



# **ibaPDA v8.4.0**

## **New Features**

02.08.2023  
iba AG

## Table of contents

<b>1</b>	<b>General remarks.....</b>	<b>2</b>
1.1	Default installation destination .....	2
1.2	Breaking change in DataStoreInfoHD diagnostic function .....	2
1.3	Dependencies to other iba products.....	2
1.4	iba product discontinuation .....	2
<b>2</b>	<b>New installer.....</b>	<b>3</b>
2.1	Welcome page.....	3
2.2	License agreement page .....	3
2.3	Destination location page.....	4
2.4	Components page .....	5
2.5	License information page.....	6
2.6	WIBU CodeMeter Runtime update page.....	7
2.7	Service user page.....	7
2.8	Service mode page.....	8
2.9	Additional tasks page.....	8
2.10	Installing page.....	9
2.11	Finish page .....	9
2.12	Command line options .....	9
2.13	ibaCapture installation .....	10
2.14	Windows firewall exceptions .....	10
2.15	Help .....	12
<b>3</b>	<b>Signal mapper .....</b>	<b>13</b>
3.1	Licensing .....	13
3.2	Signal mapper module configuration.....	13
3.3	Watch view .....	18
<b>4</b>	<b>ibaHD: Triggered Data Acquisition .....</b>	<b>20</b>
4.1	Trigger configuration .....	21
4.2	Start trigger.....	22
4.3	Stop trigger .....	24
4.4	Status .....	25
<b>5</b>	<b>DataStoreInfoHD diagnostic function .....</b>	<b>26</b>
<b>6</b>	<b>IEC 61850-7-2 support .....</b>	<b>27</b>
<b>7</b>	<b>Analytics tab.....</b>	<b>30</b>
<b>8</b>	<b>Support for machine learning .....</b>	<b>31</b>
8.1	InCycle ONNX .....	31
8.2	InSpectra ONNX .....	34

## **1 General remarks**

### **1.1 Default installation destination changed**

Please see chapter 2.3.

Please consider, if you use manually created shortcuts, the path for calling ibaPDA must be adjusted manually.

### **1.2 Breaking change in DataStoreInfoHD diagnostic function**

Please see chapter 5.

### **1.3 Dependencies to other iba products**

ibaHD-Server v3.2.0 is strongly recommended in case ibaHD triggered data acquisition is used.

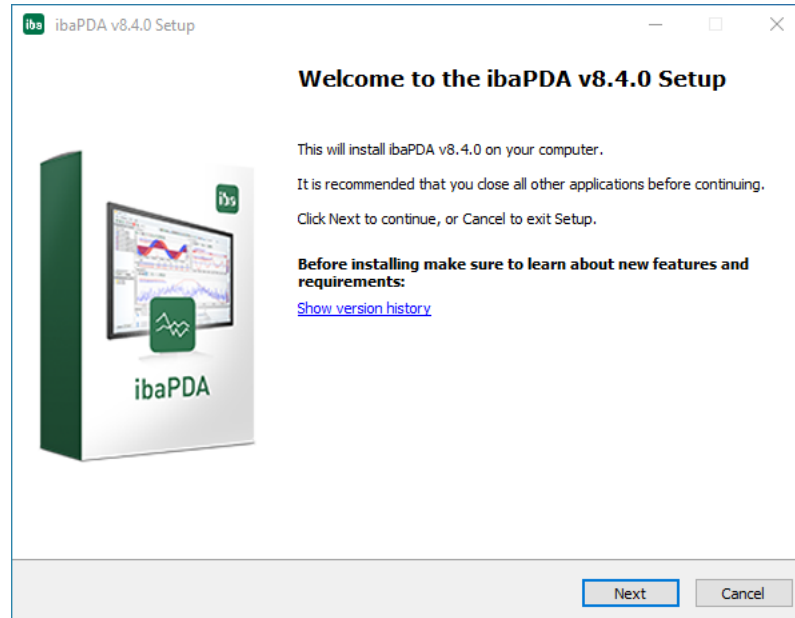
### **1.4 iba product discontinuation**

Support stopped for 12.101000 ibaCom-PCMCIA and 12.102000 ibaCom-PCMCIA-F

## 2 New installer

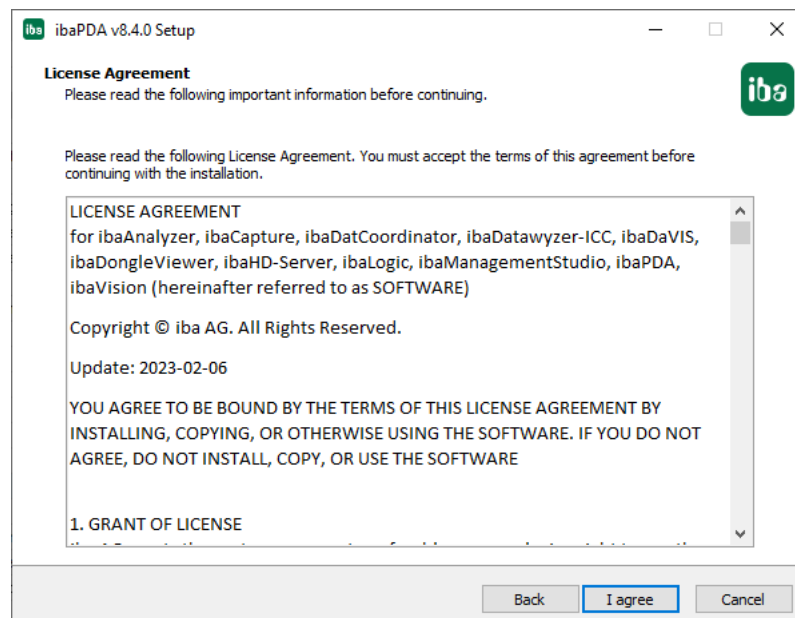
The installer has been rebuilt. Here is an overview of the different pages.

### 2.1 Welcome page



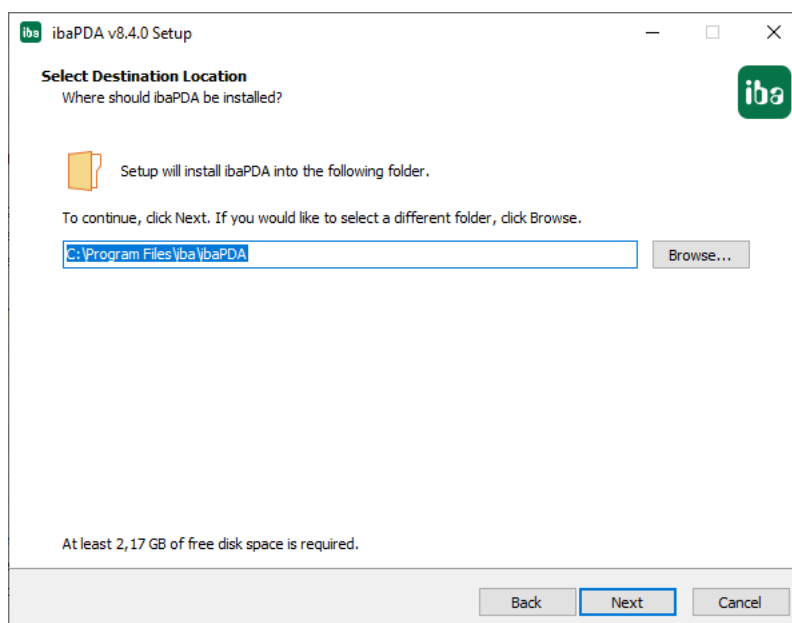
On the welcome page you can use the “Show version history” link to open the version history and review the changes that are part of the version you are about to install. Please pay attention to the breaking changes.

### 2.2 License agreement page



On the license agreement page you can read the license agreement. You have to agree with it before you can install ibaPDA.

## 2.3 Destination location page

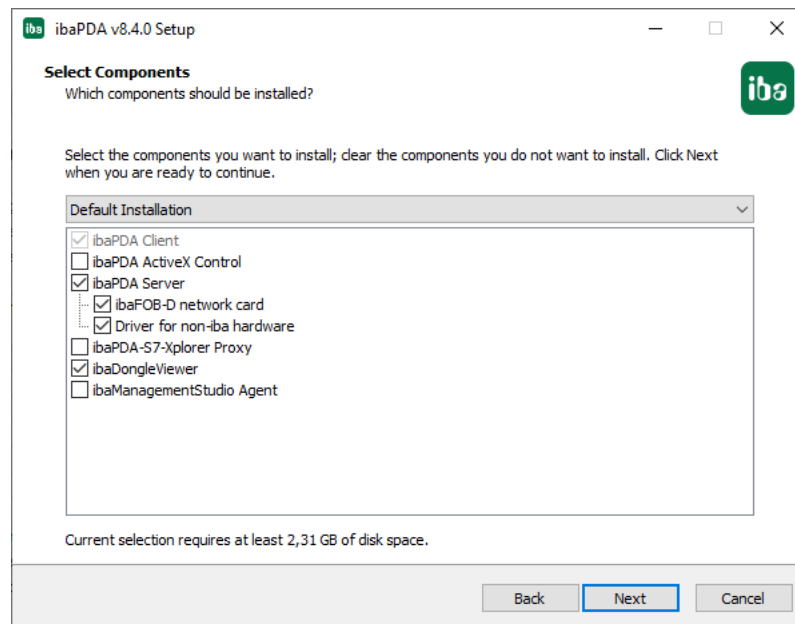


On the destination location page you can configure where ibaPDA should be installed. The default location has changed from “C:\Program Files (x86)\iba\ibaPDA” to “C:\Program Files\iba\ibaPDA”. In previous installers there were client and server subfolders created underneath the installation folder. This is no longer the case. The client and server files are both installed in the same directory. This means that less space will be taken since some files will no longer exist twice.

The server plugins that contain custom functions for virtual modules have been moved as well. They used to be loaded from {InstallDir}\Server\Plugins. Now they are loaded from %ProgramData%\iba\ibaPDA\Plugins\Server. The installer will copy all files from the old directory to the new directory. No files will be overwritten. So the copy will happen only once.

The client plugins that contain custom views have been moved as well. They used to be loaded from {InstallDir}\Client\Plugins. Now they are loaded from %ProgramData%\iba\ibaPDA\Plugins\Views. Like with the server plugins the installer will copy the view plugins from the old location to the new location.

## 2.4 Components page



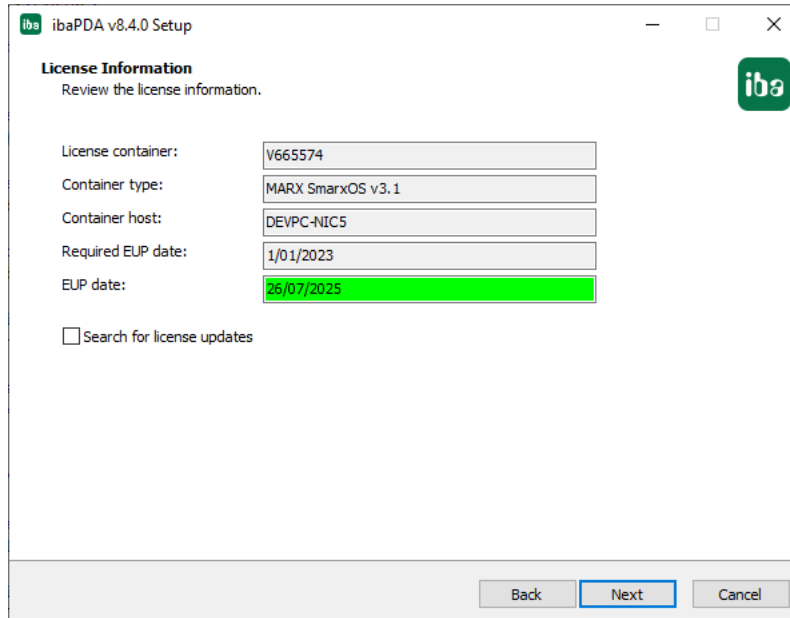
On the components page you can select which components of ibaPDA should be installed.

- ibaPDA Client: The ibaPDA client application that is used to configure ibaPDA server and to visualize the measured data. This will always be installed.
- ibaPDA ActiveX Control: The ActiveX control is a client that can be embedded in other applications. This is typically used to embed an ibaPDA client in HMIs like e.g. Siemens WinCC.
- ibaPDA Server: The ibaPDA server that is used to measure the data.
  - ibaFOB-D network card: Driver for the virtual network card that is coupled to ibaFOB-D cards installed in the system. This component is only needed when an ibaFOB-D card is installed and you want to measure data from a device connected via the ibanet 32Mbit/s flex protocol.
  - Driver for non-iba hardware: This is only required when you want to measure data via one of the following cards:
    - Reflective memory
    - CP1616 or CP1626
    - PC-Link
    - DGM200P
    - Toshiba ADMAP JAMI1
    - Scramnet+
- ibaPDA-S7-Xplorer Proxy: This component allows you to measure data from Siemens S7-PLCSim.
- ibaDongleViewer: This component provides information about the connected license containers. IbaPDA doesn't require it to work correctly. It is purely optional.
- ibaManagementStudio Agent: This component communicates with the ibaManagementStudio server. It can provide information about the installed software and

about the current state of the installed software. It also allows remote installation of iba software. It is only required if you use ibaManagementStudio.

By default the client, server and ibaDongleViewer are installed.

## 2.5 License information page

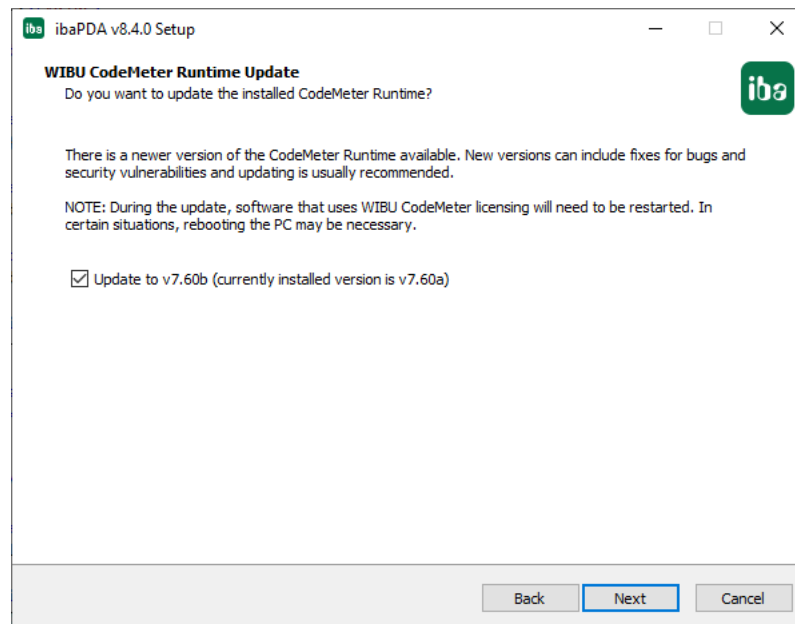


The screenshot shows the 'License Information' window of the 'ibaPDA v8.4.0 Setup' application. The window title bar includes the iba logo and standard window controls. The main content area is titled 'License Information' with the instruction 'Review the license information.' Below this, there are five input fields: 'License container:' with the value 'V665574', 'Container type:' with 'MARX SmarxOS v3.1', 'Container host:' with 'DEVPC-NIC5', 'Required EUP date:' with '1/01/2023', and 'EUP date:' with '26/07/2025'. The 'EUP date' field is highlighted in green. At the bottom left, there is a checkbox labeled 'Search for license updates' which is currently unchecked. At the bottom right, there are three buttons: 'Back', 'Next' (which is highlighted with a blue border), and 'Cancel'.

If the ibaPDA server is selected and there is a license container connected then the license information page will be shown. It shows information about the connected license. The most important information is about the EUP date. If the EUP date is equal or higher than the required EUP date then this version of ibaPDA can be installed on the system.

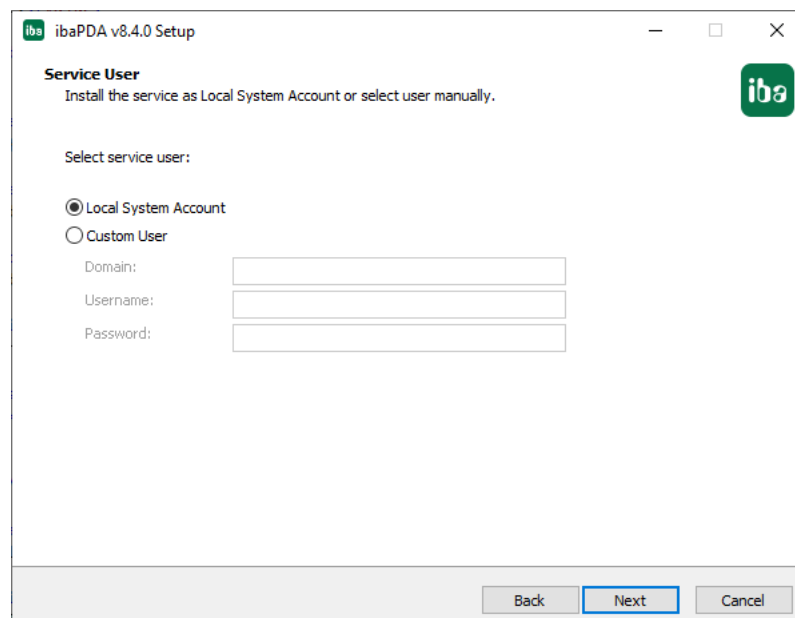
When a MARX dongle is connected then the “Seach for license updates” checkbox is shown. When it is checked then the installer will look for license updates for the connected MARX dongle in a configured (network) folder. If the folder hasn’t been configured yet then a dialog is shown where you can configure the folder. If you later want to change the folder then manually run ibaLicenseUpdater.exe from the installation directory.

## 2.6 WIBU CodeMeter Runtime update page



If the ibaPDA server is selected and the installed CodeMeter Runtime is older than the version embedded in the installer then this page will be shown. It allows you to choose whether the CodeMeter Runtime should be updated or not. It is recommended to update since the update can contain fixes for bugs and/or security vulnerabilities. You can choose not to install the update when other software that is using CodeMeter Runtime is currently running and shouldn't be stopped. This could be e.g. ibaHD or ibaCapture.

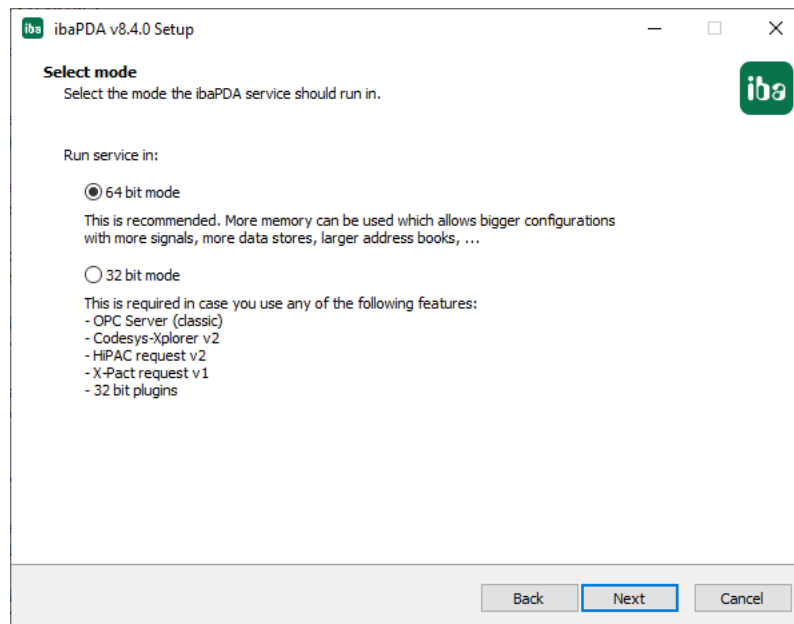
## 2.7 Service user page



On this page you can configure which user the ibaPDA service should be running as. It is recommended to use the local system account since this user account has all the privileges that the ibaPDA service needs. This page is only shown when doing a fresh installation. It is skipped when doing an update.



## 2.8 Service mode page

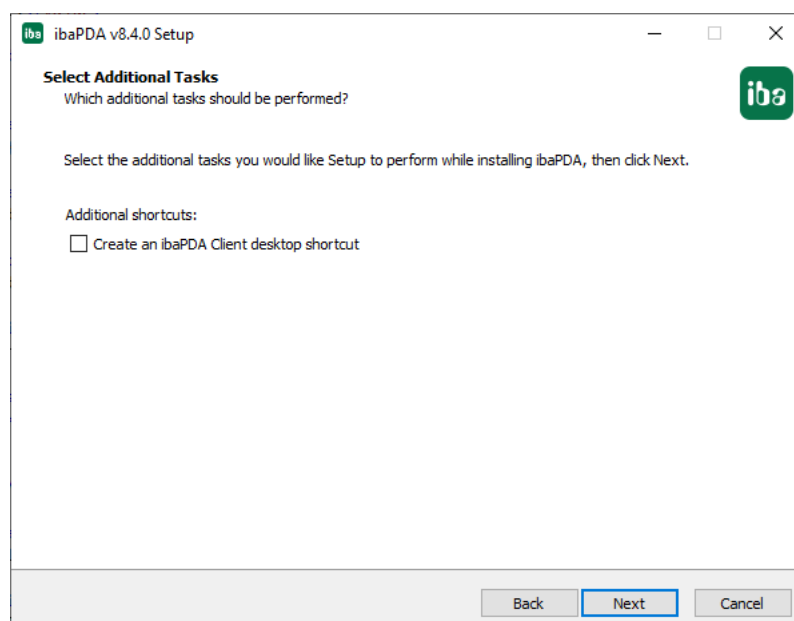


On this page you can configure if the server runs in 64 bit mode (default) or 32 bit mode. This page is only shown when installing the server on 64 bit systems. It is recommended to install the 64 bit version. You should only use the 32 bit version in case you use any of the following features:

- OPC server (classic)
- Codesys-Xplorer v2
- HiPAC request v2
- X-Pact request v1
- 32 bit server plugins

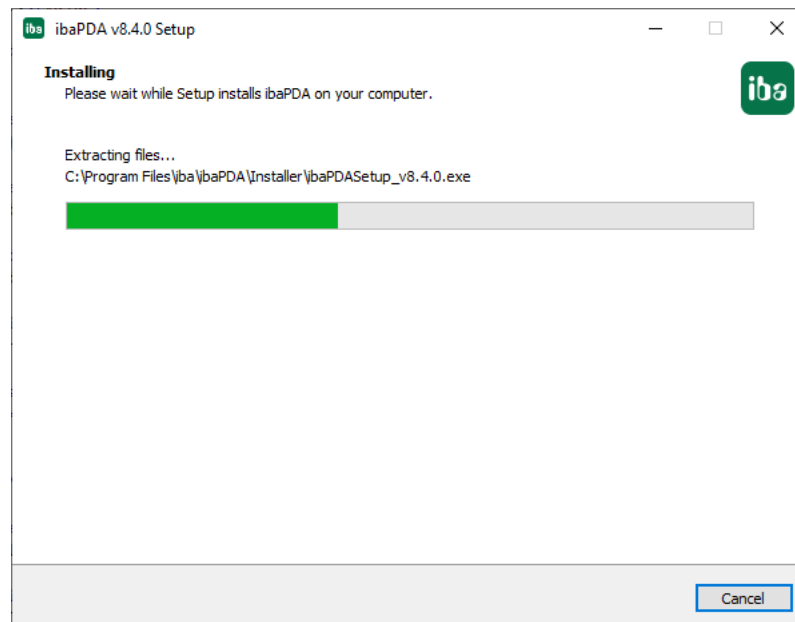
This page is only shown when doing a fresh installation. It is skipped when doing an update.

## 2.9 Additional tasks page



On this page you can decide whether or not a shortcut to the ibaPDA client should be created on the desktop.

## 2.10 Installing page



This page shows the progress while installing ibaPDA.

## 2.11 Finish page



This page is shown when the installation is finished. If an error occurred while installing the service then this error is shown in red. If the “*Open ibaPDA Client*” checkbox is checked then the ibaPDA client will be started when you click the *Finish* button. There is also a hyperlink you can click to subscribe to the iba newsletter.

## 2.12 Command line options

The following (case-insensitive) command line options are supported:

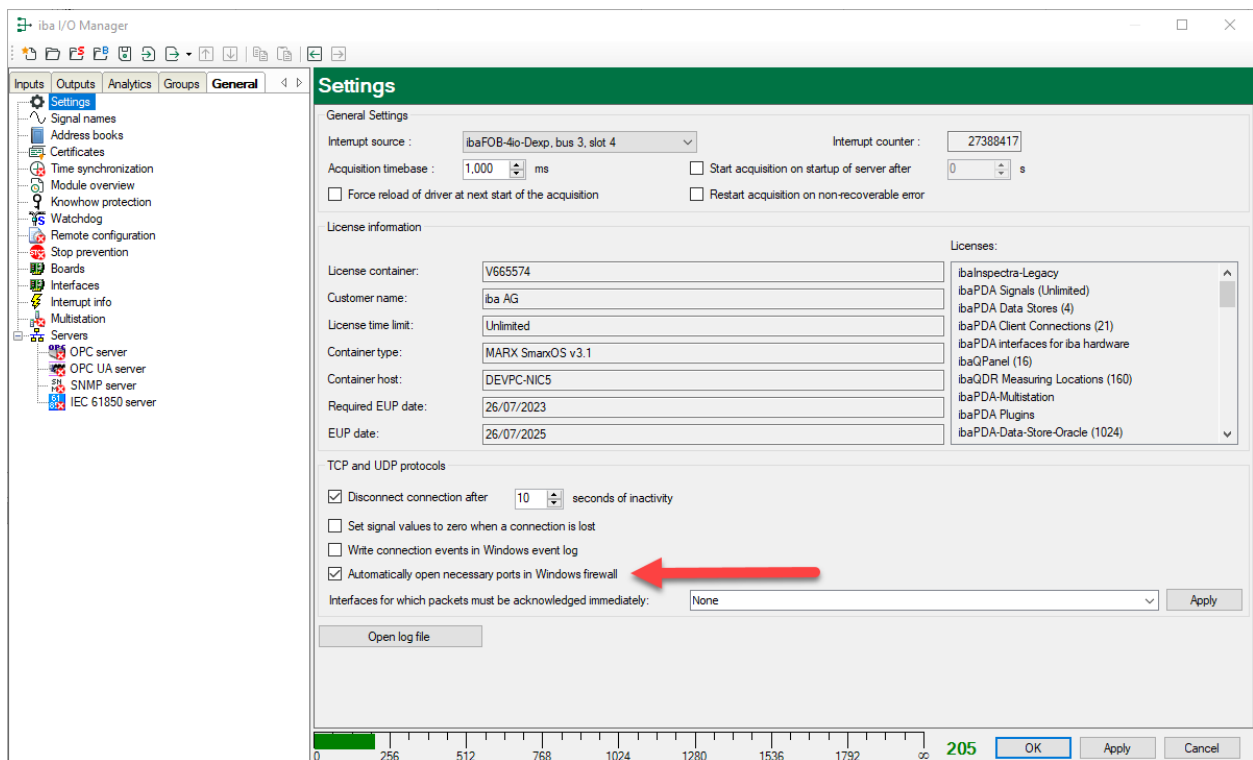
- /HELP: Shows a message box containing all supported command line parameters.
- /SILENT: All pages except for the *Installing* page are skipped. There are also no message boxes shown.
- /VERYSILENT: Nothing is shown.
- /COMPONENTS="comma separated list of component names": This installs only the components in the list. The component names are:
  - ibaPDAClient
  - ibaPDAActiveXControl
  - ibaPDAServer
  - ibaPDAServer\ibaFOBNetwork
  - ibaPDAServer\NonIbaHW
  - ibaPDAS7XplorerProxy
  - ibaDongleViewer
  - ibaManagementStudioAgent
- /LANG=id: Force the language of the installer. By default the language of the operating system is used. The following values are allowed as id:
  - de: German
  - en: English
  - es: Spanish
  - fr: French
  - it: Italian
  - ja: Japanese
  - pt: Portuguese
  - ru: Russian
  - zh: Chinese

## 2.13 ibaCapture installation

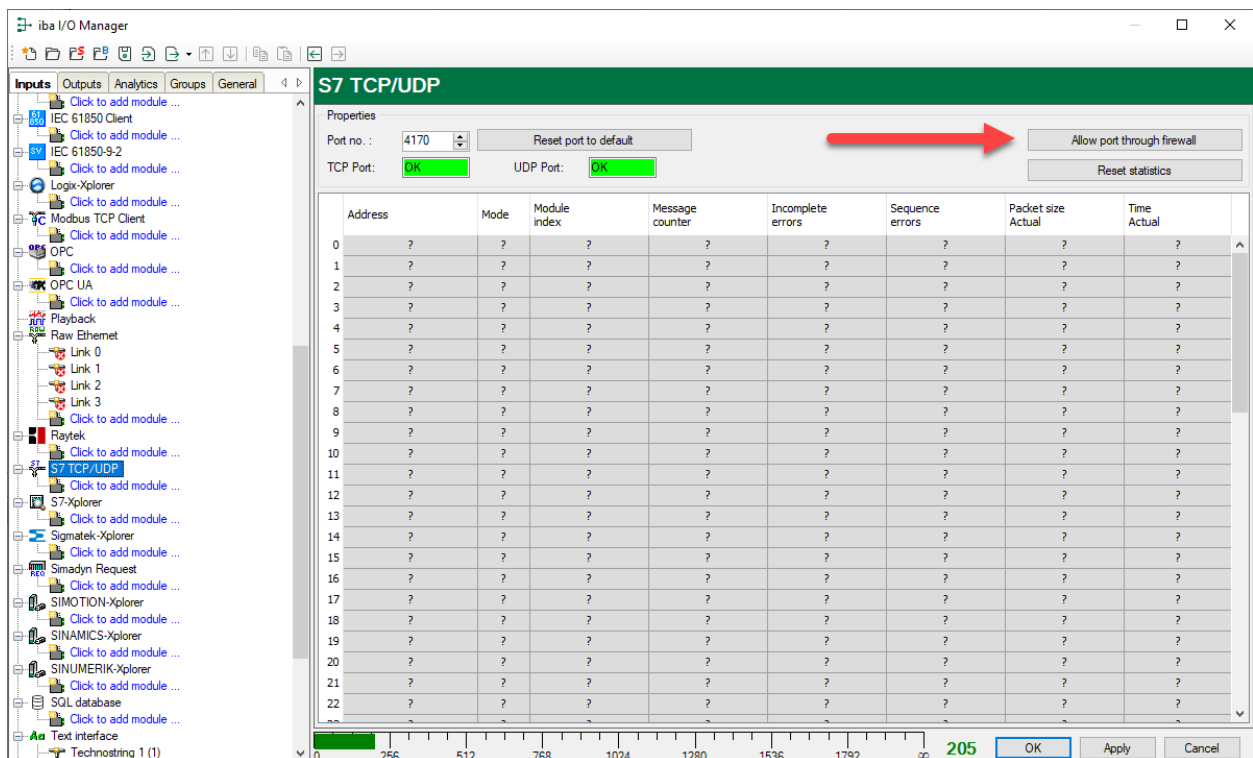
The ibaCapture camera view and player have been integrated in the ibaPDA installer. They are installed in the ibaPDA installation folder. These versions are only used by ibaPDA client and server. They are not used by other iba products installed on the system like e.g. ibaAnalyzer. If you install ibaCapture itself then ibaPDA will still keep using its local camera view and player and not the one from the ibaCapture installation.

## 2.14 Windows firewall exceptions

The new installer adds application exceptions for ibaPDA client, server and the S7-Xplorer proxy to the Windows firewall. It does not add exceptions for network protocols that are handled by the ibaPDA driver. The previous installer added exceptions for those protocols even if they were not used. In ibaPDA there is now an option to automatically add port exceptions for the protocols that are used. By default, this option is enabled.



If you disable this option then you can manually add port exceptions to the firewall by clicking the “Allow port through firewall” button on each interface that requires a listen port.



## 2.15 Help

The installer installs a new help system for ibaPDA. It is HTML-based. Anywhere in the ibaPDA client you can press F1 to open the help in the default browser. The help will go to the page that is relevant for the currently focused window.

The screenshot shows the ibaPDA Help system interface. The top navigation bar includes the iba logo, "ibaPDA Help", and links for Info, Introduction and Installation, I/O Manager, Data Interfaces and Modules, Expression Builder, Data Storage, Data Visualization, and Appendix. A search bar is located below the navigation bar. The main content area is titled "Common and general module settings" and is divided into two columns. The left column contains a tree view with categories like "Data Interfaces and Modules", "Basics of modules", "Module settings and features", and "Common and general module settings". The right column displays the content for "Common and general module settings", including sections for "Basic settings", "Locked", "Enabled", "Name", and "Module No."

You can also use the search function on the help page to look for topics.

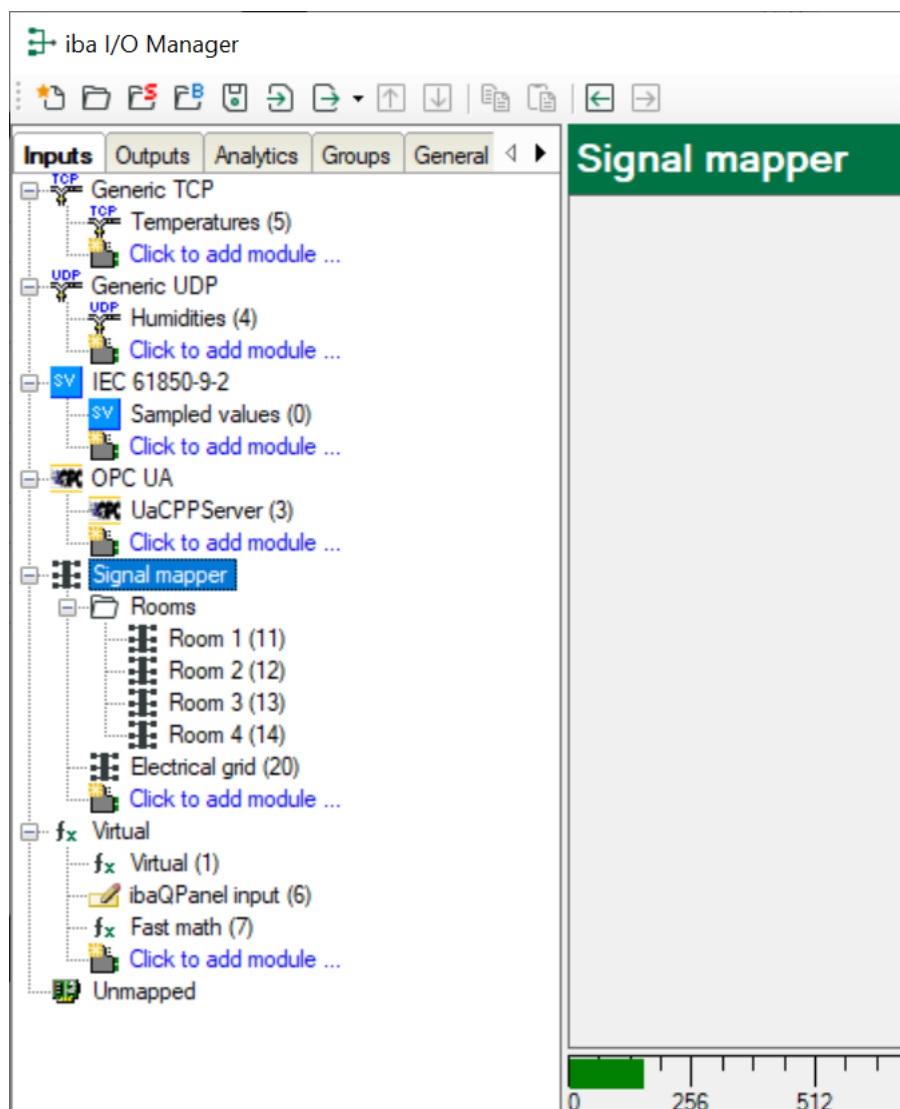
### 3 Signal mapper

A signal mapper module basically creates an abstraction layer between the input layer (e.g. Generic UDP and TwinCAT-Xplorer signals) and the visualization or processing layer. One of the main goals is to create some form of standardization between several ibaPDA systems while another goal is to allow offline preparation of layouts: these can now be created and linked to signal mapper modules and only at the end is there a need to configure the actual source signals (by linking them to the mapped signals). To accommodate testing of I/O configuration and layouts it is also possible to run a configuration in ibaPDA where no source signals are assigned to the mapped signals. The default value of mapped signals without source signals is 0 but can be changed using new functionality in the Watch view.

#### 3.1 Licensing

Since signal mapper modules do not provide additional connectivity but are merely a means to organize an I/O configuration, these modules are completely license-free: even without a valid ibaPDA license it is possible to add as many signal mapper modules as required and furthermore, they can be referenced without limitation in ibaQPanels or other views.

#### 3.2 Signal mapper module configuration



A new interface called “Signal mapper” is available in the Inputs tab of the I/O Manager. There it is possible to add new signal mapper modules. As is the case for most other interfaces, it is also possible to create folders for additional structure.

**Room 1 (11)**

**General** Analog Digital

**Basic**

Module Type	Signal mapper
Locked	False
Enabled	True
Name	Room 1
Module No.	11
Use module name as prefix	False

**Profile**

Profile Room

**Source signals**

Hide source signals

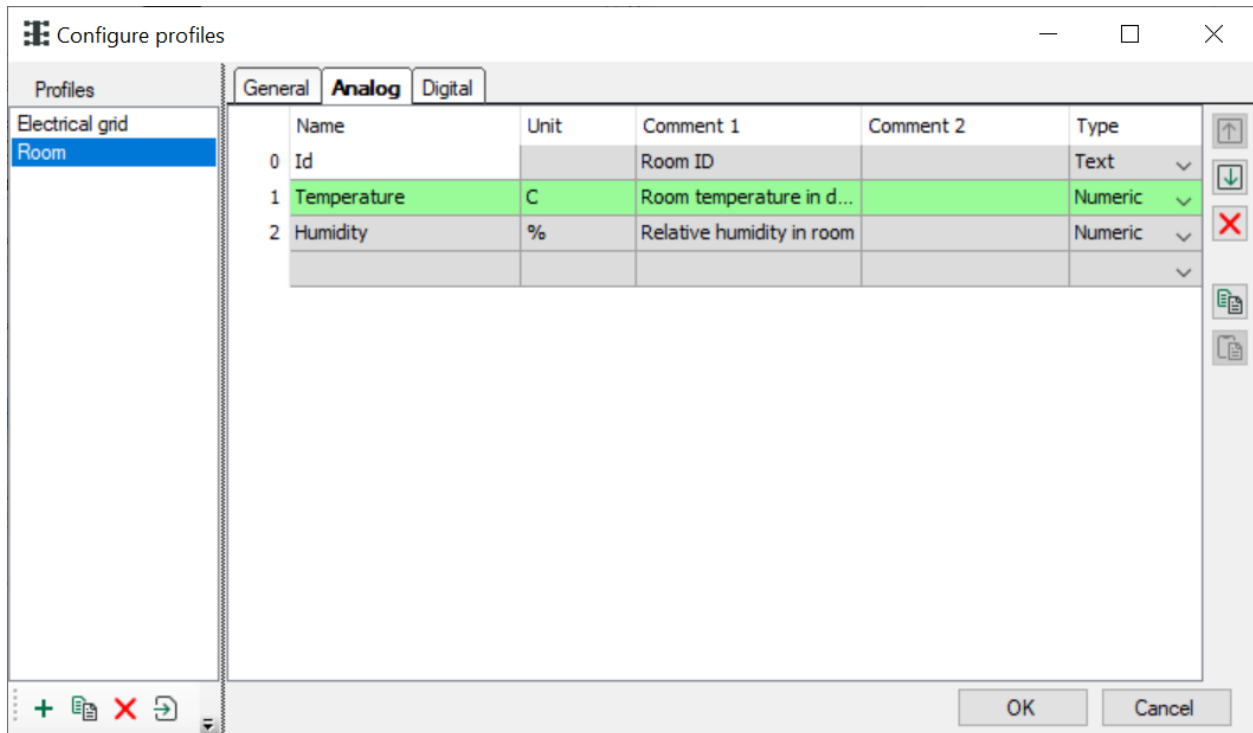
<No profile>  
Electrical grid  
Room  
<Edit profile...>  
<New profile...>

**Profile**  
The signal mapper profile that contains the signal definitions.

[Configure profiles](#)

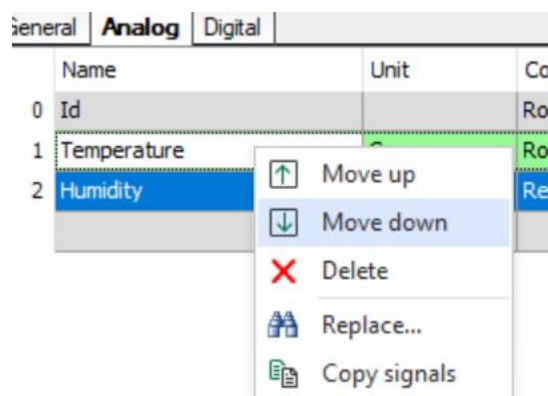
Signal mapper modules are profile-based which means that its signals are automatically generated based on a profile that can be reused across modules. For example, in the above illustrations the profile “Room” is used for modules “Room 1” to “Room 4”. Changes in a profile automatically reflect on the modules referencing that profile.

Profiles can be configured by either clicking the Configure profiles link at the bottom left or by using the drop-down menu of the module’s Profile property.



In the Analog and Digital tabs a signal grid can be found where you can define the signals that will end up in the actual signal mapper modules referencing this profile. The standard fields which can be found in the signal grid of other modules are available here: Name, Unit (only for analog signals), Comment 1 and Comment 2. The Type field is also available but has different values than those of regular analog signals: in this case it is only required to define whether the signal should be a text signal or a numeric signal. The actual data type will be the one of the source signal configured in the signal mapper module using the profile.

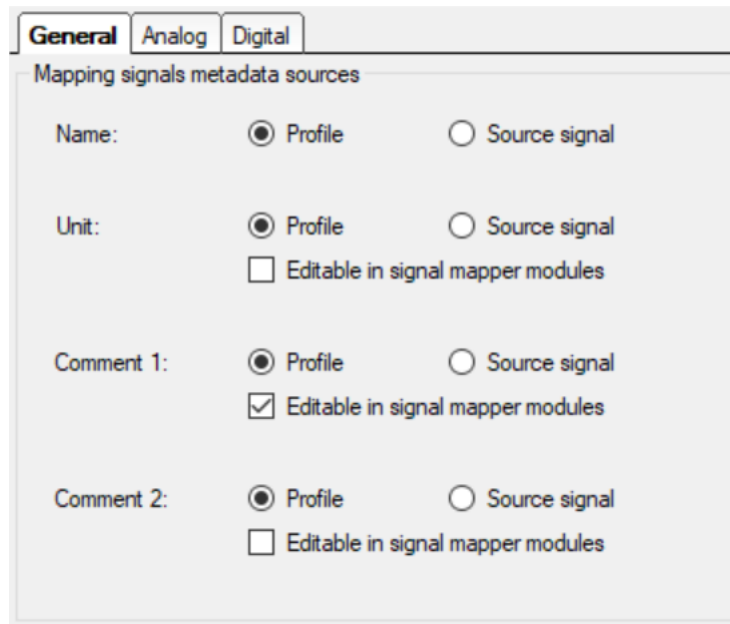
It is possible to select multiple signal rows and open a context menu by right-clicking on one of the selected rows.



The selected signals can be moved up or down or deleted. The selected rows can also be copied to the clipboard by clicking "Copy signals".

The same functionality can be achieved by using the buttons at the right side. The Copy button however copies the entire grid to the clipboard instead of only the selected rows.





**General** Analog Digital

Mapping signals metadata sources

Name: ☒ Profile ☐ Source signal

Unit: ☒ Profile ☐ Source signal  
☐ Editable in signal mapper modules

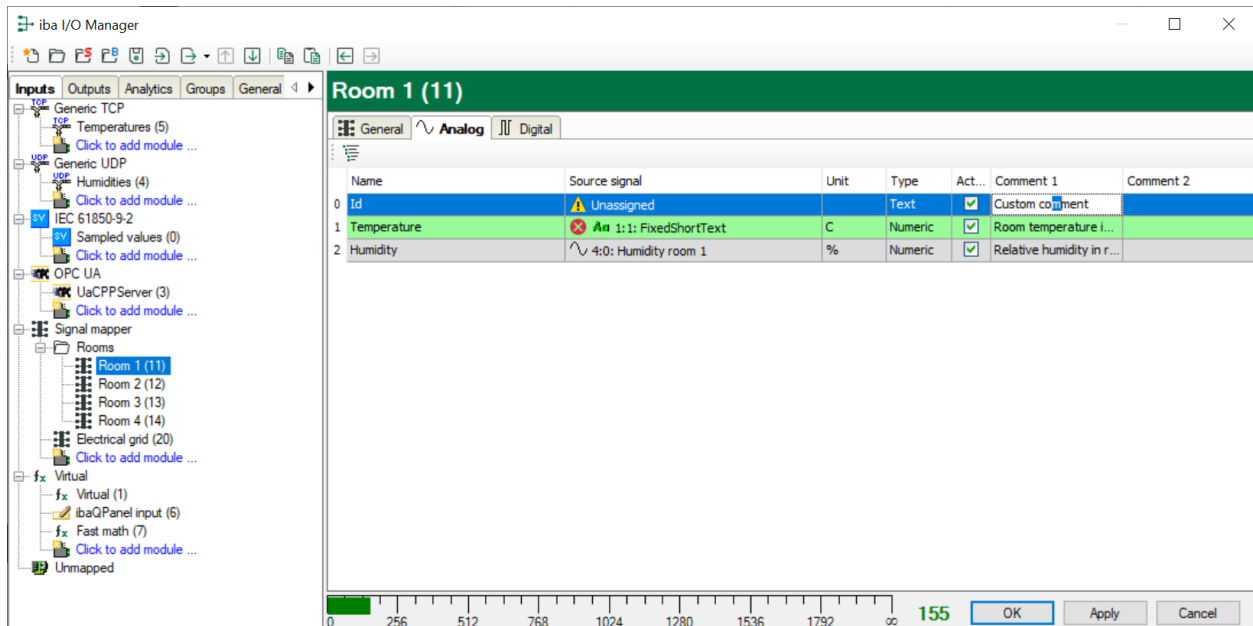
Comment 1: ☒ Profile ☐ Source signal  
☒ Editable in signal mapper modules

Comment 2: ☒ Profile ☐ Source signal  
☐ Editable in signal mapper modules

In the General tab of a profile you can configure where the metadata of mapping signals (i.e. the signals of a signal mapper module) should be sourced from. For the signal name, unit and comments 1 and 2 it is possible to select whether the values defined in the profile should be used or the values defined in the source signal. Furthermore it is possible to configure whether or not to allow signal mapper modules referencing this profile to override (i.e. provide a custom value) the metadata coming from either the profile or source signal. Note that the signal name cannot be overridden and that the signal unit is only relevant for analog signals.

Finally, as is the case for all ibaPDA modules using profiles it is possible to add, clone, delete, import and export profiles using the buttons at the bottom left of the “Configure profiles” dialog.

Once you have defined a profile make sure it is selected in the list on the left side and click OK.



The Analog and Digital tabs of the signal mapper module will now be automatically updated. All signal fields defined in the profile are available as read-only fields (unless the profile is configured to allow overriding of Unit, Comment 1 or Comment 2). An additional field is the Source signal: here you can define the source signal (i.e. the signal that is being mapped). The source signal's type must match that of the mapped signal. When this is not the case an error symbol will be displayed, as can be seen in the above figure for signal 1, and a validation error will occur. It is allowed to have no source signal linked to the mapped signal (i.e. Unassigned). This is particularly useful for preparing I/O configurations and layouts without knowing where the actual source data will come from. In that case, the mapping signal will use the default acquisition timebase and in case of a numeric signal will have the data type float. A source signal can also be used multiple times in the same signal mapper module or across other signal mapper modules.

A signal mapper module has no timebase of its own and can therefore map signals with different timebases in the same module (including signals with a timebase lower than 1 ms or the interrupt cycle time).

Since mapping signals can also be used in the data storage configuration it might occur that same signal data is recorded twice: once as the original (source) signal and once as the mapping signal. To avoid this redundancy automatically the option "Hide source signals" can be activated. When enabled, the source signal will no longer show up in the signal tree used for layouts and data storage configuration. In case of a conflicting property for a certain signal (e.g. module 1 references signal A and has "Hide source signals" enabled and module 2 also references signal B but has "Hide source signals" disabled) the source signal will not be hidden.

When mapping signals are written into DAT files specific infofields belonging to the source signal will be written as well along with the source name and ID as extra infofields.

Name	Source signal	Unit	Type	A...	Comment 1	Comment 2
0 Id	1:5: Room1 Id		Text	<input checked="" type="checkbox"/>	Room ID	
1 Temperature	1:4: Sine	C	Num...	<input checked="" type="checkbox"/>	Room temper...	
2 Humidity	4:0: Humidity room 1	%	Num...	<input checked="" type="checkbox"/>	Relative humi...	

To facilitate assigning source signals to mapping signals there's a possibility to show the current signal tree by clicking the button in the toolbar at the top left in the Analog or Digital tab. Signals can easily be dragged and dropped into a cell of the Source signal column. Note that non-text signals cannot be dragged and dropped to rows corresponding to text signals (and vice versa).

### 3.3 Watch view

Mapping signals behave like most other ibaPDA signals when used in expressions and views so they will behave the same in the Watch view. However, mapping signals that do not have a source signal assigned to them are handled slightly different in the Watch view.

Signal	Value	New value	Apply
Mapped signals			
[11:0] Id	Room 001		<input type="checkbox"/>
[11:1] Temperature	3.681245565 C		<input type="checkbox"/>
[14:1] Temperature	0 C		<input type="checkbox"/>
[14:2] Humidity	0 %		<input type="checkbox"/>
[11:0] IsOccupied	0		<input type="checkbox"/>
[12:0] IsOccupied	0		<input type="checkbox"/>
[13:0] IsOccupied	0		<input type="checkbox"/>
[14:0] IsOccupied	0		<input type="checkbox"/>
ibaQPanel inputs			
[6:0] TestAnalog	0		<input type="checkbox"/>
[6:0] TestDigital	0		<input type="checkbox"/>
*			<input checked="" type="checkbox"/>

When the Watch view contains such mapping signals without source signal an additional button and two additional columns will be shown: *New value* and *Apply*. This allows a value to be forced upon those signals.

First, you should select which signals you actually want to force a value on by checking the Apply checkbox in the appropriate rows. Then you can prepare a new value in the corresponding column of the grid. Once all values are prepared, press either the lightning button at the right or press F6 on the keyboard (make sure that you're not editing a cell value or the hotkey won't work).

Digital signals can easily be flipped by pressing F8 on the keyboard. Only signals for which the Apply checkbox has been checked will be flipped.

The *New value* and *Apply* columns support header clicking meaning that when clicking the column header, the contents of the selected cell for that column will be copied to the subsequent rows.

When using a mix of writable and non-writable signals a new value or the checkbox in the Apply column cannot be set for the non-writable signals (as is also indicated by the gray cell background).

Note that ibaQPanel input signals can also be written using the Watch view now.

## 4 ibaHD: Triggered Data Acquisition

In earlier versions, ibaPDA continuously sends data to the configured ibaHD servers – even if there is no data to be recorded.

This is not always optimal and may result in unnecessary bandwidth usage, storage usage and more irrelevant data.

Starting with version 8.4.0, ibaPDA supports trigger-based ibaHD time recording.

This new feature has numerous advantages, such as

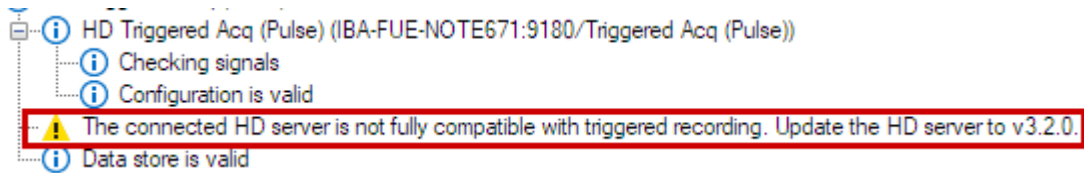
- Reduced bandwidth and storage usage
- Easier and faster analytics of important situations because of less irrelevant data
- Highly configurable for perfect adaption to specific use cases

### Remark:

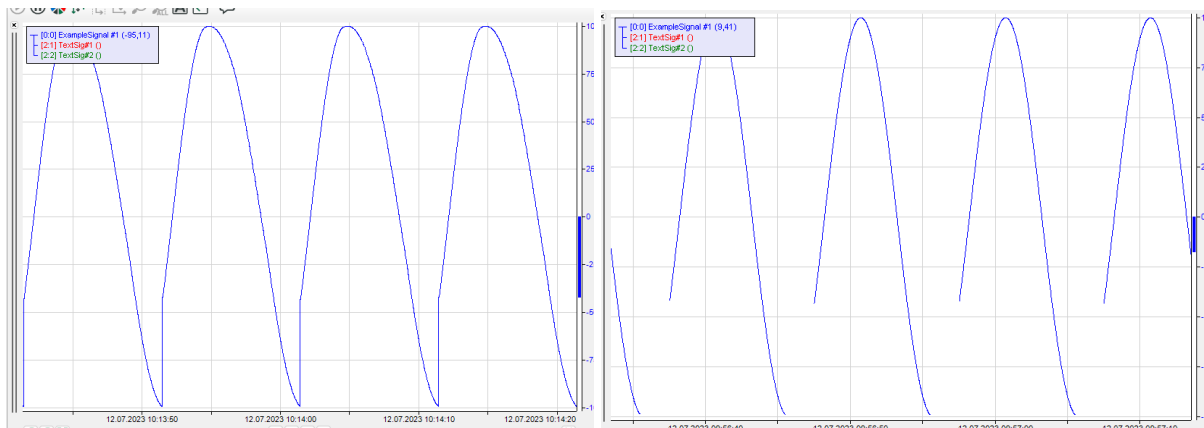
While trigger-based time recording works with previous ibaHD server versions, it is not fully compatible with them.

To ensure full compatibility, at least ibaHD server version 3.2.0 is required. Otherwise the ibaHD server may have issues with the gap handling.

This is also indicated with a warning when validating the configuration.



Older versions handle short gaps incorrectly (left image) compared to the right handling in the newer versions (right image).



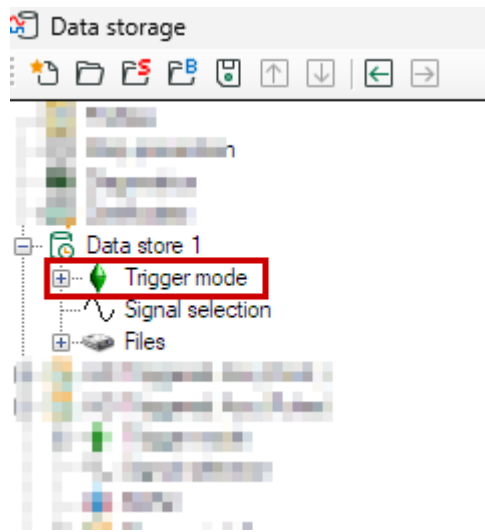
Additionally, older versions log a gap warning.

```

2023-07-03 10:33:09.147 [Debug][60:copy(Triggered Data Acq)] Resume seg 00000103 (l=5 c=22 rs=22 s=12)
2023-07-03 10:35:24.927 [Debug][60:copy(Triggered Data Acq)] Input gap detected in 17 buffers (max = 00:00:20.0126212)
2023-07-03 10:36:57.196 [Debug][60:copy(Triggered Data Acq)] Input gap detected in 17 buffers (max = 00:01:00.0197026)
2023-07-03 10:37:52.945 [Debug][60:copy(Triggered Data Acq)] Input gap detected in 17 buffers (max = 00:00:40.0104555)
2023-07-03 10:39:02.567 [Debug][60:copy(Triggered Data Acq)] Input gap detected in 17 buffers (max = 00:00:40.0195639)
2023-07-03 10:39:32.093 [Debug][60:copy(Triggered Data Acq)] Input gap detected in 2 buffers (max = -00:00:25.9399725)
2023-07-03 10:39:32.095 [Debug][60:copy(Triggered Data Acq)] Input gap detected in 1 buffers (max = 00:00:05.9402406)
  
```

## 4.1 Trigger configuration

To configure trigger-based time recording in the ibaPDA data storage configuration, ibaHD time stores provide the new *Trigger mode* node.



The *Trigger mode* node is similar to the well known *Trigger mode* node of the normal file based data storage.

For each ibaHD time store, the conditions of the start– and stop trigger can be defined.

By default, the settings are already set to the continuous recording – just like in previous ibaPDA versions.

## 4.2 Start trigger


As the name suggests, the *start trigger* defines the beginning of the data transfer to the ibaHD server.

The *trigger type*, *pre-trigger time* and the *pre-trigger buffer location* can be set.

### 4.2.1 Trigger type

*Trigger type* defines under which circumstances data should be sent to the IbaHD server.

Trigger type:

☒ Unconditional  
☐ On signal       Unassigned      rising edge  
☐ Trigger every      60      minutes starting at      00:00  
☐ Use start trigger pool  
☐ When a time period starts

The following options are available.


- Unconditional
  - There is no trigger, data should be sent continuously. This setting also disables all other settings.
- On Signal
  - *On Signal* is used to start the recording based on the selected signal.
  - For analog signals, a *level type* (rising edge, falling edge, above level or below level) and the *level* itself can be selected. Once the value of the signal matches the selected arguments, the recording is started.
  - For digital signals *level type* and *level value* are hidden. The trigger will match if the digital value is set to 1.
- Trigger every (period) minutes starting at (time)
  - This setting can be used to periodically start the recording
  - The *period* refers to the current system time and not the time when the PDA acquisition started
  - With *time* the recording is limited to only run after the selected time
- Use start trigger pool
  - Using this option, every signal in the start trigger pool can start the recording if triggered.
- When a time period starts
  - With this option the recording starts as soon as any time period in the same ibaHD time store receives a start trigger.

By default, *Unconditional* is selected.

### 4.2.2 Pre-trigger time

With triggered recording, only data after the trigger occurred is recorded, but sometimes it is also interesting to know what happened before that trigger.

To support this, a *pre-trigger time* (in seconds) can be set.



If a *pre-trigger time* greater than 0 is set, ibaPDA precedes the live data with the buffered data. The saved pre-trigger buffer is continuously overwritten and always contains the latest data.

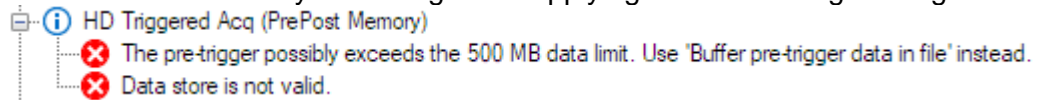
The buffer can either be stored in memory or in a file.



#### - Memory

- With *memory*, the pre-trigger data is buffered in RAM.
- There is a fixed limit of about *500 MB* of data that can be stored in the *memory* pre-trigger. If the selected options (pre-trigger time in combination with the number of selected signals) possibly exceeds that limit, the *file* option has to be used instead.

This is also indicated by a warning when applying the data storage configuration.



#### - File

- In this case the pre-trigger data is stored in a file located in the configured directory.
- The directory will be created if it does not exist.
- The filename will start with the name of the data store.



## 4.3 Stop trigger

For the stop trigger, a *trigger type* and the *post-trigger time* can be set.

If *Unconditional* is selected as the start trigger, all options of the stop trigger are deactivated and no stop trigger is defined. This mimics the behavior of the continuous ibaHD recording.

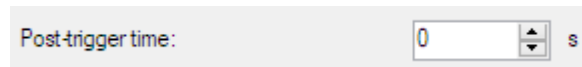
### 4.3.1 Trigger type

Contrary to the *start trigger type*, the *stop trigger type* defines the circumstances under which the recording is stopped.

- After (time)
  - The recording is stopped as soon as the given time is exceeded after starting the recording.
- On Signal
  - Like the *On Signal* option of the *start trigger type*, this option stops the trigger based on the selected signal.
  - For analog signals, a level type (rising edge, falling edge, above level or below level) and the level itself can be selected. Once the value of the signal matches the selected arguments, the recording is stopped.
  - For digital signals the level type and level value are hidden. The trigger will match if the digital value is set to 1.
- On reception of text signal
  - This option stops the recording after the selected signal receives a new text value.
- Use trigger pool
  - Using this option, every signal in the stop trigger pool can stop the recording if triggered.
- When all time periods have stopped
  - The recording stops as soon as all time periods have stopped.

### 4.3.2 Post-trigger time

With a *post-trigger time* greater than 0, the ibaHD server keeps recording after a stop trigger is received until the defined time elapsed.

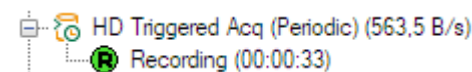


## 4.4 Status

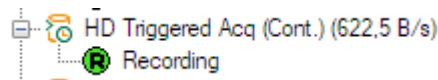
The recording status of each ibaHD time data store can be seen in the data storage status window.

### 4.4.1 Recording

If the status is set to *Recording*, the ibaHD data acquisition is active and is recording data. The time in the brackets indicates how long the recording has been running.



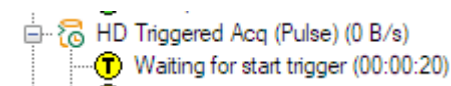
In case of a continuous recording, the time in the brackets is omitted.



### 4.4.2 Waiting for start trigger

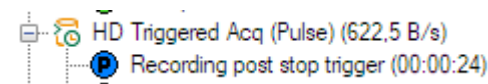
With *Waiting for start trigger*, ibaPDA is waiting for the configured start trigger to occur.

If a pre-trigger time is configured, the value in the brackets displays the already recorded time. Otherwise, the value is omitted.



### 4.4.3 Recording post stop trigger

If a post-trigger is configured, the status changes to *Recording post stop trigger* after the stop trigger occurred. As with the recording, the time in the brackets displays how long the recording is already running – including the post-trigger.



## 5 DataStoreInfoHD diagnostic function

### **This is a breaking change.**

The disconnected status has been removed from the info type 0 (status).

To get the connection status, it is recommended to use info type 2 (Is server connected?) instead.

In order to support all recording states of the triggered ibaHD recording, the DataStoreInfoHD diagnostic function now uses the same values for info type 0 (status) as the other DataStoreInfo functions.

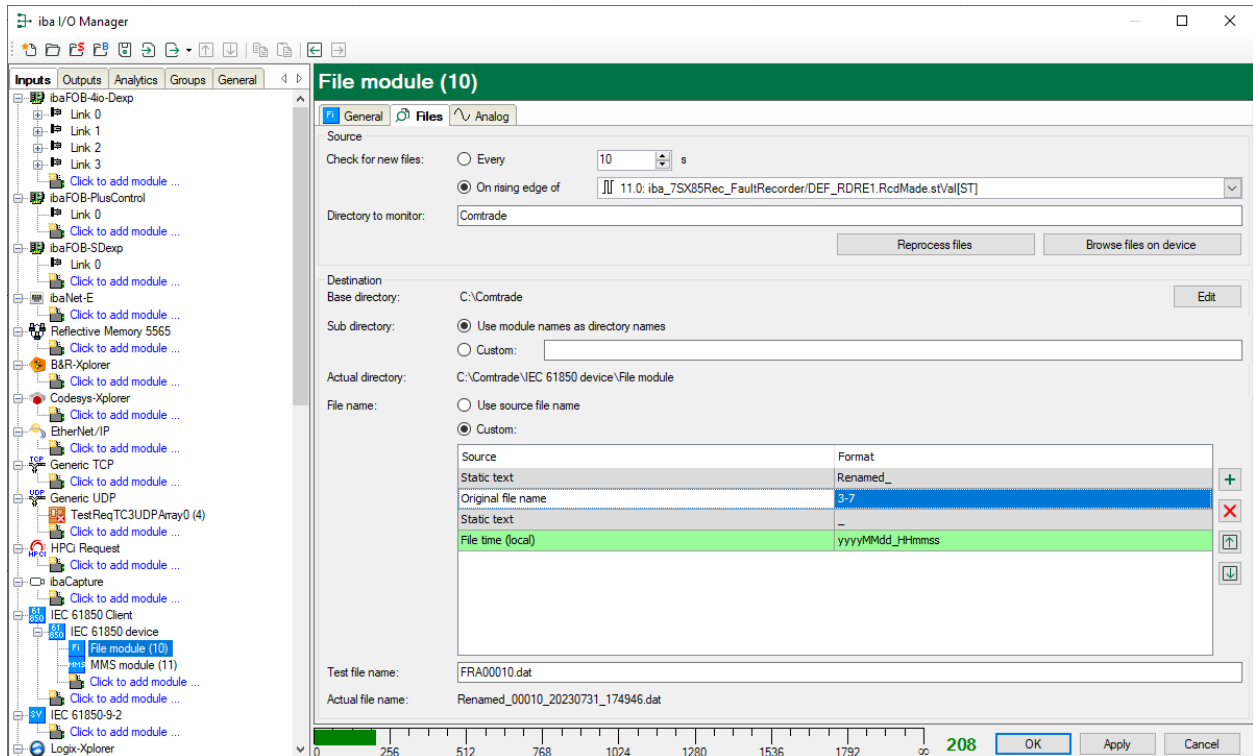
The following values are available:

Status	Meaning
1	Stopped
2	Waiting for trigger
3	Recording
4	Post trigger recording

## 6 IEC 61850-7-2 support

Part 7-2 of the IEC 61850 standard describes the handling of files on IEC 61850 devices. An IEC 61850 device can create Comtrade files when a trigger occurs. IbaPDA is now able to download such Comtrade files, rename them and transfer them to a destination directory.

In IbaPDA you can configure a file module underneath an IEC 61850 device. On the files tab of the file module you can configure which files to monitor and where to transfer them to.



You can use a periodic trigger or use a signal as trigger (e.g. the RcdMade.stVal data attribute of the RDRE logical node). You can configure a directory on the device to monitor or leave it empty to monitor all files. The "Browse files on device" button can be used to browse the current files on the device so you have an idea what is on it when you are configuring the module.

Name	Date modified	Size (KB)
COMFEDE\COMFEDE.ced	31/07/2023 17:51:41	0
COMTRADE\FRA00001.cfg	3/05/2023 12:31:05	0
COMTRADE\FRA00001.dat	3/05/2023 12:31:05	0
COMTRADE\FRA00001.inf	3/05/2023 12:31:05	0
COMTRADE\FRA00001.hdr	3/05/2023 12:31:05	0
COMTRADE\FRA00002.cfg	3/05/2023 12:48:41	0
COMTRADE\FRA00002.dat	3/05/2023 12:48:41	0
COMTRADE\FRA00002.inf	3/05/2023 12:48:41	0
COMTRADE\FRA00002.hdr	3/05/2023 12:48:41	0
COMTRADE\FRA00003.cfg	3/05/2023 12:49:41	0
COMTRADE\FRA00003.dat	3/05/2023 12:49:41	0
COMTRADE\FRA00003.inf	3/05/2023 12:49:41	0
COMTRADE\FRA00003.hdr	3/05/2023 12:49:41	0
COMTRADE\FRA00004.cfg	12/05/2023 13:01:42	0
COMTRADE\FRA00004.dat	12/05/2023 13:01:42	0
COMTRADE\FRA00004.inf	12/05/2023 13:01:42	0
COMTRADE\FRA00004.hdr	12/05/2023 13:01:42	0
COMTRADE\FRA00005.cfg	12/05/2023 13:02:12	0
COMTRADE\FRA00005.dat	12/05/2023 13:02:12	0

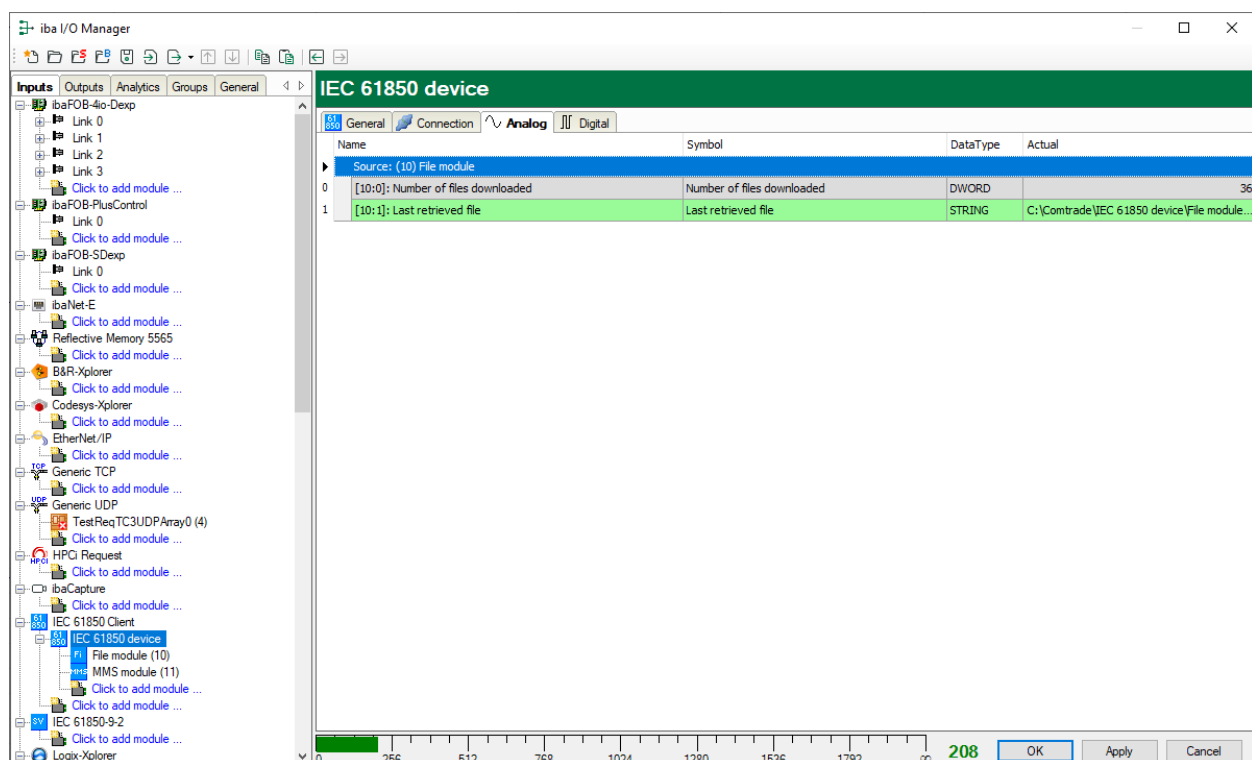
The "Reprocess files" button can be used to process all files on the device again because you maybe changed some destination settings. You can press this button when the acquisition is running.

In the destination section you configure to which directory the files should be transferred and how the files should be named. The destination directory consists of 2 parts:

- The base directory configured on the IEC 61850 Client interface. You can use the "Edit" button to go to the IEC 61850 Client interface.
- The sub directory that can be derived from the module names or that can be entered manually. If you enter a full path in custom (e.g. C:\Test) instead of a relative one (e.g. Test) then the absolute path will be taken and the base directory will be ignored.

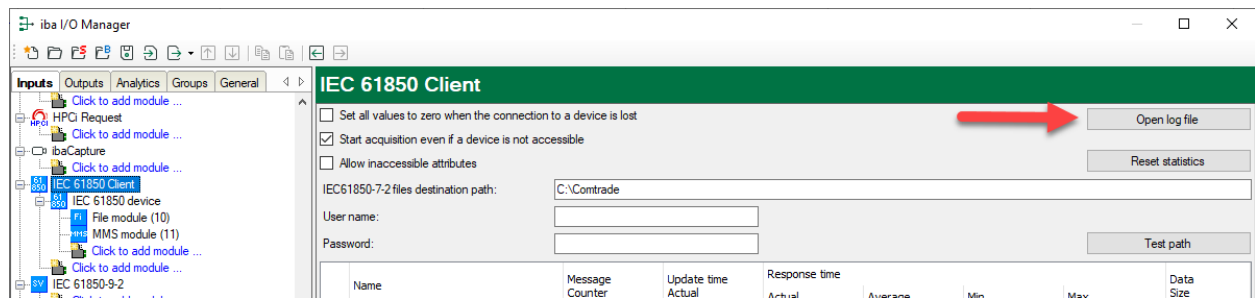
The actual directory shows the result of what you have configured.

The file name can use the source file name or it can be renamed using parts of the original file name, static parts and timestamp parts. These parts you can configure in the grid. The buttons next to the grid can be used to add and remove parts and to change their order. At the bottom you can enter a test file name and the actual file name will show the result of the renaming.



The module has 2 diagnostic signals:

- Number of files downloaded: this is a 32 bit integer counter that is incremented for each file that gets downloaded.
- Last retrieved file: This is the full file name of the last file that was saved in the destination.

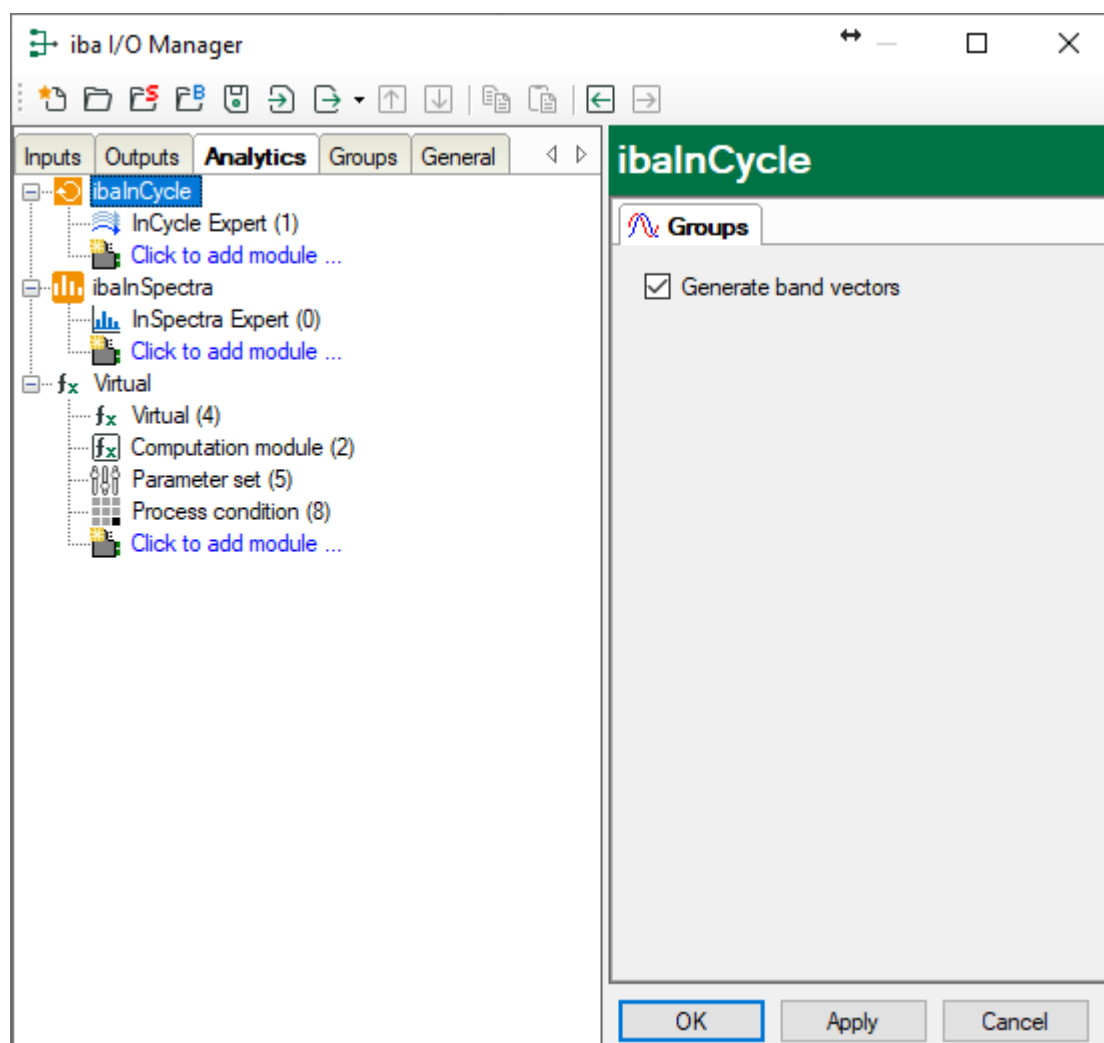


On the IEC 61850 Client interface you can open the log file. The log file will contain entries for each file that is transferred and for any error that might have occurred.

## 7 Analytics tab

In the I/O manager, the following interfaces have been moved to a new tab which is named *Analytics*:

- InCycle
- InSpectra
- Virtual



## 8 Support for machine learning

A machine learning model is an algorithm that can find patterns or make decisions from a previously unseen dataset. The model transforms a predefined number of inputs into a predefined number of outputs.

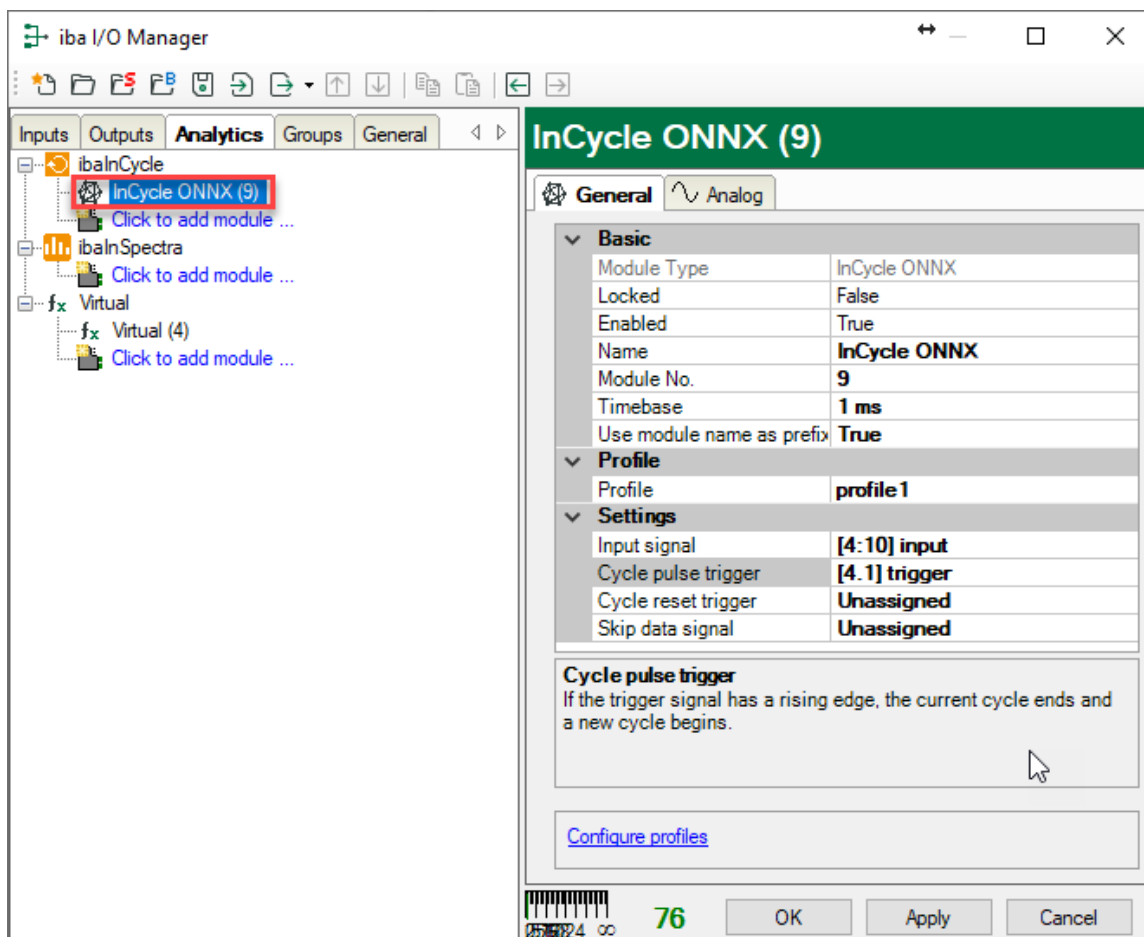
ONNX is an open format built to represent such machine learning models. These models are saved in an .onnx file.

This version has two new modules that support the use of ONNX models:

- InCycle ONNX
- InSpectra ONNX.

### 8.1 InCycle ONNX

The InCycle ONNX module is very similar to the existing InCycle Expert module:



The *General* tab of both modules is the same and both modules allow:

- Sampling a single signal using time synchronous averaging
- Sampling a vector of signals at one specific moment in time

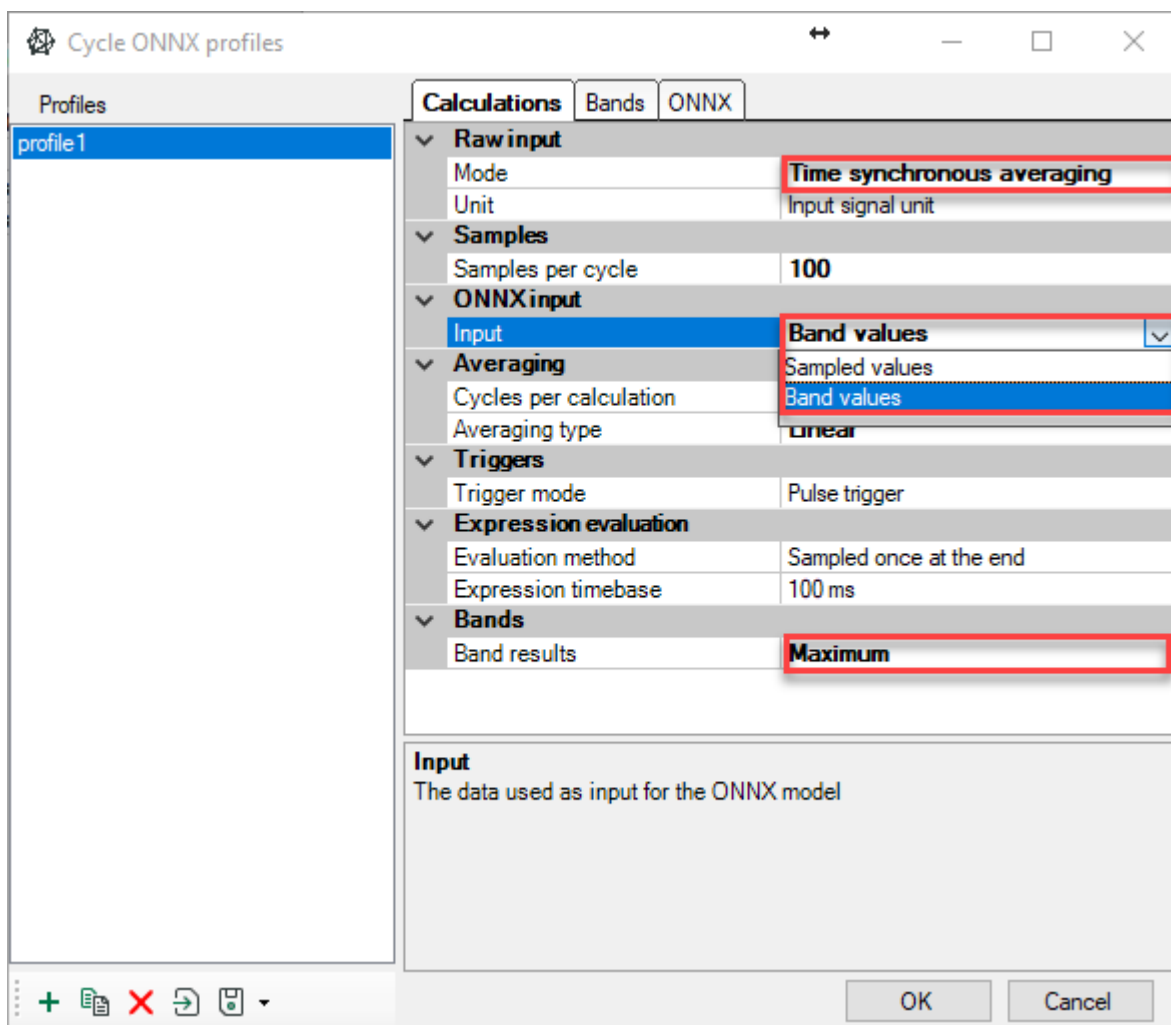


In both cases the result is a data block, i.e. a fixed number of samples. This data block can be used as input for the configured ONNX model. Each time there is a new data block, the ibaPDA server calculates the output of the model and the results are saved in the signals of the module.

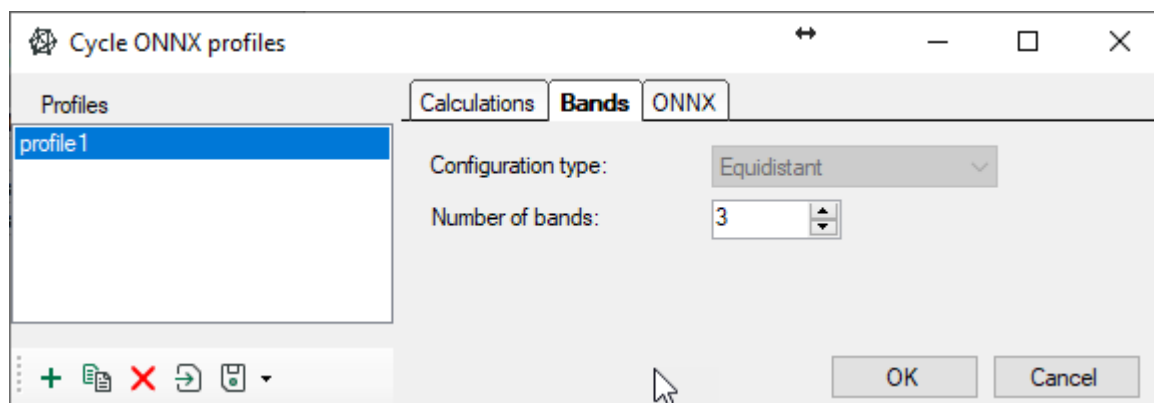
In the profile, two ways to feed the inputs of the ONNX model can be configured:

- Sampled values: Directly use the samples from the data block
- Band values: Combine the samples into a number of equidistant bands using one of the following calculation methods:
  - Minimum
  - Maximum
  - Average
  - RMS
  - Standard deviation
  - Range
  - Change
  - ...

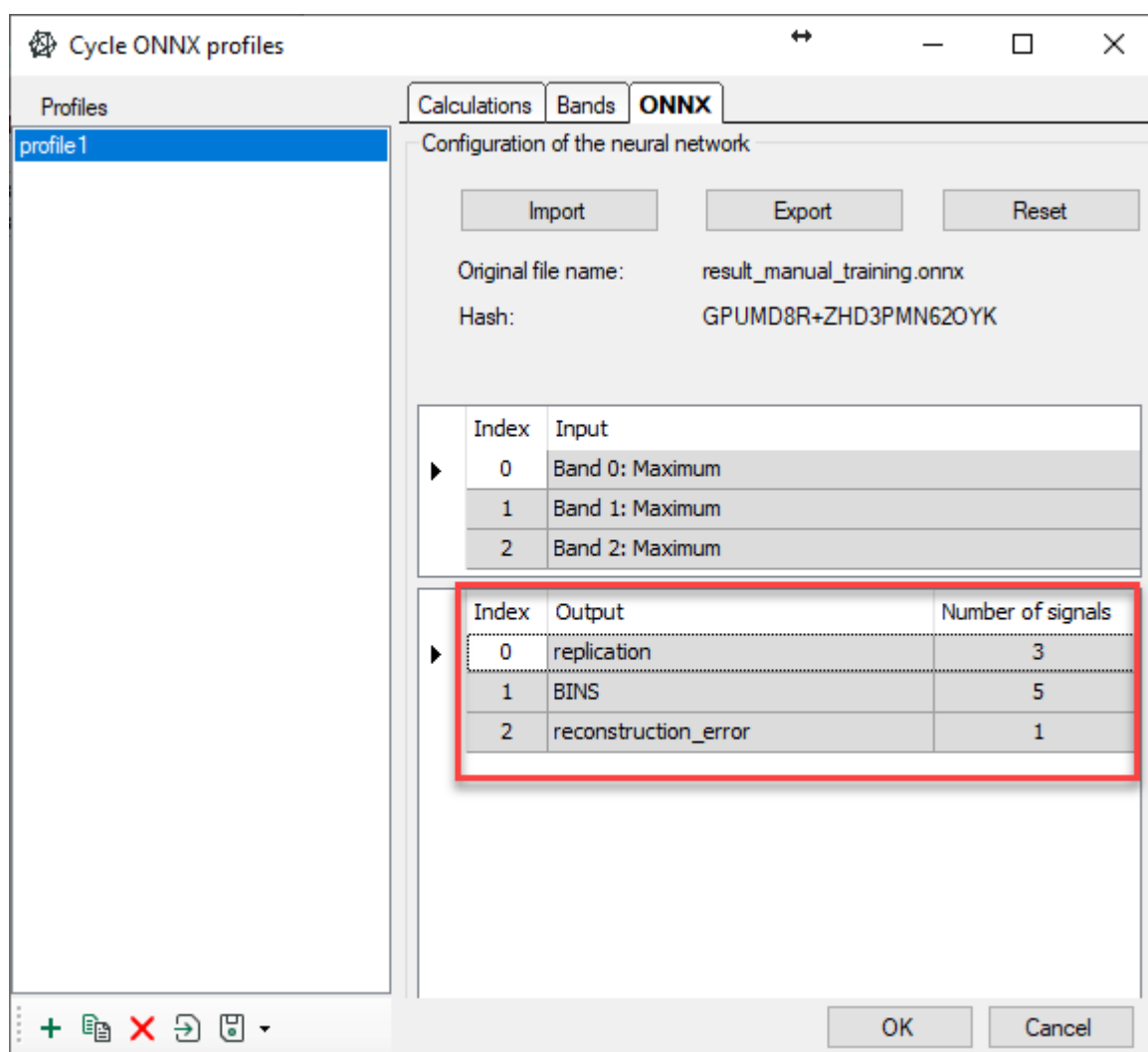
Note it is also possible to combine multiple band results by enabling multiple band results.



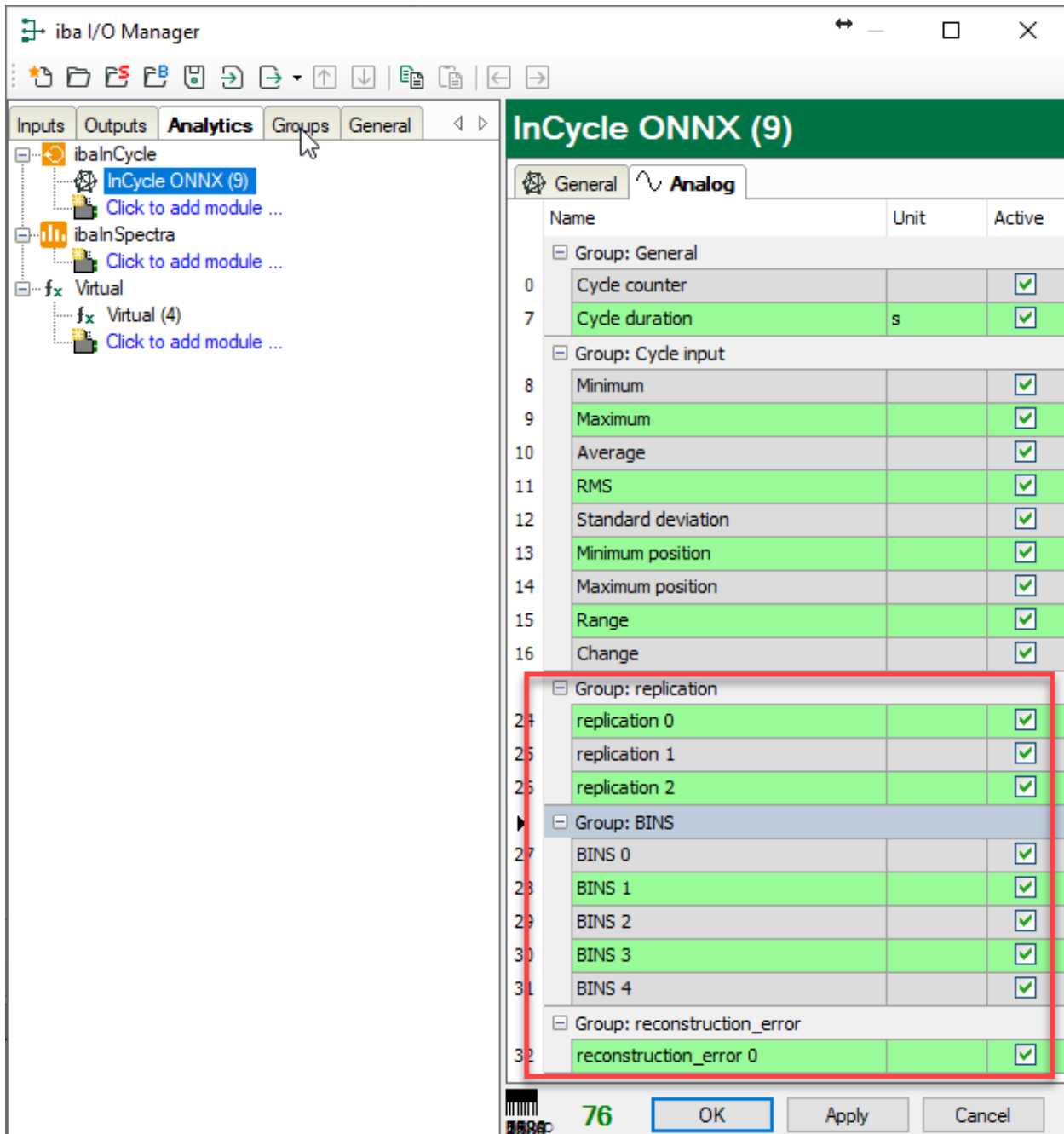
In case one is using band values, the number of bands must be configured in the *Bands* tab:



In the *ONNX* tab of the profile dialog, you can import any ONNX file. The content of the ONNX file is saved inside the profile.

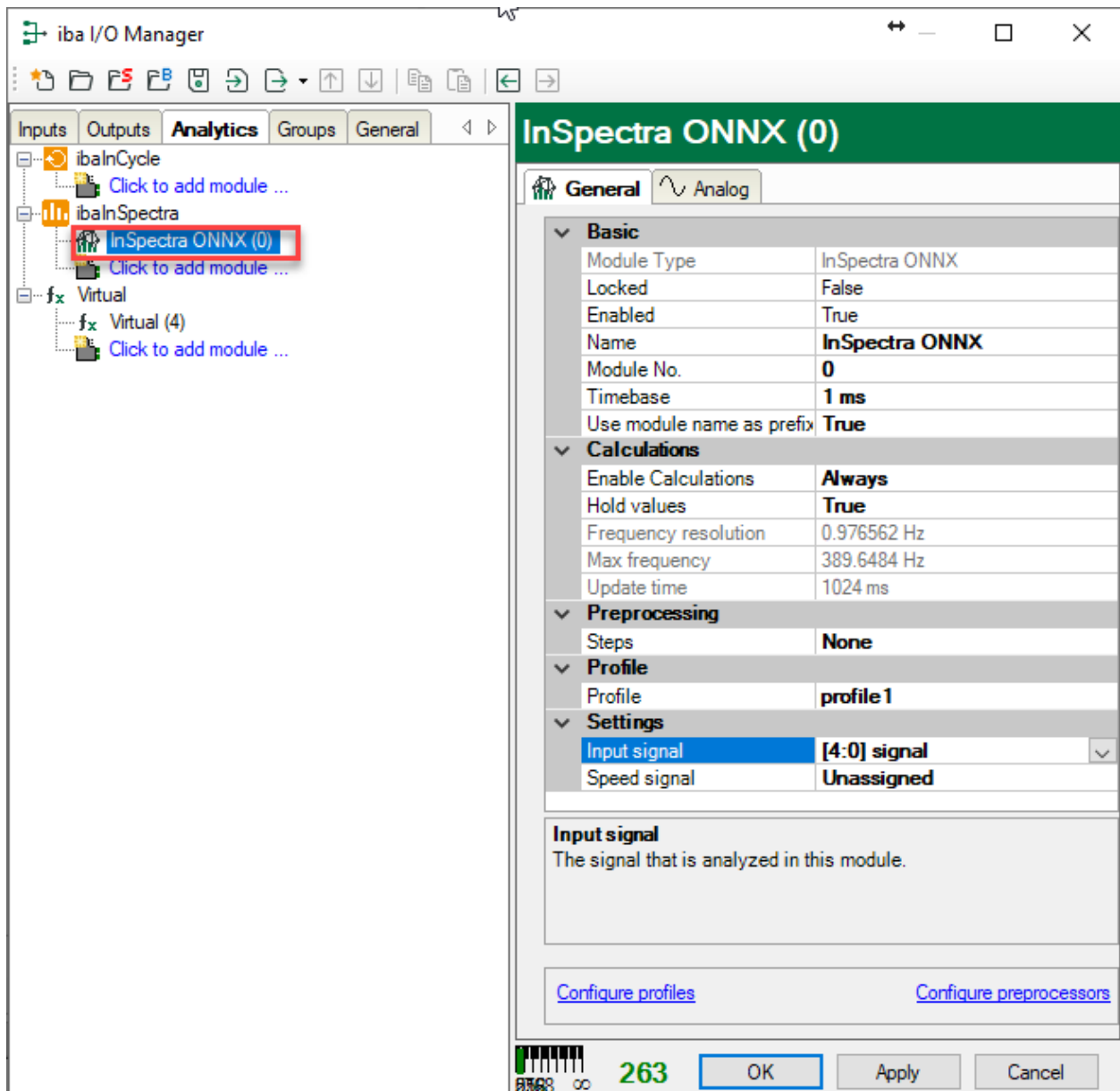


The results of the ONNX model can be measured as signals:



## 8.2 InSpectra ONNX

The InSpectra ONNX module is very similar to the existing InSpectra Expert module:



In the profile, two ways to feed the inputs of the ONNX model can be configured:

- FFT values: The values of the calculated spectrum
- Band values: One of the following band calculation methods:
  - Peak
  - Peak frequency
  - RMS
  - Peak phase
  - Crest

Note it is also possible to combine multiple band results by configuring *Band function 2* and *Band function 3*.

**Spectrum ONNX profiles**

Profiles: profile 1

**Calculations** | Bands | Placeholders | ONNX

**ONNX input**

Input: **Band and characteristic values**

**Sensor units**

Sensor type: FFT values

Sensor unit: **Band and characteristic values**

**Spectrum units**

Spectrum type: No integration

Multiplication factor: 1

Spectrum unit: Input signal unit

**Frequency unit**

Frequency unit: Hz

**Speed**

Speed type: Direct speed

Speed unit: Speed signal unit

Smoothing interval: 0 ms

**Order**

Enable order: False

**Acquisition**

Number of samples: 1024

Number of lines: 400

Overlap percentage: 0 %

Skip spectrum calculation: 0

**Calculation**

Suppress DC: False

Detrend raw data: False

Window type: Rectangular

Normalized: False

Spectrum method: Magnitude

RMS method: True RMS

**Bands**

Band function 1: **RMS**

Band function 2: None

Band function 3: None

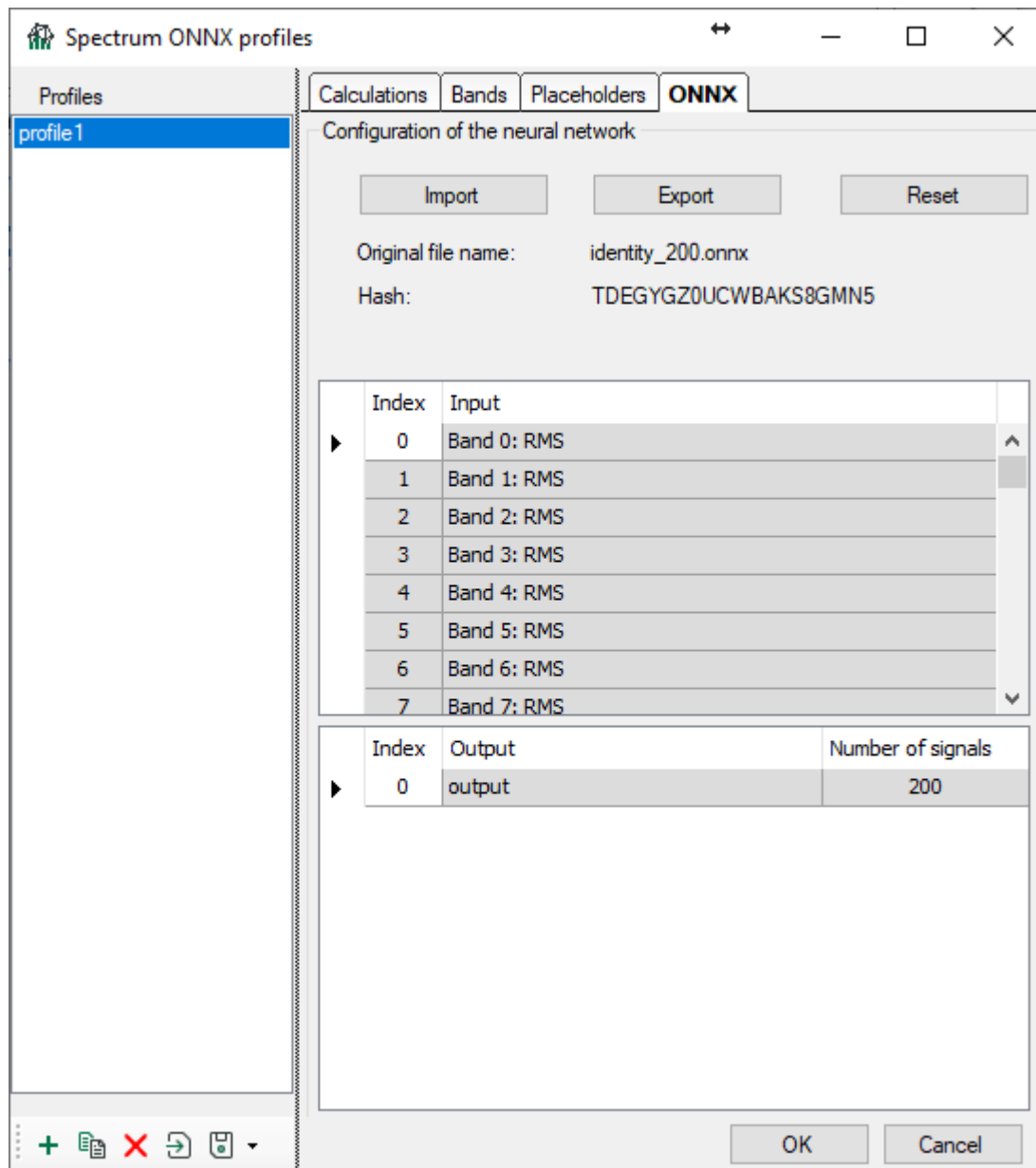
**Averaging**

**Input**

The data used as input for the ONNX model

OK Cancel

The ONNX file must be imported into the profile. The model is fully saved in the profile.



The results of the ONNX model can be measured as signals.