



ibaPDA v8.0.0

New Features

22.06.2022

iba AG

Table of contents

1	General remarks.....	3
1.1	Supported Windows Operating Systems.....	3
1.2	End of support for ibaCapture-CAM v1.x and v2.x	3
1.3	Migrating from ibaPDA v7 to ibaPDA v8	3
2	WIBU CodeMeter licensing	5
3	New layout manager	7
3.1	Overview.....	7
3.2	Layout manager.....	9
3.3	Toolbar.....	10
3.4	General Settings	14
3.5	Display Style	17
3.6	Toolbars and menus	18
3.7	Layouts	20
3.8	Startup layout.....	27
3.9	User assignment.....	29
3.10	Layout conversion from ibaPDA v6 and v7 to ibaPDA v8	34
3.11	Overview of previous layout management	42
4	I/O Manager	45
4.1	Tab structure.....	46
4.2	Module overview.....	47
4.3	Navigation buttons	48
4.4	Module folders	49
5	NMEA 0183 decoder module	54
6	Bit decoder modules	58
7	Extended ABB-Xplorer Interface	59
7.1	Using direct access.....	59
7.2	Using outputs.....	61
8	MQTT interface outputs.....	63
9	MQTT timebased data store	64
9.1	Buffered writing.....	64
9.2	Protocol Buffers Encoding.....	65
9.3	Separator for text formats	66
10	InfluxDB timebased data store	67
10.1	Licensing	67
10.2	Preparation InfluxDB server.....	67
10.3	Configuration in ibaPDA.....	67
10.4	Example data models	72
10.5	Diagnostics	75

11 ibalnCyle and ibalNSpectra79

 11.1 Vector mode for ibalnCyle.....79

 11.2 Auto-Adapting modules: endless learning and exponential learning80

12 ibaQDR optional measuring locations82

1 General remarks

1.1 Supported Windows Operating Systems

For **Windows Server 2008 R2 (x64)** and **Windows 7 (x86/x64)** the Extended Support period ended on Jan 14, 2020. No security updates are provided by Microsoft anymore since then. ibaPDA v8.0.0 and higher no longer support these two operating systems.

The following operating systems are currently supported:

- Windows 8.1 (x86/x64)
- Windows 10 (x86/x64)
- Windows 11 (x64)
- Windows Server 2012 (x64)
- Windows Server 2012 R2 (x64)
- Windows Server 2016 (x64)
- Windows Server 2019 (x64)
- Windows Server 2022 (x64)

1.2 End of support for ibaCapture-CAM v1.x and v2.x

ibaPDA v8.0.0 and higher do not support old ibaCapture-CAM systems with versions v1.x and v2.x anymore. You need to migrate the old ibaCapture-CAM system to an up-to-date ibaCapture system.

Please contact your regional iba representative for information and assistance on this. You will find your regional iba representative listed on our website (<https://www.iba-ag.com/en/contact/>).

1.3 Migrating from ibaPDA v7 to ibaPDA v8

License ibaPDA v7 vs. ibaPDA v8

- No new license is required to use ibaPDA v8 but a valid EUP date is mandatory. If you do not have a maintenance contract and require an EUP date extension please contact your regional iba representative for this.

You will find your regional iba representative listed on our website (<https://www.iba-ag.com/en/contact/>).

- MARX dongles can still be used. New systems are only delivered with WIBU license. The WIBU license by default is provided on a USB dongle but can also be stored in a software container.
- Customers who are interested in migrating from a MARX USB dongle to a WIBU soft-license can do this for a service fee. Please contact your regional iba representative for this.

You will find your regional iba representative listed on our website (<https://www.iba-ag.com/en/contact/>).

- ibaPDA license products are renamed: The version code "V7" has been removed from the product names resulting in neutral version independent license products.

- For more details on WIBU licensing please refer to chapter 2 WIBU CodeMeter licensing.

Project impacts opening v7 projects with v8

- Configuration data (io & data store configuration, address books, log files, certificates etc.) in ibaPDA v8 is no longer stored in the ibaPDA server installation directory. This data is now stored in the Windows ProgramData directory.
- When migrating from v7 to v8 the configuration data will be automatically moved from the old location to the new location and a backup of the original data will be created.
- In case of a manual deinstallation of ibaPDA v8 the user can decide if he wants to restore the backed up v7 configuration data or not.
- A major change in ibaPDA v8 is the way layouts are managed and stored. Please carefully review chapter 3 and more section 3.10 below for details on this topic. This is especially important for systems where the user management is used and layouts are stored on the server for various users.

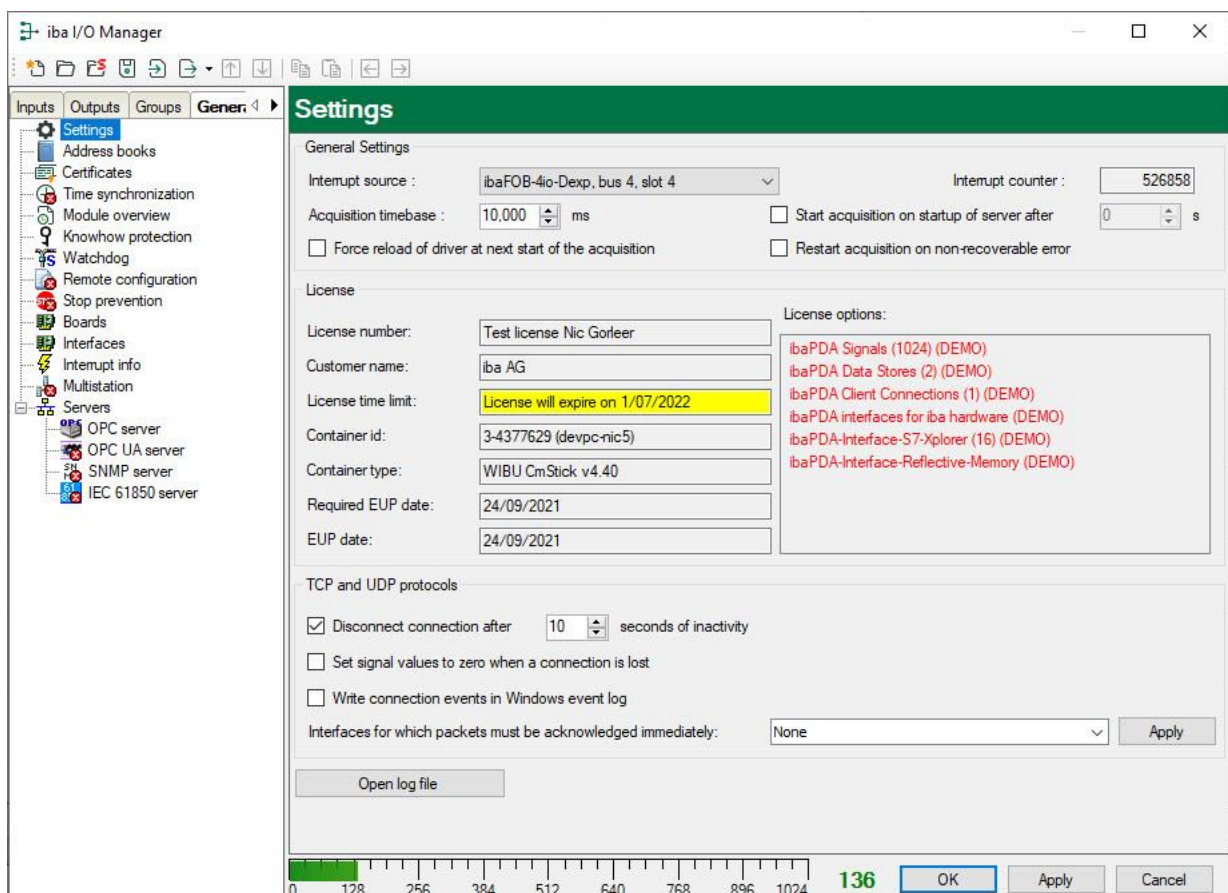
2 WIBU CodeMeter licensing

Up to now ibaPDA licenses were stored in dongles from the company MARX. In ibaPDA v8 the licenses can additionally be stored in license containers from the company WIBU. There are 2 types of license containers supported:

- CmStick: This is a USB dongle.
- CmActLicense: This is a so-called “soft container”. It is a file on disk that is coupled to some system IDs.

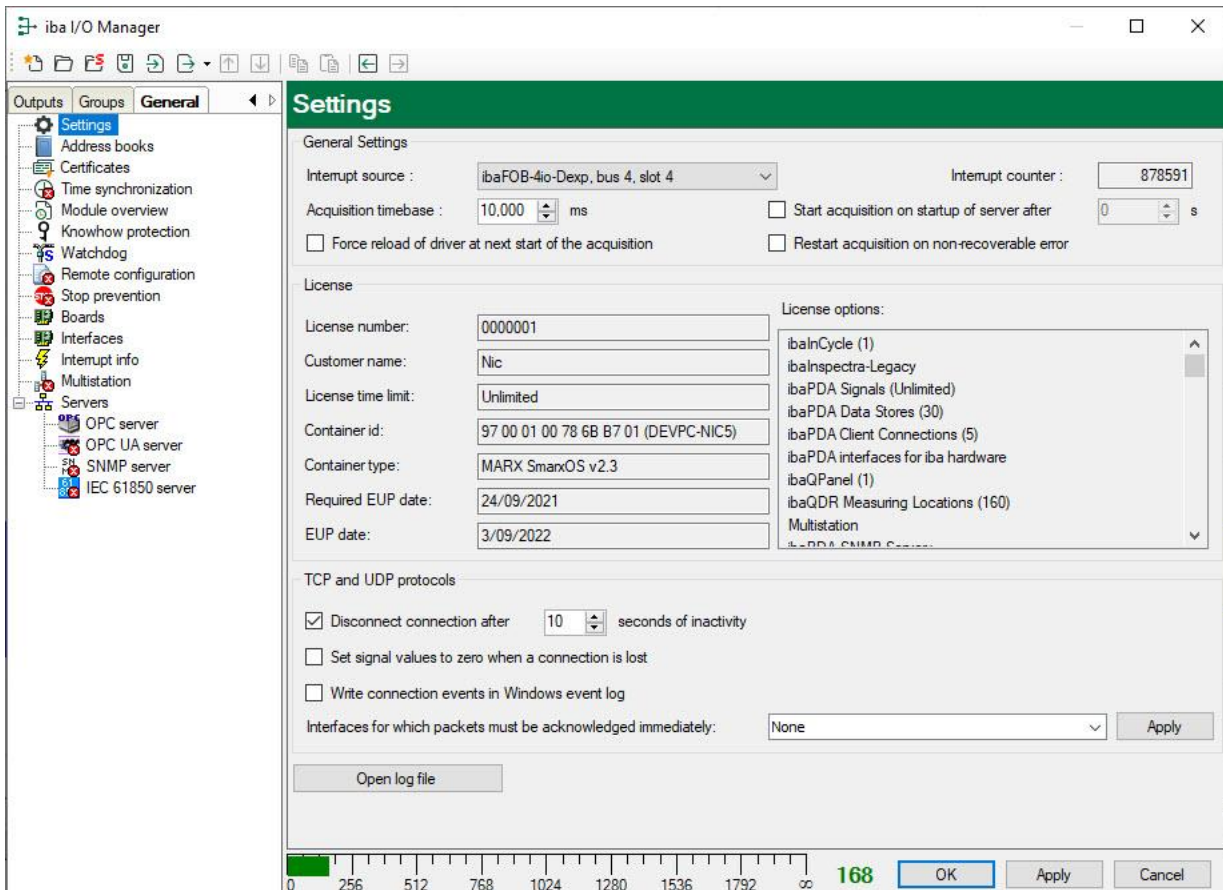
In order to use WIBU licenses the CodeMeter runtime needs to be installed. This is normally automatically done during the installation of ibaPDA.

On the Settings node in the General tab of the I/O Manager you can get information about the used licenses.



The screenshot above shows a WIBU USB dongle with container ID 3-4377629. It is attached to PC devpc-nic5. WIBU licenses can be shared across the local network. A separate manual is available that explains how to share WIBU licenses.

The license options give a list of all available licenses on the dongle. The number between brackets corresponds to the number of available licenses, e.g. ibaPDA Signals (1024) means that you have a license for 1024 signals. Demo licenses are marked in red. In case of a demo license the license time limit tells you when the license will expire. WIBU demo licenses have a fixed expiry date. MARX demo licenses have a number of usage days.



The screenshot above shows a MARX USB dongle.



The about dialog also shows information about the used license.

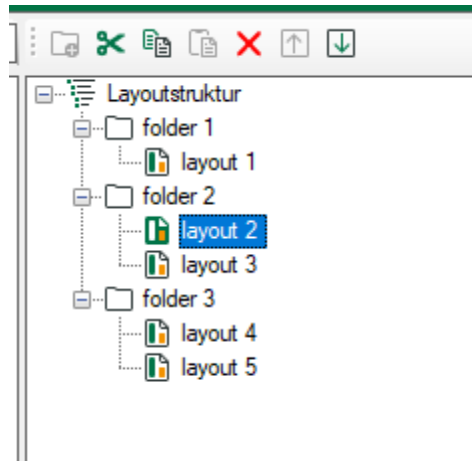
3 New layout manager

3.1 Overview

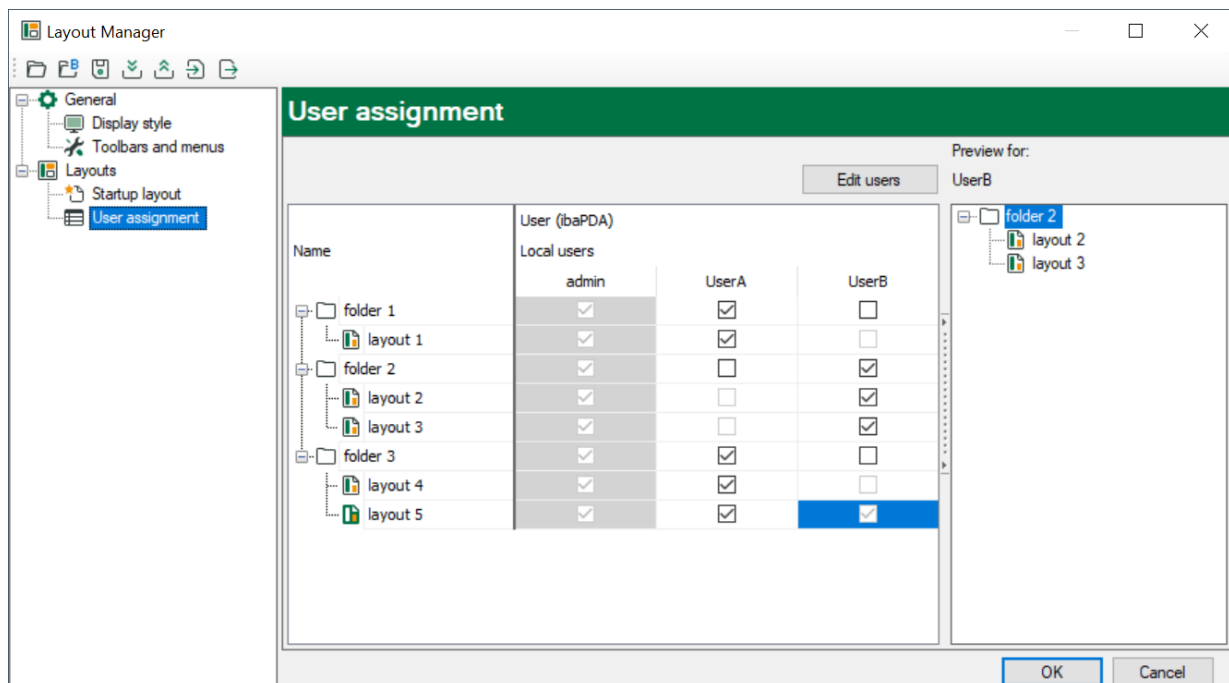
The layout manager was revised to meet increased requirements. The most important change is that there is now only one layout set. Layouts can be assigned to users so each user is able to see only a subset of layouts.

Here is a list of the biggest changes in the new layout manager. These changes are described in detail in the next chapters.

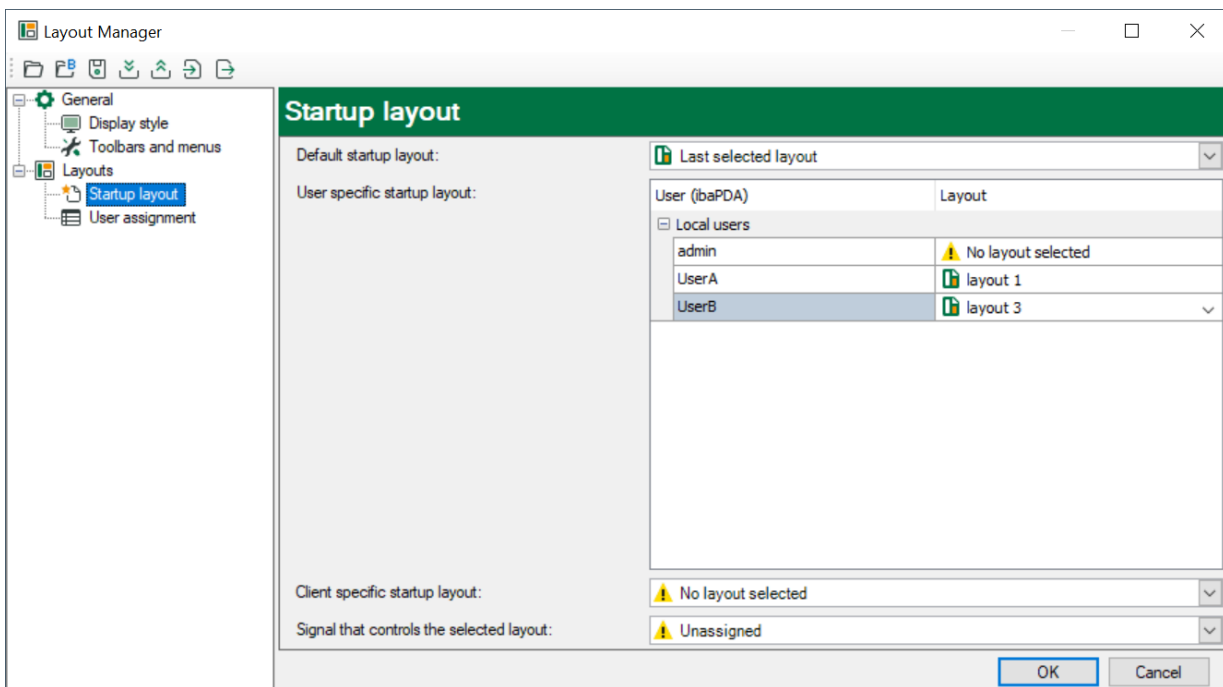
Structure layouts in folders



Assignment of layouts to users

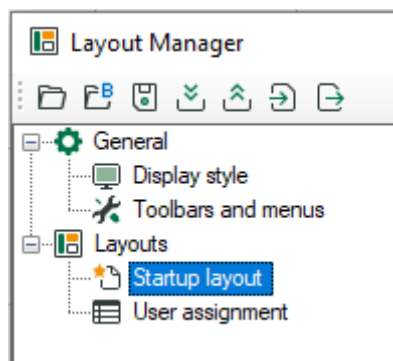


Define startup layouts for users and clients



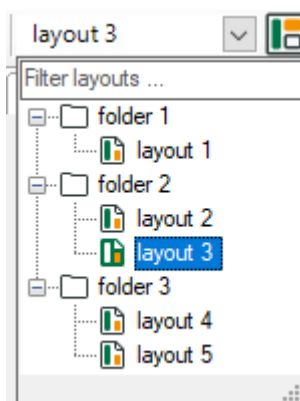
All layout related settings in one place

All layout related settings are now accessible in one place, the new layout manager.



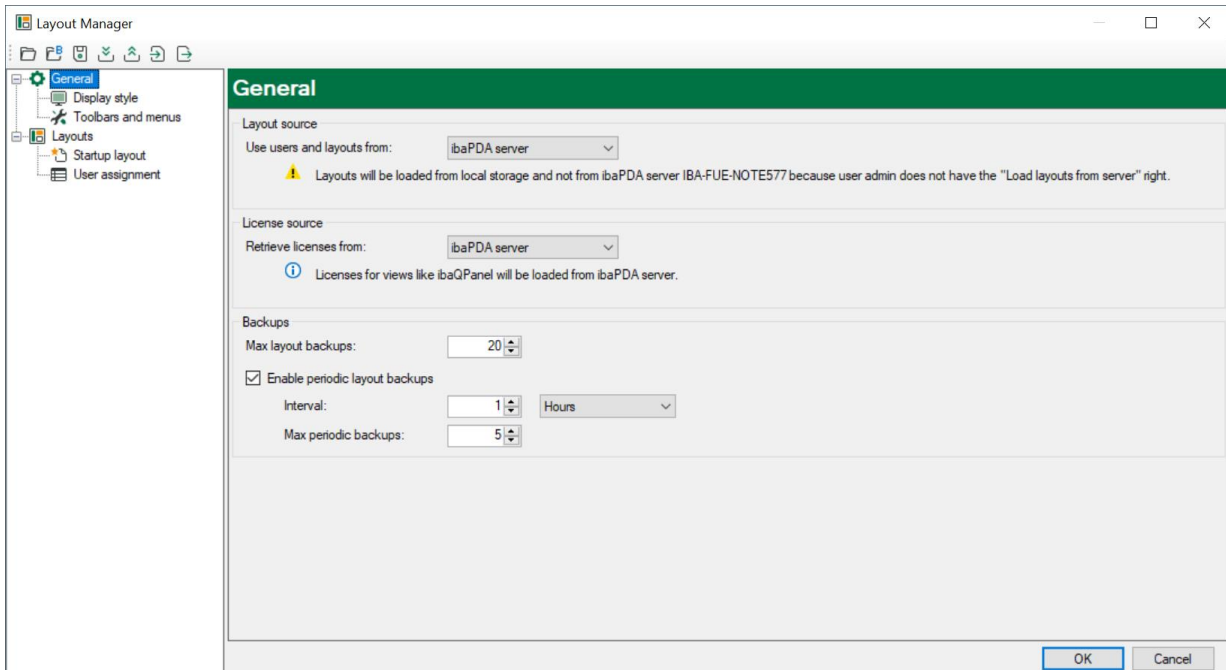
New layout selection drop-down field

Due to the innovations, a user now only sees his assigned layouts in the ibaPDA layout drop-down field and these can also be structured in folders.



3.2 Layout manager

The layout manager can be opened by the icon  on the toolbar, by the “*Layout Manager*” entry in the configuration menu and by the shortcut “Ctrl+L”.



All layout related settings are accessible in this one form. This form manages a layout set. A layout set means a whole .layouts file that includes a number of layouts and the layout configuration. In previous ibaPDA versions a layout set was saved in a .lay file.

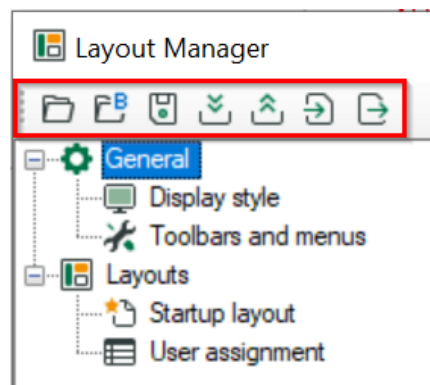
The layout manager form works on a copy of the current layout set. So if you leave with Cancel then nothing will have changed. Only when you leave the form with OK the layout set will be applied and it will be automatically saved at %localappdata%\iba\ibaPDA\CurrentLayouts.layouts.

The complete state of the layout manager form is saved when it is closed and it will be restored when it is opened again. The state consists of the form's position and size, the expanded/collapsed state of the tree nodes on the left side and the last selected node.

The next chapters describe the different parts of the form.


3.3 Toolbar

In the top toolbar you will find all the functions for saving and loading layout sets. These were previously found in the View menu in ibaPDA.




Note: You can open v7 layouts or v8 layouts. v7 layouts are converted automatically. That means .lay and .layouts files can be read in.

3.3.1 Load layouts from disk

With the button  a layout set can be loaded from the file system. You can load a layout set from a new .layouts file or an old .lay file. It is only applied when you close the form with the OK button.


3.3.2 Load layouts from backup directory

The button  opens the folder directory on the file system, where the backups are stored. Here the user can select one of the automatic or periodic (P_) backups.


» Dieser PC » Windows (C:) » Benutzer » wplass » AppData » Local » iba » ibaPDA » backup

Neuer Ordner					
	Name	Änderungsdatum	Typ	Größe	
ostics					
loaded Installations					
io-updater					
tedDiagnostics					
ant					
\$					
thing					
onfig					
LER					
o.NET					
le					
	Currentlayout_2022_03_14_12_12_49.lay	14.03.2022 12:12	LAY-Datei	25 KB	
	Currentlayout_2022_03_14_12_15_31.lay	14.03.2022 12:15	LAY-Datei	29 KB	
	Currentlayout_2022_03_14_14_13_26.lay	14.03.2022 14:13	LAY-Datei	40 KB	
	Currentlayout_2022_03_14_16_04_10.lay	14.03.2022 16:04	LAY-Datei	46 KB	
	Currentlayout_2022_03_15_17_15_09.lay	15.03.2022 17:15	LAY-Datei	53 KB	
	Currentlayout_2022_03_16_07_30_23.lay	16.03.2022 07:30	LAY-Datei	53 KB	
	CurrentLayouts_2022_03_11_11_24_13.lay...	11.03.2022 11:24	LAYOUTS-Datei	5 KB	
	CurrentLayouts_2022_03_11_14_59_04.lay...	11.03.2022 14:59	LAYOUTS-Datei	6.240 KB	
	CurrentLayouts_2022_03_14_07_43_05.lay...	14.03.2022 07:43	LAYOUTS-Datei	6.240 KB	
	CurrentLayouts_2022_03_14_11_22_16.lay...	14.03.2022 11:22	LAYOUTS-Datei	6.273 KB	
	CurrentLayouts_2022_03_16_08_15_32.lay...	16.03.2022 08:15	LAYOUTS-Datei	6.247 KB	

3.3.3 Save layouts to disk


The button  opens a dialog to save the current layout set on the file system.

3.3.4 Load layouts from server

With the button  the user can download the layouts from the corresponding server. This is only allowed if you have the right "Load layouts form server" in your user rights or you are the admin user. The layout set will be fetched from the configured server (General node), the existing layout set will be overwritten.

Note: In ibaPDA versions older than v8, the user specific layouts were also stored on the server but as a complete layout set per user (see chapter 3.10 Layout conversion from ibaPDA v6 and v7 to ibaPDA v8).


3.3.5 Save layouts on server

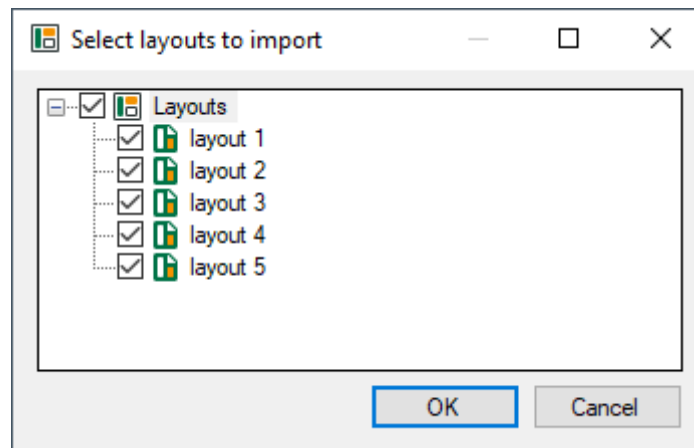
With the button  the user can save the current layout set on the corresponding server. This is only allowed if you have the right "Save layout on server" in your user rights or you are the admin user. The layout set will be saved on the configured server (General node), the existing layout set on the server will be overwritten.

Only the entire set of layouts is stored. Individual layouts cannot be stored on the server.

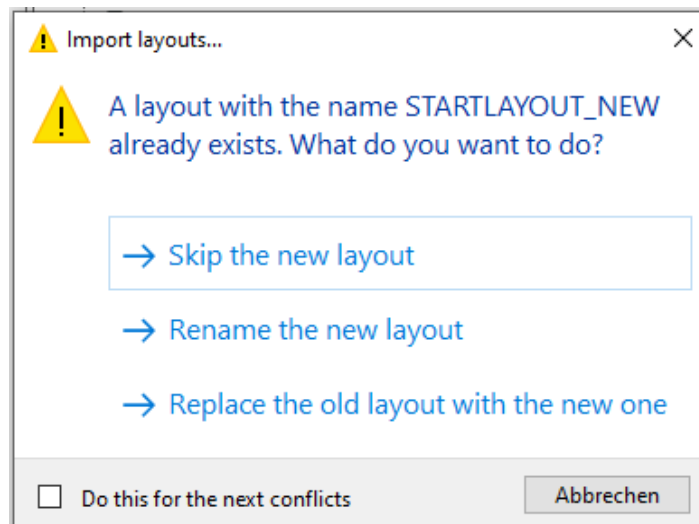
If there are logged in users in other ibaPDA clients with the right to "Load layouts from server", they will be informed that there is a new layout set available. They can then choose to load the layouts from the server.

3.3.6 Import layouts from disk

With the button  the user is able to import layouts from the file system. Layouts can be imported, i.e. added to the existing layout set. These can be old layout files (.lay) or new layout files (.layout, .layouts). The layouts to be imported can be selected from this layout collection via a dialog.




If a layout already exists with an equal name, a dialog is shown to resolve this import conflict.

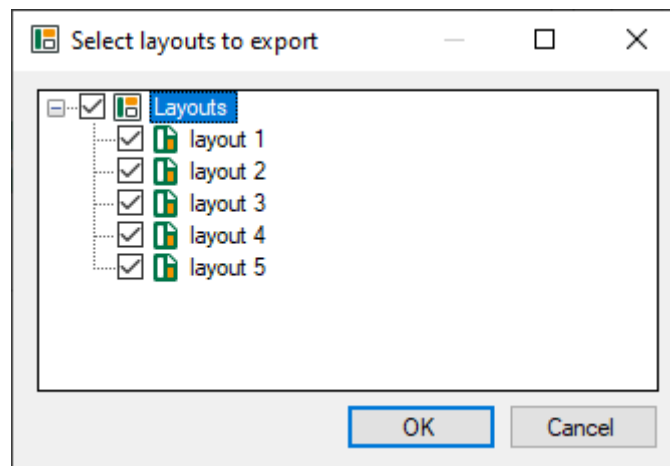


There are 3 options to choose from:

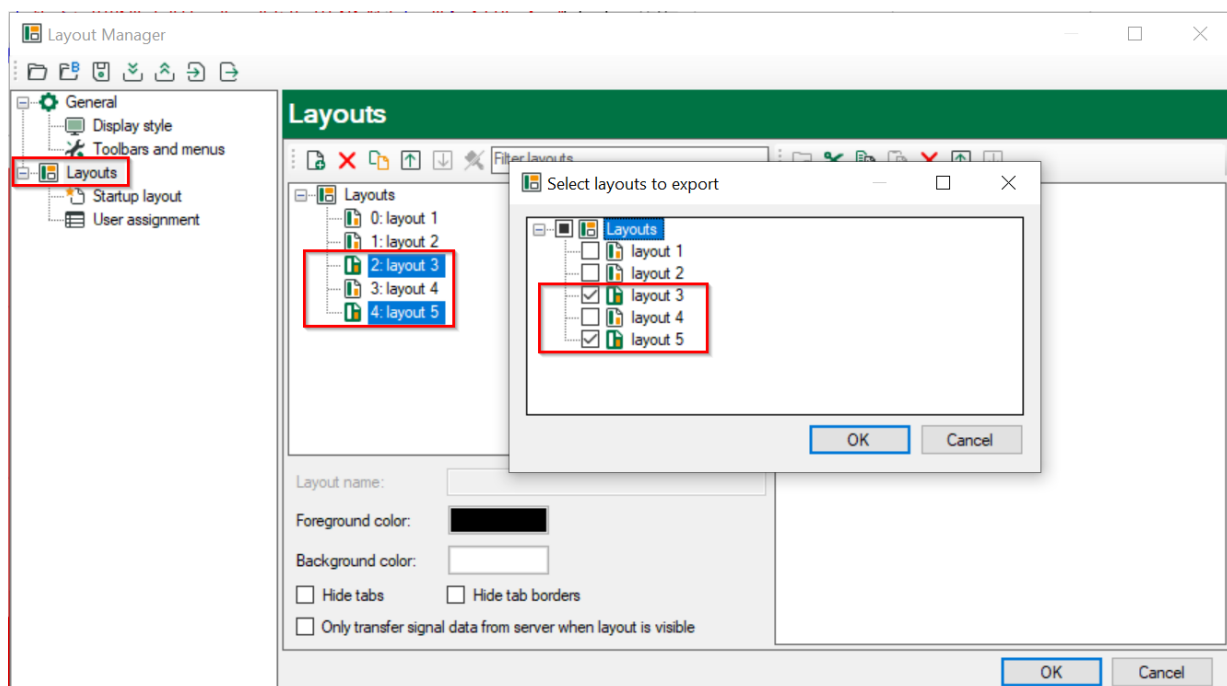
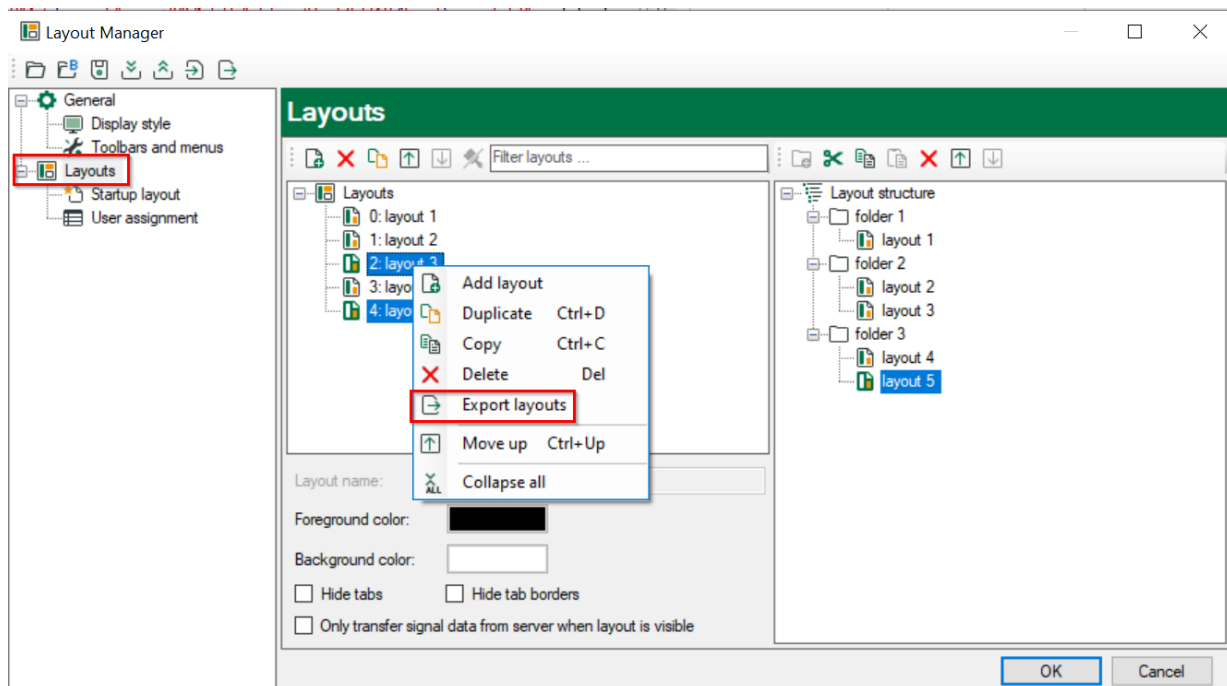
1. Skip the new layout: The layout will not be imported. The original layout with the same name will remain as is.
2. Rename the new layout: The new layout will be imported and a „-copy“ suffix will be added to its name.
3. Replace the old layout with the new one: The new layout will be imported and the original one will be removed.

3.3.7 Export layouts to disk

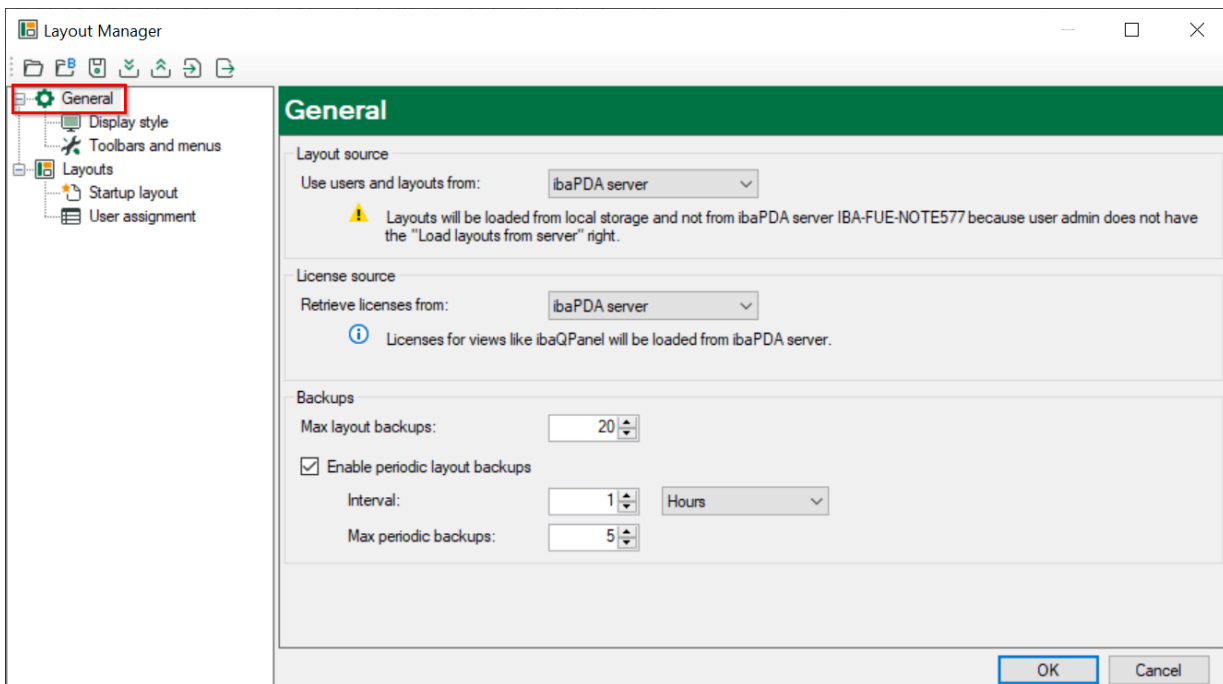
With the button  the user can export a number of layouts to the file system. A dialog for selecting the desired layouts appears.



Note: If you want to export already selected layouts, use the context menu in the layouts view then the selected layouts are automatically marked.

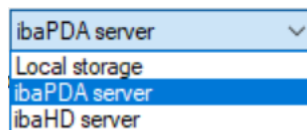


3.4 General Settings




3.4.1 Layout source


You can select here where the ibaPDA client, you are currently working in, loads and saves its layouts.



- Local storage: Layouts are only stored locally on the client
- ibaPDA server: ibaPDA users are used and layouts are stored on the currently connected ibaPDA server
- ibaHD server: ibaHD users are used and layouts are stored on the currently connected ibaHD server

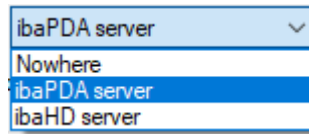
This setting is important in relation to the “Save layouts on server” and “Load layouts from server” rights in the user management.

When you have the “Save layouts on server” right and the layouts source is set to that server type then you are allowed to manually save the layouts on the server via the  button. When you close the client then ibaPDA will also ask you if you want to save the layouts on the server.

When you have the “Load layouts from server” right and the layouts source is set to that server type then ibaPDA will automatically load the layouts from the server when you connect to the server. You can also later on manually load the layouts from the server via the  button.

3.4.2 License source

Here you can specify from where the ibaQPanel license is fetched.



- Nowhere: no ibaQPanel license is fetched for this client
- ibaPDA server: License is fetched from the connected ibaPDA server
- ibaHD server: License is fetched from the connected ibaHD server

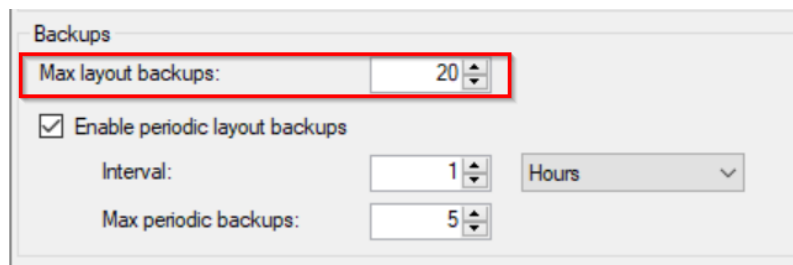
3.4.3 Backups

There are two kinds of backups.

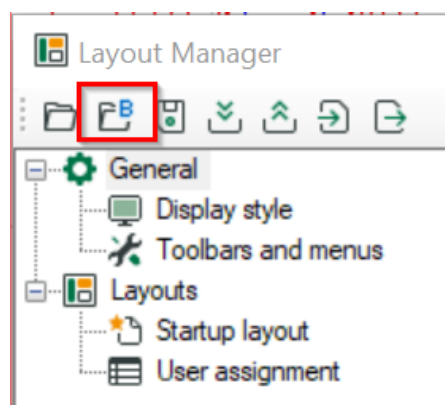
Automatic backups

These backups are automatically generated by the ibaPDA client when:

- the ibaPDA client is closed
- a project is opened
- a support file is created
- the layout manager is closed and changes took place
- layouts are saved on the server
- a connection to an ibaPDA server is established
- a connection to an ibaHD server is established



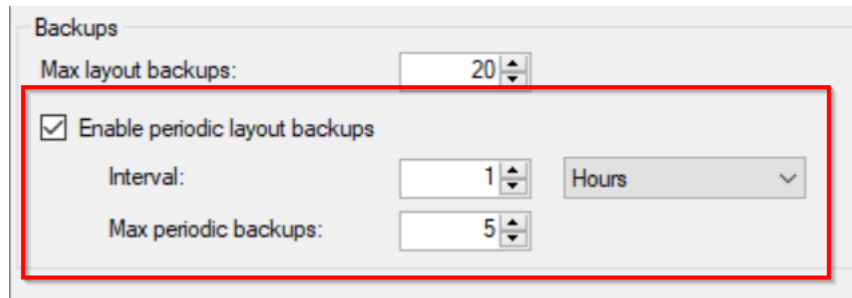
Maximum layout backups: Can be set from 5 to 100 (default value is 20). The backups are stored in the backup folder and are accessible via the toolbar.



Periodic backups

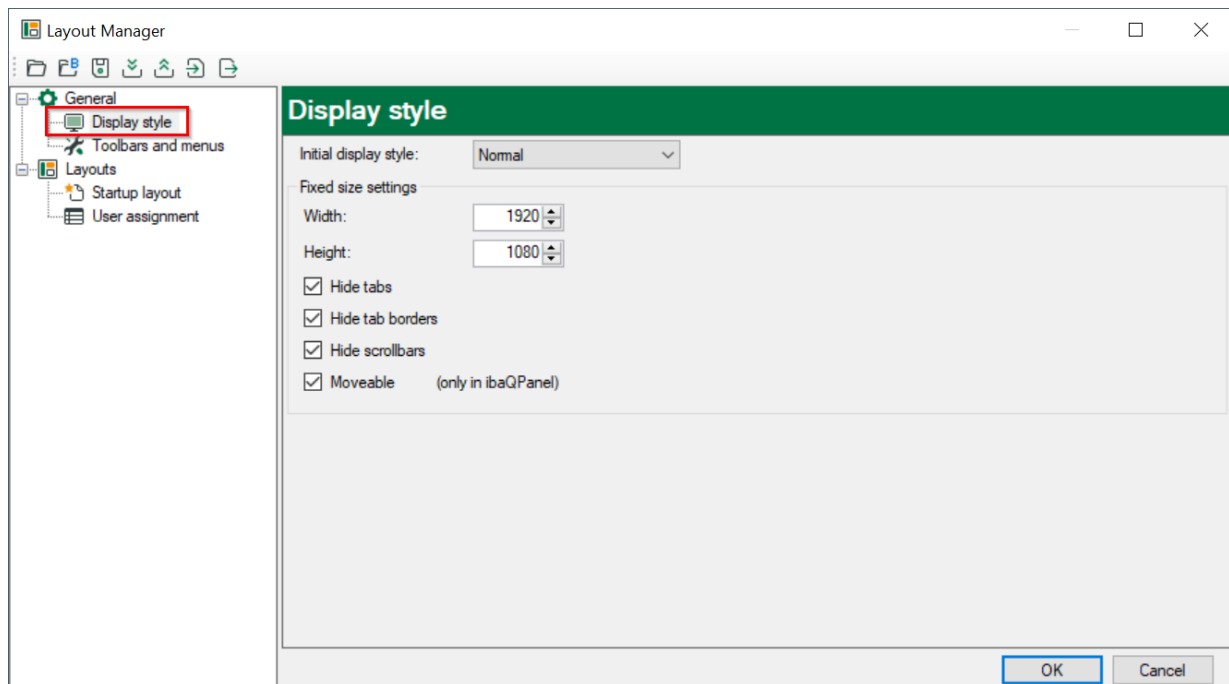
The periodic backups are created periodically in the background. The period can be expressed in minutes, hours and days. These backups are optional. They can be used to prevent work loss when you are editing layouts and the client for some reason isn't shutdown properly. You can configure the maximum number of periodic backups.

Note: No backup is created while the layout manager is opened. After exiting, the cycle starts again.



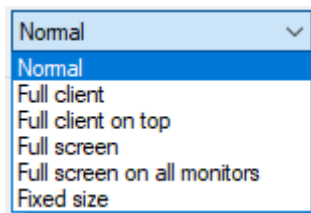
The periodic backups are stored in the same backup folder as the automatic backups. Periodic backups get the file prefix "P_".

3.5 Display Style



3.5.1 Initial display style

The initial display style can be set to the following values:



Here you set a desired display style with which the ibaPDA client should generally open.

Note: If you have also entered a display style for a user in the user management, the user management settings will have priority.

3.5.2 Fixed size settings

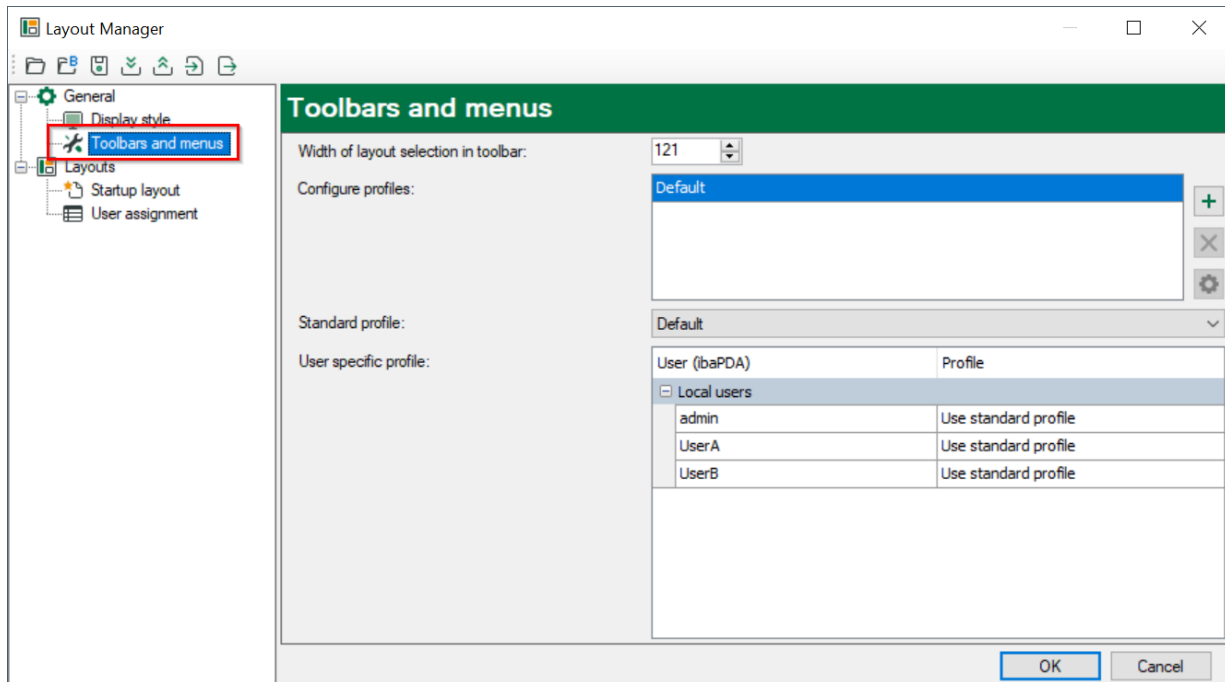
In this section some settings for the “Fixed size” display style can be made. These settings are only applied if the “Fixed size” display style is selected.

3.6 Toolbars and menus

The ibaPDA client toolbar was configured individually per layout in ibaPDA client v7 and older. This resulted in ever-changing toolbars per layout, which were often not desired.



The toolbar is now a fixed toolbar in which certain elements can be hidden. The order of the elements is always fixed. It is now possible to define a toolbar for a user.



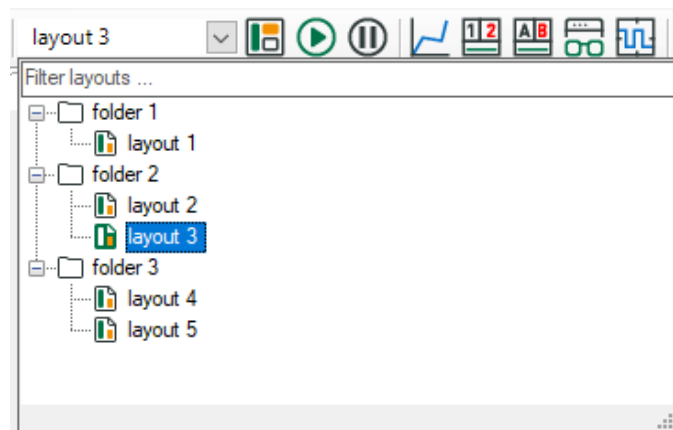
3.6.1 Width of the layout selection in toolbar

This sets the width of the layouts dropdown box (Layout list) in the ibaPDA client toolbar.




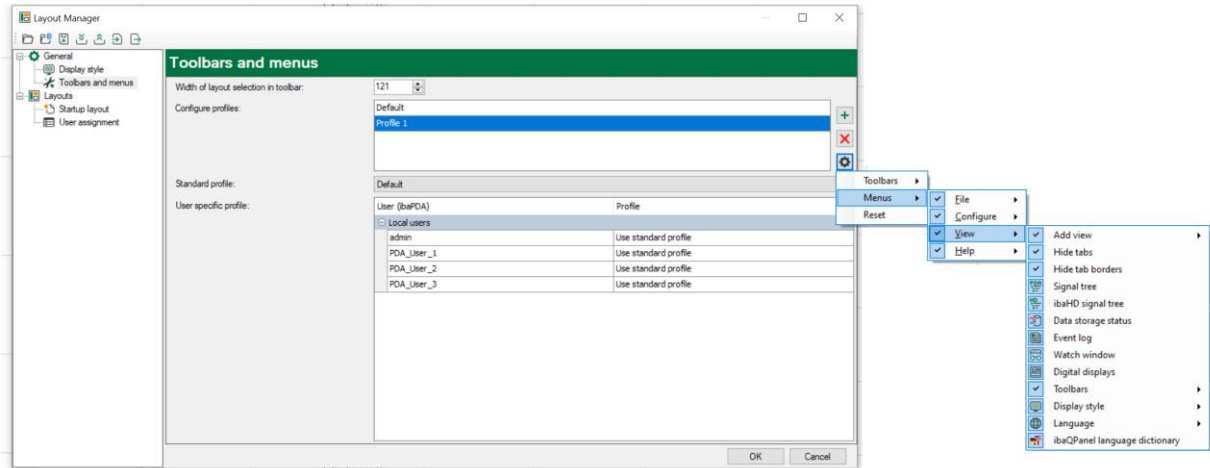
When you change the value in the edit field then the dropdown box's width in the toolbar will change immediately so that you can see what the effect is.


Note: The dropdown box window size can also be changed by the mouse when you open the dropdown. You can drag the lower right corner. This resize is only temporary.



3.6.2 Configure profiles

Profile: A new profile can be created with the + sign. The corresponding toolbar for a selected profile can be configured via the  symbol. The default profile is always available and cannot be edited.



The current toolbar for a profile can be viewed in the ibaPDA client if a profile is selected and the  button is activated. This will open a context menu and you can click toolbar buttons and menu commands to hide/show the corresponding elements. The *Reset* command can be used to reset the toolbars and menus back to the default state.

3.6.3 Standard profile

It is the standard profile that will be used by any user that doesn't have a specific profile assigned.

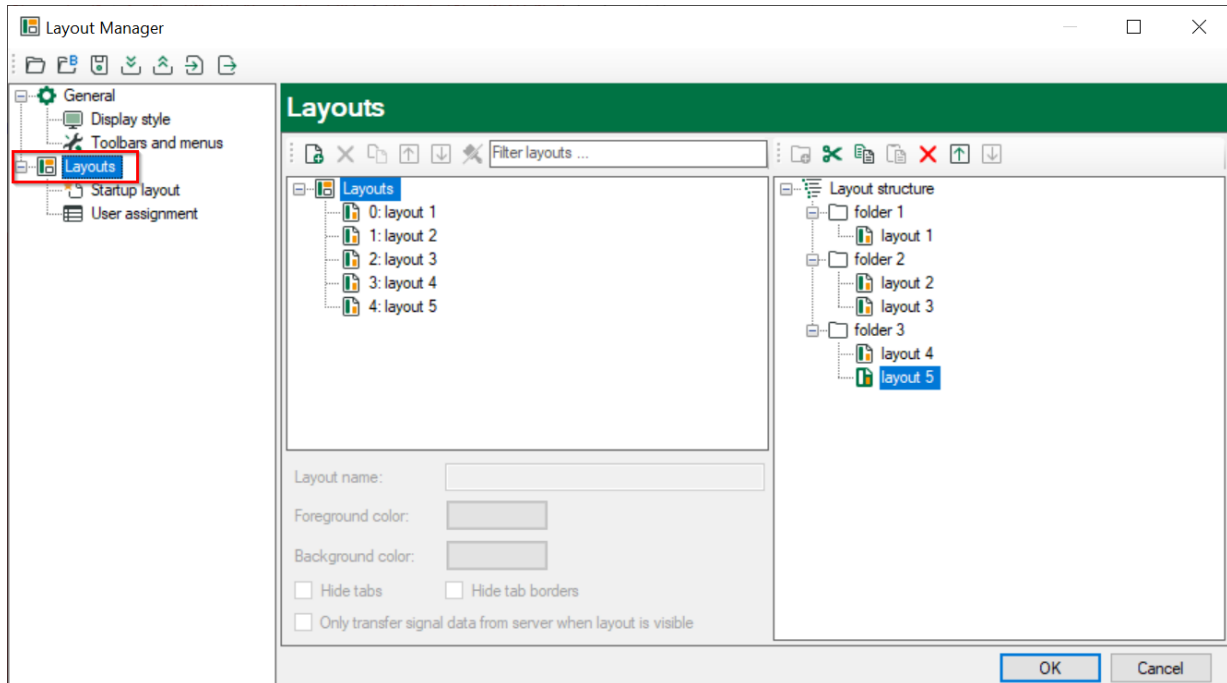
3.6.4 User specific profile

In this table profiles can be assigned to users.

3.7 Layouts

The layout control offers the following functionality:

- Manage a set of layouts (create / duplicate / delete layouts)
- Create a layout structure
- Modify layout properties



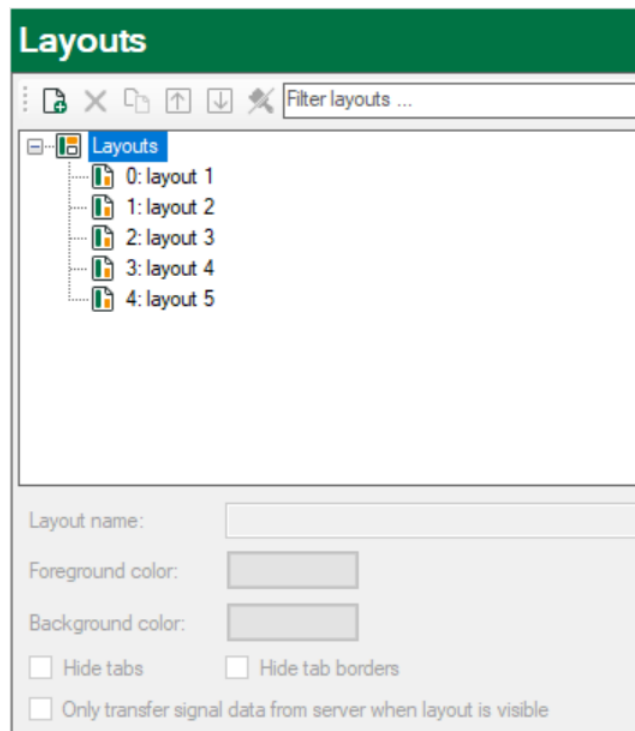
The tree on the left side (layout pool) contains all available layouts. It corresponds to the layout list from the previous layout management.

The tree on the right side (layout structure) offers the possibility to structure the layouts into folders and sub layouts.

While layouts can only appear once in the layout pool, they can appear multiple times in the layout structure.

3.7.1 Layout pool

Multi selection of layouts is possible with SHIFT and CTRL.



Layout pool toolbar

In the toolbar of the layout pool the following actions can be done:

- Add new layout
- Delete selected layouts
- Duplicate selected layouts
- Move selected layouts up
- Move selected layouts down
- Remove color settings
- Filter layouts

The functions can also be reached via the context menu.

Add new layout

The name for the new layout is generated by the following pattern: "layout **id**". If a layout is added it will also be added to the layout structure on the first level at the end.

Delete selected layouts

If a layout is deleted in the layout pool, it will be also deleted in the layout structure. So it is generally removed from the set of layouts.

Duplicate selected layouts

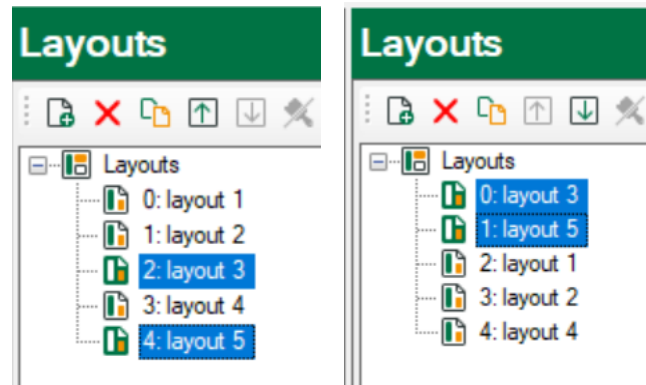
If existing layouts are duplicated, the suffix "- copy" is added to the new ones.

Moving layouts

Layouts can be moved via drag and drop, with the corresponding toolbar / context menu entries and shortcuts CTRL+Up and CTRL+Down.

If you move layouts that have a distance to each other with the move function (toolbar buttons or shortcuts), they always move with the same distance.

When dragging and dropping with the mouse, however, they end up together one behind the other. If a layout is dropped on the “Layouts” node it will be added at the end of the layout pool.



Remove color settings

The color settings can be removed for one or multiple layouts. The colors shouldn't be required in the future due to the new layout structure functionality.

Filter layouts

With the “Filter layouts...” textbox in the toolbar, the displayed layouts can be filtered by the layout name (the layout index number is not part of the name).

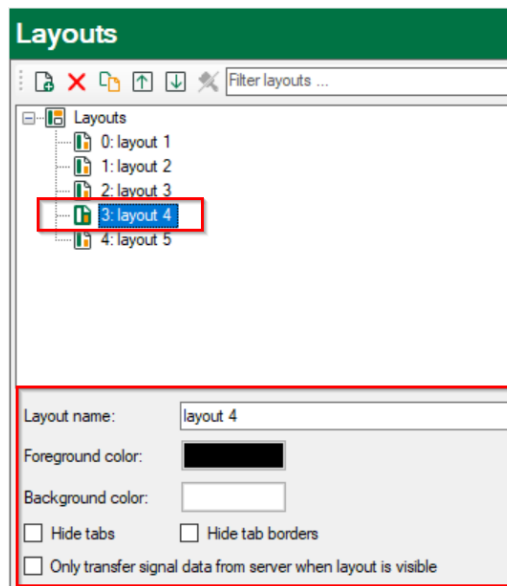
Layout naming

All layout names are allowed. Duplicate layout names are also possible, due to backwards compatibility. When you exit the layout manager, you will be informed that duplicate layout names exist and you can correct this manually if necessary.

A layout can be renamed by context menu, F2 or inside the properties section.

Properties section

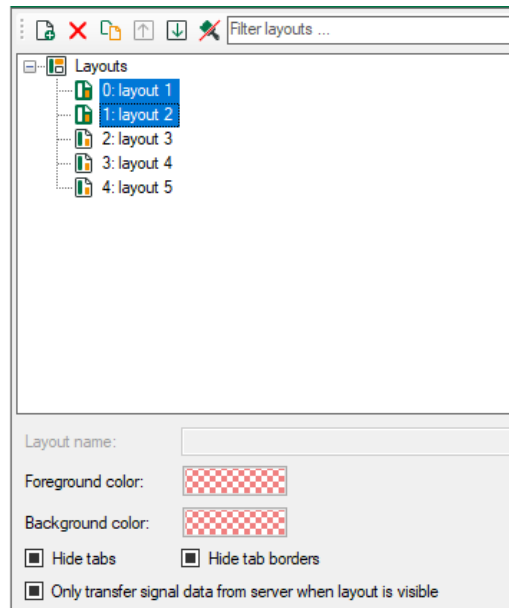
The properties for layouts can be changed in the properties section under the layout pool tree.



The following properties of one or multiple layouts can be changed:

- **Layout name:** Change the layout name is only allowed if one layout is selected
- **Foreground color:** Changes the foreground color of the layout node in the layout pool and layout structure
- **Background color:** Changes the background color of the layout node in the layout pool and layout structure
- **Hide tabs / Hide tab borders:** These values can also be changed in the standard view menu as before
- **Only transfer signal data from server when layout is visible:** Normally the data for all signals referenced in all layouts is transferred from server to client. This is done so that you always have historical data for all signals when you switch from one layout to another no matter if the signals were visible in the previous layout or not. If you enable "only transfer ..." then data will not be transferred for signals that are not visible. This reduces the amount of data transferred between server and client. This also means that you won't have historical data when you switch to a new layout for signals that weren't visible before.

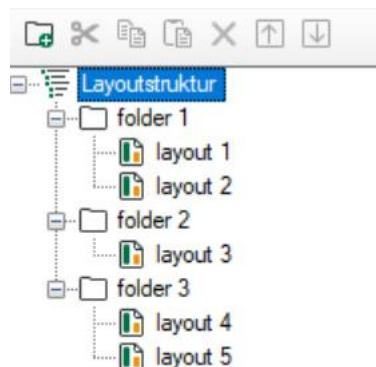
If there are multiple layouts with different settings selected the properties are displayed like in the following screenshot.



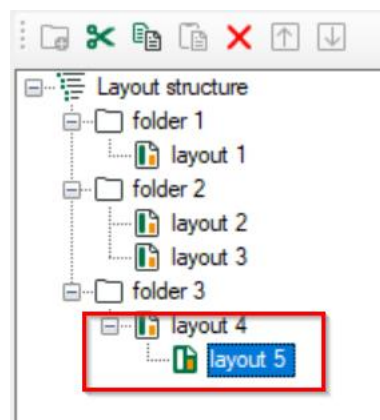
3.7.2 Layout structure

It is not necessary to make changes in the layout structure. By default that layouts are ordered in a flat list like in ibaPDA v7. In this case the layout structure is identical to the layout pool.

However, if you want to structure your layouts, you now have the possibility to do it here.



In the layout structure folders and subfolders can be created. It is also possible to assign layouts below layouts.



This can be helpful, for example, to display the structure of sub-layouts that are grouped together in the layout tab in ibaQPanel.

Note: Currently, this can only be done manually. There is no connection to the actually used sub-layouts in ibaQPanel, i.e. no automatic transfer of the parameterisation from ibaQPanel to the layout manager, etc.

Layout structure toolbar

In the toolbar of the layout structure the following actions can be done:

- Add new folder
- Cut selected elements
- Copy selected elements
- Paste copied elements
- Delete selected elements
- Move selected elements up
- Move selected elements down

These functions can also be reached via the context menu.

Add new folder

The name of the new folder is generated by the following pattern: „folder **id**“. It is not possible to create folders below layout nodes.

Cut selected elements

The selected elements will be cut out and are available on the clipboard.

Copy selected elements

The selected elements will be copied and are available on the clipboard.

Paste copied elements

The elements from the clipboard will be pasted.

Delete selected elements

The selected elements will be removed, this has no impact on the layout pool.

Moving elements

Elements can be moved via drag/drop, with the corresponding toolbar / context menu entries and shortcuts.

Drag/drop is only allowed if the move is possible for all selected nodes.

To move layouts below other layouts, the CTRL modifier has to be pressed during the drag/drop action.

It is possible to drag layouts from the layout pool to the layout structure. Also it is possible to copy selected layouts from the layout pool to the clipboard and paste it in the layout structure.

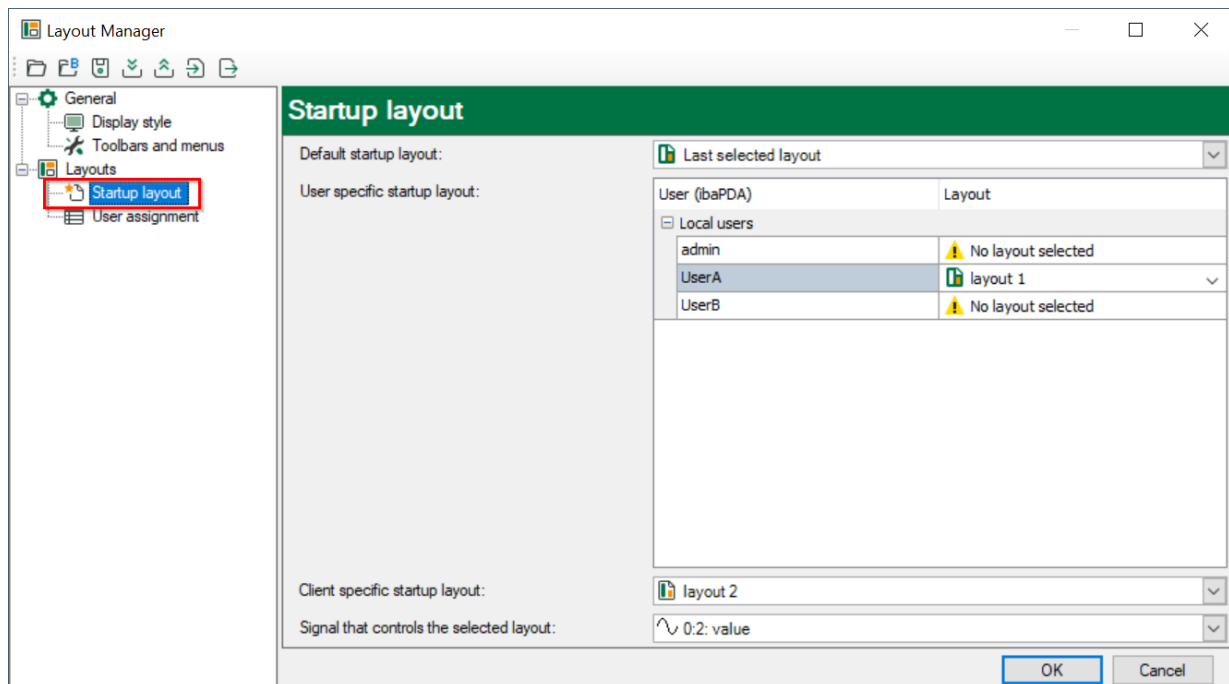
Folder naming

All folder names are allowed. Duplicate folder names are also possible. When you exit the layout manager, you will be informed that duplicate folder names exists and you can correct this manually if necessary.

Structuring rules

- Folders are displayed on the top before the layout nodes
- A layout node can only be once inside a folder or below another layout node

3.8 Startup layout



In this control the startup layout can be configured. The following startup layouts are available for this purpose. Please note that the priority of the entries increases from top to bottom. For example, a signal overwrites the previous startup layout settings.

3.8.1 Default startup layout

The last selected layout is generally set as the startup layout. You can also select a specific layout so that ibaPDA client will always start up with the same layout no matter which layout was selected when the client was closed.

3.8.2 User specific startup layout

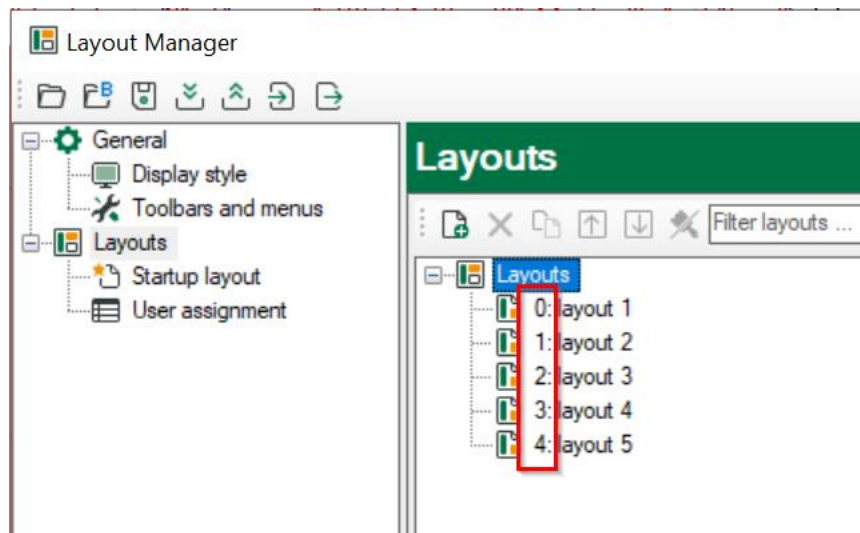
For each user (see 3.4.1 Layout source) a startup layout can be configured. When the user logs in the configured layout will be selected. It is possible to assign layouts that are not visible for the user. In this case a warning is displayed.

3.8.3 Client specific startup layout

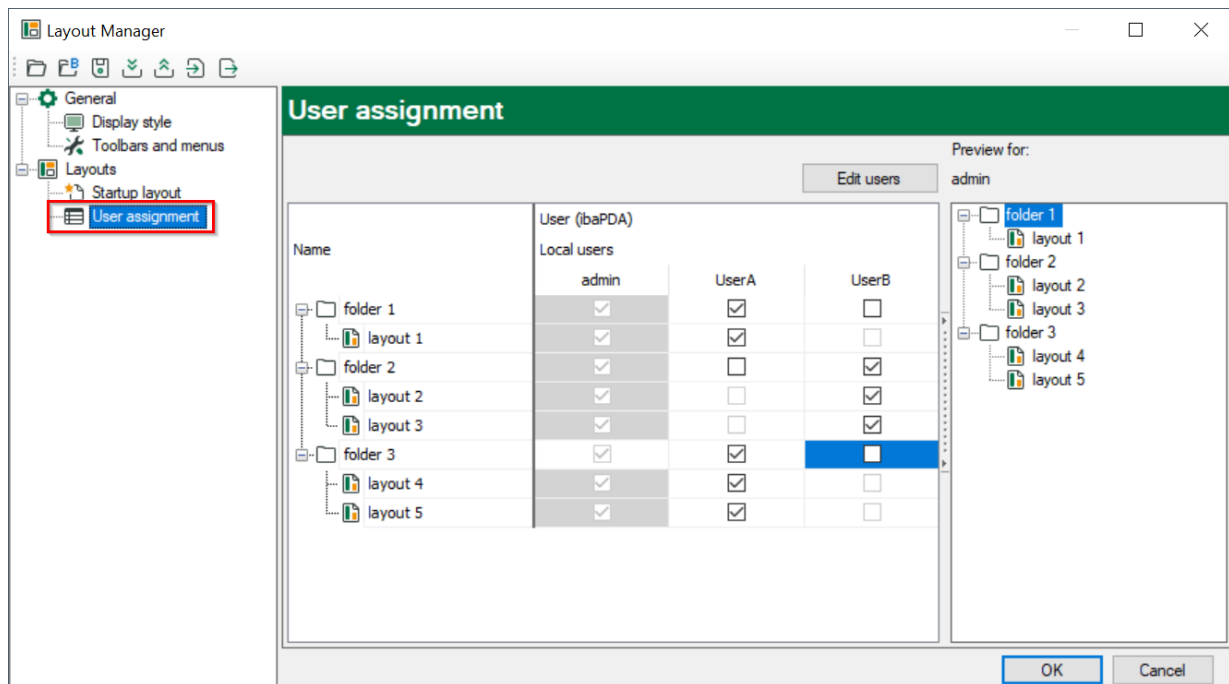
Here you can define the startup layout for a specific client/computer. This function was previously found in the preferences as "Force startup layout".

3.8.4 Signal that controls the selected layout

Here you can specify an analog signal, the value 0...n that comes via this signal selects the corresponding layout. The assignment of number to signal value can be seen in the list of layouts.



3.9 User assignment



3.9.1 User layout mapping

Depending on the layout source (ibaPDA, ibaHD) in the general node (3.4.1 Layout source) these users are shown here.

On the left-hand side the layout structure, defined in 3.7.2, is shown. You can now assign the individual layouts to the users. To be able to assign layouts from folders, the folder itself must also be assigned.

Note: For example, to temporarily deprive a user of all layouts in a folder, it is sufficient to deselect the folder, then this user will no longer see any of the sub-layouts. But the assignment of the sub-layouts remains, so that by reactivating the folder all sub-layouts become visible to the user again.

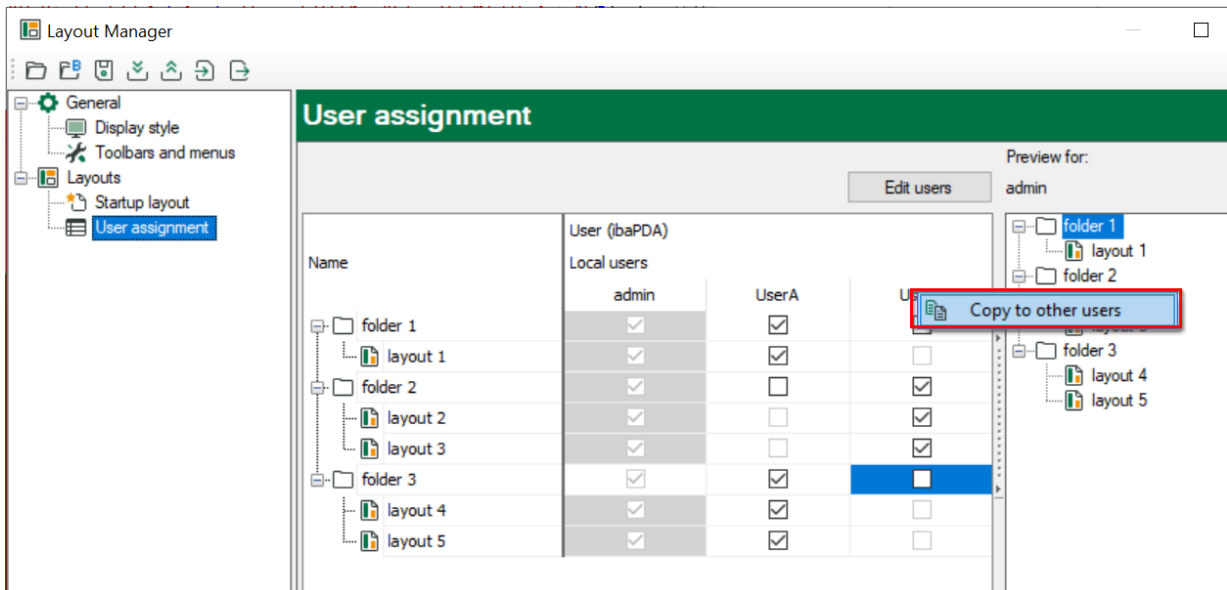
The user can select multiple cells and set the layout assignment with one click.

3.9.2 Preview

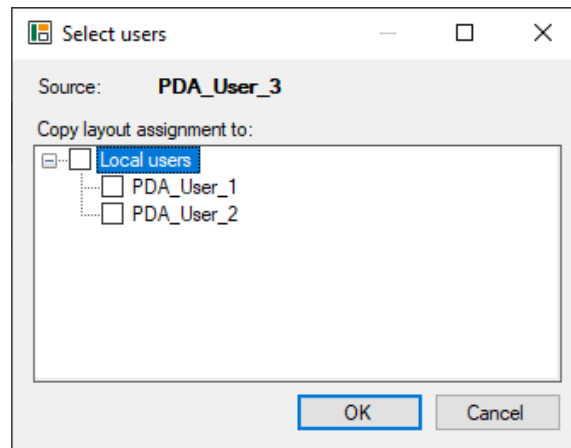
On the right-hand side, you can see the preview of the dropdown layout list in the ibaPDA client toolbar. The preview is shown for the user from the selected user column.

3.9.3 Copy to other users and layouts

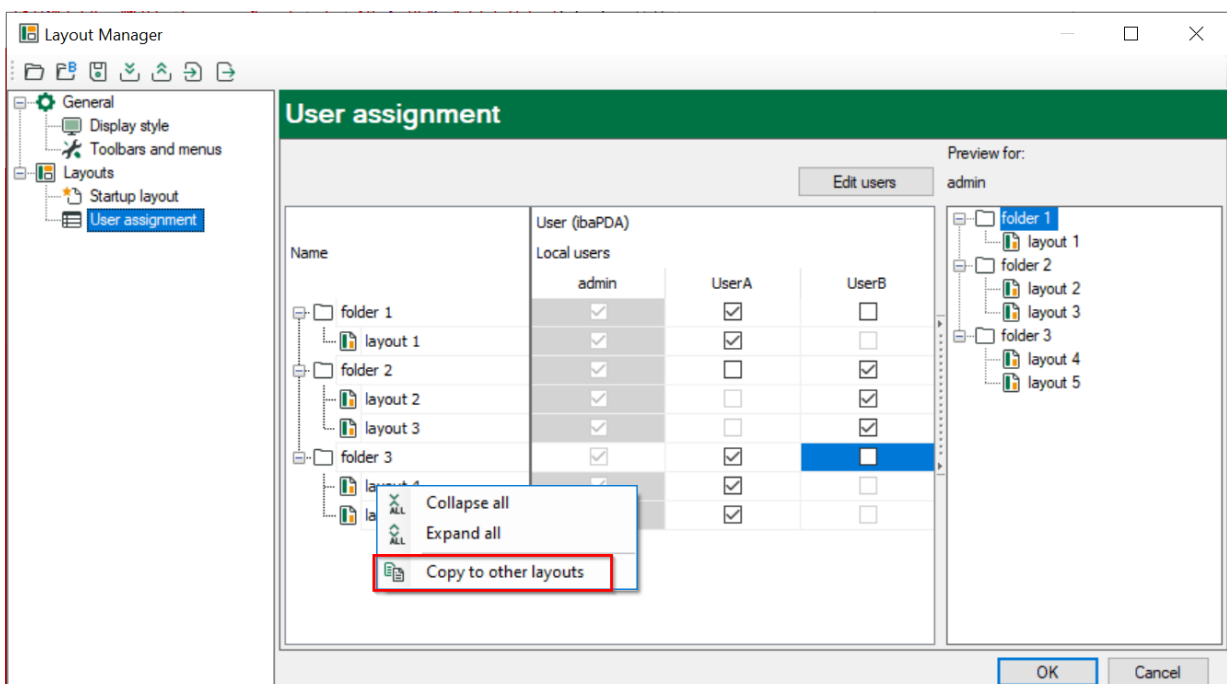
Settings for a user can be copied to other users via the context menu.



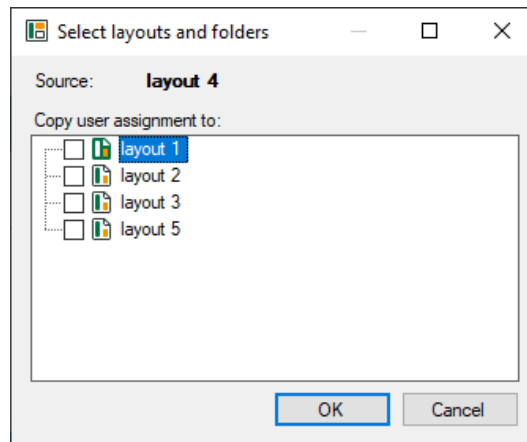
In this dialog the target users can be selected, the admin user is readonly and cannot be selected.



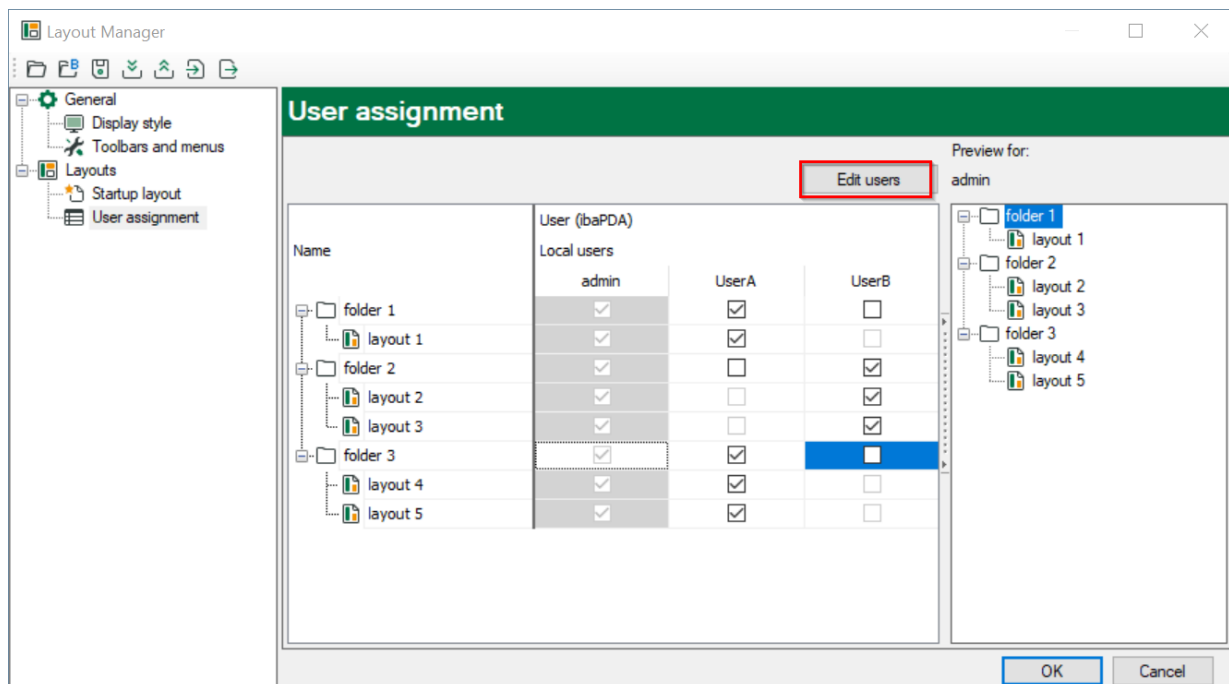
Likewise, the user assignment of layouts and folders can be copied to other layouts and folders.



In this dialog the target layouts and folders can be selected.

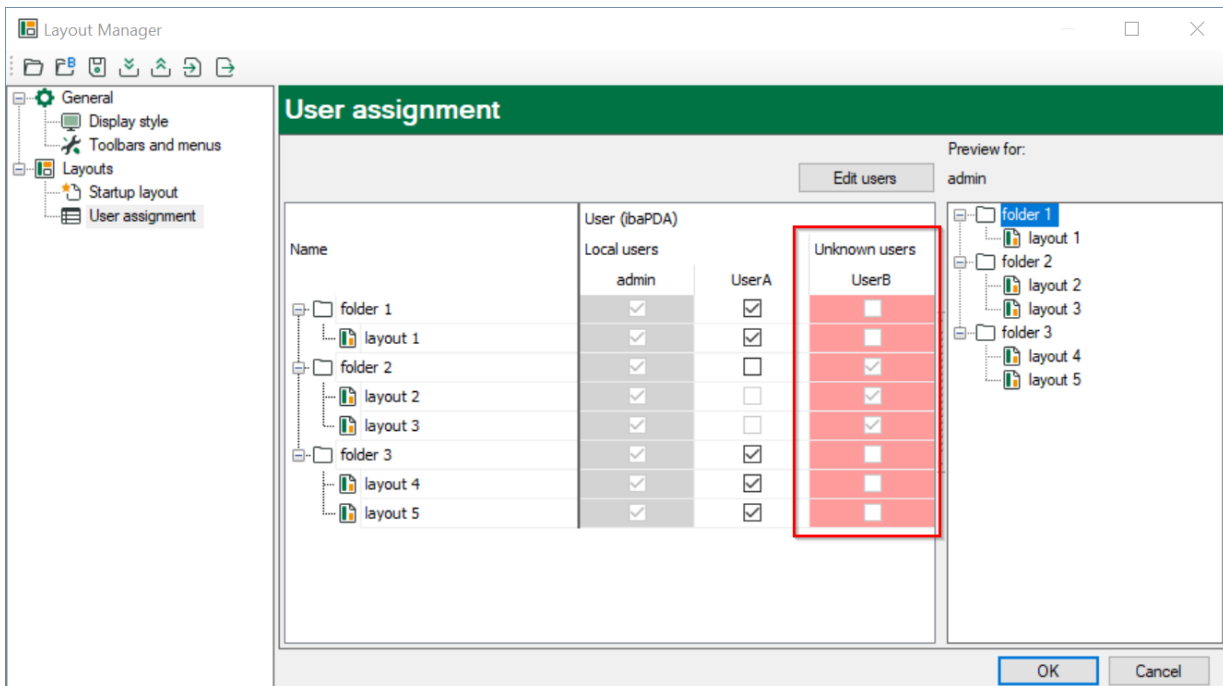


With the button “Edit users” the user management can be opened directly. Here the user management settings can be modified without leaving the layout manager.



3.9.4 Unknown users

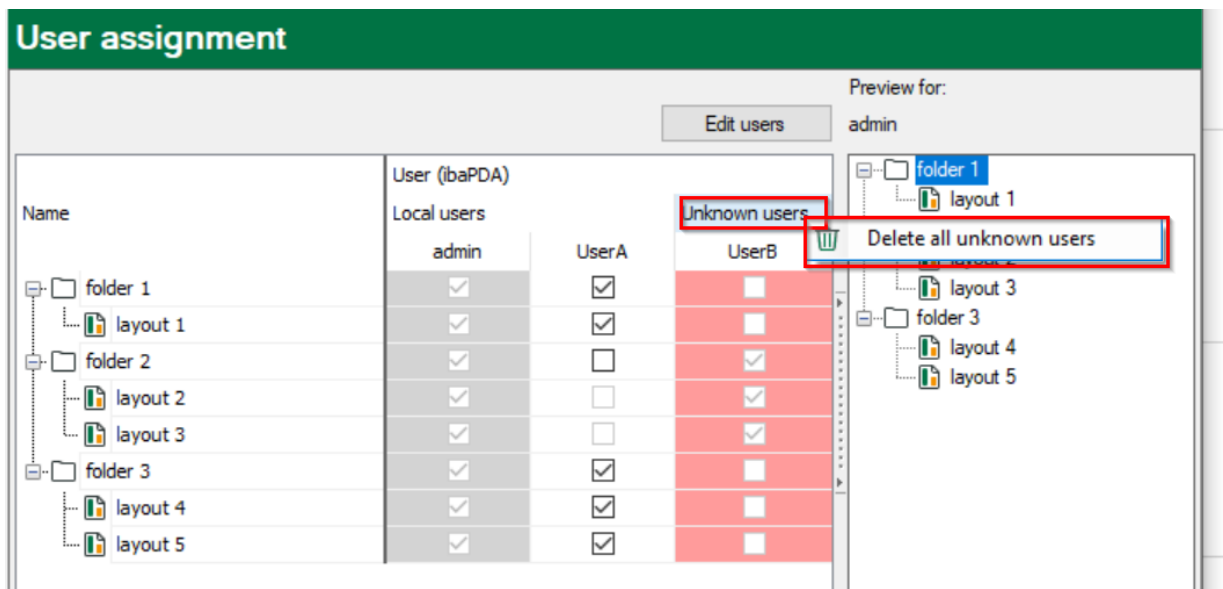
If there are non-existing users with defined user assignments, these user columns are highlighted in red.



This behaviour can happen in the following scenarios:

- Delete users, that have layouts assigned, in the user management
- Change the layouts source (ibaPDA, ibaHD) to another server (3.4.1)
- Open a layout set (.layouts) with layouts assigned to users, that are not available in the current server user set

If the users is required it can be created in the user management, then the unknown user becomes known. Otherwise a single unknown user can be deleted by the context menu on the user column or all users can be deleted by the context menu on the users column.



Preview for: admin

Edit users

Name	User (ibaPDA)		
	Local users	UserA	Unknown users
folder 1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
layout 1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
folder 2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
layout 2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
layout 3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
folder 3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
layout 4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
layout 5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Copy to other users

Delete unknown user

3.10 Layout conversion from ibaPDA v6 and v7 to ibaPDA v8

3.10.1 Overview

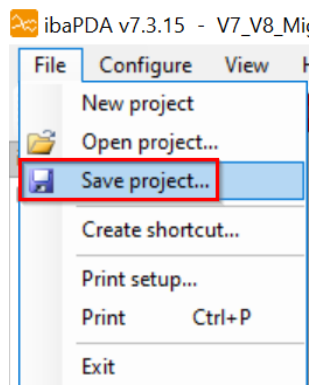
Due to the changes/improvements in layout management, there are a number of points that need to be considered during the upgrade to ibaPDA v8 from older versions.

In this document, different scenarios are presented and a recommended procedure is described.

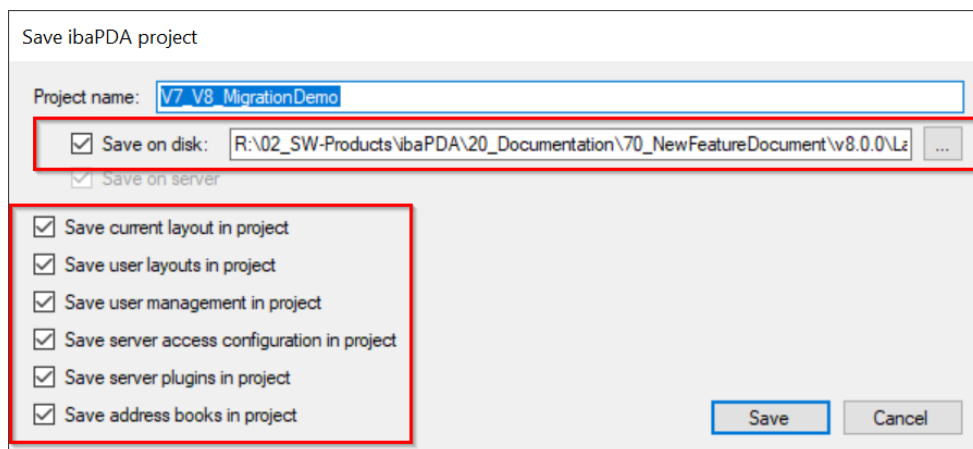
3.10.2 Getting started

Please save your whole project in v7 before starting the upgrade to v8.

You can do this by saving your project via the file menu:



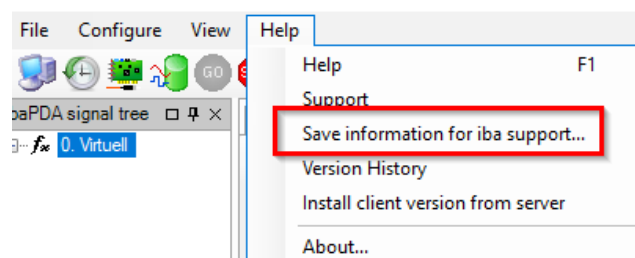
Save it somewhere on your disk (**NOT IN THE ibaPDA INSTALLATION FOLDER!**).



Make sure that all checkboxes are checked to get all information into the project.

In ibaPDA v8 the checkbox "Save user layouts in project" is removed. The user layouts are now stored together with the user management.

Another possibility is to create a support file from the Help menu:



3.10.3 Internal structure

As a prerequisite for the changeover, the internal file structure, i.e., the structure and file type of the layout files, was improved.

Whereas previously there was one large file containing all corresponding layouts (.lay), there is now a zip file (.layouts) containing all single layout files (.layout) for each layout set.

There is an index file that manages all corresponding layouts (LayoutConfiguration.config).

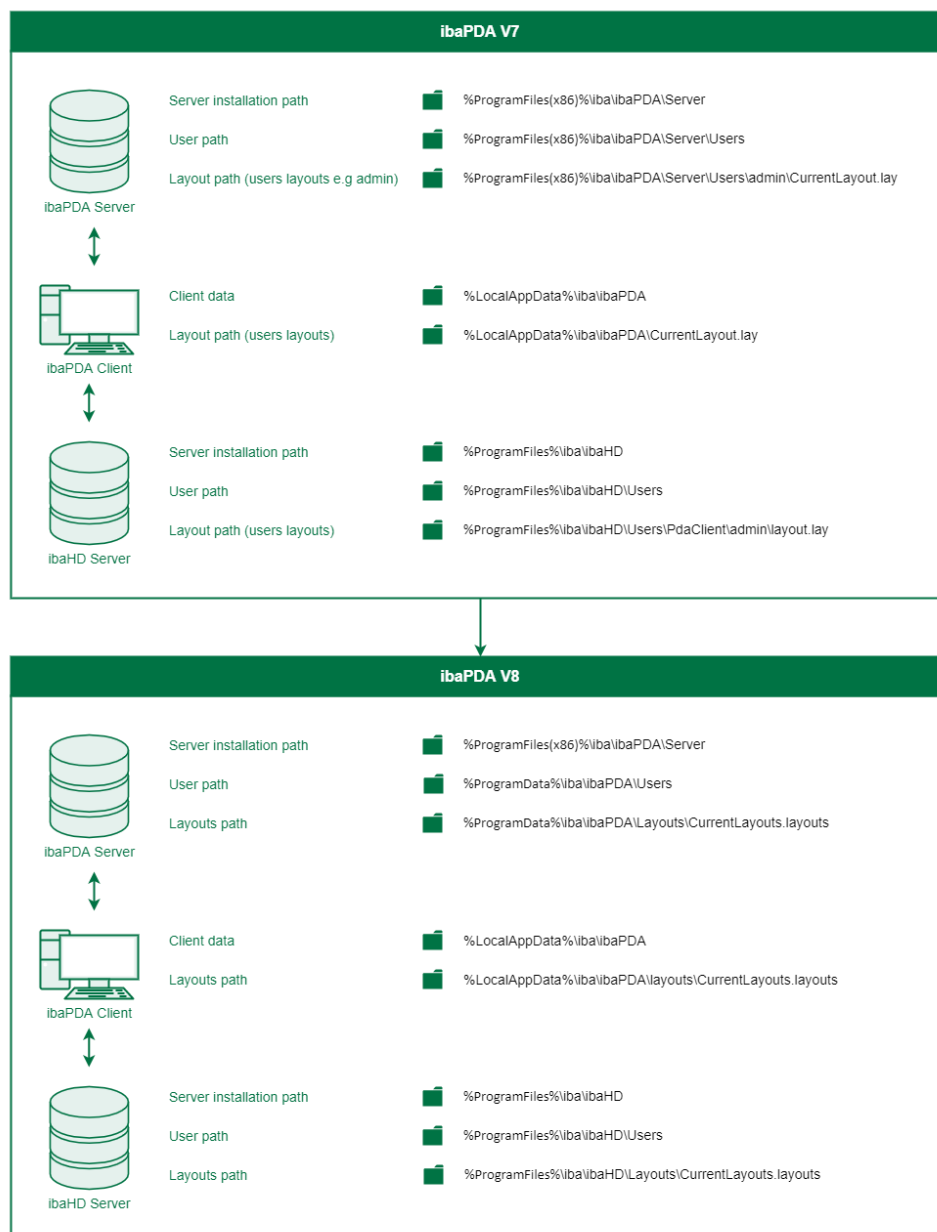
The images of ibaQPanels are stored in a backup storage file (BackupStorage.storage).

The ibaQPanel text translations are stored in a dictionary file (Dictionary.dict)

All files are saved in XML format.

As before, one has to distinguish between storage of the layout files on the client, on the ibaPDA server or on the ibaHD server.

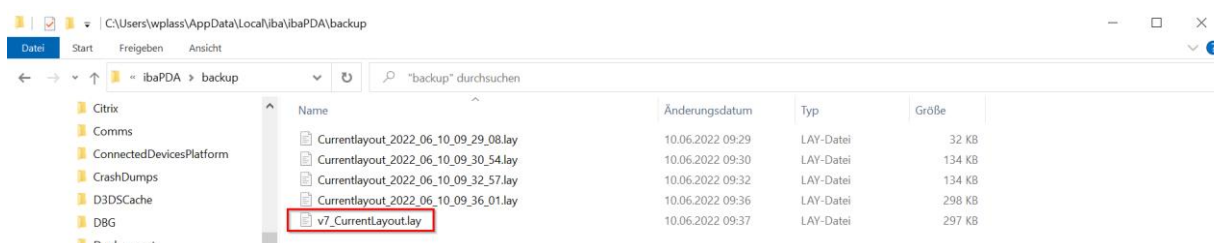
Here is an overview of the storage locations in v7 and in v8:



3.10.4 Upgrade a simple ibaPDA system with layouts (only local, no user layouts)

If you have a simple ibaPDA system where the layouts were only saved locally on the client or via the Windows file system then there is not much to consider when changing over.

When you upgrade to v8, the local layouts are automatically taken and converted into the new form. This happens once, from this point on the new layouts format is used. The original v7 layouts are automatically stored in the backup directory.



In the layout manager you will see all your layouts as before, including any color settings.

In this case the layout pool and layout structure will be identical. Since no user management has been used, the layouts dropdown will also show the layouts as usual.

Note: Toolbars could previously be set per layout. This is no longer the case, as the toolbars can now be set per user.

During the changeover, the toolbar settings of the first layout are taken and stored as admin profile.

All existing layout settings are converted into the v8 format and are accessible on the corresponding dialogs.

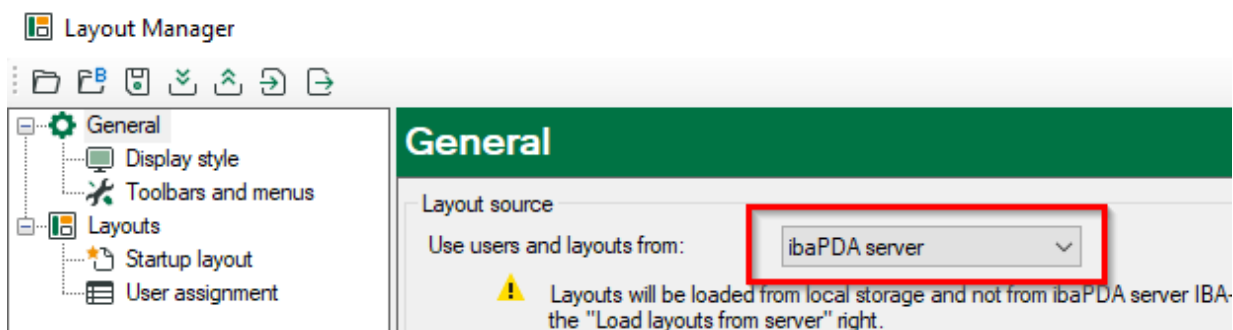
3.10.5 Upgrade from an ibaPDA system with user layouts on the server

Upgrade from a system with user layouts on the ibaPDA and/or ibaHD server will have another approach to get a safe upgrade.

First a normal upgrade can be done. Make sure to log in as admin.

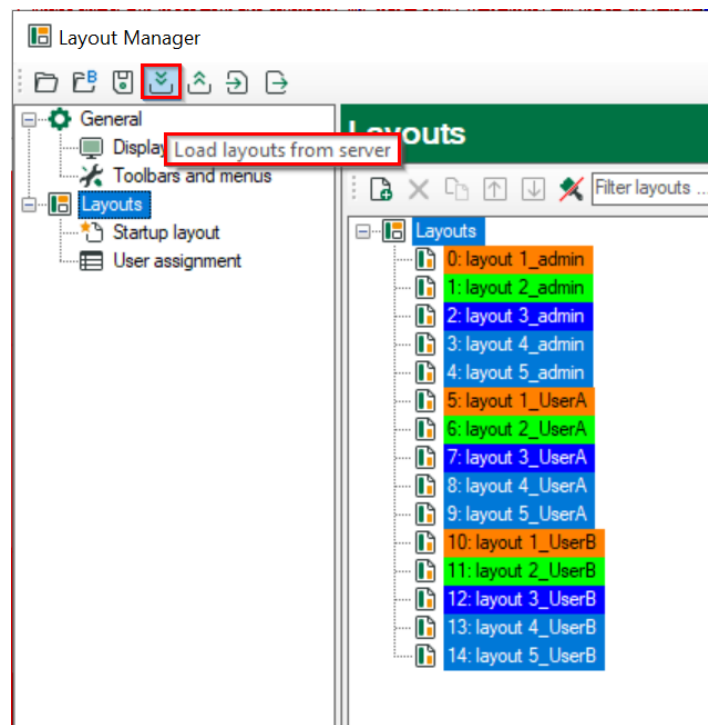
3.10.5.1 Migrate layouts from one server

Choose your desired layouts/users source in the general dialog (3.4.1).

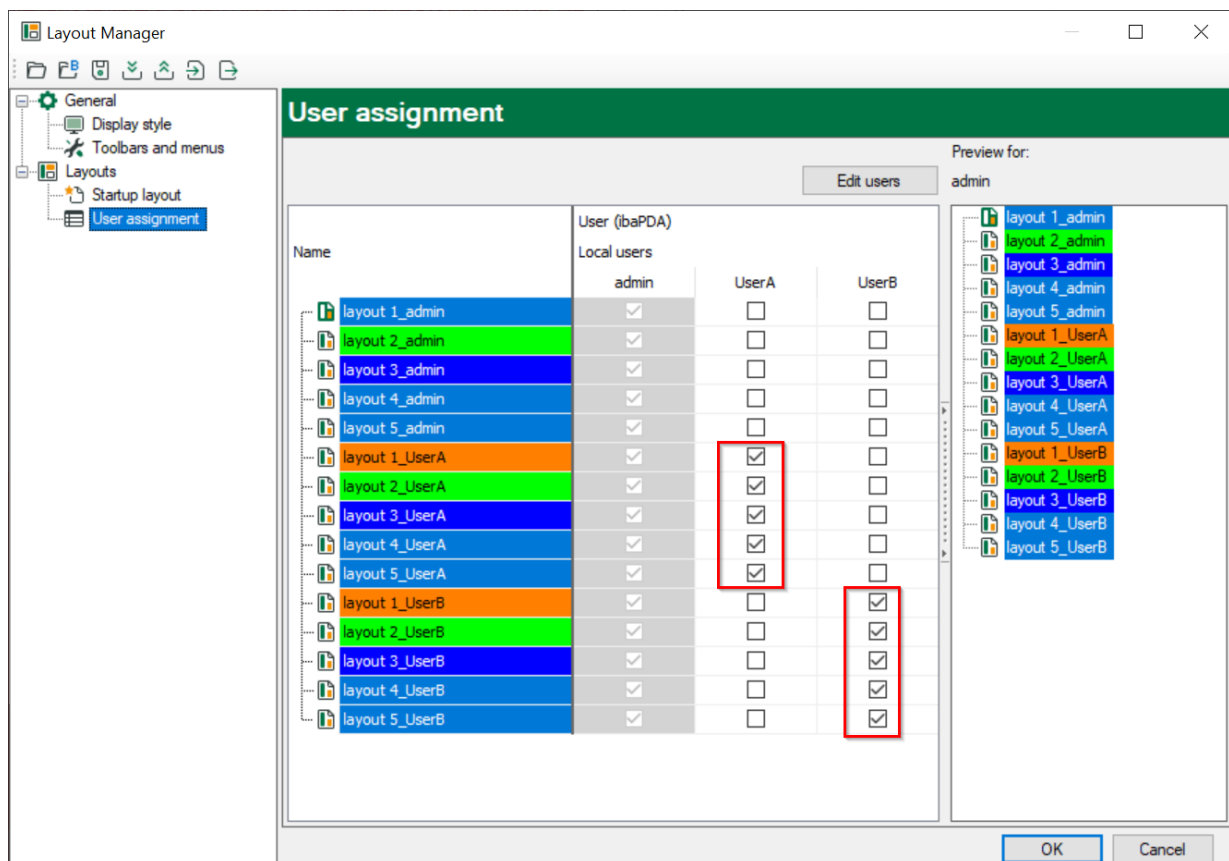


Note: in v8 only one source is allowed. So, choose “ibaPDA server” or “ibaHD server” as the place where your layouts should be stored, so that all clients can load them if there are changes. If you have only one station with ibaPDA without other clients you can choose “local storage”. Because you have only local layouts in that case.

Note: “ibaPDA server” is the default settings here. Only if in the v7 preferences the license source was set to “iba HD server” this setting will automatically switch to “ibaHD server”.



Now load the layouts from the server. The layouts list should now contain all the layouts that were present for all users. In v7 each user had his own layout set. These sets have been merged into 1 set. The layouts are renamed. They now have the user's name appended to it.



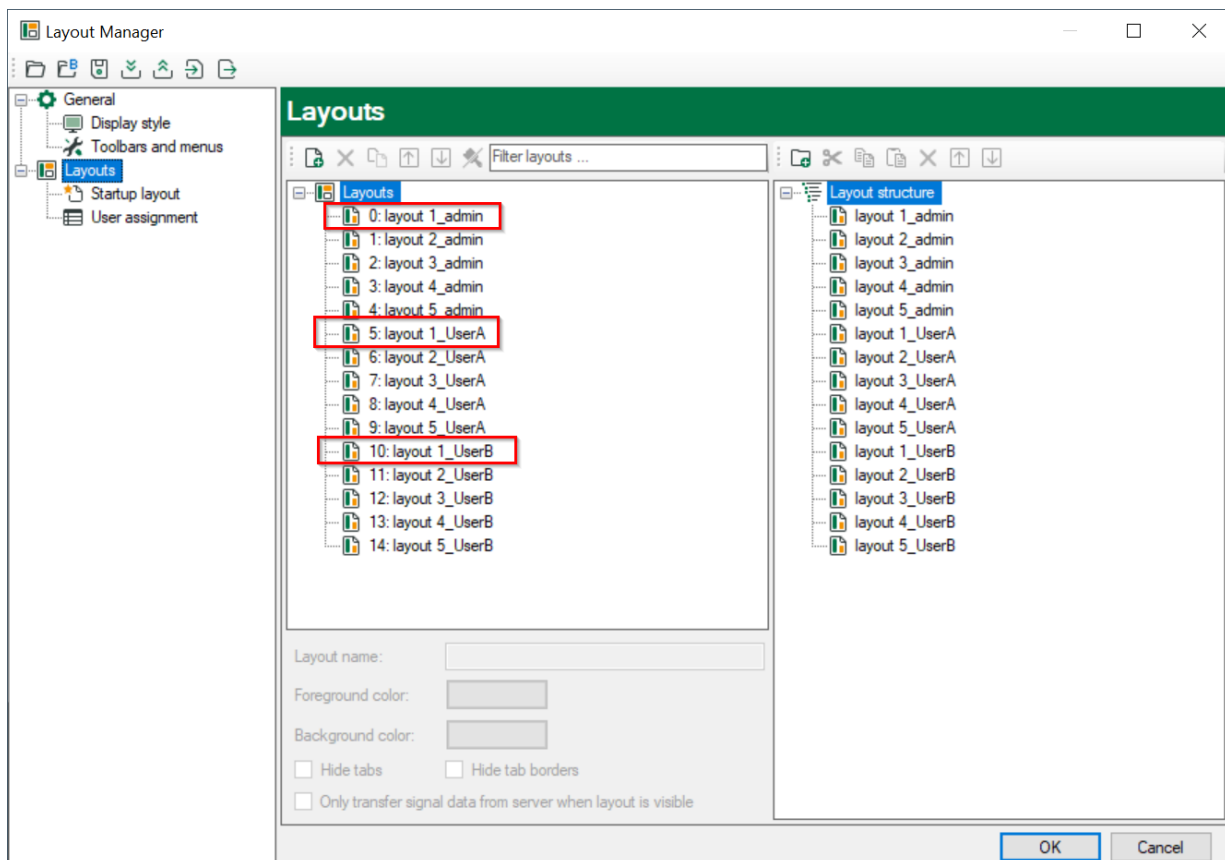
In the user assignment all the layouts for a particular user should be assigned to it. The screenshot shows that UserA has all the UserA layouts assigned to it and UserB has all the UserB layouts assigned to it.

If you would now log in as UserA you will see all your layouts as before. Only the layout names will have an additional _UserA suffix. You won't see any layouts of UserB. You should be able to work with these settings like before.

In this layout configuration set you will probably have multiple layouts that are equal but only differ in name. Probably "layout 1_UserA" is the same as "layout 1_UserB". Therefore it is recommended to clean up the layouts. This is described in the next chapter.

3.10.5.2 Cleaning up the layouts

After updating to ibaPDA v8 all layouts are available, including the users name at the end.



Now as an example we look at the layout with the name "layout 1".

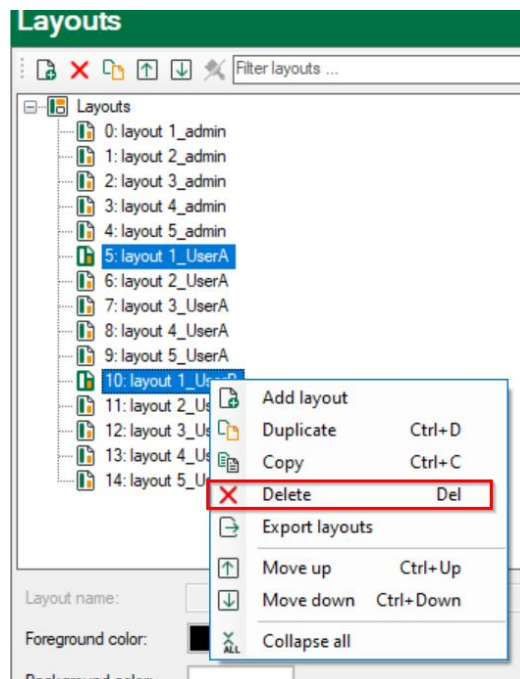
If there were layouts stored on the server for different users, all these layouts are in the layout pool. IbaPDA doesn't know if they are equal or not. That knowledge has only the designer of the layouts.

Note: the server layouts should be stored on local disk, so it will be accessible after cleaning up the layouts.

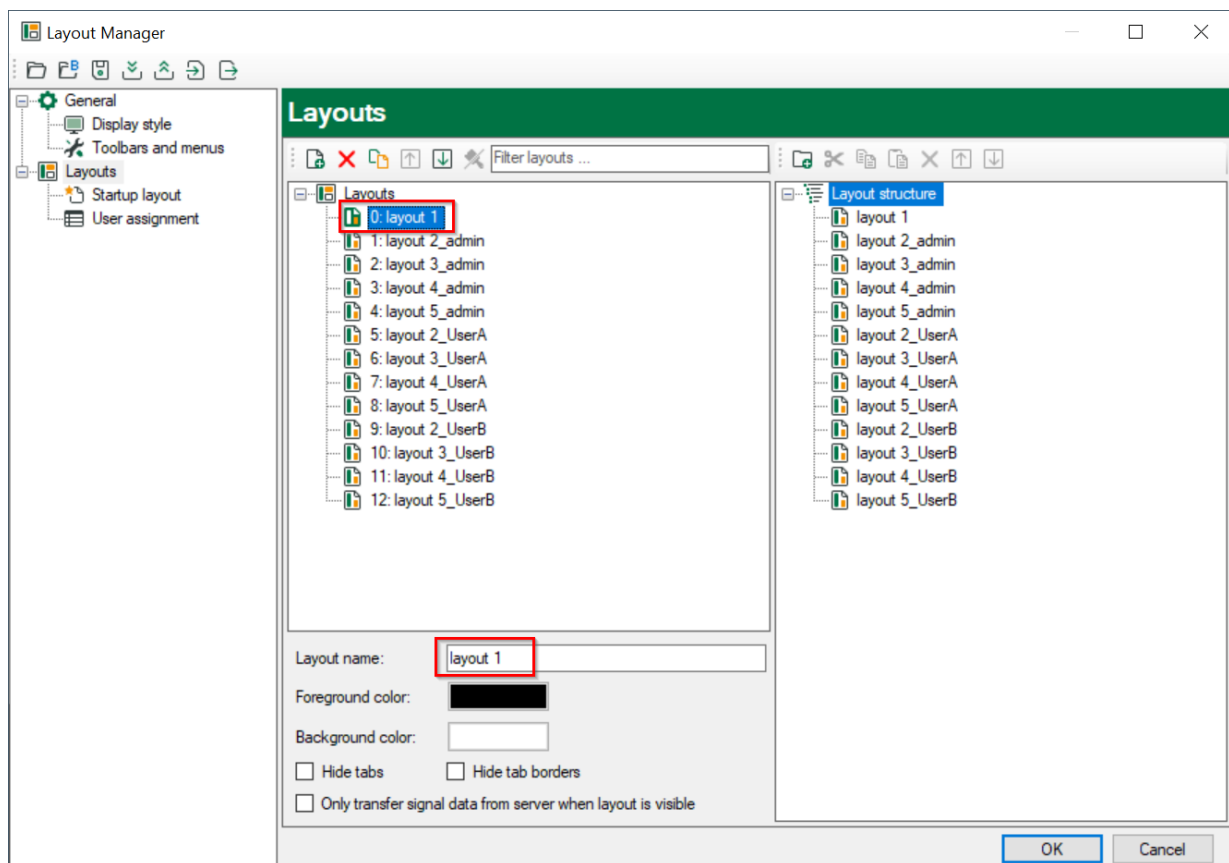
Use case: Equal user layouts

The designer is sure that these three layouts are the same (see screenshot above).

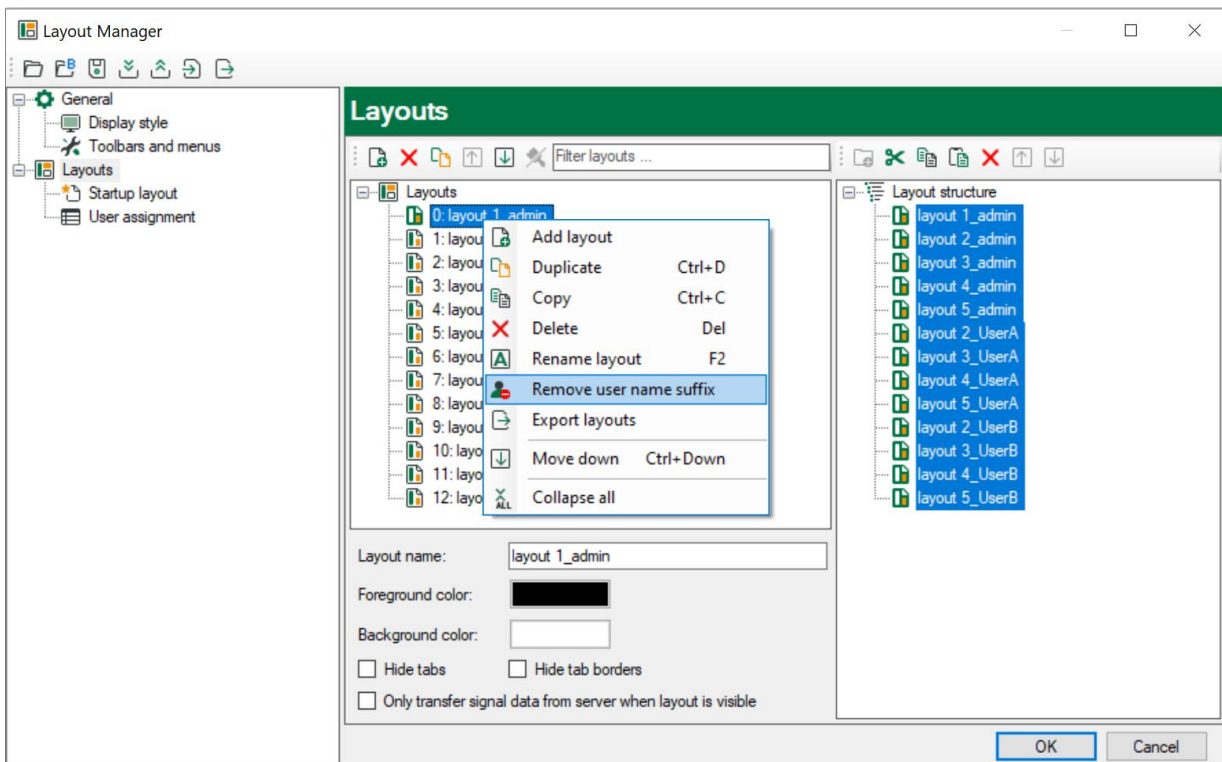
So, keep one of the layouts e.g., the "layout 1_admin" and delete the user layouts "layout 1_UserA" and "layout 1_UserB".



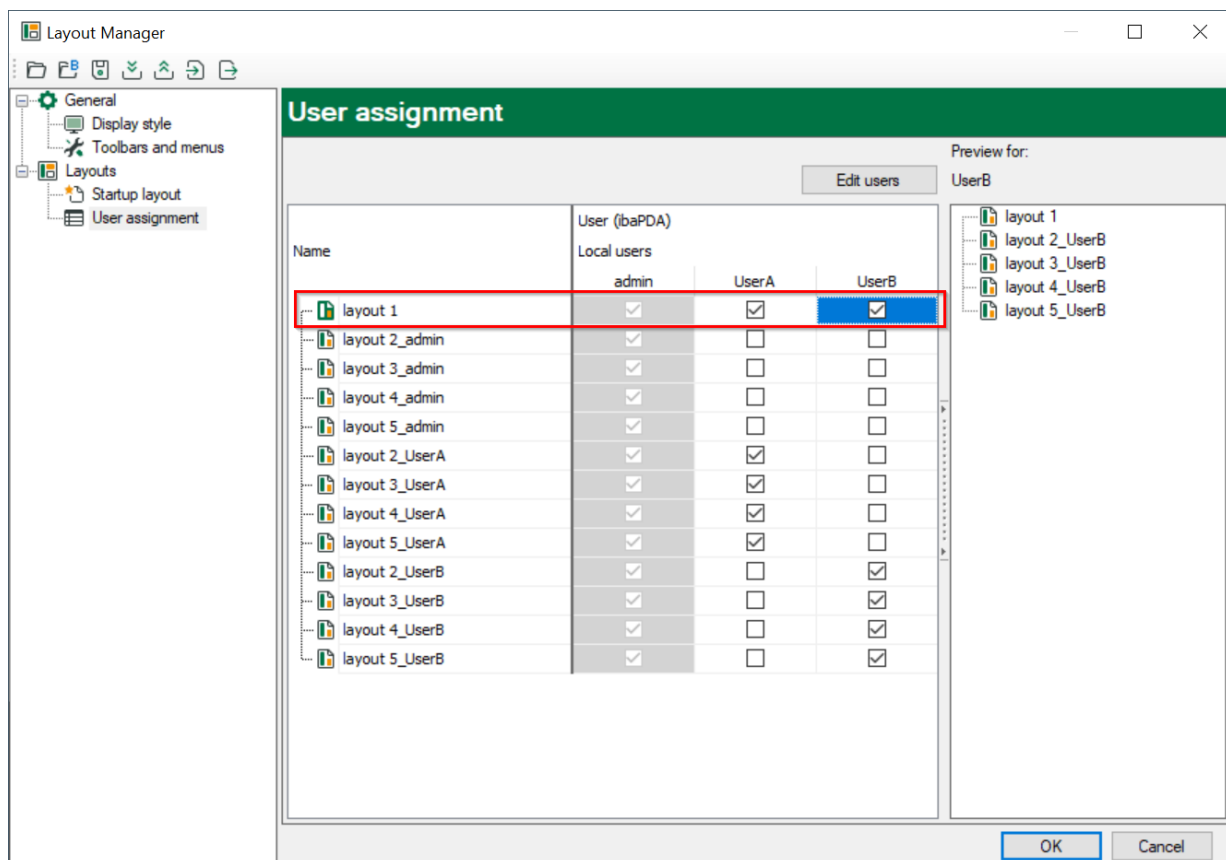
Rename the remaining one „layout 1_admin“ to the original name „layout 1“. Here are two options available. The first is to rename the layout manually.



The second option is to remove the user name suffix with an entry in the context menu.



Because the “layout 1” belongs also to “UserA” and “UserB” you now have to assign them in the user assignment to these users.



Now you are ready. One layout is cleaned up. You can proceed with the other ones. In order to do this faster you can make use of multiselection in the pool tree and the copy feature in the user assignment.

Use case: Different user layouts

The layouts are not equal for “admin”, “UserA” and “UserB”.

In this case the only step that the user can do is to change the layout name to an individual name or don't change anything at all.

3.10.5.3 Migrate layouts from different sources

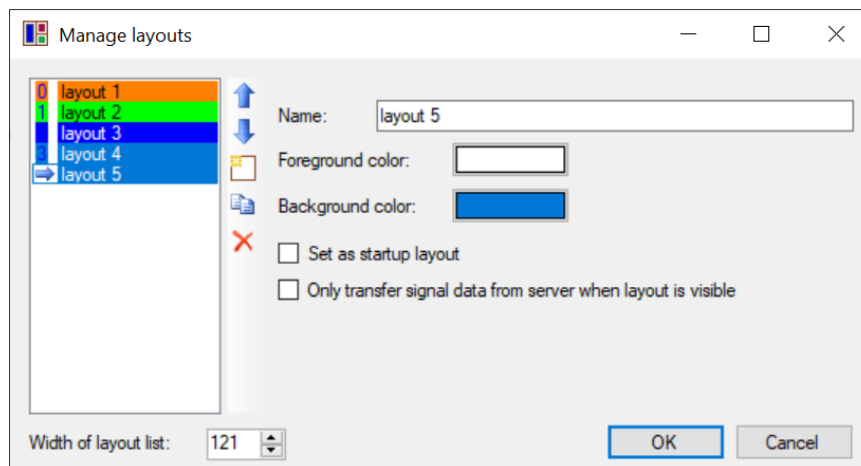
If there are layouts on both ibaPDA and ibaHD servers and you wish to have all of them then you should follow this procedure:

1. Select ibaPDA as layouts source.
2. Load layouts from ibaPDA server and do the conversion as described in 3.10.5.1.
3. Save the converted layouts set to disk.
4. Select ibaHD as layouts source.
5. Load layouts from ibaHD server and do the conversion as described in the previous chapter.
6. Import the saved layouts from step 3 with the new import function.
7. Select the desired layouts source
8. Save the merged layouts to the server

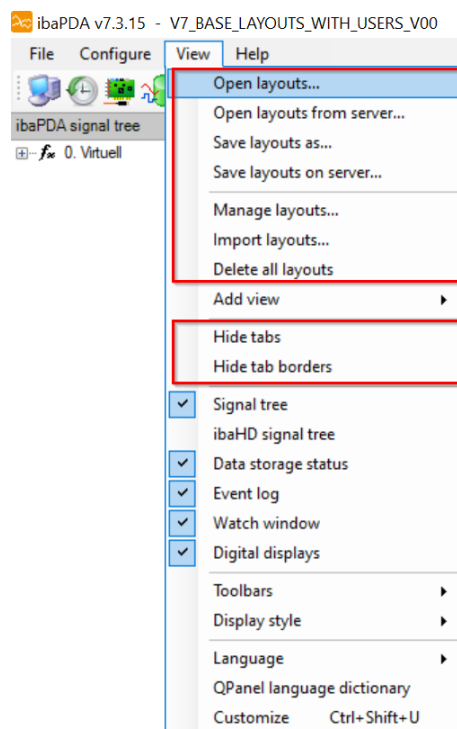
3.11 Overview of previous layout management

For the new layout management, the following parts of the previous user interface were taken into account:

The previous layout management with its possibility to create layouts and to define color properties for the layout tree, etc.

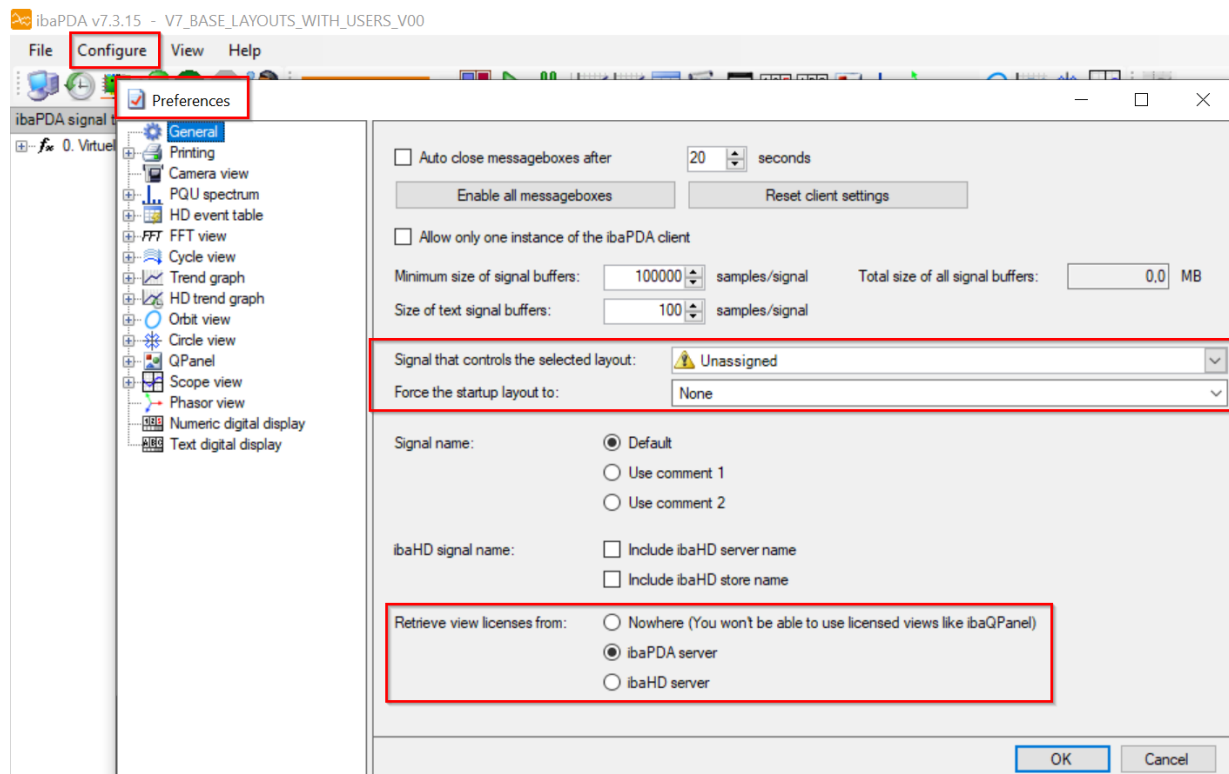


The actions for the layouts like “Open layouts ...”, “Open layouts from server ...”, “Save layouts as ...”, “Save layouts on server ...”, “Import layouts ...” and “Delete all layouts” are moved to the new layout manager dialog.

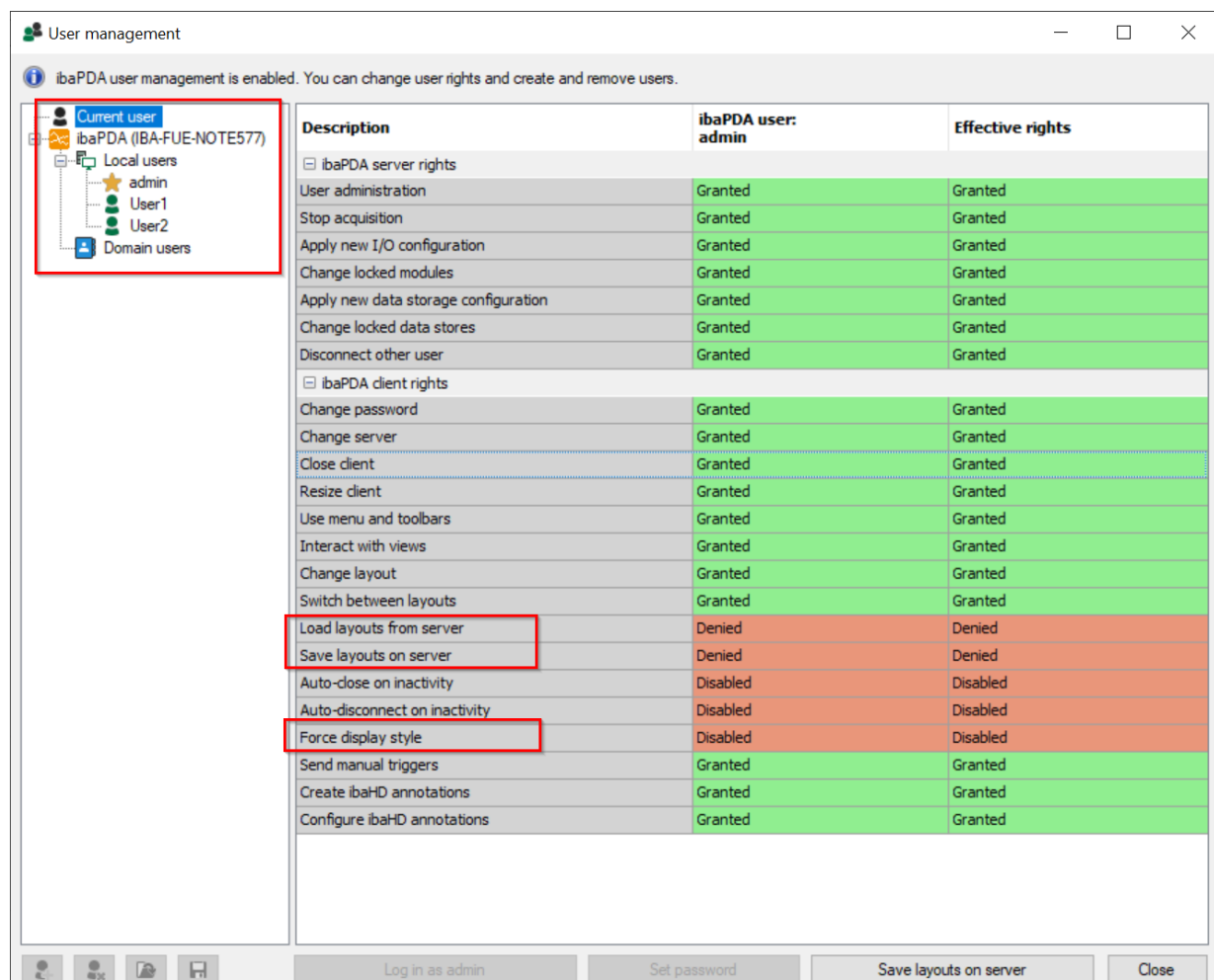


The options to set the visibility of tab and tab borders are added to the new layout manager but they are also accessible in the shown menu.

The preferences shown in the following figure are moved to the new layout manager and no longer accessible in the preferences.



There are also some layout related settings in the user management.



In the user management there are some user rights for “Load layouts from server” and “Save layouts on server”.

Also an individual forced display style for the users can be configured.

In ibaPDA v7 layouts could be stored on the ibaPDA server or on the ibaHD server, or on both in parallel. This also led to problems of understanding. It was also difficult to keep track of the layouts that were assigned to certain users, as the layouts could be saved to many users and everyone has then his own special layout even if they have the same name.

The new layout management in ibaPDA v8 considers these points and offers a new solution and a better and clearer view to the user.

4 I/O Manager

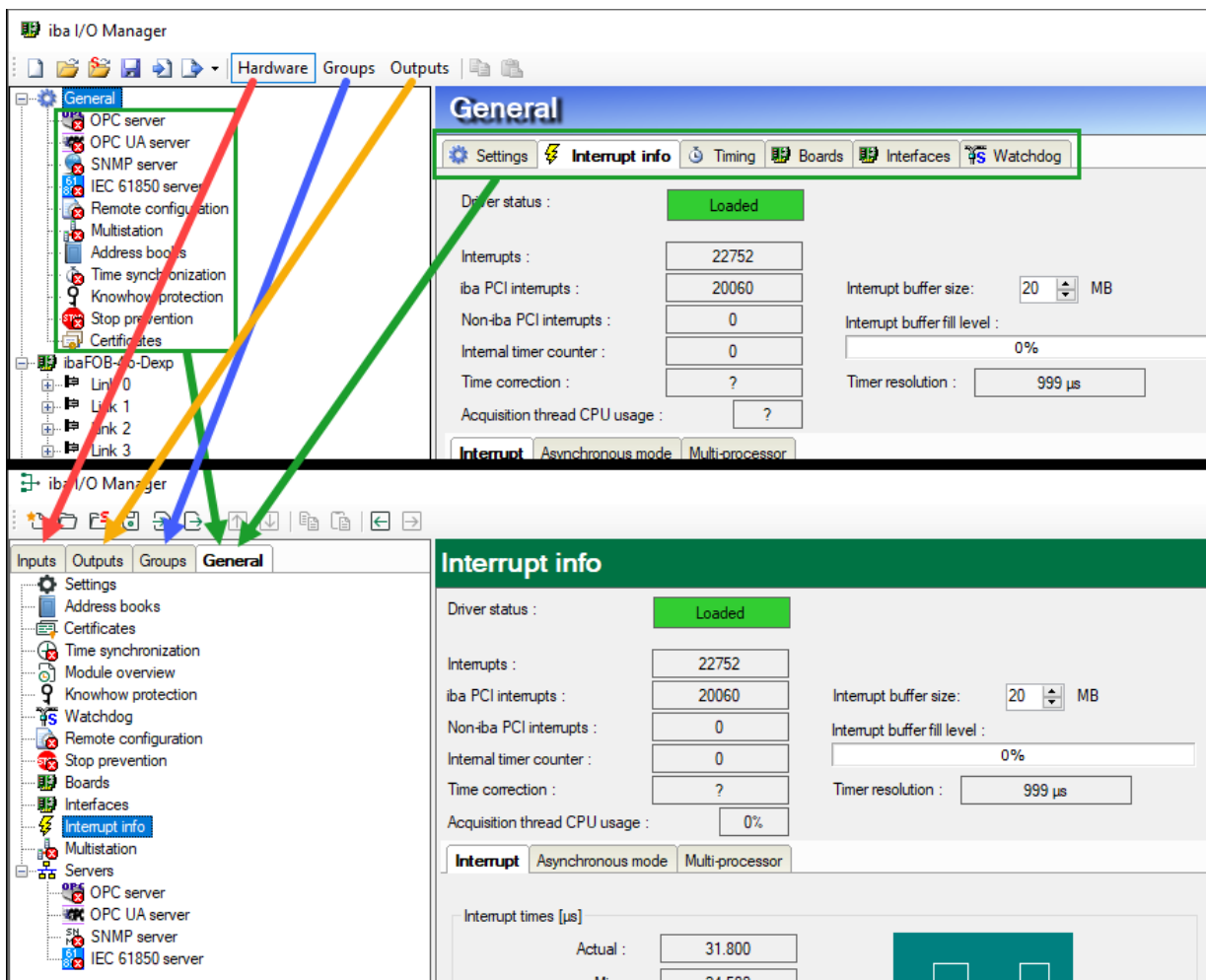
The I/O Manager has undergone a few visual and functional changes which will be described in the following sections.

The screenshot displays the 'iba I/O Manager' window. On the left, a tree view shows the system structure under 'ibaFOB-4io-Dexp'. The 'Volando (0)' module is selected. The right pane shows the 'General' tab for this module, containing a table of inputs.

Name	Unit	Input ra...	Min	Max	Filter	Act...	Actual
0 Vengo	V	± 24.0 V	-24	24	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
1	V	± 24.0 V	-24	24	<input type="checkbox"/>	<input type="checkbox"/>	
2	V	± 24.0 V	-24	24	<input type="checkbox"/>	<input type="checkbox"/>	
3 Voy	V	± 24.0 V	-24	24	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0

At the bottom of the window, a status bar shows a value of 54, along with 'OK', 'Apply', and 'Cancel' buttons.

4.1 Tab structure



In previous versions of ibaPDA the tree structure at the left in the I/O Manager could be in **Hardware**, **Groups** or **Outputs** mode, depending on which button was active in the toolbar. These 3 buttons have now been removed from the toolbar and instead, for better visibility, a tab structure was introduced just above the tree structure.

The **Hardware** mode has been renamed to (the more general) **Inputs**; **Groups** and **Outputs** remain the same but have switched positions.

All contents of the **General node**, available in previous versions of ibaPDA when in **Hardware** mode are now available in a separate **General tab** as is clear from the picture above. The tabs that could be seen on the right-hand side of the I/O Manager when selecting the **General node** now all have their own node in the **General tab**. Furthermore, all server interfaces (e.g. OPC UA, SNMP) have been grouped in a **Servers** node in the **General tab**.

4.2 Module overview

Module overview

Acquisition timebase: 10 ms Minimum output time: 50 ms
 Interrupt cycle time: 10 ms Client update time: 200 ms

Direction: Inputs Hide disabled modules
 Interface: All Hide modules with a fixed timebase

Module Number	Name	Type	Timebase	Active signals	Configured signals
				Analog Digital	Analog Digital
	ibaBM-DP	ibaBM-DP	10 ms	0 0	0 0
10	Sniffer	ibaBM-DP/Sniffer	10 ms	0 0	64 64
0	Volando	ibaPADU-4-AI-U	10 ms	4 0	4 0
1	Water meter	Generic TCP	10 ms	1 0	1 0
2	Electricity meter	Generic TCP	10 ms	6 0	6 0
3	Motion sensors	Generic TCP	10 ms	0 4	0 4
9	Extra motion sensors	Generic TCP	10 ms	0 8	0 8
8	Temperature	Raw Ethernet	10 ms	8 0	8 0
117	Helikopter	Virtual	10 ms	2 0	2 0

COUNT=9 SUM=21 SUM=12 SUM=85 SUM=76

The **Timing** tab from previous ibaPDA versions has been expanded and renamed to **Module overview** (available in the **General** tab). As the name implies this gives an overview of all configured modules in the I/O Manager along with their time base and number of active and configured signals.

In the grid it's possible to change the time base of modules that allow it; grey colored rows indicate that a module's time base cannot be changed (because it is e.g. depending on the time base of a parent module). Multiselect is supported so that you can change the time base of multiple modules at once.

At the bottom the totals per column are displayed. Note that only the currently displayed modules in the grid are taken into account.

Module overview

Acquisition timebase: 10 ms Minimum output time: 50 ms
 Interrupt cycle time: 10 ms Client update time: 200 ms

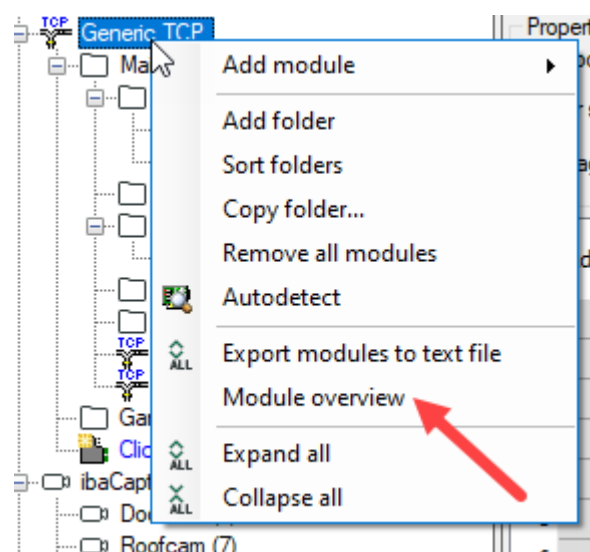
Direction: Inputs Hide disabled modules
 Interface: All Hide modules with a fixed timebase

Module Number	Name	Type	Timebase	Active signals	Configured signals
				Analog Digital	Analog Digital
0	Volando	ibaPADU-4-AI-U	10 ms	4 0	4 0
1	Water meter	Generic TCP	10 ms	1 0	1 0
2	Electricity meter	Generic TCP	10 ms	6 0	6 0
8	Temperature	Raw Ethernet	10 ms	8 0	8 0
117	Helikopter	Virtual	10 ms	2 0	2 0

COUNT=5 SUM=21 SUM=0 SUM=21 SUM=0

Filter: Analog > 0 Edit Filter

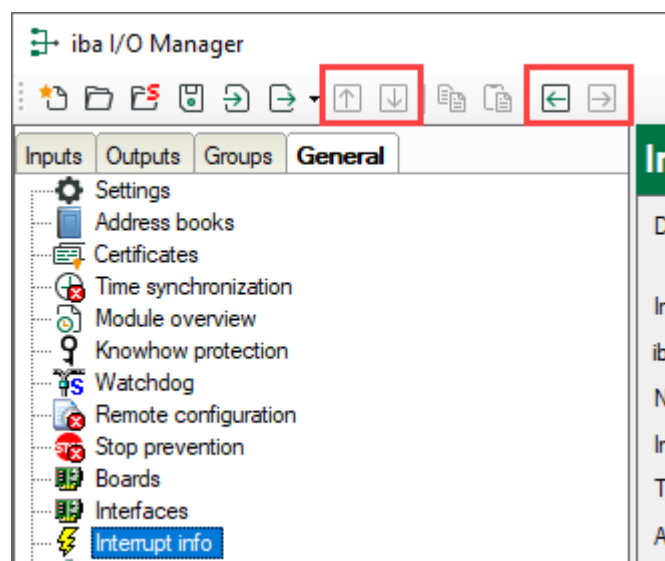
Apart from being able to filter by **Interface** and **Direction**, it's also possible to apply a custom filter to the grid by clicking the icon in a column header. In the above picture only modules are displayed that have more than 0 active analog signals. The totals at the bottom are also updated according to the currently displayed modules in the grid.



The Module overview can also be reached through the context menu of an interface in the I/O Manager; the Module overview will automatically only show modules belonging to the selected interface.

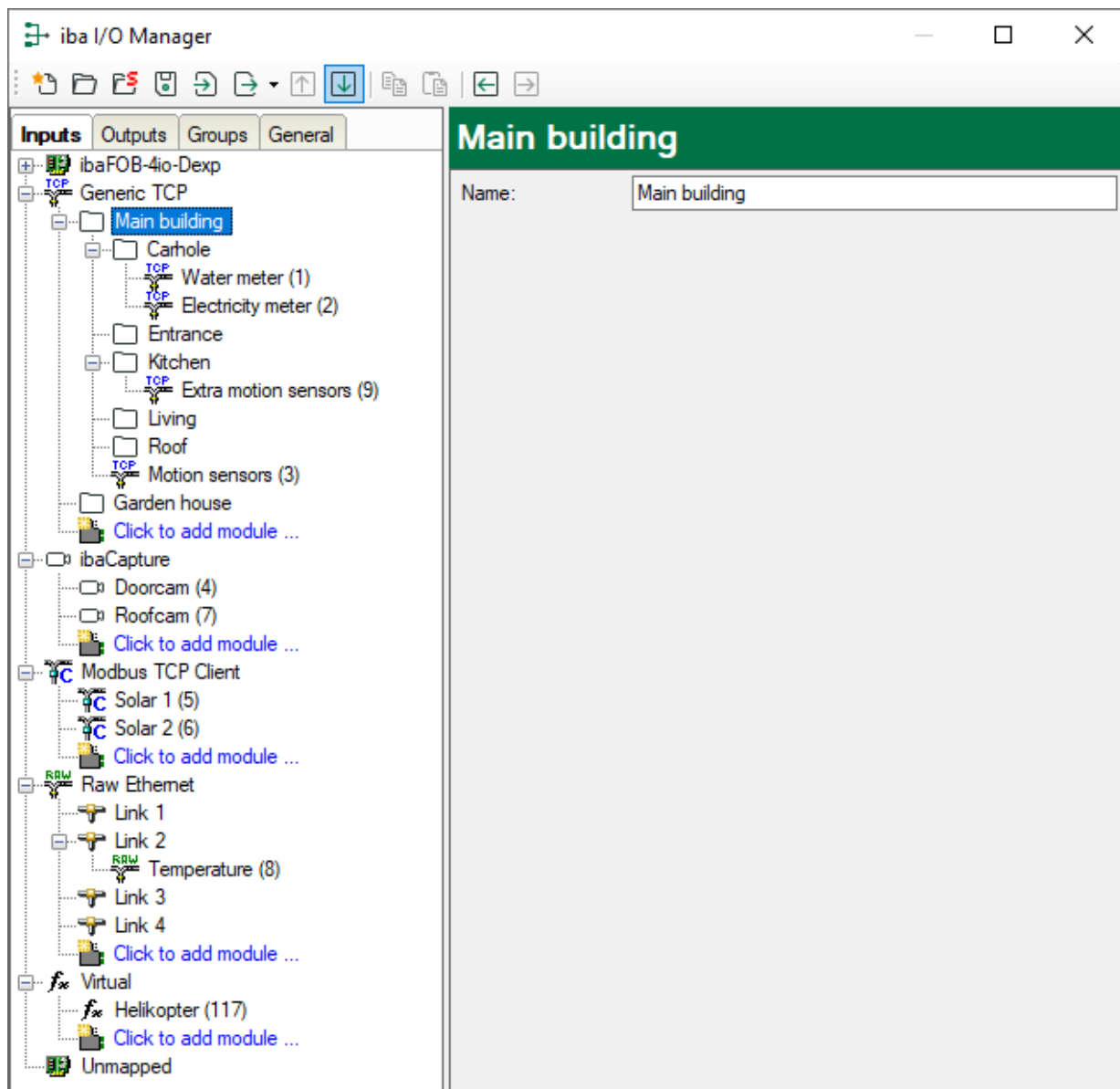
Another shortcut to the Module overview is using the general context menu of the **Inputs** or **Outputs** tab (this context menu can be activated by right-clicking somewhere where there's no node)

4.3 Navigation buttons



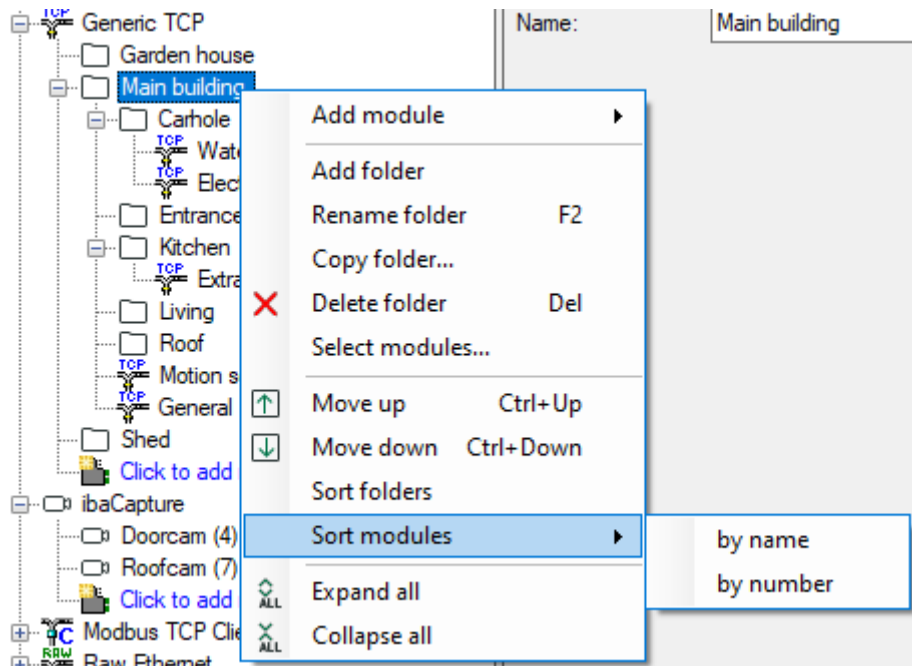
In the I/O Manager's toolbar 4 new buttons are now available. The up and down arrows can be used to move a module or folder up or down. The left and right arrows can be used to cycle through previously visited nodes (similar to e.g. an internet browser or Windows Explorer).

4.4 Module folders



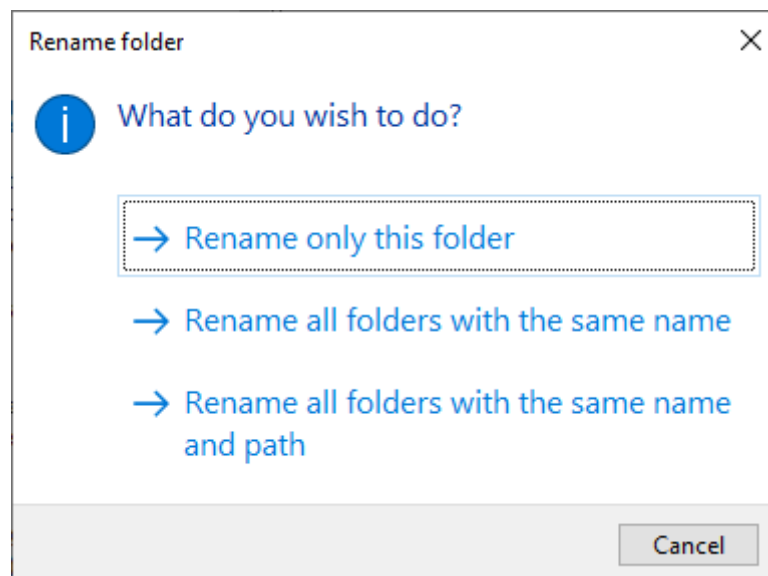
For most interfaces in the I/O Manager it is now possible to group modules into folders to help create a better overview or logical structure that's not strictly related to the input interface type. Each folder can contain 0 or more modules and 0 or more subfolders. An interface or interface link acts as a root folder.

Interfaces that do not support folders are e.g. ibaFOB boards because modules there are already ordered by link and slot number.



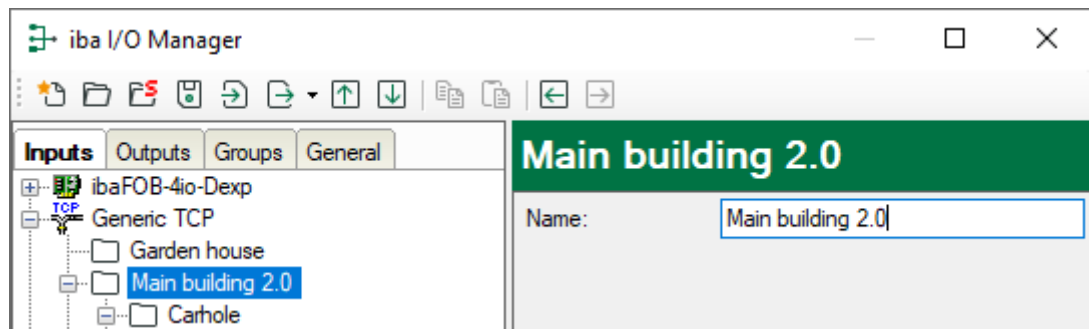
Most folder operations can be accomplished using a folder's (or interface (link)'s) context menu:

- **Add folder:** a new folder will be created as a subfolder of the selected folder or interface (link). The newly created node in the tree structure will be editable so you can immediately assign a name
- **Rename folder:** rename the current folder and optionally folders bearing the same name. After confirming the new folder name, the I/O Manager will check the entire configuration for folders with the same (new) name. In case duplicates are found the following dialog will appear:

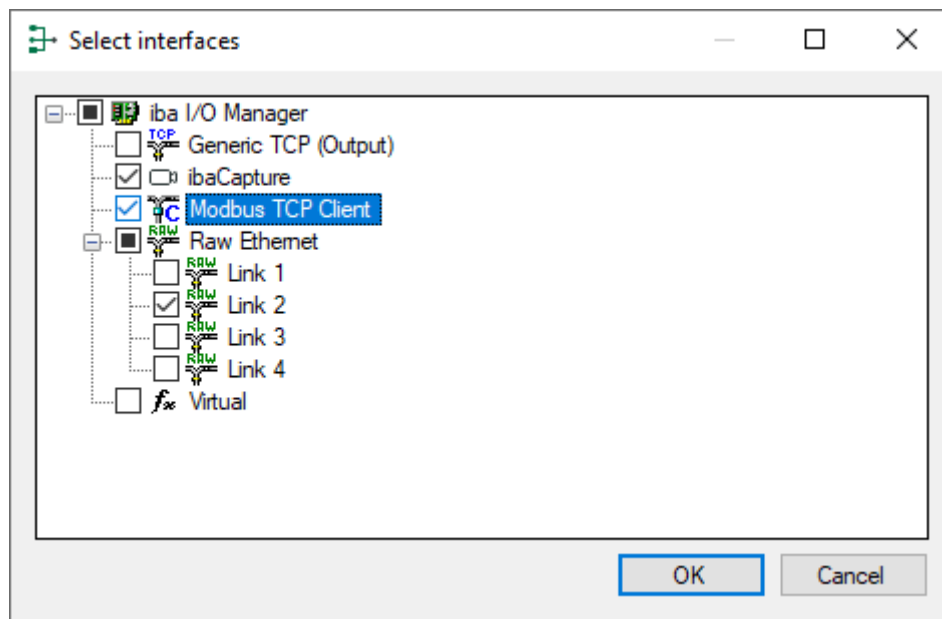


Since a folder is uniquely defined only by its name, two folders with the same name on the same level are not allowed.

Renaming a folder can also be done by selecting a folder and editing its name in the right panel:

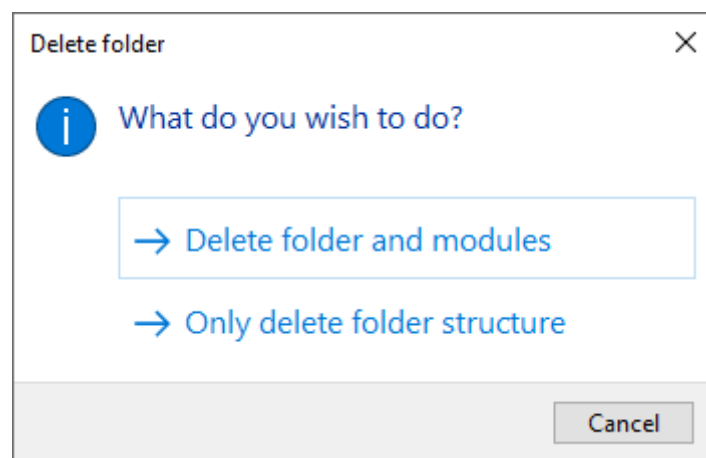


- **Copy folder...:** copies a folder and its subfolders to another interface (link). A dialog will appear to select to which interfaces or interface links the folder(s) should be copied.



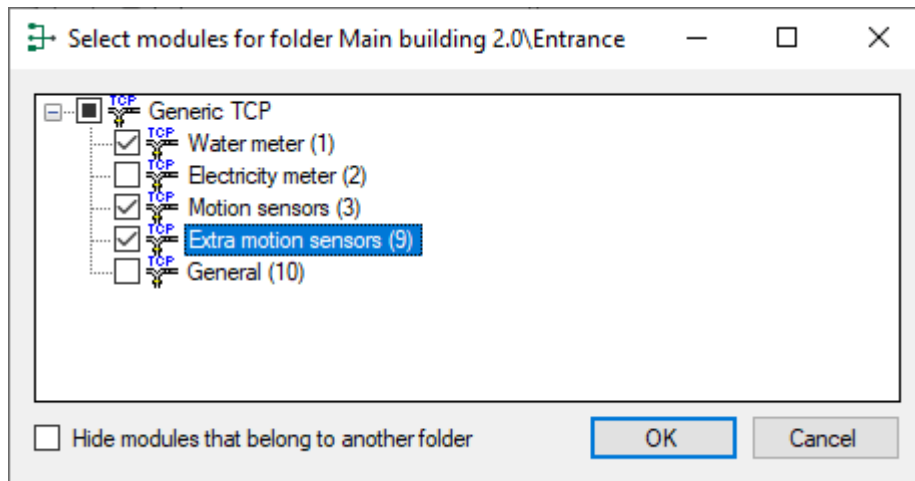
Since most modules are interface-specific, they are not copied to the new interfaces or interface links.

- **Delete folder:** deletes a folder and its subfolders and optionally its modules. In case the folder contains one or more modules the following dialog will appear:



In case only the folder structure is deleted the modules are moved to the selected folder's parent folder (or the interface (link)).

- **Select modules...:** instead of dragging and dropping module per module to a folder it's possible to do a batch move using this functionality

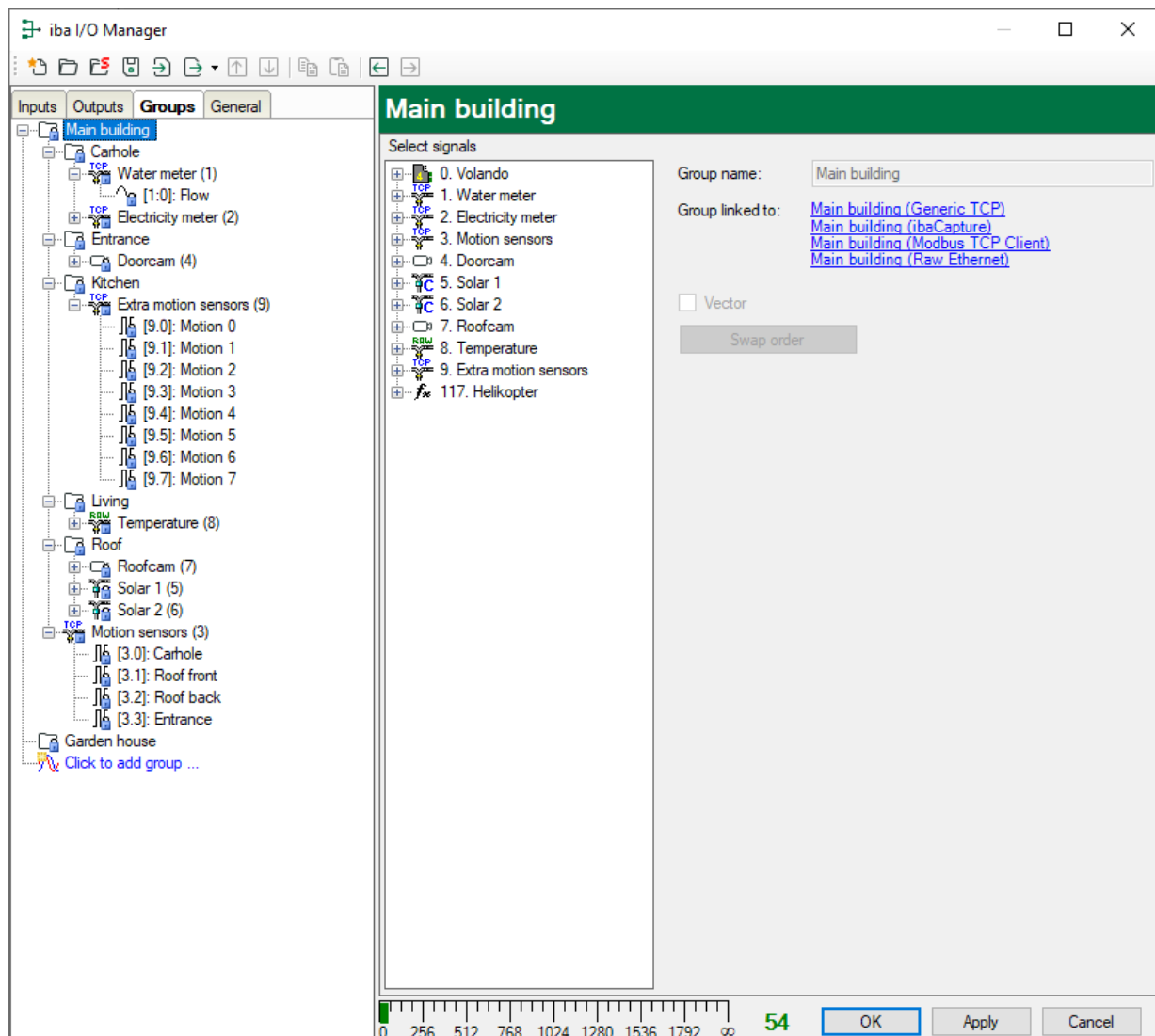


By default only modules that do not yet belong to any folder are listed. To show all modules, uncheck **Hide modules that belong to another folder** at the bottom left

- **Move up/move down:** moves a folder one position up/down if possible. This is now also possible for modules. The CTRL+Up and CTRL+Down keys can also be used to move folders and modules.
- **Sort folders:** sorts a folder's subfolders in alphabetical order
- **Sort modules:** sorts a folder's modules according to its name or its module number

Apart from those operations it is also possible to drag and drop folders between folder levels or to other interfaces or interface links. When moving a folder containing modules to another interface (link), the modules will first be moved to the folder's current parent folder (similar to **Copy folder** functionality).

Folders not only have the potential to create a better or more structured overview in the I/O Manager, they are also automatically converted into signal groups:

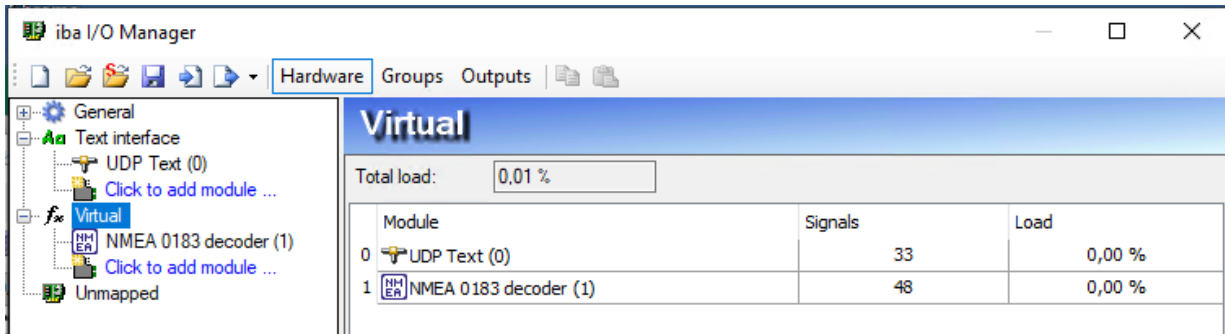


Signals in folders with the same path (relative to the root folder; i.e. the interface or interface link) are added to the same signal group.

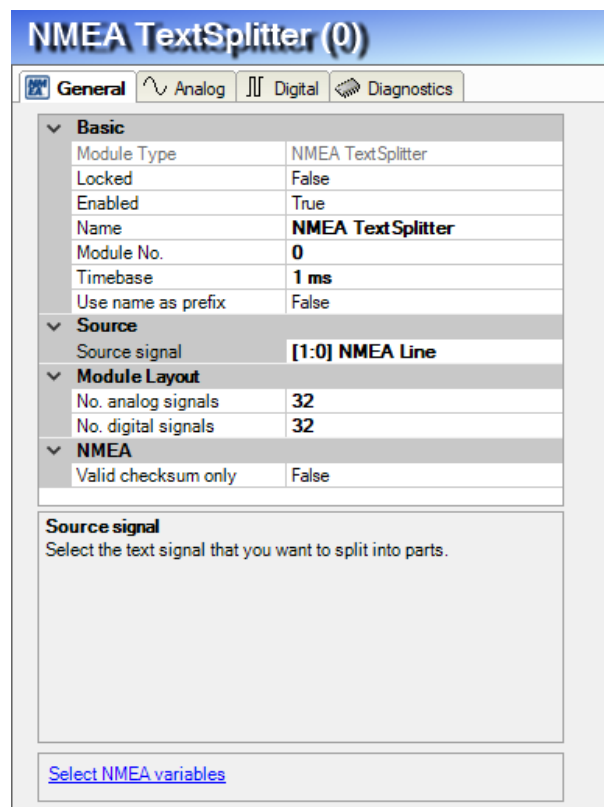
5 NMEA 0183 decoder module

The NMEA 0183 decoder in ibaPDA is a virtual module and used to analyze received NMEA 0183 strings and convert the contained values into formats usable in ibaPDA. The strings may be received from NMEA senders, e.g. by a Serial Text module or a TCP/UDP Text module.

The NEMA 0183 decoder module is a **virtual module**.

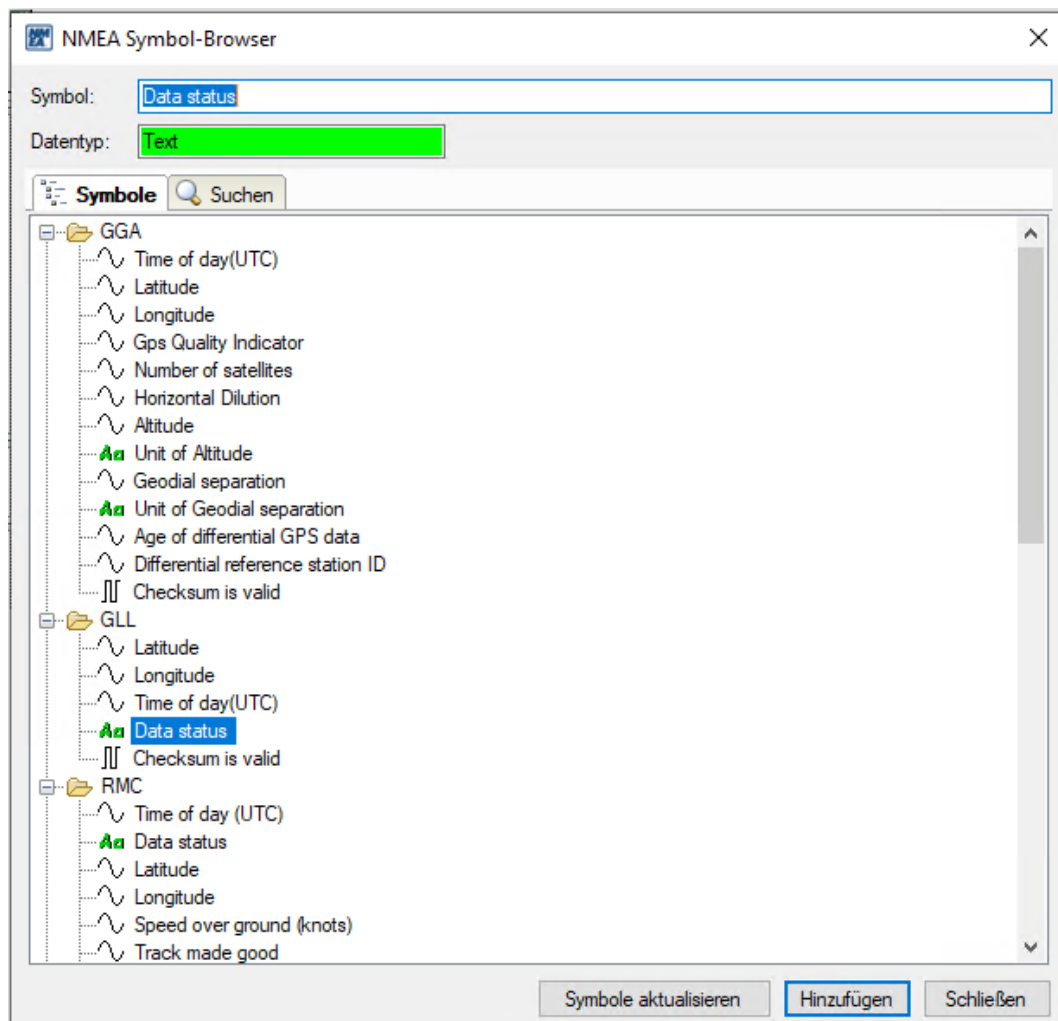


The **General** tab of a NMEA 0183 decoder module looks as follows:



Apart from the standard options that can be found on most other modules, the **Source signal** has to be configured. Optionally all messages with invalid check sum can be filtered. The validity of the checksum is also available as a digital signal.

New signals can be added by clicking **Select NMEA variables** at the bottom. This list is filled from a text file, which can be adapted to add other NMEA sentences. By default, the NMEA sentences GGA, GLL and RMC are available.



With the NMEA symbol browser you can easily add analog or digital signals to the NMEA 0183 decoder module, by double-clicking any variable, or selecting multiple variables and clicking on "Add".

NMEA TextSplitter (0)

General Analog Digital Diagnostics

	Name	Unit	Sentence	Field	NMEA data type	Active	Actual
4	GGA: Horizontal Dilution		GGA	8	Real	<input checked="" type="checkbox"/>	
5	GGA: Altitude		GGA	9	Real	<input checked="" type="checkbox"/>	
6	GLL: Latitude		GLL	1	PositionConverted	<input checked="" type="checkbox"/>	51,0266421
7	GLL: Longitude		GLL	3	PositionConverted	<input checked="" type="checkbox"/>	3,4511332
8	GLL: Time of day(UTC)		GLL	5	TimeConverted	<input checked="" type="checkbox"/>	62478
9	GLL: Data status		GLL	6	Text	<input checked="" type="checkbox"/>	A
10	GGA: CRC Valid		GGA	-1	Bool	<input type="checkbox"/>	
11	RMC: Time of day (UTC)		RMC	1	TimeConvertedAsString	<input checked="" type="checkbox"/>	
12	RMC: Data status		RMC	2	Text	<input checked="" type="checkbox"/>	
13	RMC: Longitude		RMC	5	PositionConverted	<input checked="" type="checkbox"/>	
14	RMC: Latitude		RMC	3	PositionConverted	<input checked="" type="checkbox"/>	
15	RMC: Speed over ground (knots)		RMC	7	Real	<input checked="" type="checkbox"/>	
16	RMC: Track made good		RMC	8	Real	<input checked="" type="checkbox"/>	
17	RMC: Date		RMC	9	DateConvertedAsString	<input checked="" type="checkbox"/>	
18	RMC: Magnetic variation (degrees)		RMC	10	Real	<input checked="" type="checkbox"/>	
19	RMC: Magnetic variation direction		RMC	11	Hemisphere	<input checked="" type="checkbox"/>	
20	RMC: 4		RMC	4	HemisphereConverted	<input checked="" type="checkbox"/>	
21	RMC: 6		RMC	6	HemisphereConverted	<input checked="" type="checkbox"/>	
22	RMC: Date		RMC	9	DateConverted	<input checked="" type="checkbox"/>	
23	GSV: 3		GSV	3	Int	<input checked="" type="checkbox"/>	
24				24	Invalid	<input type="checkbox"/>	

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.

```

$ G N G L L
5 1 0 2 . 6 6 4 2 1
N
0 0 3 4 5 . 1 1 3 3 2
E
1 7 2 1 1 8 . 0 0
A
D

```

The above figure shows an example of the Analog tab of an NMEA 0183 decoder module. You can open the NMEA symbol browser using the “...” button in the sentence column, as shown in line 16. You can also manually define a parameter not present in the symbol browser by typing in the sentence name and the field index, and selecting an appropriate NMEA data type, like it is done for line 23. Same behavior is available in the Digital tab.

As with other text splitter values, the definition tab offers a preview of the values based on the current input string.

NMEA TextSplitter (0)

General Analog Digital **Diagnostics**

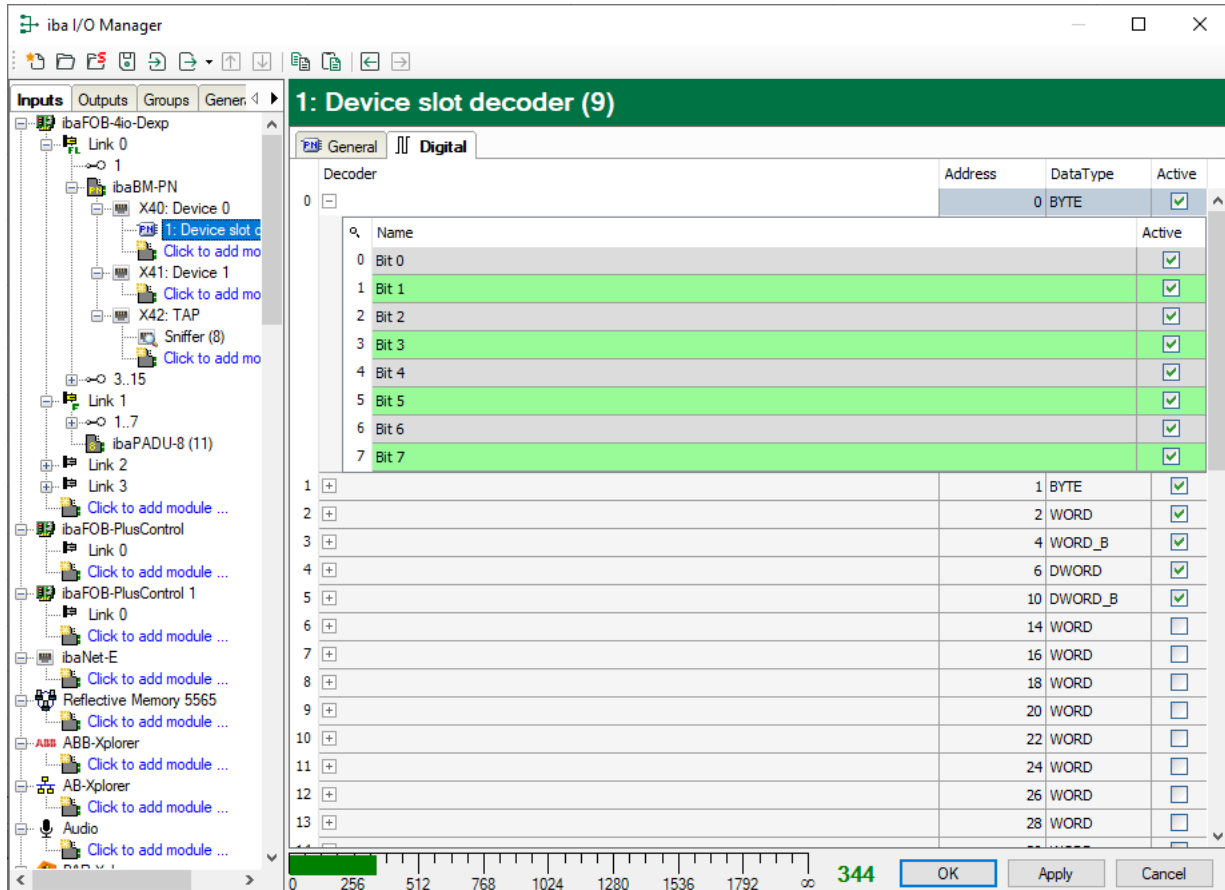
Analog values Digital values

	Name	Symbol	Datatype	Value
0	GGA: Time of day(UTC)	GGA.1	STRING	172159.00
1	GGA: Latitude	GGA.2	DOUBLE	51,0267
2	GGA: Longitude	GGA.4	DOUBLE	3,45113
3	GGA: Gps Quality Indicator	GGA.6	DINT	2
4	GGA: Horizontal Dilution	GGA.8	DOUBLE	0,86
5	GGA: Altitude	GGA.9	DOUBLE	22,3
6	GLL: Latitude	GLL.1	DOUBLE	51,0267
7	GLL: Longitude	GLL.3	DOUBLE	3,45113
8	GLL: Time of day(UTC)	GLL.5	DOUBLE	62519
9	GLL: Data status	GLL.6	STRING	A
10	GGA: CRC Valid	GGA.-1		
11	RMC: Time of day (UTC)	RMC.1	STRING	17:21:59,0000
12	RMC: Data status	RMC.2	STRING	A
13	RMC: Longitude	RMC.5	DOUBLE	3,45113
14	RMC: Latitude	RMC.3	DOUBLE	51,0267
15	RMC: Speed over groun...	RMC.7	DOUBLE	0,035
16	RMC: Track made good	RMC.8	DOUBLE	NaN
17	RMC: Date	RMC.9	STRING	19.02.2021
18	RMC: Magnetic variation ...	RMC.10	DOUBLE	NaN
19	RMC: Magnetic variation ...	RMC.11	DOUBLE	NaN

The current values of the requested topics can be monitored in the **Diagnostics** tab. There a further distinction is made between **Analog values** and **Digital values**.

6 Bit decoder modules

Some bit decoder modules have been extended with support for data types. This data type can be configured per decoder. This allows you to combine 8-, 16- and 32-bit decoders in a single module. You can also configure swapping by selecting the big-endian data types WORD_B and DWORD_B.



This feature is supported on the following module types:

- ibaBM-PN device slot decoder
- ibaBM-PN sniffer decoder
- ibaBM-ENetIP EtherNet/IP sniffer decoder

7 Extended ABB-Xplorer Interface

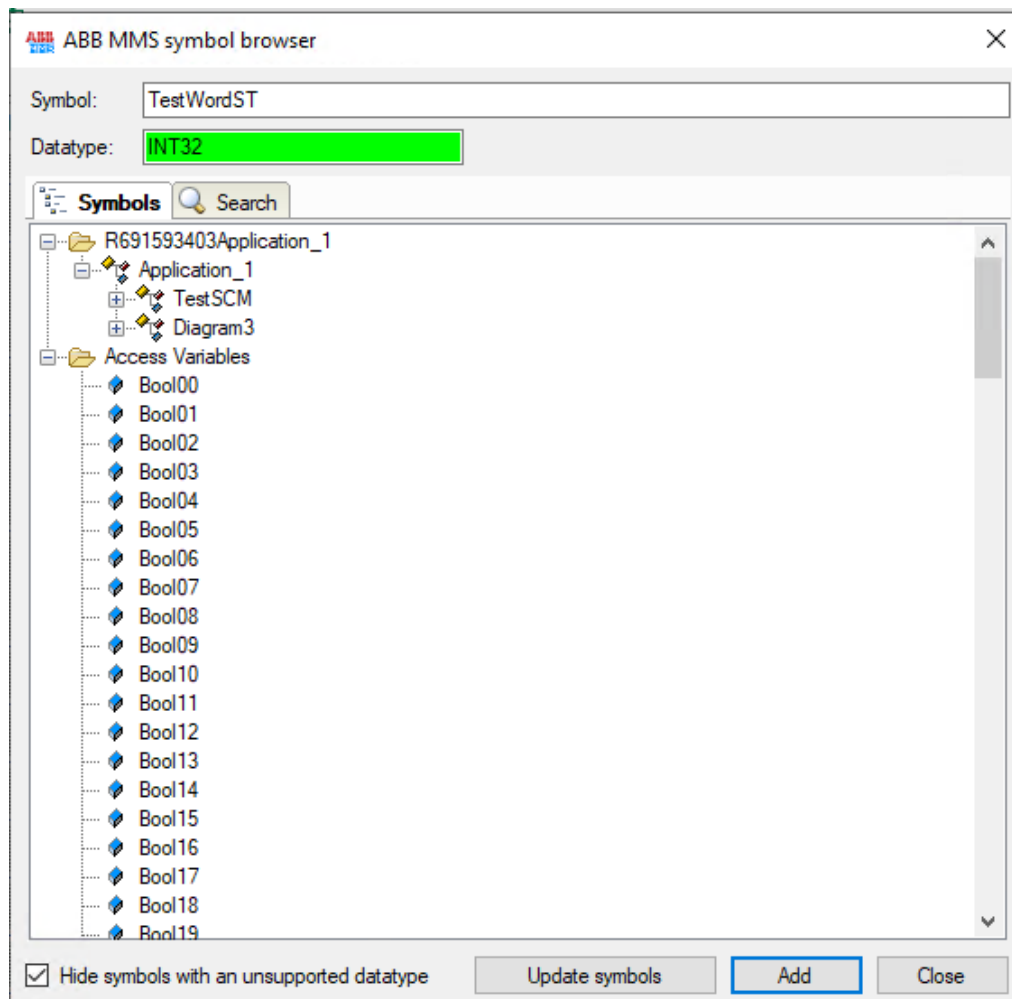
7.1 Using direct access

The ABB-Xplorer interface in ibaPDA was extended to use the project files of the ABB Compact control builder to access internal variables of the PLC. To use this, additional parameters have to be set in the connection tab of the ABB-Xplorer module. At first the direct access has to be enabled, and the base path to the project files has to be entered.

The screenshot shows the 'ABB-Xplorer MMS (0)' window with the 'Connection' tab selected. The 'Address' field contains '192.168.51.125', 'Timeout (s)' is '5', and 'Maximum number of objects to read in a single command' is '64'. The 'Base path to Control Builder projects' field contains 'C:\Users\daq\Documents\ABB-Projekte'. The 'Enable direct access' checkbox is checked. The 'Create address book offline' button is visible. The 'Test' button is also present.

On the PC where the ABB Compact control builder is installed, this path is usually “C:\ABB Industrial IT Data\Engineer IT Data\Compact Control Builder AC 800M\Projects”. This folder can either be used directly, if ibaPDA is running on the same computer, or by sharing the folder over network. For this, username and password to access the share can be entered in the dialog too. If the folders are copied to another place for ibaPDA to use them, the user has to make sure to update those copies whenever changes are done in the ABB Compact control builder and update the address book in ibaPDA, or access to the variables will fail, showing a message that the address book must be updated.

The folder selected must be the base path containing all the projects, the appropriate project folder is then selected automatically based on the information from the PLC.

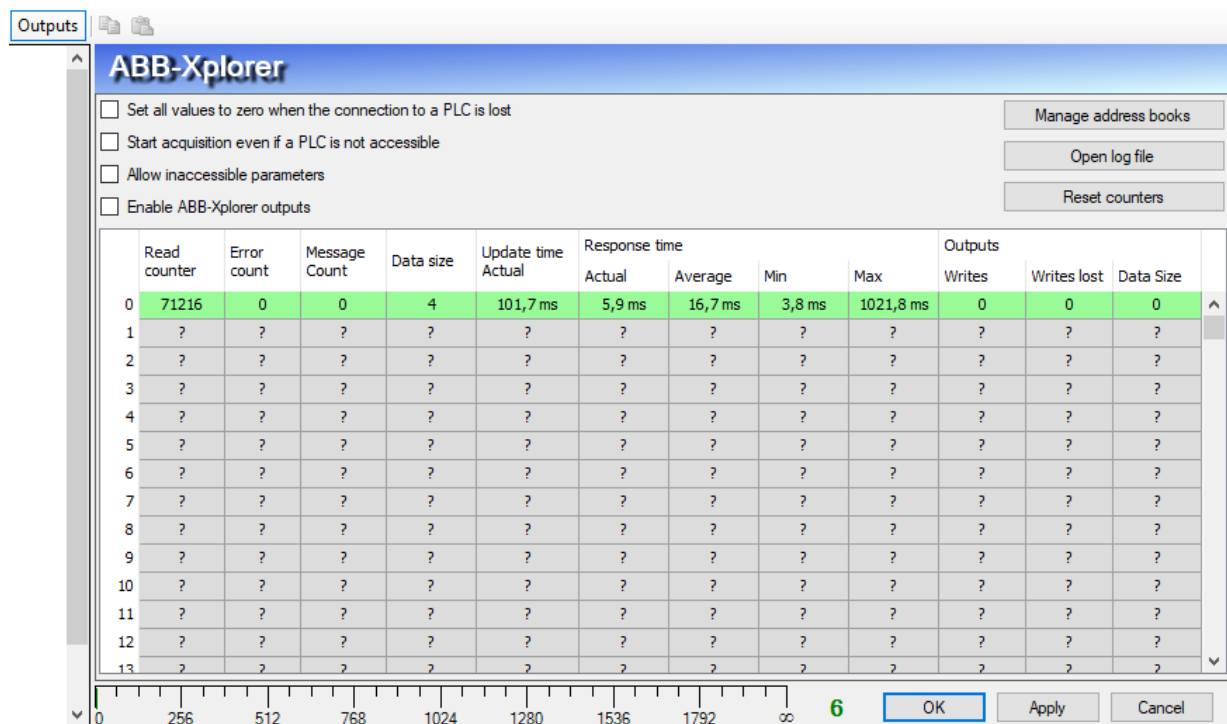


After creating the new address book, an additional branch will show up in the symbol browser containing the variables of the imported project, including named elements of structures. The symbol created from this contains the full path of the variable, in contrast to the direct name of the access variables.

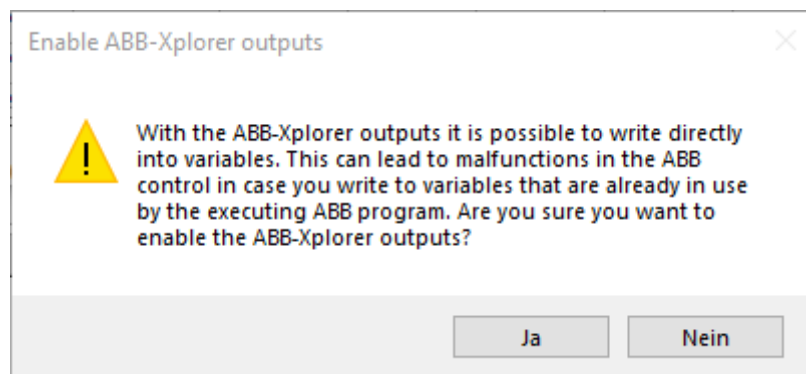
ABB-Xplorer MMS (0)							
	Name	Unit	Gain	Offset	Symbol	DataType	Active
275	Application_1.TestSC...		1	0	R691593403Application_1\Application_1.TestSCM._...	FLOAT	<input checked="" type="checkbox"/>
276	Application_1.TestSC...		1	0	R691593403Application_1\Application_1.TestSCM._...	FLOAT	<input checked="" type="checkbox"/>
277	Application_1.TestSC...		1	0	R691593403Application_1\Application_1.TestSCM._...	FLOAT	<input checked="" type="checkbox"/>
278	Application_1.TestSC...		1	0	R691593403Application_1\Application_1.TestSCM._...	FLOAT	<input checked="" type="checkbox"/>
279	Application_1.TestSC...		1	0	R691593403Application_1\Application_1.TestSCM._...	FLOAT	<input checked="" type="checkbox"/>
280	Application_1.TestSC...		1	0	R691593403Application_1\Application_1.TestSCM._...	FLOAT	<input checked="" type="checkbox"/>
281	TestDintFast		1	0	TestDintFast	DINT	<input checked="" type="checkbox"/>
282	TestDintNormal		1	0	TestDintNormal	DINT	<input checked="" type="checkbox"/>
283	TestDintST		1	0	TestDintST	DINT	<input checked="" type="checkbox"/>
284	TestDwordFast		1	0	TestDwordFast	DINT	<input checked="" type="checkbox"/>
285	TestDwordNormal		1	0	TestDwordNormal	DINT	<input checked="" type="checkbox"/>
286	TestDwordST		1	0	TestDwordST	DINT	<input checked="" type="checkbox"/>
287	TestIntFast		1	0	TestIntFast	DINT	<input checked="" type="checkbox"/>
288	TestIntNormal		1	0	TestIntNormal	DINT	<input checked="" type="checkbox"/>

7.2 Using outputs

When switching to outputs, the ABB-Xplorer module can also be used to send data to the PLC. To enable this feature, check the “Enable ABB-Xplorer outputs” checkbox in the interface control.



A warning to the user shows up, explaining that the ABB system contains no access control, so all variables can be written, which may cause malfunctions of the program running on the PLC. The user is fully responsible for this, and must choose only variables intended for writing from ibaPDA.



To define the output signals, the symbol browser can be used, which fills in the access path, a symbol name based on the path, and the data type associated with this symbol. Now standard expressions can be used to supply values to this output.

ABB-Xplorer MMS (0)

General

Connection

Analog

Digital

Diagnostics

	Name	Expression	Symbol	DataType	Active
0	Real255	f_{sc} GenerateSignal(0,10,1,1)	R691593107Application_1\Application_1.TestSC...	FLOAT	<input checked="" type="checkbox"/>
1		f_{sc}		INT	<input type="checkbox"/>
2		f_{sc}		INT	<input type="checkbox"/>
3		f_{sc}		INT	<input type="checkbox"/>
4		f_{sc}		INT	<input type="checkbox"/>
5		f_{sc}		INT	<input type="checkbox"/>
6		f_{sc}		INT	<input type="checkbox"/>

02565127681024128015361792∞

6

OK

Apply

Cancel

8 MQTT interface outputs

The MQTT interface now supports outputs.

MQTT (13)															
<div>General Connection Analog Digital Diagnostics</div>															
<div>Basic</div> <table> <tr><td>Module Type</td><td>MQTT</td></tr> <tr><td>Locked</td><td>False</td></tr> <tr><td>Enabled</td><td>True</td></tr> <tr><td>Name</td><td>MQTT</td></tr> <tr><td>Module No.</td><td>13</td></tr> <tr><td>Calculation timebase</td><td>10 ms</td></tr> <tr><td>Use name as prefix</td><td>False</td></tr> </table>		Module Type	MQTT	Locked	False	Enabled	True	Name	MQTT	Module No.	13	Calculation timebase	10 ms	Use name as prefix	False
Module Type	MQTT														
Locked	False														
Enabled	True														
Name	MQTT														
Module No.	13														
Calculation timebase	10 ms														
Use name as prefix	False														
<div>Module Layout</div> <table> <tr><td>No. analog output signals</td><td>32</td></tr> <tr><td>No. digital output signals</td><td>32</td></tr> </table>		No. analog output signals	32	No. digital output signals	32										
No. analog output signals	32														
No. digital output signals	32														
<div>Mqtt</div> <table> <tr><td>Send Bool as number</td><td>Float</td></tr> <tr><td>Write mode</td><td>On change</td></tr> </table>		Send Bool as number	Float	Write mode	On change										
Send Bool as number	Float														
Write mode	On change														
<div>Processing</div> <table> <tr><td>Decimal point</td><td>Point</td></tr> <tr><td>Message Format</td><td>Text</td></tr> </table>		Decimal point	Point	Message Format	Text										
Decimal point	Point														
Message Format	Text														
<div>Name</div> <p>The name of the module.</p>															

Apart from the standard settings, some additional options show up when switching to the output view. The number of analog and digital outputs topics can be configured here. There are 3 possible write modes:

- Cyclic: every cycle based on the calculation timebase
- On change: whenever the value of the signal changes
- On trigger: on every rising edge of the configured trigger signal

For boolean values using the message format "Text" it can be configured if they are written as float values ("1.0" and "0.0") or as boolean values ("true" and "false").

9 MQTT timebased data store

9.1 Buffered writing

For the data formats *JSON* and *Protobuf* the MQTT data store now supports combining multiple values over a period of time in one message.

MQTT 1 2883 - Topics

+ ✕ 📄 📄 ⬆ ⬇

Send digital signals in JSON as true / false

Decimal point in text format Point

☐ Send only changed values

☐ Compact JSON format

Number of active topics: 1

Number of active signals: 3

Active	Name	Metadata	Signal reference	Data format	Combine values (ms)	Retain	Signals
<input type="checkbox"/>	Topic_100ms	Timestamp, Signal...	Signal name	JSON (per signal)	0	<input type="checkbox"/>	14A + 2D = 16
<input checked="" type="checkbox"/>	Topic_1s	Timestamp, Signal...	Signal name	JSON (grouped)	3000	<input checked="" type="checkbox"/>	3A + 0D = 3

If you enter a value greater than 0 in *Combine values* column then the data format changes and data from more than one time stamp is written to the message. A setting of 0 disables this feature.

9.1.1 Data format JSON (grouped) with combined values

Example: Combining values from 3 signals over 3000ms

```
{
  "Millis.ID": "[0:1]",
  "Millis.Name": "Millis",
  "Seconds.ID": "[0:2]",
  "Seconds.Name": "Seconds",
  "Minutes.ID": "[0:3]",
  "Minutes.Name": "Minutes",
  "Values": [
    {
      "Timestamp": "2022-06-03T10:43:06.0291901Z",
      "Millis": 29,
      "Seconds": 6,
      "Minutes": 43
    },
    {
      "Timestamp": "2022-06-03T10:43:07.0292059Z",
      "Millis": 29,
      "Seconds": 7,
      "Minutes": 43
    },
    {
      "Timestamp": "2022-06-03T10:43:08.0292182Z",
      "Millis": 29,
      "Seconds": 8,
      "Minutes": 43
    }
  ]
}
```

9.1.2 Data format JSON (per signal) with combined values

Example: Combining values from one signal over 3000ms.

```
{
  "Signal": "Seconds",
  "ID": "[0:2]",
  "Name": "Seconds",
  "Values": [
    {
      "Timestamp": "2022-06-03T10:45:06.0308501Z",
      "Value": 6
    },
    {
      "Timestamp": "2022-06-03T10:45:07.0308624Z",
      "Value": 7
    },
    {
      "Timestamp": "2022-06-03T10:45:08.0308765Z",
      "Value": 8
    }
  ]
}
```

9.2 Protocol Buffers Encoding

The MQTT timebased data store is extended to optionally use protobuf encoding. To activate that, the data format “Protobuf (grouped)” has to be selected in the topic definition.

MQTT timebased data store 1 - Topics

Send digital signals in JSON as:

Decimal point in text format:

☐ Send only changed values

☒ Compact JSON format

Number of active topics:

Number of active signals:

Acti...	Name	Metadata	Signal reference	Data format	Combine values ...	Retain	Signals
<input checked="" type="checkbox"/>	Topic	Unit, Comment 1, Comme...	Signal ID	Protobuf (grouped)	0	<input type="checkbox"/>	3A + 0D = 3
<input type="checkbox"/>	Topic_1	Unit, Comment 1, Timesta...	Signal ID	Single value as text	0	<input type="checkbox"/>	3A + 0D = 3

In the signal definition, select the analog and digital signals to be stored in the topic. Additional metadata can be selected. Timestamp and Identifier are not available for selection, as they are an integral part of the protobuf definition. Protobuf supports the buffered writing, using a structure similar to the PDA .DAT-file format, where a block starts with a timestamp, followed by multiple values with a fixed time base defined by the time base of the used profile.

The protobuf definition file that can be used for decoding the values is installed with the ibaPDA server, under “C:\Program Files (x86)\iba\ibaPDA\Server\AuxiliaryFiles\Protobuf\ibaPDA_data.proto”.

9.3 Separator for text formats

When writing floating point values to an MQTT broker, ibaPDA can be configured which decimal point character to use. In the topic configuration you can select if the decimal point character is either “Point” or “Comma” for topics using a data format “Single Value as text”, “JSON (grouped)” or “JSON (per signal)”.

10 InfluxDB timebased data store

The new *InfluxDB timebased data store* allows you to write timebased signal data to an InfluxDB Server. InfluxDB v2.x is supported. The data is transferred using the InfluxDB line protocol.

10.1 Licensing

The licensing concept is identical to other existing streaming data stores (e.g. MQTT, Kafka etc.). Different licenses are available allowing you to write a certain total number of signals. The number of data stores you use for writing is not limited. All signals written in all InfluxDB data stores are added and counted against the license. The following license products are available:

- *ibaPDA-Data-Store-InfluxDB-64* (30.671060), allows you to write 64 signals in total
- *ibaPDA-Data-Store-InfluxDB-256* (30.671061), allows you to write 256 signals in total
- *ibaPDA-Data-Store-InfluxDB-1024* (30.671062), allows you to write 1024 signals in total

Licensing is possible on MARX and WIBU dongle systems.

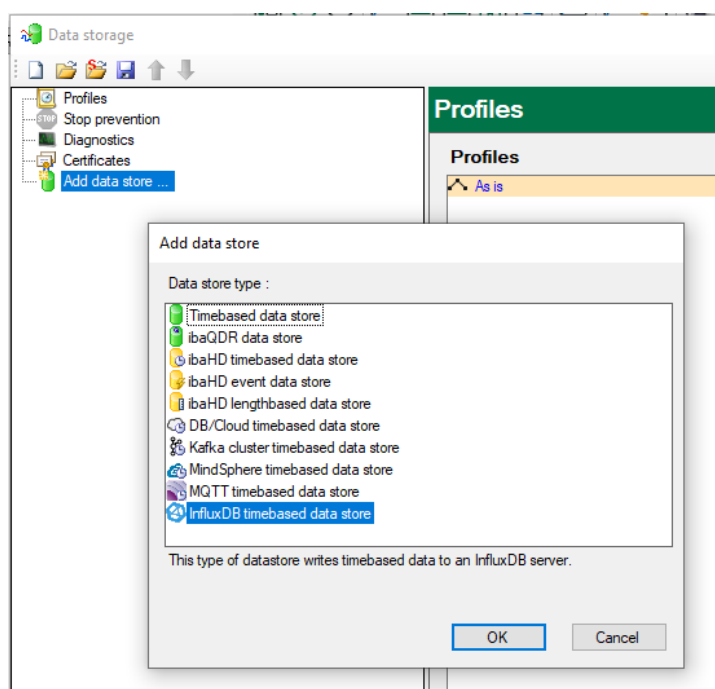
10.2 Preparation InfluxDB server

In order to be able to write data to an InfluxDB, some information regarding your InfluxDB server is required to do the necessary configurations in ibaPDA.

- The InfluxDB server address and port (default 8086) needs to be known to establish a connection from ibaPDA
- The InfluxDB organization needs to be known.
- Within your InfluxDB organization you need to prepare the buckets you want to use to store the data.
- An API token with read and write rights on the buckets is required. Access via user credentials is not supported in ibaPDA.

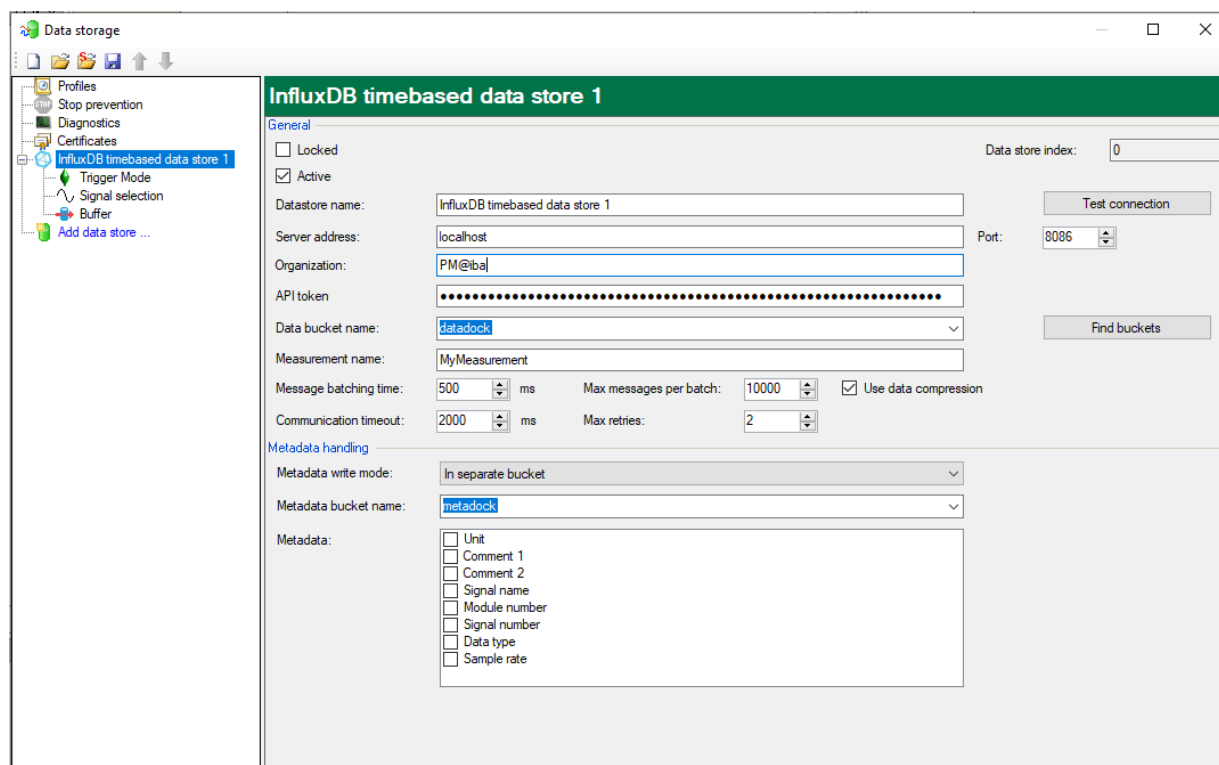
10.3 Configuration in ibaPDA

To add an *InfluxDB timebased data store* click on *Add data store...* in the Data storage configuration, select the InfluxDB timebased data store and click <OK>.



10.3.1 InfluxDB specific settings

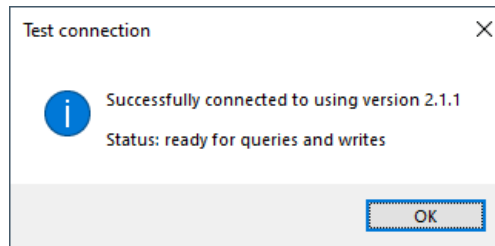
Several general configuration options have to be filled in the newly added InfluxDB timebased data store:



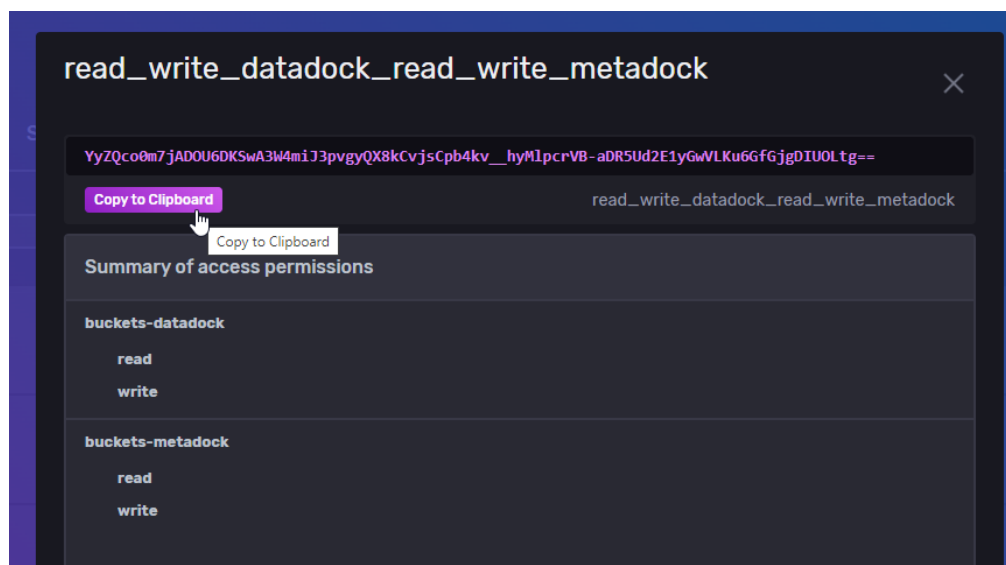
- **Locked:** data store can be locked in order to prevent an accidental or unauthorized change of settings.
- **Active:** A data store must be enabled in order to work. However, you can configure various data stores and disable data stores that are not required.

- **Data store index:** Unique index of all existing InfluxDB data stores. You need to reference this index e.g. in the virtual function *DataStoreInfoInflux()* for generating diagnostic data for a specific InfluxDB data store.
- **Data store name:** You can enter a name for the data store here. Data store names have to be unique.
- **Server address:** The IP address or hostname of the InfluxDB server
- **Port:** Port to use for the connection. The standard port is 8086.

Click on the button <Test connection> to verify if ibaPDA can establish a connection to your InfluxDB server using the configured server address and port number.



- **Organization:** The organization in your InfluxDB you want to use
- **API token:** The API token you enter here needs to have read and write rights to the data bucket and, if used, the metadata bucket. The easiest way to enter the API token is to log into the InfluxDB web interface, go to the API token configuration and use the <Copy to clipboard> button to copy & paste your API token into the ibaPDA configuration.



- **Data bucket name:** The name of the bucket you want to store the data in.
Use the button <Find buckets> to fill the drop-down list with all available buckets. Then select the data bucket you want to use from the drop-down list.
Only buckets will be found (and can be used) that belong to the configured organization and have read rights assigned for the API token in use.
- **Measurement name:** Name for the current measurement
- **Message batching time, Max messages per batch:** To reduce the number of single telegrams sent to the InfluxDB server, multiple data points are collected and sent as a

packet. This is done when either the message batching time expires, or the number of messages exceeds the max messages per batch number.

- **Use data compression:** to reduce the size of the data packets transferred, data compression can be activated.
- **Communication timeout:** Time until a telegram sent to the server is regarded as not successfully delivered.
- **Max retries:** Number of retries for a not successfully delivered message until the connection to the InfluxDB server is regarded as broken.

In the lower section it is configured how metadata should be handled.

Metadata in general is superposed data describing other data, in our case the individual signals written to the InfluxDB.

Metadata write mode: There are three ways to treat metadata. All have pros and cons. It depends on the use case which one to use.

1. **No metadata:** Only the data bucket is used. No signal metadata is written, except for the ibaPDA signal ID.

The signal ID is written as a tag key not as a field key. Tag keys are indexed in InfluxDB. Therefore, queries on tags are faster than queries on fields. In this write mode the least data is transmitted via the line protocol and the least space is required for data storage in InfluxDB. The filtering options in queries are limited on the other hand, since only the signal ID is available as meta data.

2. **In data bucket:** Here you can choose to write additional metadata (the signal ID by default is always written). All additional meta data will be written into the data bucket as extra tag keys. Choose via checkboxes which metadata you want to include:

- Unit
- Comment1 and Comment2
- Signal name
- Module number
- Signal number
- Data type
- Sample rate in ms

In this write mode all meta data needs to be transmitted via the line protocol for each data point. Also, more space is required for data storage in InfluxDB. The advantage is that time series data as well as all metadata is kept in one single bucket.

3. **In separate bucket:** The additional metadata and the signal ID is written into a separate bucket. Metadata is written only once when either the IO configuration or the data store configuration is applied. Via the signal ID and the timestamp, the metadata and the time series data from both buckets can be correlated.

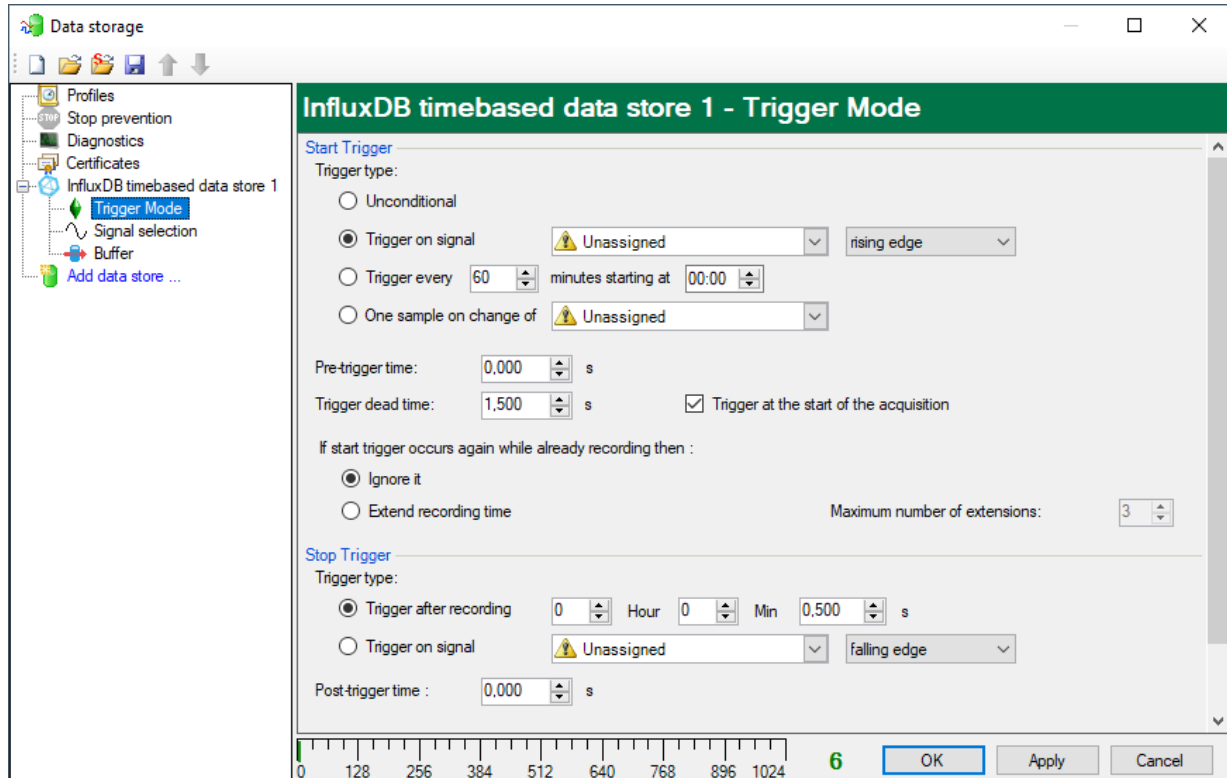
This way less data needs to be transferred and stored compared to the write mode “In data bucket”. The downside is that the analysis e.g. via Flux is more complex.

Metadata bucket name: The name of the bucket you want to store the metadata when using the write mode “In separate bucket”.

Use the button <Find buckets> to fill the drop-down list with all available buckets. Then select the metadata bucket you want to use from the drop-down list.

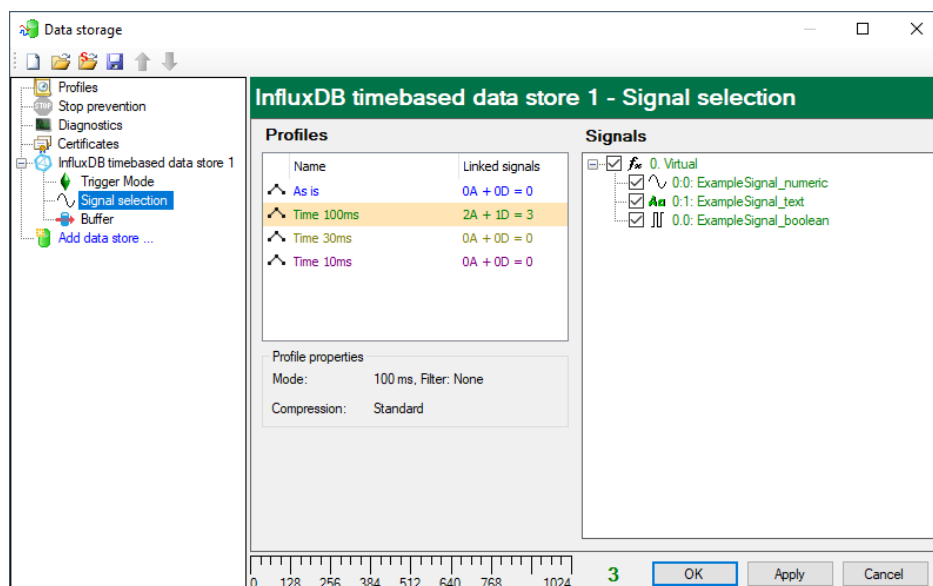
10.3.2 Trigger Mode

In the node *Trigger Mode* you adjust if the data is written continuously, or only on certain conditions. The available options are the same as for other data store types.



10.3.3 Signal selection

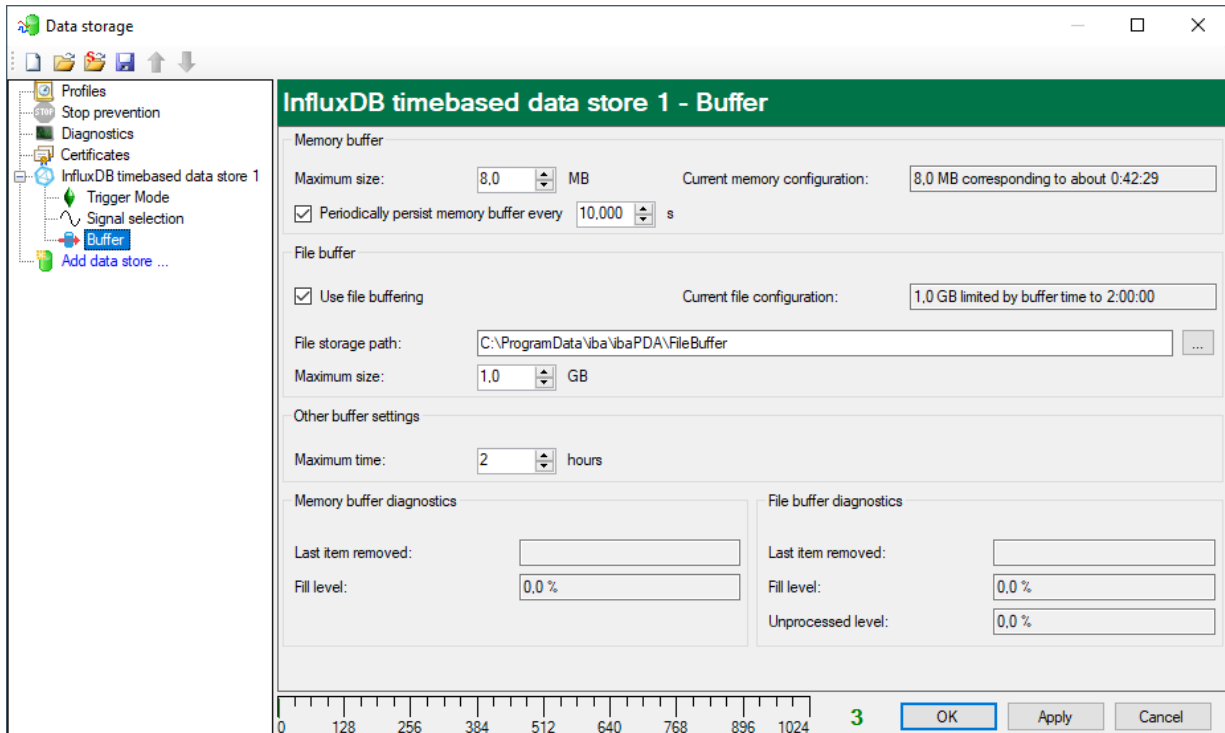
In the node *Signal selection* you select which of the signals you want to write into the InfluxDB using the selected profile. The standard profile type *Time* can be used for the signal selection. The functionality is identical to other data store types.



In general the trigger mode, the number of signals as well as the time base (defined within the profile) should be chosen according to the use case, considering network loads and the amount of data generated in the target InfluxDB.

10.3.4 Buffer

Like for other data store types used for streaming data to external systems a buffer functionality is available to overcome connection losses and to balance the network overloads. The functionality is identical to other data store types.



10.4 Example data models

The following small example demonstrates what the data in InfluxDB will look like when using the different metadata write modes. Three example signals are configured in a module with number 0 and will be written to InfluxDB applying the different meta data write modes. The three signals are:

- The signal with ID [0:0] has a numeric value
- The signal with ID [0:1] has a text value
- The signal with ID [0:0] has a boolean value

Virtual (0)							
	Name	Expression	Unit	Active	Actual	Comment 1	Comment 2
0	ExampleSignal_numeric	f_x GenerateSignal(0)	mm	<input checked="" type="checkbox"/>	-3,03035 mm	MyComment1 [0:0]	MyComment2 [0:0]
1	ExampleSignal_text	f_x GetSystemTimeAsText()		<input checked="" type="checkbox"/>	2022-01-13 18:38:03	MyComment1 [0:1]	MyComment2 [0:1]
		f_x		<input checked="" type="checkbox"/>			

Virtual (0)					
General		Digital			
Name	Expression	Active	Actual	Comment 1	Comment 2
0 ExampleSignal_boolean	GenerateSignal(0) > 5		1	MyComment1 [0.0]	MyComment2 [0.0]

Metadata write mode 'No metadata'

The column `_value` contains the signal value for each written timestamp. The field key (column `_field`) shows an entry `value`, `value_t` or `value_b` depending on if it is a numeric, text or boolean signal value. There is just one tag key `SignalId` filled with the signal ID. A tag key is indexed for faster query execution.

_time	_value	SignalId	_field	_measurement
13/01/2022 18:47:33.810	-2.36499	[0:0]	value	MyMeasurement
13/01/2022 18:47:33.910	-7.624425	[0:0]	value	MyMeasurement
13/01/2022 18:47:34.010	-9.971589	[0:0]	value	MyMeasurement
13/01/2022 18:47:34.110	-8.509945	[0:0]	value	MyMeasurement
13/01/2022 18:47:34.210	-3.797791	[0:0]	value	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 18:47:33.810	2022-01-13 18:47:33	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:33.910	2022-01-13 18:47:33	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:34.010	2022-01-13 18:47:34	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:34.110	2022-01-13 18:47:34	[0:1]	value_t	MyMeasurement
13/01/2022 18:47:34.210	2022-01-13 18:47:34	[0:1]	value_t	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 18:47:33.810	false	[0:0]	value_b	MyMeasurement
13/01/2022 18:47:33.910	false	[0:0]	value_b	MyMeasurement
13/01/2022 18:47:34.010	false	[0:0]	value_b	MyMeasurement
13/01/2022 18:47:34.110	false	[0:0]	value_b	MyMeasurement
13/01/2022 18:47:34.210	false	[0:0]	value_b	MyMeasurement

Metadata write mode 'In data bucket'

As an example three additional metadata values are written in the data bucket (Unit, Comment 1, Signal name):

Metadata handling

Metadata write mode: In data bucket

Metadata:

- ☒ Unit
- ☒ Comment 1
- ☐ Comment 2
- ☒ Signal name
- ☐ Module number
- ☐ Signal number
- ☐ Data type
- ☐ Sample rate

Compared to the first example additional tag key columns for the metadata values appear: *Unit*, *Comment1*, *SignalName*. Normally the metadata doesn't change very often. This means unchanged data is transmitted and stored repetitively.

_time	_value	Comment1	SignalId	SignalName	_field	_measurement	Unit
13/01/2022 18:58:...	-8.589945	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	-3.797791	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	2.36499	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	7.624425	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm
13/01/2022 18:58:...	9.971589	MyComment1 [0:0]	[0:0]	ExampleSignal_nu...	value	MyMeasurement	mm

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 18:58:22.1...	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22...	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22...	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22...	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement
13/01/2022 18:58:22...	2022-01-13 18:58:22	MyComment1 [0:1]	[0:1]	ExampleSignal_text	value_t	MyMeasurement

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 18:58:22.1...	false	MyComment1 [0:0]	[0:0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22...	false	MyComment1 [0:0]	[0:0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22...	false	MyComment1 [0:0]	[0:0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22...	true	MyComment1 [0:0]	[0:0]	ExampleSignal_boole...	value_b	MyMeasurement
13/01/2022 18:58:22...	true	MyComment1 [0:0]	[0:0]	ExampleSignal_boole...	value_b	MyMeasurement

Metadata write mode 'In separate bucket'

Again, three additional metadata values are written, but this time in a separate metadata bucket (Unit, Comment 1, Signal name):

Metadata handling

Metadata write mode: In separate bucket

Metadata bucket name: metadock

Metadata:

- ☒ Unit
- ☒ Comment 1
- ☐ Comment 2
- ☒ Signal name
- ☐ Module number
- ☐ Signal number
- ☐ Data type
- ☐ Sample rate

The data bucket now again looks the same as when using the write mode 'No metadata':

_time	_value	SignalId	_field	_measurement
13/01/2022 19:08:57.950	-7.624425	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.050	-9.971589	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.150	-8.589945	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.250	-3.797791	[0:0]	value	MyMeasurement
13/01/2022 19:08:58.350	2.36499	[0:0]	value	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 19:08:57.950	2022-01-13 19:08:57	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.050	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.150	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.250	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement
13/01/2022 19:08:58.350	2022-01-13 19:08:58	[0:1]	value_t	MyMeasurement

_time	_value	SignalId	_field	_measurement
13/01/2022 19:08:57.950	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.050	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.150	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.250	false	[0.0]	value_b	MyMeasurement
13/01/2022 19:08:58.350	false	[0.0]	value_b	MyMeasurement

The metadata bucket shows now a single metadata entry per signal for the point of time where the acquisition was started. Metadata can change only when the configuration was changed.

The column *_value* contains the signal ID. The field key (column *_field*) is always *SignalId*. The tag key *SignalId* always exists. The other tag key columns depend on the metadata that was configured. In this example again three additional tag key columns *Unit*, *Comment1*, *SignalName* exist.

_time	_value	Comment1	SignalId	SignalName	_field	_measurement	Unit
13/01/2022 19:0...	[0:0]	MyComment1 [0:0]	[0:0]	ExampleSignal_...	SignalId	MyMeasurement	mm

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 19:08:4...	[0:1]	MyComment1 [0:1]	[0:1]	ExampleSignal_text	SignalId	MyMeasurement

_time	_value	Comment1	SignalId	SignalName	_field	_measurement
13/01/2022 19:08:4...	[0:0]	MyComment1 [0:0]	[0:0]	ExampleSignal_bool...	SignalId	MyMeasurement

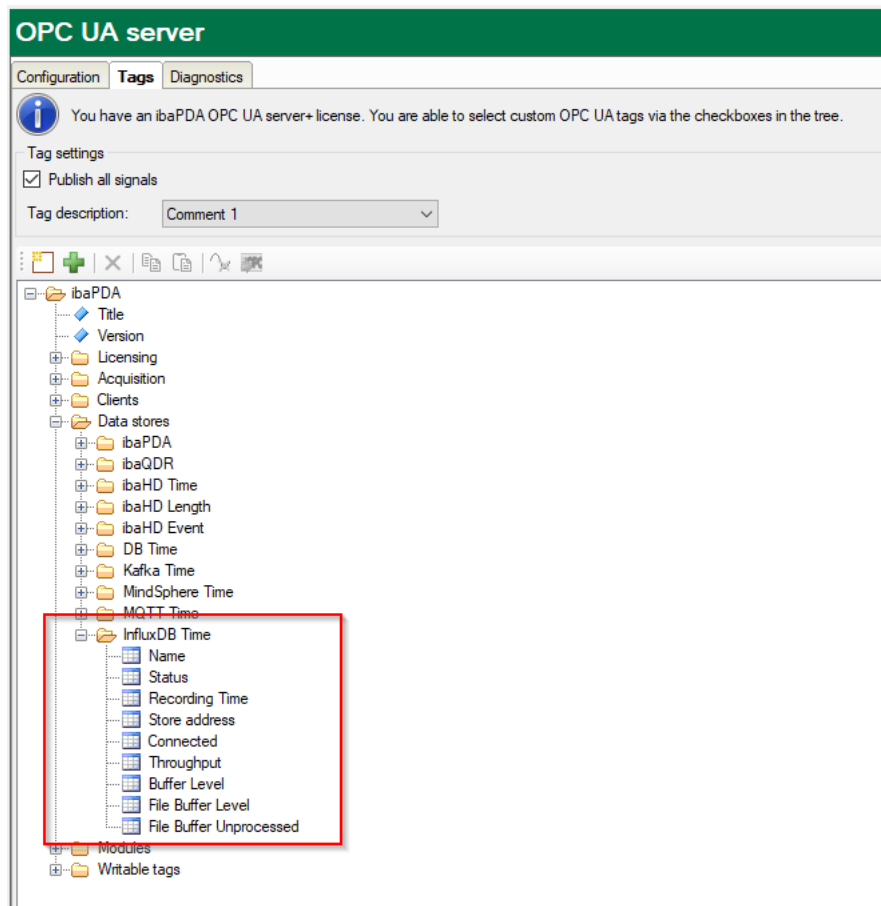
Less metadata is transmitted this way but the analysis is more complex since the data from two buckets needs to be correlated.

10.5 Diagnostics

Like for other data store types various diagnostic possibilities are available to monitor the status of the InfluxDB data store.

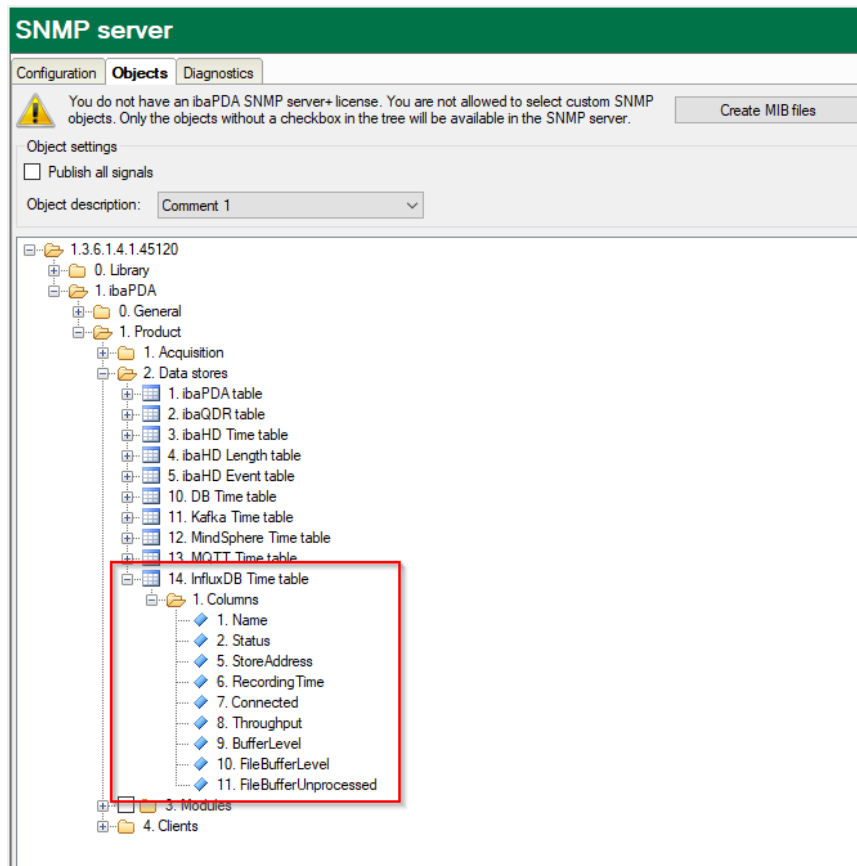
10.5.1 OPC UA server

The ibaPDA internal OPC UA server was extended and now also provides diagnostic data regarding the InfluxDB data stores. Diagnostic data can always be used without having an additional OPC UA server license.



10.5.2 SNMP server

The ibaPDA internal SNMP server was extended and now also provides diagnostic data regarding the InfluxDB data stores. Diagnostic data can always be used without having an additional SNMP server license.



10.5.3 Virtual function *DataStoreInfoInfluxDB()*

A new virtual function DataStoreInfoInfluxDB() is available to generate diagnostic data related to InfluxDB data stores as signal data for further processing within ibaPDA.

DataStoreInfoInflux('DatastoreIndex*', 'InfoType')**

This function returns information about the selected InfluxDB data store.
'DatastoreIndex' >= 0

The following info types are supported:

0: Recording status:

- 0=Stopped
- 1=Waiting for trigger
- 2=Recording
- 3=Posttrigger recording

1: Data throughput in KB/s

2: Is server connected?

3: Recorded time since last start trigger expressed in seconds. This will be constant 0 for continuous recording.

5: Current buffer usage (in %)

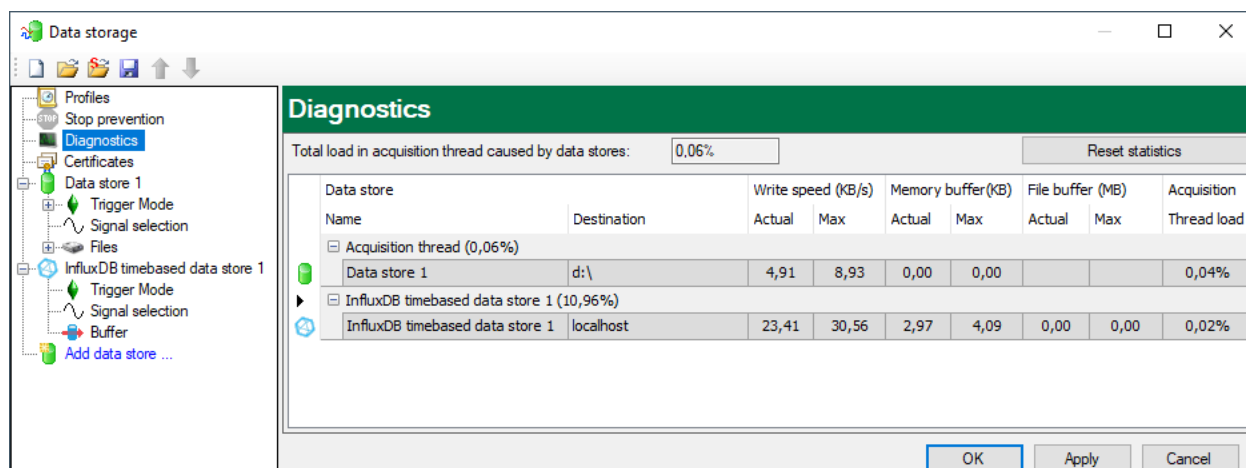
6: Current file buffer usage (in %)

7: Unprocessed bytes in file buffer (in %)

Parameters ending with * are only evaluated once at the start of the acquisition.

10.5.4 Diagnostics grid data storage manager

The diagnostics grid in the data storage manager has been extended with the InfluxDB data store as well. The group shows the load of the writer thread that writes the data to the InfluxDB server. The destination column shows the address of the database. The write speed shows how fast data is written to the database. The buffer data shows how much data is currently buffered



11 ibalnCicle and ibalNSpectra

11.1 Vector mode for ibalnCicle

The InCycle Expert and InCycle Auto-Adapting modules can be configured to have their input from a vector. This has to be configured in the calculation settings of the profile:

Calculations		Bands
<div> <div>Input</div> <div> <div>Mode</div> <div>Unit</div> </div> </div>		
		Vector
		Time synchronous averaging
<div>Averaging</div> <div> <div>Cycles per calculation</div> <div>Averaging Type</div> </div>		

Each signal in the vector corresponds to one data point in a cycle, so the number of samples of a cycle equals the number of signals in the vector. The cycle is a snapshot of all signals in the vector. The cycle is triggered periodically or on a rising edge of a trigger signal.

The data that precedes the sampling time of the cycle can also be taken into account by configuring a sampling method.

- Sample once (default, no data from before is taken into account)
- Average (average since the end of the previous cycle)
- Peak hold (maximum since the end of the previous cycle)
- Minimum hold (minimum since the end of the previous cycle)

Calculations		Bands
Input		
Mode	Vector	
Unit	Input signal unit	
Averaging		
Cycles per calculation	10	
Averaging Type	Linear	
Triggers		
Trigger mode	Pulse trigger	
Sampling method	Sample once	
Bands		
Band results	Sample once Average Peak hold Minimum hold	

Sampling method
 Defines how the data for a cycle is captured. Note that in all cases the sampling of each signal in the vector is handled individually:

- Sample once: the signal is sampled once. No data from before is taken into account (*).
- Average: the average of the signal since the end of the previous cycle is calculated (*)(**).
- Peak hold: the maximum of the signal since the end of the previous cycle is calculated (*)(**).
- Minimum hold: the minimum of the signal since the end of the previous cycle is calculated (*)(**).

(*) When the skip data signal is false, the data is not taken into account. If skipped data is false for all data, no cycle is created.
 (**) If a reset trigger occurs in the time frame, only the data later than this reset is taken into account.

The cycles with the vector data can be visualized in the Cycle view.

11.2 Auto-Adapting modules: endless learning and exponential learning

The InCycle Auto-Adapting and InSpectra Auto-Adapting modules allow learning a specific reference. A new setting 'endless learning' is available now in the profile. If endless learning is configured, one does not have to configure how many spectra/cycles are used for the learning the reference as the learning never stops:

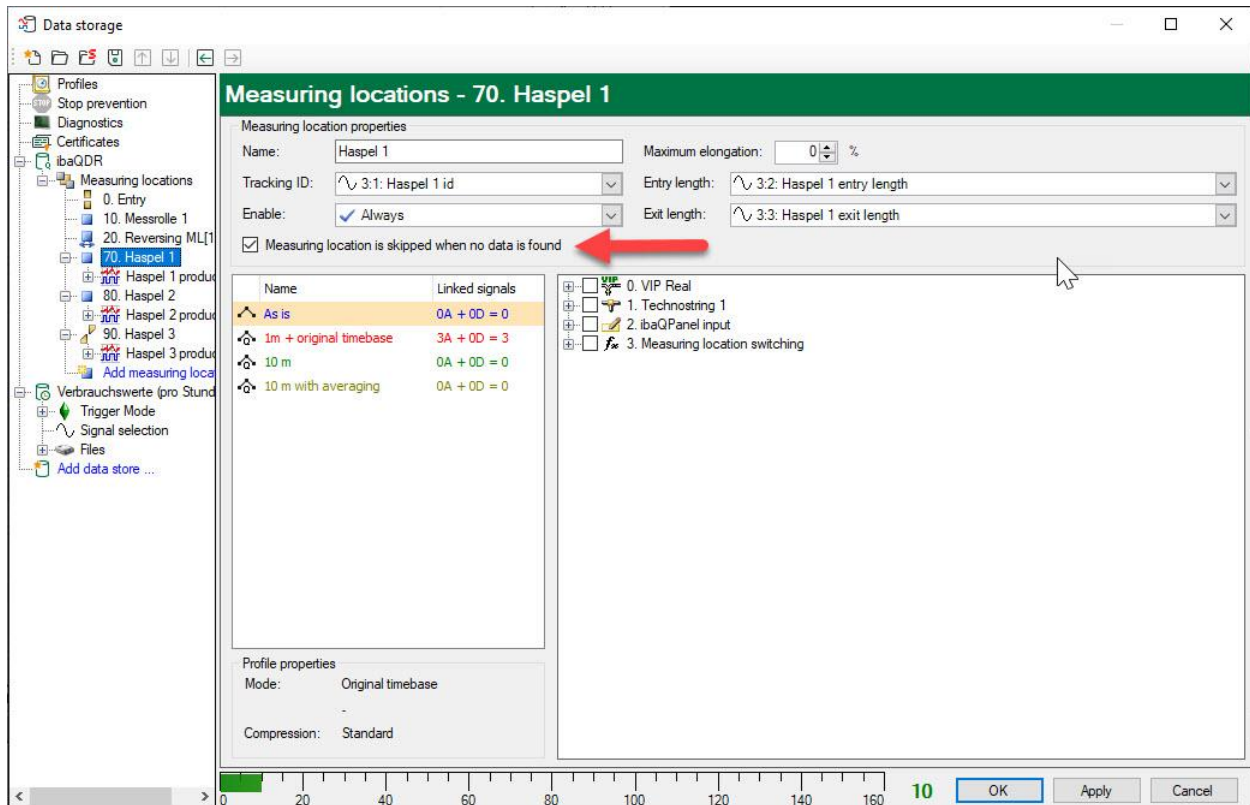
Teach settings		Calculations	Bands
Limits			
Monitoring mode	Maximum		
Learning			
Learning averaging type	Linear		
Endless learning	True		
Events			
Event status signals	Digital		
Event mode	Lower and upper		
Endless learning			

From this version on, it is possible to configure the 'Learning averaging type'. By default, this is 'Linear', which means that the reference curve is the average of all spectra/cycles during the

learning. The learning option 'Exponential' is new. In that case one also has to configure the 'Exponential learning percentage' which is the percentual weight of the most recent curve.

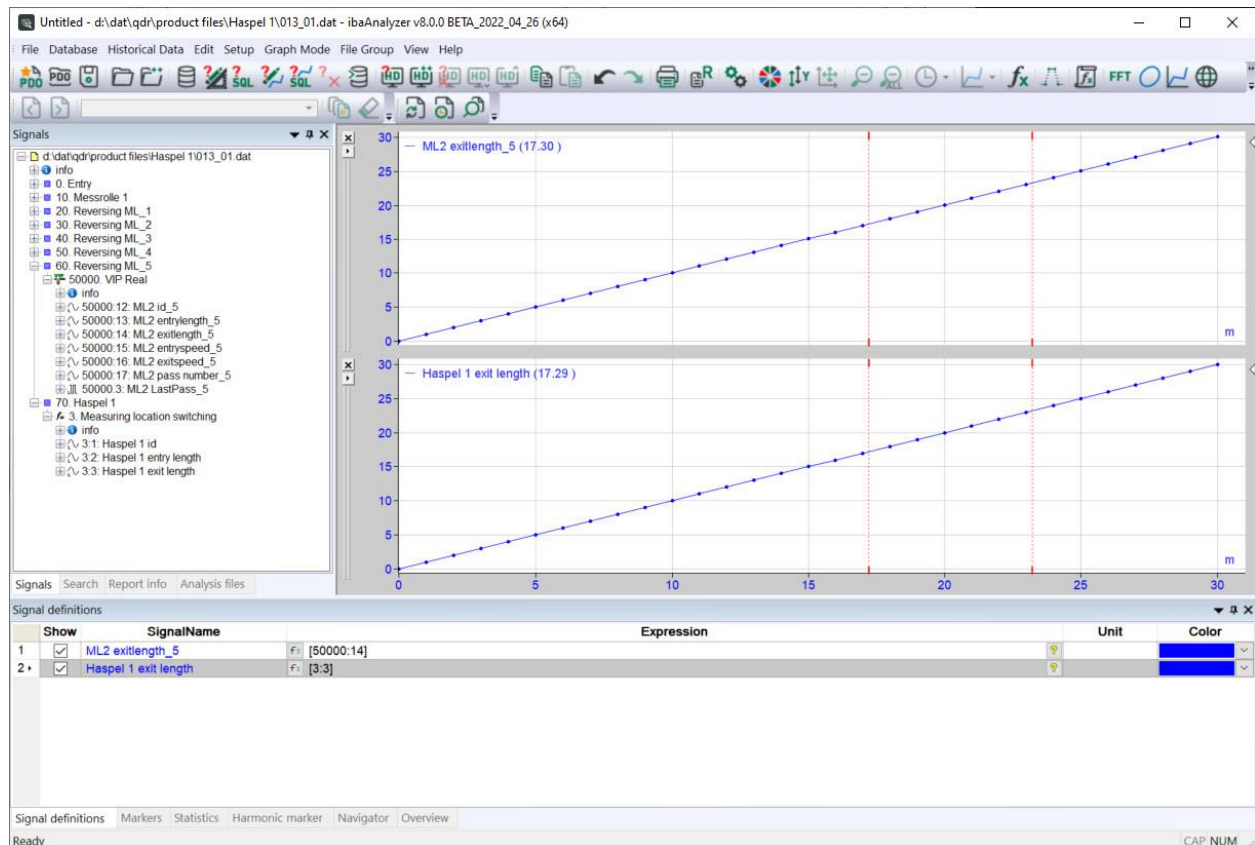
Teach settings		Calculations	Bands
Limits			
Monitoring mode	Maximum		
Learning			
Learning averaging type	Exponential		
Endless learning	Linear		
Averaging			
Nr. curves to learn	5		
Exponential learning percentage	10 %		
Events			
Learning averaging type			

12 ibaQDR optional measuring locations



In ibaQDR you can now mark standard measuring locations and measuring locations with an offset as optional. This means that if a tracked part doesn't pass such a measuring location that ibaQDR will skip the measuring location and will continue with the previous measuring location.

A typical use case is a situation where you have multiple coilers at the end of a processing line and only one of them is used for a coil. You can configure a measuring location per coiler and mark it as optional. You will also have to add a product file per coiler. When the first coiler is used then a product file will be generated that contains no data for the other coilers.



When the last coiler is used then a product file will be generated that contains signals of the previous coils but they won't have data.

