and mostly apply to input and output modules.

Nevertheless, there are some specific settings that are different for the outputs. In the basic settings, for example, there is the "Calculation time base" instead of the "Time base".

Basic settings

Module Type (information only)

Indicates the type of the current module.

Locked

A module can be locked in order to prevent change of module settings by accident or unauthorized users.

Enabled

Disabled modules are excluded from the signal acquisition.

Name

The plain text name should be entered here as the module designation.

Module No.

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

Calculation timebase

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

The calculation timebase and update timebase are NOT the same!

For interfaces whose modules have an input and an output side (same module number), the calculation timebase is identical to the module timebase on the input side. Hence, changing the calculation timebase also changes the module timebase on the input side and vice versa.

For interfaces with separate input and output modules (different module numbers), the calculation timebase can be changed independently.

The smallest possible calculation time base, i.e., the fastest possible update of the outputs, results from the smallest common multiple of all module time bases, and is at least 50 ms. .

This setting does not exist for the E-mail, OPC output and SQL command output modules.

Minimum output timebase (information only)

Smallest timebase for updating the outputs.

This value is calculated automatically by the system considering the current I/O configuration. It is displayed for information only. The minimum output timebase depends on the least common multiple of all module timebases, with a minimum of 50 ms.

Use name as prefix

Puts the module name in front of the signal names.

Module structure

Number of analog/ digital signals

Defines the number of configurable analog and digital signals in the signal tables. The default value is 32 for each. You can change the number. The maximum value is 1000.

7.3.2 OPC output module

The OPC output module can be used to write values in OPC signals provided by an OPC server. The OPC output module in *ibaPDA* is an OPC slave.

7.3.2.1 General tab

It is not possible to set the time base, since this results from the smallest common multiple of all module time bases and is at least 50 ms. Higher time bases will be calculated automatically.

In the *General* tab, you should

- select the OPC server computer
- select the OPC server (service)
- enter user account data (if required)
- enter a group name (optional)

For notes on the general module settings see ➤ *General module settings, page 141*.

In addition, you can still configure the following settings:

Force data type

If this option is enabled (True), then the analog signals are requested with data type R4 (4 byte floating point value) and the digital signals with data type BOOL. Use this option if the OPC server can dynamically change the data type of its signals.

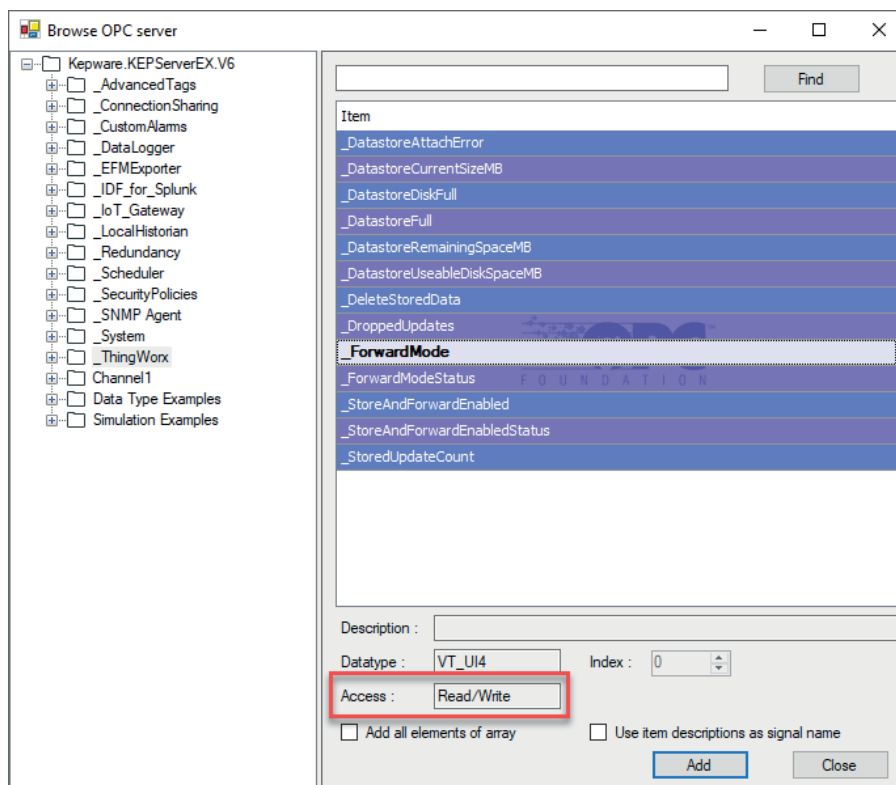
Do initial read

When this option is enabled (True) *ibaPDA* will wait until the first read of data from the OPC server is finished before the acquisition starts.

Add item attempts

Certain OPC servers do not accept items until they have loaded their complete configuration. Via this setting you can determine how often *ibaPDA* should try to add the requested OPC items to the group. There is a waiting time of 1 s between two attempts.

After you have made the settings, click on the blue hyperlink *Connect* at the bottom of the tab. As soon as the connection is established, you can click on the available hyperlink *Add signals* in order to browse the OPC signals. Only items with read/write or write access apply for selection. Selecting OPC items at this stage will put them automatically into the correct signal table (analog/digital).



7.3.2.2 Analog and Digital tab

In the signal tables on the *Analog* and *Digital* tabs, you can define the output signal in the *Expression* column.

The result of the expression will be written to the item of the OPC server selected in the "*Item ID*" column next to it.

For selection of OPC items, you can open the browser also from the *Item ID* column.

7.3.3 Email

Using the email output interface, emails can be sent each time a trigger signal is triggered and whenever acquisition is started and stopped.

Therefore, users can be notified automatically, e.g., in the case of an alarm. You can also use current signal values and text in the email as well as send attachments (still images from an iba-Capture system or other files).

The email interface is part of the standard scope of *ibaPDA* and does not require an additional license. *ibaPDA* has an integrated email client that can connect to any SMTP server on the network or the Internet. The system supports TLS communication to SMTP servers as well as OAuth 2.0 authentication.

The interface supports up to 256 different emails that use different accounts and can be addressed to various addressees and/or have different content. Various emails can be sent with the same trigger signal or individual trigger signals. A module must be configured for each email.

The accounts, as well as some placeholder fields, must be set up on the “Email” interface node beforehand to transmit selected signal values.

7.3.3.1 Basic settings (accounts and fields)

Accounts tab (interface)

In this tab, you configure the email accounts to be used by *ibaPDA* to send emails.

Add an account using the button **+** and enter the required information. If available, multiple accounts can be configured and used. Which account will be used for which email is determined when you configure the email module. You can duplicate an account in the list by using the copy button. This is useful if, e.g., you want to set up different senders.

If you want to configure a standard SMTP server, select "Other" in the *Provider* field. Alternatively, you can select other servers, like for instance *Microsoft Exchange Online*. The input mask changes accordingly to enable configuring the selected provider. For such providers *ibaPDA* supports the authentication protocol OAuth 2.0.

Optionally, you can declare an account as the default account so that this account is automatically entered when a module is added.

You can also enable logging of SMTP communication for each individual account.

Fields tab (interface)

Accounts Fields		
Name	Channel	Format
Hot coil number	 3:17: 113 Hot coil number	1.00
TS_Customer	 11:2: TS_Customer 	
	 Unassigned	

In this tab, you can define fields for signal values (numeric and text) that can be used as placeholders in emails. Thus, with the fields, current analog and digital signals or calculated values can be inserted into the message. Accordingly, you can insert values of text signals into the message.

First enter the name of the desired placeholder in the *Name* column. Then you can open a drop-down list in the *Channel* column, from which you select the desired signal or text channel.

In the numeric fields, you can even specify the number format in the *Format* column. Click on the <...> button to open the settings dialog.

The names of the fields must **not** contain the following characters:

- Square brackets [] (indicate a placeholder)
- **IT** at the beginning of the name (reserved for image triggers, see below)
- **DS** at the beginning of the name (reserved for data records, see below)

Names may not be given twice.

7.3.3.2 Email module settings

General tab

For email modules, only the standard information must be entered in the general module settings.

For notes on the general module settings see  *General module settings, page 141*.

Message tab

In this tab, you will create the actual message.

In the window on the left, the fields for the signals defined in the *Fields* tab are displayed in the form of a signal tree.

Below this, all projected data stores and image triggers are displayed. These data records and image trigger fields always refer to the path of the last data file or the last images.

These data stores and image trigger fields always refer to the path of the last data file or the last images.

Select the desired account and enter the standard information such as *recipient*, *CC* and *subject* as well as the message text. You can also use fields with text signals in the lines *To*, *Cc* and *Bc*, and can thus enter the addressees dynamically. For example, if an email is to be sent to a different recipient on weekdays than on weekends.

For the *email trigger*, select whether the sending of email should be triggered at the start or stop of acquisition or by a digital signal.

A dead time (default 10 seconds) prevents the email from being sent multiple times should the trigger signal be triggered several times in quick succession.

With the *Send retry period* and *Total retry time* settings, you can determine the intervals at which, and over how long in total, a failed transmission of an email is repeated.

If you enable the option *Mark retries with [Delayed message...] in subject*, then this note is automatically inserted in the subject line if the email could only be sent after several failed attempts.

With the <Send test email> button, you can send the email without the trigger signal being triggered or the measurement running. If the measurement is not running and the configured fields have no values then the names of the fields are entered in the email.

Using the fields

In addition to static input, you can add fields as placeholders in various places, thus dynamically shaping your email.

The following table shows where you can add which fields and what impact this has.

Field	Use in...	Result when sending the email
Analog and digital signals	Subject	Value is applied
	Message text	Value is applied
	Attachment	Value is applied and interpreted as part of a path or file name. If the file exists, it is attached.
Text signals	To/Cc/Bcc:	Text is applied
	Subject	Text is applied
	Message text	Text is applied
	Attachment	Text is applied and interpreted as part or all of a path or file name. If the file exists, it is attached.
Data storages	Subject	Name of the data file (full path) is applied.
	Message text	Name of the data file (full path) is applied.
Image trigger	Subject	Name of the image file (full path) is applied.
	Message text	Name of the image file (full path) is applied.
	Attachment	Image file is attached
Clients	Message text	The information of the selected fields referring to the last start and/or last stop of the acquisition is applied.

Table 9: Use of fields / placeholders in email

A field can be added at the desired location in the entry fields, *Subject*, *Text* and *Attachment*, in the following ways:

- By double-clicking on the field; the field will be inserted in the text at the cursor position
- By dragging and dropping
- By manual entry of the field name in square brackets

The fields are highlighted in the text with red, bold characters to indicate that these points are later replaced by actual values.

Data storages cannot be added as an attachment as data files are often too large for emailing and have not yet been concluded by *ibaPDA* when the email is sent.

If you want to send multiple attachments, they must be separated by semicolons.

Entries in the *Attachment* field must always be the full path and file name of the files that you want to attach.

The path and file name can be formed in different ways:

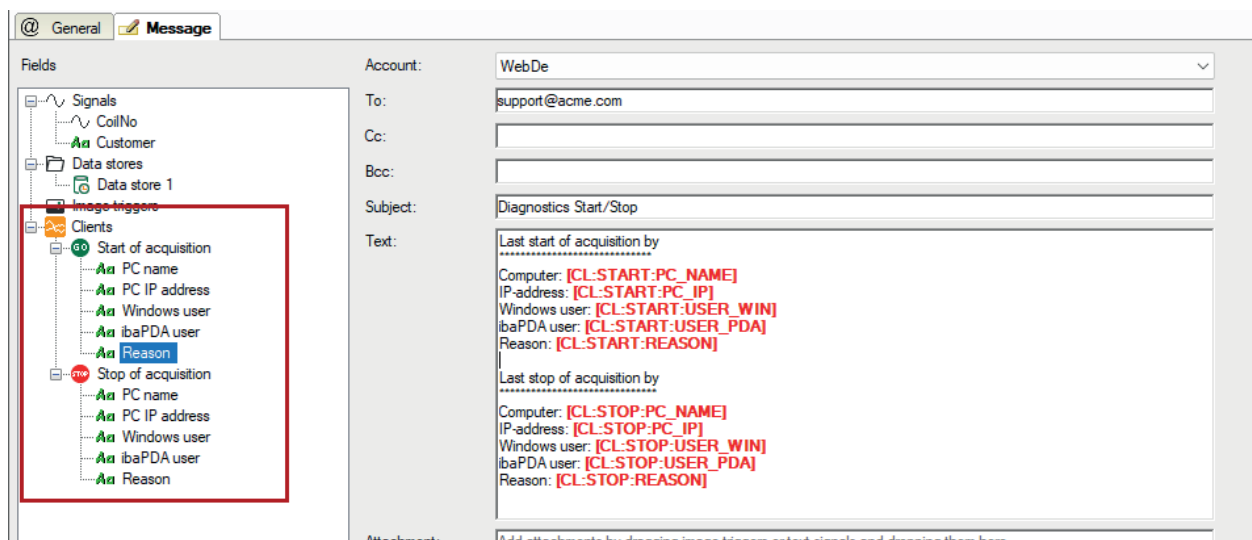
Type	Example	Comment
Entered fully without placeholders	D:\xyz\agb.pdf	Only useful if the path and file-name never change.
Only a text field	[Report] [Imagetrigger_ABC]	Full path and file name must be included in the text box. (Automatically the case for image triggers.)
Combination of static text and text field	\\DATASRV\reports\[Shiftreport]	Text field only contains the file name.
Combination of text and numeric fields	[Destination] [Prefix] [Number] [Extension]	[Number] is a numeric field with a consecutive number, [Destination], [Prefix] and [Extension] are text fields.
Combination of static text and text and numeric fields	D:\abc\[PathA]\[Reportname]_ [No].pdf	-

Table 10: Examples of entries in the attachment line

Example for a message about the last start and last stop of acquisition

The message can be sent at the time when the acquisition was started or stopped, in order to send an immediate notification, or at any time.

Dedicated fields related to the client where the last start or stop of acquisition was initiated provide more detailed information.



- PC name: Name of the client computer
- IP address: IP address of the client computer
- Windows user: Windows user name that was used to log into Windows on the client
- ibaPDA user: *ibaPDA* user name that was used to log into *ibaPDA*
- Reason: Reason, why the acquisition started or stopped

Possible reasons are:

- Interactive start (start by user operation)
- Interactive stop (stop by user operation)
- Interactive apply new configuration (I/O configuration changed and applied by user)
- Automatic start (e. g. when a disconnected PLC returns)
- Start after remote configuration
- Start triggered by other synchronized system (e. g. in a Multistation network)
- ibaPDA service is shutting down
- PC is going to sleep (computer switches to power saving or standby mode)
- Error occurred (unknown error)
- License changed (license container disconnected or license changed)

7.3.4 ibaNet750-BM, -BM-D output modules

For each link of an ibaFOB output card, you can add one ibaNet750 module with up to 8, 16 or 32 analog and digital output signals, depending on the address method selected.

When using a ibaFOB...-D card or ibaFOB-io-ExpressCard, the device ibaNet750-BM-D can also be used in 32 Mbit flex mode.

7.3.4.1 General tab

Basics

For notes on the general module settings see [↗ General module settings, page 141](#).

Note



You may add an ibaNet750 module either on the input side (Inputs tab) or on the output side (Outputs tab) and add input terminals as well as output terminals. The module number of the ibaNet750 module will always be the same on both sides.

Depending on the connection mode of the module (see description below), the module is available on just one or both sides.

ibaNet750

Address mode

The address mode setting must comply with the switch setting (S1) on the device.

Mode	Switch positions on device	Addresses	Number of analog and digital signals
1	1– 8	1– 8	8/8
2	C – F	1 - 2, 3 - 4, 5 - 6, 7 - 8	16/16
4	A, B	1 - 4, 5 - 8	32/32

Table 11: ibaNet750 address modes

ECO mode

The ECO mode setting (TRUE or FALSE) must comply with the switch (S2) position on the device.

Connection mode

With the connection mode setting, you can control the mapping of the input and output fiber optical links of an *ibaFOB* card to an *ibaNet750-BM* device.

Mode	Display	Comment
I	Inputs	Module is only visible in the <i>Inputs</i> tab of the I/O manager, even with output terminals, if configured. Module is not visible in the <i>Outputs</i> tab. FO output link is not mapped to <i>ibaNet750-BM</i> device.
O	Outputs	Module is only visible in the <i>Outputs</i> tab of the I/O manager, even with input terminals, if configured. Module is not visible in the <i>Inputs</i> tab. FO input link is not mapped to <i>ibaNet750-BM</i> device.
IO	Inputs and Outputs	The module will be visible in the <i>Inputs</i> and <i>Outputs</i> tabs. FO input and output links are mapped to the <i>ibaNet750-BM</i> device.

Table 12: ibaNet750 connection modes

7.3.4.2 Analog and Digital tab

The signal tables for analog and digital signals are available as soon as a corresponding output terminal has been added to the ibaNet750 module.

The signals are divided into groups according to the terminal name.

In the signal tables on the *Analog* and *Digital* tabs, you can define the output signal in the *Expression* column.

7.3.5 ibaL2B... output modules

3 output module types are available for ibaL2B:

Output modules	Data type	GSD file
 L2B integer I/O (5)	32 inputs/outputs - integer	iba_0F08.gsd
 L2B real I/O (6)	32 inputs/outputs - real	iba_0F09.gsd
 L2B S7 real I/O (7)	28 inputs/outputs - real (S7-400)	ibaF0b_4.gsd
	28 inputs/outputs - real (S7-300)	ibaF0b_3.gsd

Table 13: L2B output modules

With ibaL2B i/o modules, there are 32 analog (28 for S7 Real only) and 32 digital signals available per module, corresponding to the capacity of a PROFIBUS slave.

7.3.5.1 General tab

In the *General* tab, you should

- enter a calculation time base (optional)
- set the slave number according to your PROFIBUS project
- set the timeout (optional)
- select byte order

For notes on the general module settings see [General module settings, page 141](#).

7.3.5.2 Analog and Digital tab

In the signal tables on the *Analog* and *Digital* tabs, you can define the output signal in the *Expression* column.

7.3.6 Reflective Memory output module

7.3.6.1 General tab

For notes on the general module settings see [↗ General module settings, page 141](#).

No further general settings are required. The main setup of the Reflective Memory interface must have been made before in the *Inputs* tab of the I/O manager.

7.3.6.2 Analog and Digital tab

In the signal tables on the *Analog* and *Digital* tabs, you can define the output signal in the *Expression* column.

Remember to set the address of each analog signal or address and bit number for digital signals.

7.3.7 EtherNet/IP I/O output modules

Modules that have already been defined at the input level can also be seen in the tree of the output interfaces and can be used for output signals.

The general module settings are identical to those on the input side and are retained. Only the number of analog and digital signals can be set differently.

For notes on the general module settings see [↗ General module settings, page 141](#).

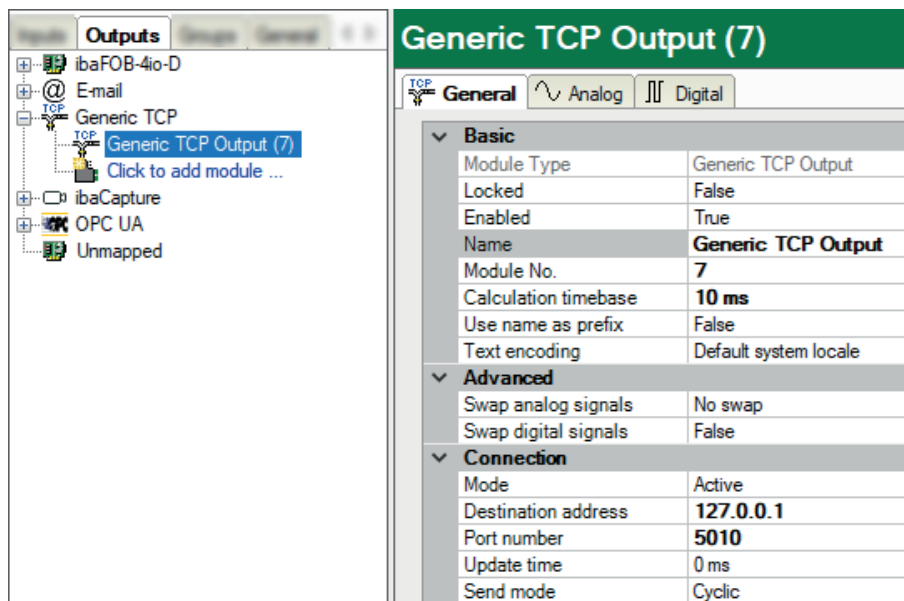
Configure the desired output signals in the *Analog* and *Digital* tabs. Enter a name, expression and address and select a data type for analog signals. The following data types are available for analog signals:

Data type		Description	Value range:
Big Endian	Little Endian		
BYTE	BYTE	8 bit without positive or negative sign	0 ... 255
INT_B	INT	16 bit with positive or negative sign	-32768 ... 32767
WORD_B	WORD	16 bit without positive or negative sign	0 ... 65535
DINT_B	DINT	32 bit with positive or negative sign	-2147483647 ... 2147483647
DWORD_B	DWORD	32 bit without positive or negative sign	0 ... 4294967295
FLOAT_B	FLOAT	IEEE754; single precision; 32 bit floating point value	±3,402823 E+38 ... ±1,175495 E-38

Table 14: Data types (Big and Little Endian)

7.3.8 Generic TCP output module

By using the *Generic TCP Output* module, you can transmit analog and digital signals via TCP/IP from *ibaPDA* to other systems.



7.3.8.1 General tab

The basic settings and the settings under *Advanced* can be adjusted as usual.

For notes on the general module settings see ↗ *General module settings, page 141*.

Text encoding

ibaPDA supports different text encodings for the output of texts or strings.

- Default system locale As specified in the Windows settings
- Western European (1252): Code page for Western Europe
- UTF-8: Unicode

Select the desired encoding here.

Connection settings

The following settings must be made for the connection:

Mode

Connection setup mode

- Active: *ibaPDA* establishes the connection to the port at the specified address.
- Passive: *ibaPDA* listens on the specified port and waits for a request to establish the connection. Only one connection is allowed in this mode.

Destination address

Here specify the destination address of the system with which the connection should be established.

Port number

Here specify the port via which the target system should communicate.

Update time

Update time determines in which cycle *ibaPDA* sends telegrams. If the update time is set to 0 ms, then *ibaPDA* sends a telegram as soon as output data is available. As for all output modules, the shortest cycle is 50 ms (see calculation time base).

Send mode

The selected send mode determines when *ibaPDA* should send the data.

- **Cyclic:** A telegram is sent at the set update time, at least every 50 ms.
- **On change:** A telegram is sent each time the signal data is changed.
- **On trigger:** A telegram is sent each time a rising edge is detected on the trigger signal.

7.3.8.2 Analog and Digital tab

You configure the analog output signals in the Analog tab.

In the Expression column, enter a signal or, if necessary, using the expression editor, a mathematical expression, the result of which is to be displayed.

The address you enter in the Address column is the address within the telegram where the data is written.

A data type must be set for each analog signal. The following data types are available:

Data type	Description	Value range:
SINT	8 bit with positive or negative sign	-128 ... 127
BYTE	8 bit without positive or negative sign	0 ... 255
INT	16 bit with positive or negative sign	-32768 ... 32767
WORD	16 bit without positive or negative sign	0 ... 65535
DINT	32 bit with positive or negative sign	-2147483647 ... 2147483647
DWORD	32 bit without positive or negative sign	0 ... 4294967295
FLOAT	IEEE754; single precision; 32 bit floating point	±1.175495E-38 ... ±3,403E+38
DOUBLE	IEEE754; double precision; 64 bit floating point;	±2,225E-308 ... ± 1, 798E+308
STRING[32]	String with up to 32 characters	

Table 15: Data types

Digital signals are processed in packets of 16 bits (integer). Consecutive addresses therefore have a distance of 2 bytes and the individual digital signals are identified by the bit number (0 to 15).

For a signal to actually be processed, a check mark must be placed in the *Active* column. If a signal is not active, then a null is entered at the appropriate place in the telegram.

7.3.9 Generic UDP output module

By using the *Generic UDP Output* module, you can transmit analog and digital signals via UDP from *ibaPDA* to other systems.

7.3.9.1 General tab

The basic settings and the settings under *Advanced* can be adjusted as usual.

For notes on the general module settings see ➔ *General module settings, page 141*.

Text encoding

ibaPDA supports different text encodings for the output of texts or strings.

- Default system locale As specified in the Windows settings
- Western European (1252): Code page for Western Europe
- UTF-8: Unicode

Select the desired encoding here.

Connection settings

The following settings must be made for the connection:

Destination address

Here specify the destination address of the system with which the connection should be established.

Port number

Here specify the port via which the target system should communicate.

Update time

Update time determines in which cycle *ibaPDA* sends telegrams. If the update time is set to 0 ms, then *ibaPDA* sends a telegram as soon as output data is available. As for all output modules, the shortest cycle is 50 ms (see calculation time base).

Send mode

The selected send mode determines when *ibaPDA* should send the data.

- Cyclic: A telegram is sent at the set update time, at least every 50 ms.
- On change: A telegram is sent each time the signal data is changed.
- On trigger: A telegram is sent each time a rising edge is detected on the trigger signal.

7.3.9.2 Analog and Digital tab

You configure the analog output signals in the Analog tab.

In the Expression column, enter a signal or, if necessary, using the expression editor, a mathematical expression, the result of which is to be displayed.

The address you enter in the Address column is the address within the telegram where the data is written.

A data type must be set for each analog signal. The following data types are available:

Data type	Description	Value range:
SINT	8 bit with positive or negative sign	-128 ... 127
BYTE	8 bit without positive or negative sign	0 ... 255
INT	16 bit with positive or negative sign	-32768 ... 32767
WORD	16 bit without positive or negative sign	0 ... 65535
DINT	32 bit with positive or negative sign	-2147483647 ... 2147483647
DWORD	32 bit without positive or negative sign	0 ... 4294967295
FLOAT	IEEE754; single precision; 32 bit floating point	$\pm 1.175495E-38$... $\pm 3,403E+38$
DOUBLE	IEEE754; double precision; 64 bit floating point;	$\pm 2,225E-308$... $\pm 1,798E+308$
STRING[32]	String with up to 32 characters	

Table 16: Data types

Digital signals are processed in packets of 16 bits (integer). Consecutive addresses therefore have a distance of 2 bytes and the individual digital signals are identified by the bit number (0 to 15).

For a signal to actually be processed, a check mark must be placed in the *Active* column. If a signal is not active, then a null is entered at the appropriate place in the telegram.

7.3.10 Modbus-TCP Client output module

The output module is not an autonomous module, but rather an extension of the *Modbus Client* module. With the output module, you can write data from *ibaPDA* to a controller.

You can configure the module in the *Outputs* tab. You do not have to add it separately. The module will be available as soon as a *Modbus Client* module has been added.

The settings correspond to those in the *Inputs* tab, apart from the module-specific settings, and can also be configured there. The connection settings also correspond to those in the *Inputs* tab.

For notes on the general module settings see [🔗 General module settings, page 141](#)

Please note the following differences to the settings of the input modules:

- For Analog type only "Holding register", single or multiple is possible
- For Digital type only "Holding register" and "Coils", each single or multiple are possible
- The transmission cycle of the messages is determined by the following parameters:
 - Update time (in ms): If you enter 0 or a value <50, the transmission cycle equals the *ibaPDA* task cycle of approximately 50 ms
 - Send only when changed:
 - False: the output message will be sent in the above mentioned cycle (update time).
 - True: the output message will only be sent when one of the signals changes its value, but at the latest after 1 min.

7.3.11 ABB-Xplorer output module

Please consider to check the *Enable ABB-Xplorer outputs* option in the I/O Manager on the root node *ABB-Xplorer* in order to use outputs.

Modules that have already been defined at the input level can also be seen in the tree of the output interfaces and can be used for output signals.

Select the *Outputs* tab in the module tree to configure the module. You do not have to add it separately. The module will be available as soon as an *ABB-Xplorer-MMS* module has been added.

Except for the calculation timebase and the minimum output timebase the general module settings are identical to those on the input side and are retained.

The number of analog and digital signals (module layout) can be set differently.

For notes on the general module settings see ➤ *General module settings, page 141*.

MMS

Unlike to the input side there is an additional setting *Write mode*.

Send mode

Determines when new data is written to the controller:

- Cyclic: Data is written cyclically at the set update time.
- On change: Data is written each time the signal data is changed.
- On trigger: Data is written with every rising edge of the trigger signal.

All signals of a module are always written, regardless of the write mode.

Trigger signal

This field only appears when the "on trigger" send mode is selected. Select here a digital signal. A rising edge on this digital signal writes the signal values taken at the time of the rising edge.

The connection settings are identical for inputs and outputs.

Configure the desired output signals in the *Analog* and *Digital* tabs. Enter name and expression and select the desired symbol in the PLC using the symbol browser.

If you make your selection of analog signals in the symbol browser, the corresponding data type will be applied as well. Without symbol browser you can enter the symbol name and select the data type of analog signals. The following data types are available for analog signals:

SINT, BYTE, INT, WORD, DINT, DWORD, FLOAT, DOUBLE

7.3.12 OMRON-Xplorer output module

The output module is not an autonomous module, but rather an extension of the *OMRON FINS* module. With the output module, you can write data from *ibaPDA* to a controller.

Note



Depending on the configured protection-level access of the OMRON CPU, writing values to the CPU may be not possible.

You can configure the module via the *Outputs* tab. You do not have to add it separately. The module will be available as soon as an *OMRON FINS* module has been added.

The settings correspond to those in the *Inputs* tab, apart from the module-specific settings, and can also be configured there. The connection settings also correspond to those in the *Inputs* tab.

For information about the general module settings refer to ↗ *General module settings*, page 141

Module-specific settings

Update time

With the update time, you determine the speed with that *ibaPDA* tries to request the data from the OMRON PLC. The actual resulting update time may be higher, depending on the load of the device. Check the diagnostic overview for measured update rates, because an overload will cause loss of samples.

Send mode

Determines when new data is written to the controller:

- **Cyclic:** Data is written cyclically at the set update time.
- **On change:** Data is written each time the signal data is changed.
- **On trigger:** Data is written with every rising edge of the trigger signal.

All signals of a module are always written, regardless of the write mode.

Trigger signal

This field only appears when the "on trigger" send mode is selected. Select here a digital signal. A rising edge on this digital signal writes the signal values taken at the time of the rising edge.

Output signals

The OMRON-Xplorer output module does not support digital signals.

You can configure the signals to be output in each case via the expression editor. Open the expression editor via the <fx> button in each signal row.

For each signal, enter a name and select memory type, address and data type. Supported data types are SINT, BYTE, INT, WORD, DINT, DWORD, LINT, QWORD, FLOAT, DOUBLE.

Note



ibaPDA reads and writes all signals for an Xplorer module via a common connection. Therefore, the total number of configured signals influences the update time.

7.3.13 S7-Xplorer output module

Please consider to check the *Enable S7-Xplorer outputs* option in the I/O Manager on the root node *S7-Xplorer* in order to use outputs.

Only the module type *S7-Xplorer* can be used for outputs.

Modules that have already been defined at the input level can also be seen in the tree of the output interfaces and can be used for output signals.

Except for the calculation timebase and the minimum output timebase the general module settings are identical to those on the input side and are retained.

The number of analog and digital signals (module layout) can be set differently.

For notes on the general module settings see ➤ *General module settings, page 141*.

S7

Unlike to the input side there is an additional setting *Write mode*.

Send mode

Determines when new data is written to the controller:

- Cyclic: Data is written cyclically at the set update time.
- On change: Data is written each time the signal data is changed.
- On trigger: Data is written with every rising edge of the trigger signal.

All signals of a module are always written, regardless of the write mode.

Trigger signal

This field only appears when the "on trigger" send mode is selected. Select here a digital signal. A rising edge on this digital signal writes the signal values taken at the time of the rising edge.

The connection settings are identical for inputs and outputs.

Configure the desired output signals in the *Analog* and *Digital* tabs. Enter name and expression and select the desired symbol in the PLC using the symbol browser or enter the S7-operand.

If you make your selection of analog signals in the symbol browser, the corresponding data type will be applied as well. Without symbol browser you can enter the symbol name and select the data type of analog signals. The following data types are available for analog signals:

BOOL, CHAR, BYTE, WCHAR, WORD, DWORD, SINT, INT, DINT, LINT, USINT, UINT, UDINT, REAL, LREAL, S5 REAL, TIME, LTIME, TIMER, COUNTER, STRING[], WSTRING[], CHAR[], WCHAR[]

7.3.14 TwinCAT-Xplorer output module

Please consider to check the *Enable TwinCAT-Xplorer outputs* option in the I/O Manager on the root node *TwinCAT-Xplorer* in order to use outputs.

Only the module type *TwinCAT PLC* can be used for outputs.

Modules that have already been defined at the input level can also be seen in the tree of the output interfaces and can be used for output signals.

Except for the calculation timebase and the minimum output timebase the general module settings are identical to those on the input side and are retained.

The number of analog and digital signals (module layout) can be set differently.

For notes on the general module settings see ➔ *General module settings, page 141*.

PLC

Unlike to the input side there is an additional setting *Write mode*.

Send mode

You can choose between three different send modes:

- **Cyclic:** Cyclic writing according to the calculation timebase
- **On change:** Data is written whenever the value of an analog signal changes
- **On trigger:** Data is being written each time a rising edge is detected on "send trigger" signal.

The connection settings are identical for inputs and outputs.

Configure the desired output signals in the *Analog* and *Digital* tabs. Enter name and expression and select the desired symbol in the PLC using the symbol browser.

If you make your selection of analog signals in the symbol browser, the corresponding data type will be applied as well. Without symbol browser you can enter the symbol name and select the data type of analog signals. The following data types are available for analog signals:

FLOAT

7.3.15 ibaCapture output module

By using the *ibaCapture* output module, signals can be sent from *ibaPDA* to an *ibaCapture* server in order to control video recording and/or cameras.

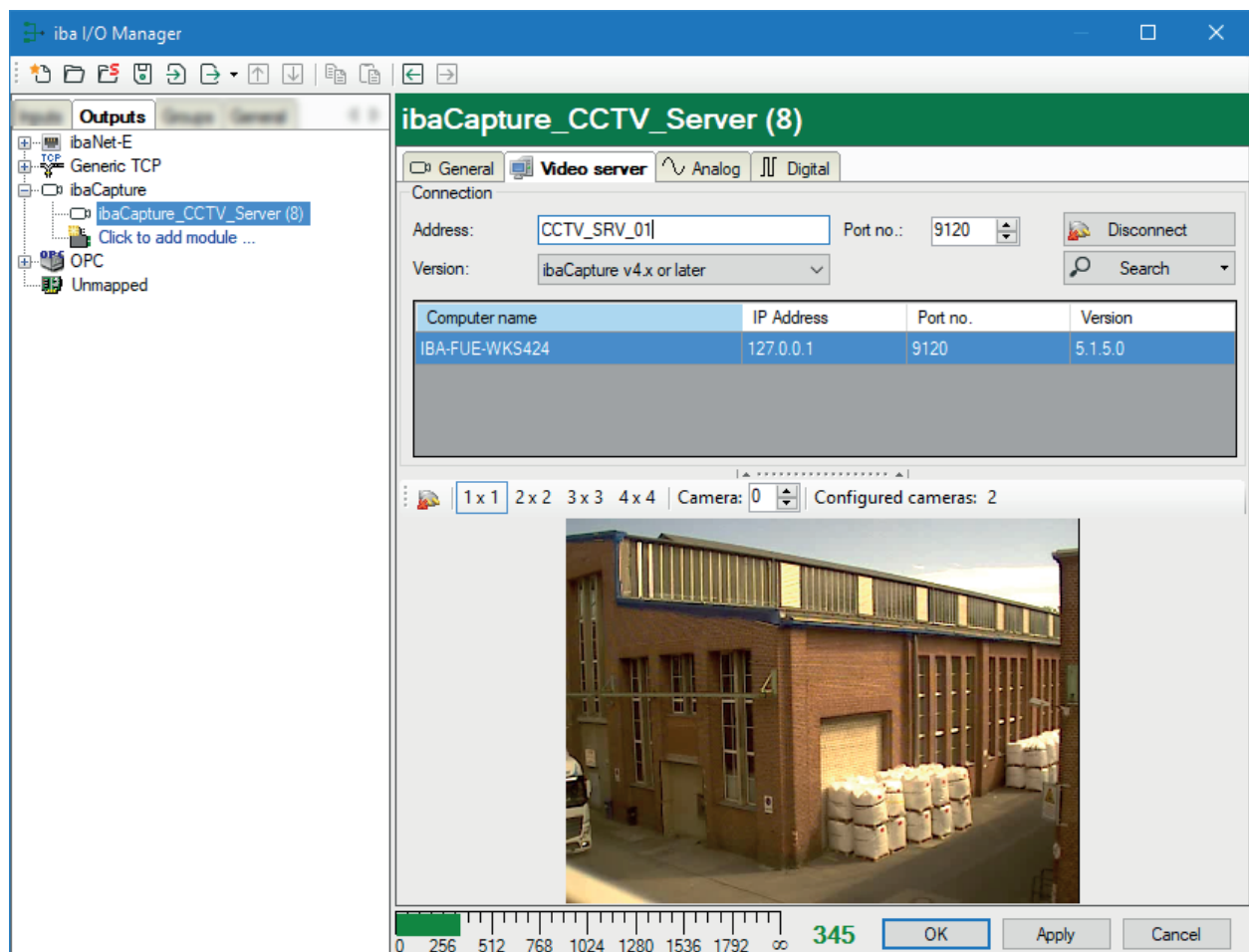
7.3.15.1 General tab

You can configure the usual basic settings in the *General* tab.

For notes on the general module settings see ➤ *General module settings, page 141*

7.3.15.2 Video server tab

Each *ibaCapture* output module must first be connected to the appropriate *ibaCapture* server. You can make the settings in the *Video server* tab of the module.



7.3.15.3 Analog and Digital tab

Controlling PTZ cameras with analog signals

ibaCapture-CAM_CC-Server (13)					
General Video server Analog Digital					
	Name	Expression	Act...	Actual	+
▶ Camera 1: Axis_Network-Cam					
5	Pan value	f_{∞}	?	☐	
6	Tilt value	f_{∞}	?	☐	
7	Zoom value	f_{∞}	?	☐	
8	Preset position index	f_{∞}	?	☐	
+ Camera 2: Sony Onvf					
+ Camera 3: Axis RISP					
+ Camera 4:					

If you use PTZ (pan tilt zoom) cameras and want to control their settings in terms of pan, tilt and zoom with *ibaPDA*, you can do this by using analog output signals. You can define the values for the pan and tilt angle as well as for the zoom factor by using the expression editor. Presets stored in the camera can also be accessed directly. A data set is available for each of an *ibaCapture* server's 16 cameras.

Controlling video recording and PTZ cameras with digital signals

ibaCapture-CAM_CC-Server (13)					
General Video server Analog Digital					
	Name	Expression	Act...	Act...	+
▶ Camera 1: Axis_Network-Cam					
0	Recording control	f_{∞}	?	☐	
5	Pan active	f_{∞}	?	☐	
6	Tilt active	f_{∞}	?	☐	
7	Zoom active	f_{∞}	?	☐	
8	Goto preset position	f_{∞}	?	☐	
+ Camera 2: Sony Onvf					
+ Camera 3: Axis RISP					
+ Camera 4:					

A data set with digital output signals is available for each camera of an *ibaCapture* server. You can use the first signal "Camera recording control ..." to start and stop a camera video recording. You can use the expression builder to configure the signal depending on the process. To control PTZ cameras, you will find the digital output signals to enable or disable the control functions (pan, tilt, zoom and preset) in this tab.

Other documentation



Detailed information on the use and configuration of output signals can be found in the *ibaCapture* product manual.

7.3.16 ibaVision output module

With this type of module, you can transfer signal values from *ibaPDA* to an *ibaVision* system.

For notes on the general module settings see ➤ *General module settings, page 141*

Other documentation



For a description of the *ibaVision* functions and configuration options for this module, see the *ibaVision* product manual.

7.3.17 FOB alarm output module

One FOB alarm module each with 64 analog/digital output signals is possible per link of an ibaFOB output card. The FOB alarm module only supports the 3 Mbit mode. If you add an FOB alarm module to a link of an ibaFOB-D card, then the link automatically switches to the 3 Mbit mode. Per FOB alarm module, you can configure up to 64 analog and 64 digital signals and send to iba-capable devices with 3 Mbit mode, such as ibaLink-SM-64-io, ibaLink-SM128-i-2o, ibaPA-DU-8-O or ibaNet750-BM.

7.3.17.1 General tab

In the *General* tab, you can change the connection mode to “Real” if the receiving system specifies real values. The default setting is “integer.”

Setting the calculation time base is not possible because the FOB alarm outputs are always processed with the lowest time base that corresponds to the smallest common multiple of all module time bases (min. 50 ms). Higher time bases will be calculated automatically.

For notes on the general module settings see ➤ *General module settings, page 141*.

7.3.17.2 Analog and Digital tab

You can configure the desired output signals in the column *Expression* on the tabs *Analog* and *Digital* in the signal tables, just like for the virtual signals. To define the output signals, use the expression builder, which will again provide you with all options for the mathematical and logical calculation. You may enter simple numerical or Boolean equations or complex mathematical expressions. The expression builder is the supporting tool for the creation of mathematical expressions. If you know the syntax, you may also manually enter the expressions in the table. Use the button <fx> to open the expression builder.

For more information about the expression builder, please refer to part 4 *Expression builder (virtual signals)*.

Remember that the result of the expression must match the signal's data type.

Virtual analog signals are always REAL type values. If the connection mode is set to “Integer,” the analog output signal will be rounded to the nearest whole number (integer). In order to get a virtual digital signal or alarm, the result of the expression must be a Boolean quantity (TRUE or FALSE).

7.3.18 Outputs to the iba modular system (ibaPADU-S)

If *ibaPDA* is connected to the central unit of an iba modular system via an *ibaFOB-D* card then analog and digital signal values can also be output to the modular system.

The following central units are supported:

Name	Function
ibaPADU-S-CM	Setting discrete analog and digital output signals using appropriate output modules
ibaPADU-S-IT-16	Setting discrete analog and digital output signals using appropriate output modules; Data supply of the application on the device, e.g., ibaLogic
ibaPADU-S-IT-2x16	Setting discrete analog and digital output signals using appropriate output modules; Data supply of the application on the device, e.g., ibaLogic
HAICMON CMU	Data supply of the application on the device
ibaCMU-S	Setting discrete analog and digital output signals using appropriate output modules; Data supply of the application on the device
ibaPQU-S	Setting discrete analog and digital output signals using appropriate output modules; Data supply of the application on the device

Table 17: Central units of the modular system with the support of outputs

Other documentation



For information about the configuration of the outputs for modular system devices, see the relevant manual for the device.

7.3.19 Outputs to iba bus modules (ibaBM-...)

If *ibaPDA* is connected to a bus module in the 32 Mbit flex mode via an ibaFOB-D card, analog and digital signal values can also be written on the connected fieldbus.

The following bus modules are supported:

Name	Function
ibaBM-DP	Sending output signals to a DP master via an active slave configured in the device.
ibaBM-PN	Sending output signals to a PN controller via a device slot configured in the device.

Table 18: Bus modules with output support

Other documentation



For information about the configuration of the outputs for bus modules, see the relevant manual for the device.

7.3.20 Outputs to iba system interconnections (ibaLink-...)

If *ibaPDA* is connected to a card for system interconnection in 32 Mbit flex mode via an ibaFOB-D card with a module, analog and digital signal values can also be written in the memory of the target system.

The following system interconnections are supported:

Name	Function
ibaLink-VME	Sending output signals to a VME-based automation system (via the card's dual-port RAM)
ibaLink-io-embedded	Depending on implementation in an external system

Table 19: System interconnections with output support

Other documentation



For information about the configuration of the outputs for system interconnections, see the relevant manual for the device.

7.3.21 OPC UA client output module

The *OPC UA client output* module is also available with the license for the OPC UA client interface.

Other documentation



Note the extensive explanations about the prerequisites and the connection set-up in the manual about the product *ibaPDA-Interface-OPC-UA-Client*.

Modules that have already been defined at the input level can also be seen in the tree of the output interfaces and can be used for output signals.

If you want to use the output signals, suitable writable tags must have been configured on the corresponding OPC UA server.

As with all output modules, the time base for the output cycle is at least 50 ms.

General module settings

The general module settings are partially identical to those on the input side and are retained.

OPC UA client (5)	
General	
Basic	
Module Type	OPC UA client
Locked	False
Enabled	True
Name	OPC UA client
Module No.	5
Calculation timebase	10 ms
Use name as prefix	False
Advanced	
Send Mode	On change
Module Layout	
No. analog output signals	32
No. digital output signals	32
OPC UA Server	
Verify tags	True

No settings with respect to the OPC server can be made for the output module.

In the *Advanced* section, you can use the Send mode to set when the output data should be written:

- On change
The output data is written when the signal values change, but not faster than the general output cycle time of the system allows (at least 50 ms).
- On trigger
The output data is written on each rising edge of the selected trigger signal (write trigger), but not faster than the general output cycle time of the system allows (at least 50 ms).

The number of analog and digital output signals is preset to 32. If necessary, you can change the number (max. 1000).

Clicking on <Select symbols> opens the OPC UA symbol browser. The prerequisite here is that the right address book of the OPC UA server was created. See all available tags here. By double-clicking on a tag or selecting the tag and clicking on <Add>, the tag is entered in the next open line of the appropriate signal table (*Analog* or *Digital*) as *node ID*.

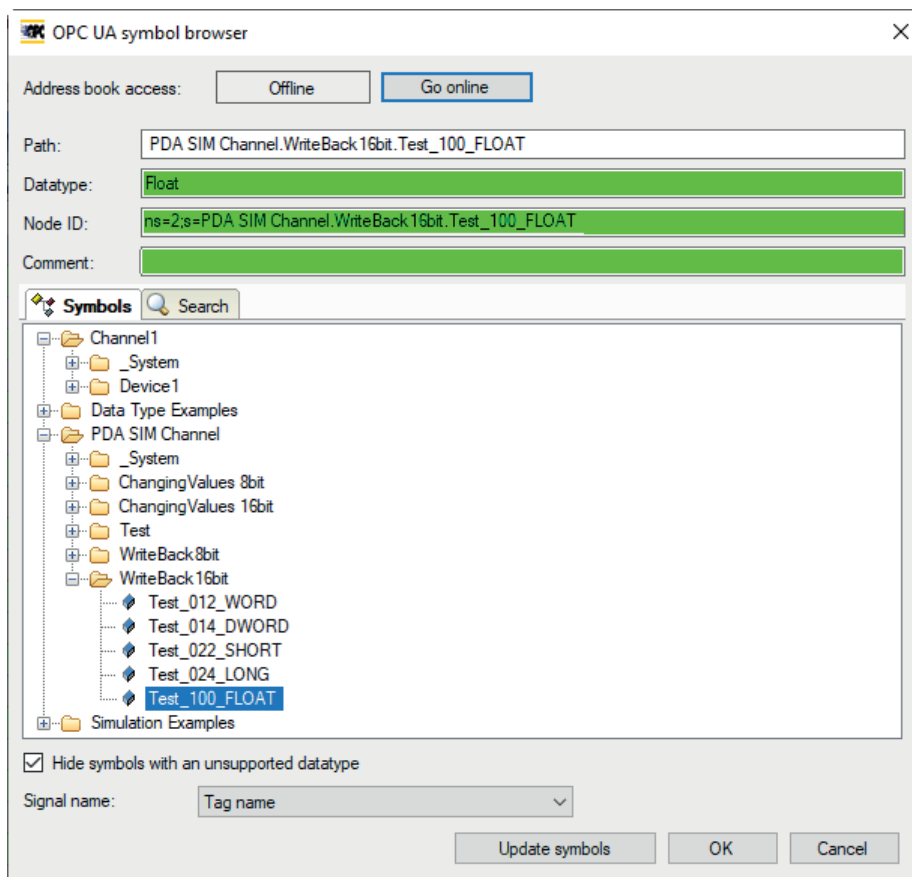
Connection

The connection settings are the same as those on the input side (OPC UA client module, I/O Manager area *Inputs*).

Analog and digital signals

In the *Analog* and *Digital* tabs, you can either select measurement signals from the signal tree in the *Expression* column and thus make them available as OPC UA output signals, or generate separate (virtual) signals using the expression builder.

To allocate the signals to the OPC UA tags of the OPC UA server, proceed in the *Node ID* column. Click on the button <...> in a table cell to open the OPC UA symbol browser, which you can use to select the desired tag.



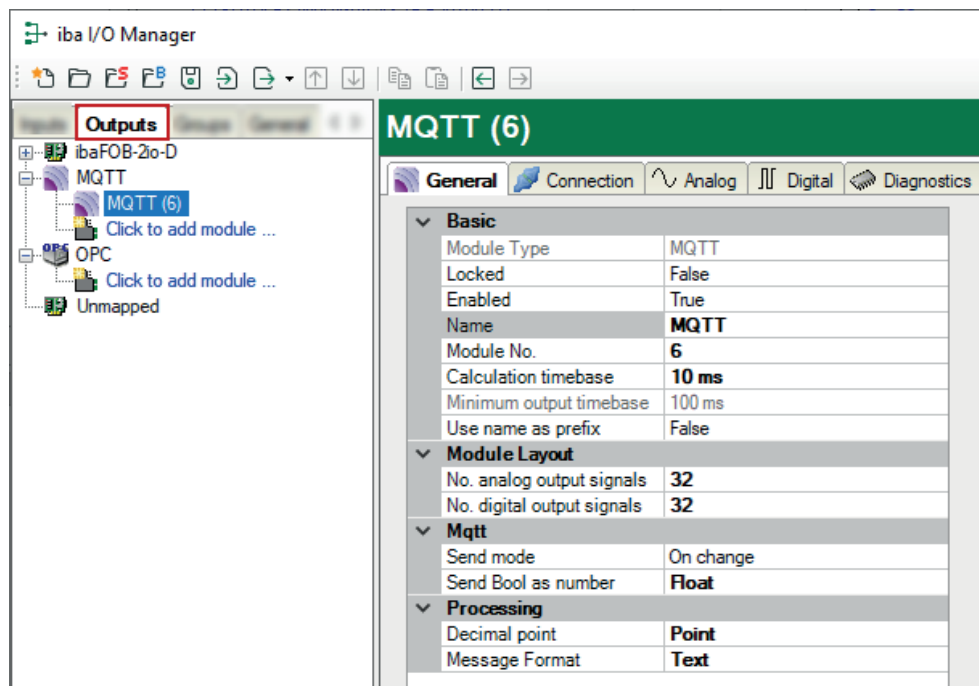
If you open the symbol browser from the signal tables instead of via the hyperlink in the *General* tab, then you will already be shown the tags with the corresponding data types.

7.3.22 MQTT output module

The output module is not an autonomous module, but rather an extension of the MQTT module. With the output module, you can write data from *ibaPDA* to an MQTT-Broker.

Select the *Outputs* tab in the module tree to configure the module. You do not have to add it separately. The module will be available as soon as an MQTT module has been added.

The following module settings can be configured on the *General* tab:



Basic settings

Module Type (information only)

Indicates the type of the current module.

Locked

A module can be locked in order to prevent change of module settings by accident or unauthorized users.

Enabled

Disabled modules are excluded from the signal acquisition.

Name

The plain text name should be entered here as the module designation.

Module No.

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

Calculation timebase

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

The calculation timebase and update timebase are NOT the same!

For interfaces whose modules have an input and an output side (same module number), the calculation timebase is identical to the module timebase on the input side. Hence, changing the calculation timebase also changes the module timebase on the input side and vice versa.

For interfaces with separate input and output modules (different module numbers), the calculation timebase can be changed independently.

The smallest possible calculation time base, i.e., the fastest possible update of the outputs, results from the smallest common multiple of all module time bases, and is at least 50 ms. .

This setting does not exist for the E-mail, OPC output and SQL command output modules.

Minimum output timebase (information only)

Smallest timebase for updating the outputs.

This value is calculated automatically by the system considering the current I/O configuration. It is displayed for information only. The minimum output timebase depends on the least common multiple of all module timebases, with a minimum of 50 ms.

Use name as prefix

Puts the module name in front of the signal names.

Module layout**No. of analog signals/ No. of digital signals**

Determines the number of analog/digital output signals in this module. The default value is 32 for each. Maximum value is 1000.

Mqtt**Send Bool as number**

Decide whether Boolean values in text mode should be written as FLOAT values (1 and 0) or as Boolean values (TRUE and FALSE).

Send mode

Determines when new data is written to the controller:

- Cyclic: Data is written cyclically at the set update time.
- On change: Data is written each time the signal data is changed.
- On trigger: Data is written with every rising edge of the trigger signal.

All signals of a module are always written, regardless of the write mode.

Trigger signal

This field only appears when the "on trigger" send mode is selected. Select here a digital signal. A rising edge on this digital signal writes the signal values taken at the time of the rising edge.

Processing

Decimal point

Configure, which character should be used as decimal point.

Message format

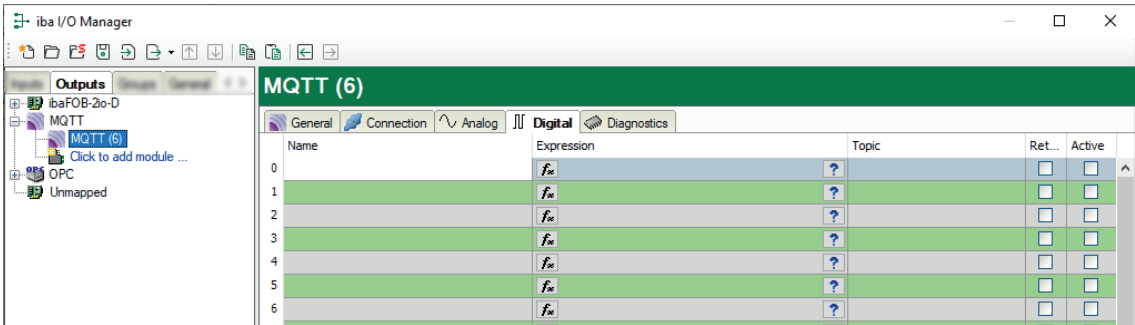
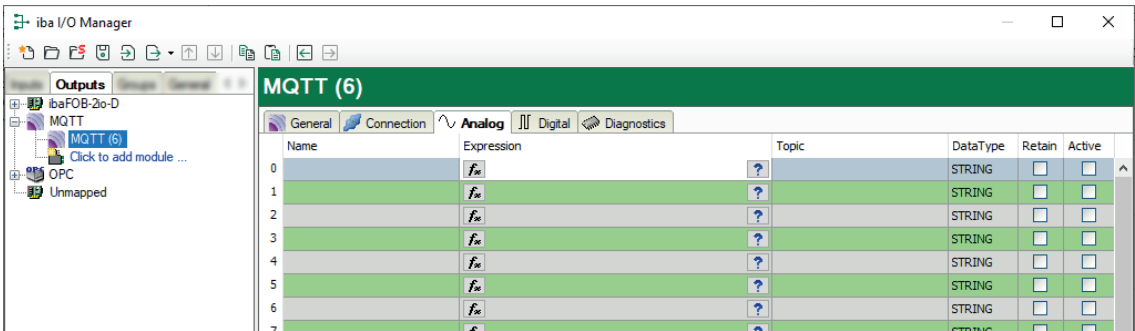
Select if values in MQTT broker are stored in text format or as binary values.

Output signals

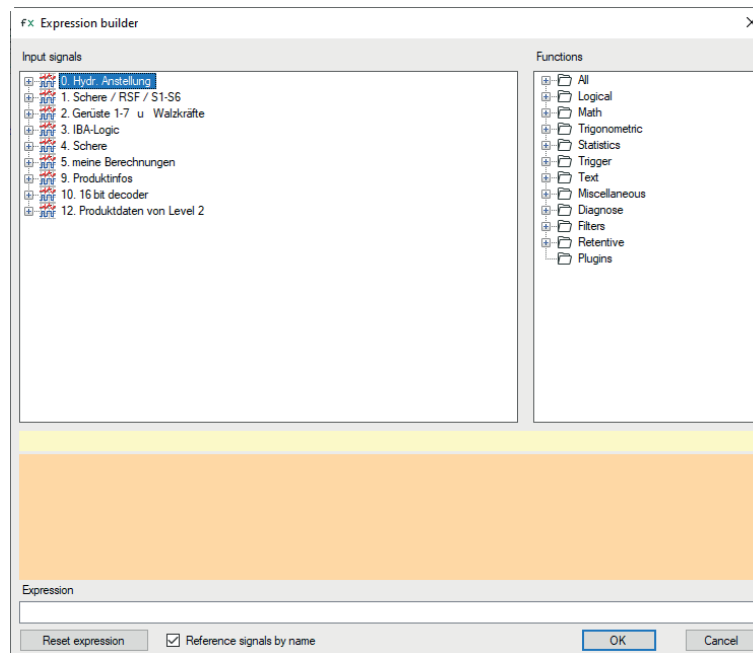
The analog and digital signals be outputs, are configured in the expression editor. The expression editor can be opened by the <fx> button from each signal row.

In the *Analog* and *Digital* tabs enter a name for each signal and a data path to the broker. The following data types are supported for analog signals: SINT, BYTE, INT, WORD, DINT, DWORD, FLOAT, DOUBLE, STRING.

If you enable the *Retain* option, the signals are stored on the broker until they are overwritten.



Expression builder:



For more detailed information about the expression builder please refer to part 4 of the *ibaPDA* manual.

7.3.23 Output module for SQL commands to SQL databases

The connections to one or more databases are configured and managed with the SQL interface. One or more connections can be configured per database. All DB connections can be used simultaneously. The databases can be of the same or different types.

Database systems from different manufacturers are supported (Maria DB, MySQL, Oracle, PostgreSQL, SAP HANA, SQL Server).

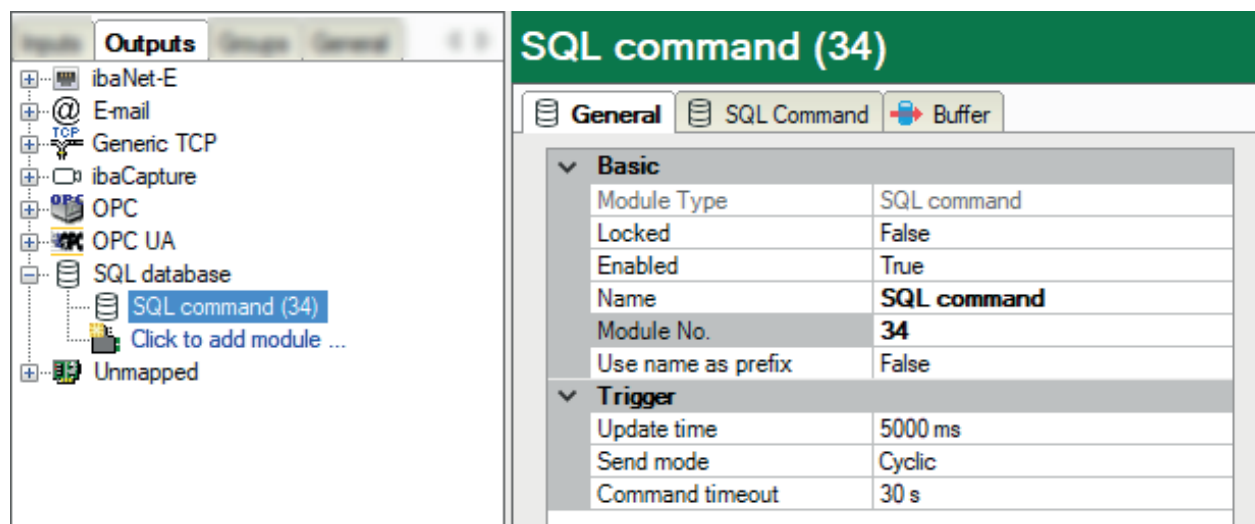
The data exchange with the databases occurs via corresponding SQL modules, which are configured according to the specific database type.

The SQL commands are configured in the *Outputs* tab. An *SQL command* module must be configured for each SQL command.

General module settings

For notes on the general module settings see ↗ *General module settings, page 141*.

The "Calculation timebase" parameter is not available here.



Trigger

- **Update time**
In the "Cyclic" send mode, the period in which the command is issued; in the "On change" or "On trigger" send mode, the dead time for suppression of closely successive changes/triggers, if not desired.
- **Send mode**
 - Cyclic: Cyclic sending of the command after the update time
 - On change: Command is sent when the value of the "Send trigger" signal changes
 - On trigger: Command is sent when "Send trigger" has a rising edge.
- **Send trigger**
Only available with send mode "On change" or "On trigger"
Select a suitable signal (analog/digital) with which you want to control the execution of the command.
- **Command timeout**
Time limit (in seconds) for the execution of a command. If a command cannot be completely executed within this time after triggering, it is aborted.

SQL command tab

You enter the SQL command in this tab.

Buffer tab

In this tab you can configure a buffer in case the database is not accessible. The memory buffer is always enabled, the file buffer can be additionally enabled and configured.

Commands that could not be executed because the connection to the database was interrupted are stored here and executed sequentially after the connection is restored.

Other documentation



A detailed description of the SQL interface and SQL modules can be found in the manual for the corresponding software product:

- ibaPDA-Interface-MySQL
 - ibaPDA-Interface-Oracle
 - ibaPDA-Interface-PostgreSQL
 - ibaPDA-Interface-SAP-HANA
 - ibaPDA-Interface-SQL-Server
-

7.3.24 ibaNet-E output modules

You can use the *ibaNet-E* interface to establish a connection to an *ibaNet-E*-compatible device.

If you use *ibaPDA* on a PC, the connection is established over the network adapter of the computer or over a special *ibaNet-E* network adapter (*ibaM-2E*). In this case you can connect to devices as follows:

- *ibaW-750* + output terminals of the WAGO I/O-System 750
- *ibaM-COM* + output modules of the *ibaMAQS* system

If you use *ibaPDA* on an *ibaM-DAQ*, then you configure the *ibaM-DAQ* device itself with its own outputs as well as attached modules of the *ibaMAQS* system on the *IO bus* (internal *ibaNet-E*). Additionally, several modules are available for selection, which can be configured as output modules:

- *ibaW-750* + output terminals of the WAGO I/O-System 750
- *ibaM-COM* + output modules of the *ibaMAQS* system or + *ibaM-FOB-2io* with *ibaPADU-/ibaBM* devices with output modules

The configuration of outputs for *ibaW-750* follows the same way like for *ibaNet-750-BM*.

When configuring the combination of *ibaM-COM* + *ibaM-FOB-2io* + *ibaPADU-/ibaBM* devices with output modules, the rules and settings of the connected devices apply.

7.3.25 ibaM-DAQ outputs

The device *ibaM-DAQ* provides one digital output, which can be configured for the *ibaM-DAQ-IO* module.

For notes on the general module settings see ➤ *General module settings, page 141*

No more module settings need to be made for the outputs.

Output signal

There is only one digital output signal.

Name

Here you can enter a signal name and if you click on the symbol in the *Name* field two more comments.

Expression

By means of the expression builder you can assign signals to the output or create logical and mathematical signal expressions.

Active

Enable/disable the signal

8 The I/O configuration file

All configuration data is stored in an XML file.

The *ibaPDA* system uses three different configuration files for I/Os, data storage and display (layout).

For the I/O configuration, *ibaPDA* always uses the *CurrentIoConfig.io* file. The file is stored in a sub-folder of the *ProgramData* folder (path: `c:\ProgramData\iba\ibaPDA\...`).

This file is loaded with each start of the system. The file is updated with each configuration change and when exiting the I/O Manager by pressing the <OK> button. Thus, the current configuration is always stored in this file without the need for explicit saving. Each time a new configuration is applied, the previous one is automatically saved in the `...\Backup` folder. The last ten configurations are always available in the backup folder.

If the most recently used configuration file is not found, the server automatically loads the most recent backup configuration.

If you have applied a valid and functional configuration, it is recommended to also store the configuration file under another name in a protected folder or disk of your choice in order to avoid unintended modifications or loss of configuration data.

8.1 Saving an I/O configuration

1. Open the I/O Manager after you have made sure that the current configuration is correct.
2. Click the button  (save configuration) in the toolbar of the I/O Manager.
3. In the dialog, select the folder where you want to store the file. We recommend choosing the path `c:\ProgramData\iba\ibaPDA`, as this is the default path of the I/O configuration files. Of course, however, you may also choose any other folder.
4. Enter a file name (prefix).
5. Choose the file name extension (default: .io)
 - .io: A simple XML file will be saved.
 - .zip: The XML file will be saved in ZIP file.
6. Click on <Save>.

8.2 Loading an existing I/O configuration

1. Open the I/O manager.
2. Click the button  (open) in the toolbar of the I/O Manager.
3. Using the browser, select the configuration file you would like to use. Usually, you will find the configuration files in the folder `c:\ProgramData\iba\ibaPDA\Server`. The browser shows only files with the extension `.io`.
The *CurrentIoConfig.io* file contains the latest configuration.
4. Click on <Open>. The configuration will be loaded and verified with the devices connected and the license information in the dongle. If no conflict occurs, the configuration is considered as valid and the acquisition will be started.

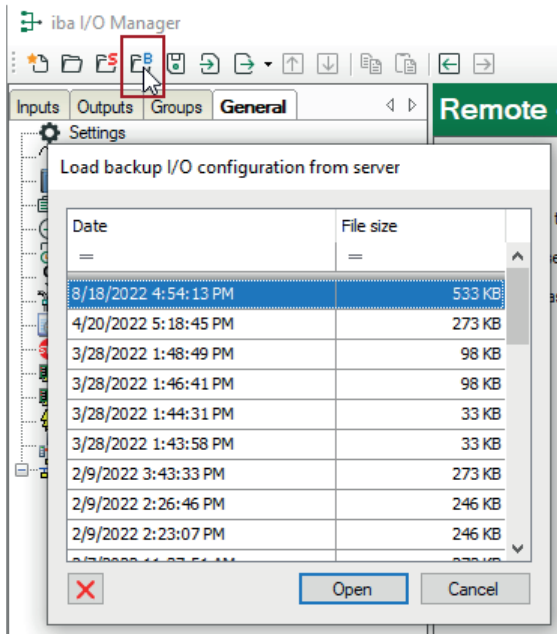
Tip



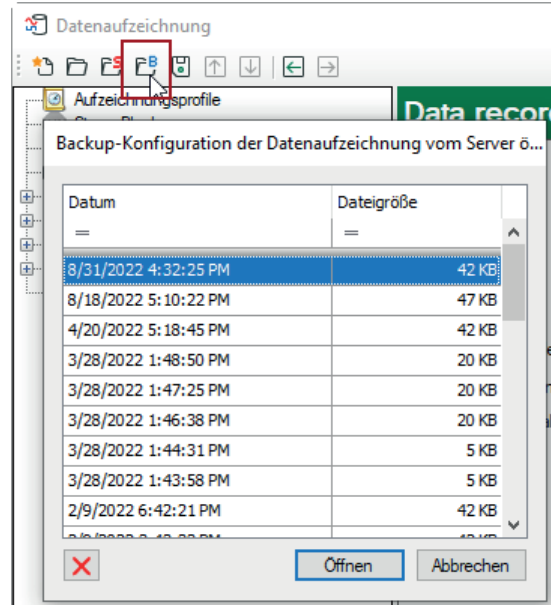
Alternatively, you can store the current configuration in a project. For more information, please refer to the "ibaPDA project management" chapter.

8.3 Loading a backup configuration file

Each time a new I/O configuration or datastore configuration is applied, *ibaPDA* automatically creates a backup of the previous configuration. These backups are stored in the path `%ProgramData%\iba\ibaPDA\Backup` on the *ibaPDA* server computer. In the toolbar of the I/O Manager or data storage manager there is a dedicated button which provides direct access to the backups.



Open backup in I/O Manager



Open backup in datastore manager

If you click on this button, a dialog window will open, which shows a list of the available backups with their date and time of creation and file size.

By selecting a row and clicking on <Open> or by doubleclicking on a row you can open the corresponding backup in the I/O Manager or data storage manager respectively.

If you want to apply the configuration from the backup, then click on <Apply> or <OK>.

In the dialog window with the backup list there is a button to remove a selected backup, if needed.

8.4 Exporting an I/O configuration to a text file

The export function generates a text file which contains modules, groups and profiles. If you have configured profile-based modules (e.g., InSpectra modules), then their profiles will also be exported.

You can only obtain a complete I/O configuration, e.g. including the settings in the *General* branch in the I/O Manager, by saving as an XML file (*.io) or using the project save function.

Modules, groups and profiles can also be exported separately.

You can edit all of the exported files and then reimport them into *ibaPDA*.

This step is sometimes necessary if you update from older *ibaPDA* versions (< v6.0/v7.0) to newer versions.

Even in the case of an initial configuration where many signals need to be configured, editing a text file (e.g., in MS Excel) is usually more convenient and efficient than using the I/O Manager dialogs, especially if you already have the necessary signal information in list form.

Simply create an empty sample configuration which consists of one module of each type.

Next, export this configuration into a text file and then open it with a standard text editor or MS Excel (for example).

You can see the structure of the information in the text file. Copy the module parts as often as needed and adjust the module numbers. You can then edit the signals for each module or insert parts of signal lists from your engineering environment via copy and paste.

Note

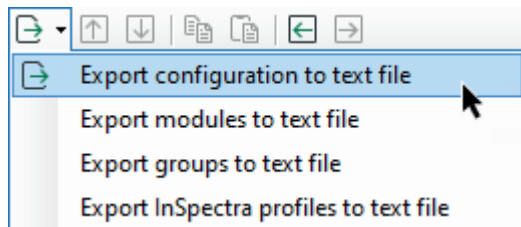


When saving the text file, make sure that a Unicode or UTF-8 data format is selected so that umlauts (mutated vowels) and special characters in signal names and comments are correctly displayed in the I/O manager afterwards.

Once you re-import this text file into *ibaPDA*, the I/O configuration for the modules and signals is complete.

Proceed as follows:

1. Open the I/O manager.
2. Load or create a configuration.
3. Click on the export button and select *Export configuration to text file*.



4. Use the browser to select the desired directory (HDD and folder) in which to store the file, and specify a prefix for the file name. The file suffix *.txt* will be added automatically by *ibaPDA*.
5. Click on <Save>.

Note



The other export options only include the elements specified as part of the overall configuration.

Tip

Export of single modules is possible, too. If there is no need to export an entire I/O configuration, you can export a single module. Simply right-click on the module in the I/O Manager tree and choose *Export* from the context menu.

If you want to export multiple modules from one interface, right-click the interface in the I/O Manager.

Other documentation

More information on exporting InSpectra profiles can be obtained from the *ibaInSpectra* manual.

8.5 Importing a configuration text file

You may import files which have been created by older ibaPDA versions (V5, V6), ibaScope or by other tools (e.g. MS Excel), provided they match the ibaPDA-format.

The files must be simple text files. No XML files!

In case of ibaScope configuration files you need files with the extension ".setup".

1. Open the I/O manager.
2. Click the button  (Import)
3. By using the browser, select the corresponding configuration text file from its storage location (local drive, network drive, floppy disk, CD-ROM etc.). By default, the browser only shows files with the extension .txt. If you are looking for ibaScope files, select the appropriate file type in the lower right corner of the dialog.
4. Click <Open>. The configuration will be loaded and verified with the devices connected and the license information in the dongle. If no conflict occurs, the configuration is considered as valid and the measurement will be started.
5. Store the configuration in the usual way, i.e. as an ibaPDA configuration file, as described above.

Note



Please note that the module and channel settings are taken over from ibaPDA-V5, however, not the archive profiles! The new data storage profiles should be defined after the import is done and the signals should be assigned to the new profiles.

9 Support and contact

Support

Phone: +49 911 97282-14

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.

Product Security Incident Response Team (PSIRT) at iba

Current advisories: www.iba-ag.com/en/security

Email: psirt@iba-ag.com

Contact

Headquarters

iba AG
Gebhardtstrasse 10–20
90762 Fuerth
Germany

Phone: +49 911 97282-0

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