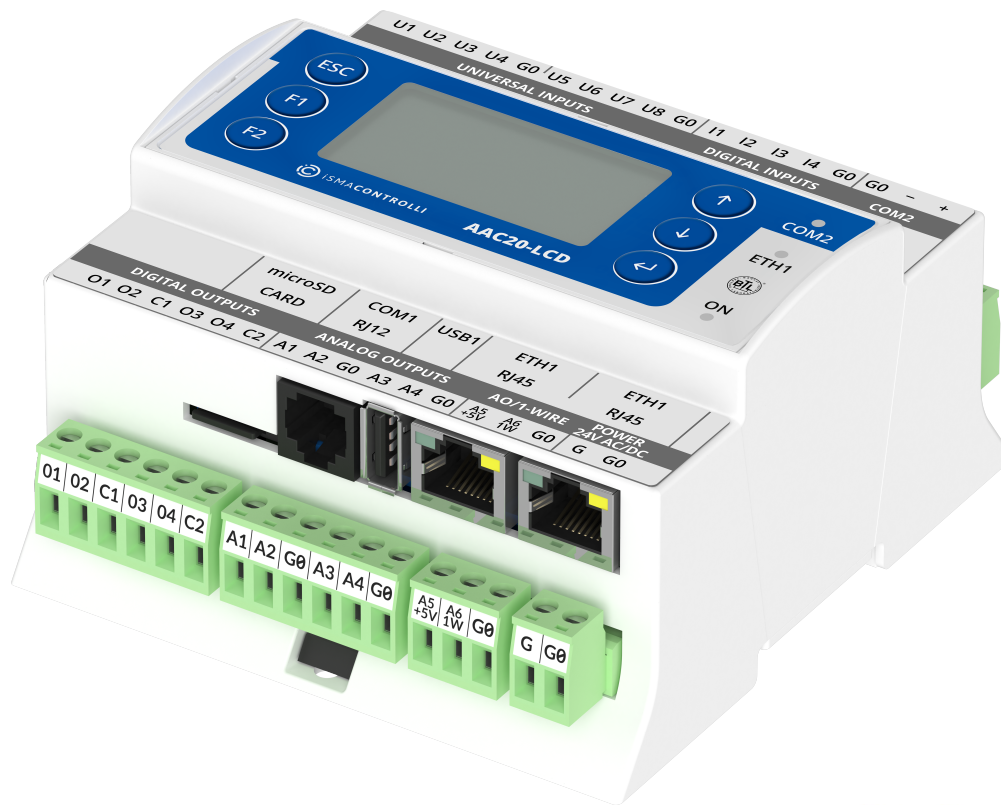


iSMA-B-AAC20

User Manual

Platform



Powered by
sedona
FRAMEWORK™

Table of Contents

1	Introduction	4
1.1	Revision History.....	4
2	Sedona in iC Tool.....	6
2.1	iC Tool Installation	6
2.2	Importing Kits to iC Tool.....	7
2.3	Connecting to Sedona Device	8
3	License	10
3.1	Updating License Online.....	10
3.2	Updating License Offline.....	10
4	Sedona Kits	12
4.1	Kit Manager	12
4.1.1	Using Kit Manager	12
4.1.2	Operations on Kits.....	13
4.2	Application Manager	17
4.2.1	Using Application Manager	18
4.2.2	Operations on Applications	18
5	App.....	21
5.1	Plat	22
5.1.1	Changing IP Address.....	23
5.2	Users Service	24
5.2.1	User Rights and Permissions.....	25
5.3	DateTime	25
6	NV Components.....	26
6.1	NVBooleanWritable.....	26
6.2	NVIntegerWritable	28
6.3	NVNumericWritable	29
6.4	NVMultiStateWritable	30
7	History Extension.....	33
7.1	History Service.....	33
7.2	COV	33
7.3	Interval.....	34
7.4	COV_Interval.....	35
8	Alarm Extension	37
8.1	Alarm Service	37
8.2	Alarm Boolean Points	37

8.3	Alarm Numeric Points	38
9	Scheduler.....	40
10	SOX Protocol.....	41
11	1-Wire	42
11.1	Adding 1-Wire Network.....	42
11.2	Discovering 1-Wire Sensors	42
11.3	Adding 1-Wire Sensors Manually.....	43
12	LocalIO Driver	44
12.1	LocalIOConfig.....	45
12.2	LocalIOFolder	47
12.2.1	AODigital	47
12.2.2	AOVoltage.....	48
12.2.3	DI	49
12.2.4	DICounter	50
12.2.5	DO	51
12.2.6	UIDigital.....	52
12.2.7	UIResistance	53
12.2.8	UITemperature.....	54
12.2.9	UIVoltage	55

1 Introduction

This manual contains information about commissioning the iSMA-B-AAC20 controller.

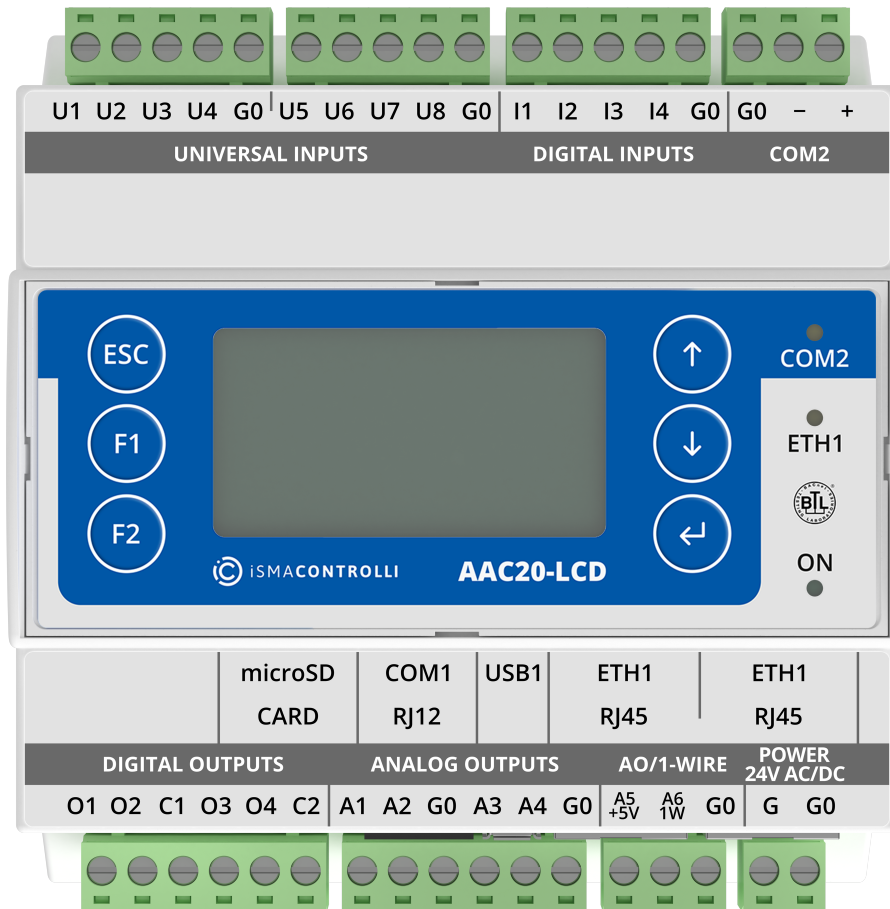


Figure 1. AAC20 controller

1.1 Revision History

Rev.	Date	Description
1.7	19 Jun 2024	<ul style="list-style-type: none"> Corrected CPU app usage description in the app component AAC20 Updater section moved to a separate document
1.6	28 Feb 2022	<ul style="list-style-type: none"> Rebranded Corrected information about the PWM mode in analog outputs
1.5	23 Jul 2020	Company data update
1.4	27 Jan 2020	<ul style="list-style-type: none"> Added chapter about NVMultiStateWritable Changed description of License Manager Replaced environment of programming from Workplace to iSMA Tool Added description of Autologin slot in AAC20 Platform Service
1.3	5 Feb 2018	Added DNS option

Rev.	Date	Description
1.2	20 Apr 2017	Added DHCP option
1.1	28 Aug 2015	First edition

Table 1. Revision history

2 Sedona in iC Tool

As a significant part of an end-to-end iSMA solution, the iC Tool gives customer a convenient way to create and manage custom applications for Sedona-based iSMA controllers.

The iC Tool covers all requirements to create a perfect application: wire sheet for convenient visual programming, property sheets for details, kit management, real-time monitoring of system states and slot values, logs and historical data, deployment, and backup.

2.1 iC Tool Installation

Note: Full description of the iC Tool functionalities is available in the iC Tool manual available at [iSMA CONTROL LI Online Docs](https://www.ismacontrolli.com).

The iC Tool is a software created for modern Microsoft Windows system, such as Windows 10. The oldest supported version of the operating system is Windows 7. The iC Tool is delivered as a compressed folder, which needs to be extracted in a chosen location on a hard drive, unless the access to the extracted folder is restricted by the system (e.g., Program Files is not a recommended location).

In order to download the iC Tool Software Bundle, which includes all files necessary to run the program efficiently (a zipped file iCTool_Vx.x.x.zip), go to the iSMA CONTROL LI web page [ismacontrolli.com](https://www.ismacontrolli.com) and to the iC Tool/Software Bundle folder.

Extracting the zipped package reveals the folders and additional files described below. In order to run the iC Tool, open the iCTool.exe file.

Nazwa	Stan	Data modyfikacji	Typ	Rozmiar
Config	✓	15.12.2023 13:05	Folder plików	
de	✓	15.12.2023 13:02	Folder plików	
es	✓	15.12.2023 13:02	Folder plików	
External	✓	15.12.2023 13:02	Folder plików	
fr	✓	15.12.2023 13:02	Folder plików	
home	✓	15.12.2023 13:04	Folder plików	
icons	✓	15.12.2023 13:03	Folder plików	
it	✓	15.12.2023 13:03	Folder plików	
ja	✓	15.12.2023 13:03	Folder plików	
Libraries	✓	15.12.2023 13:02	Folder plików	
Localization	✓	15.12.2023 13:02	Folder plików	
log	✓	15.12.2023 13:04	Folder plików	
pl	✓	15.12.2023 13:03	Folder plików	
Resources	✓	15.12.2023 13:02	Folder plików	
iC Tool	✓	15.12.2023 13:02	Aplikacja	11 287 KB
iC Tool.exe.config	✓	15.12.2023 13:02	Plik CONFIG	3 KB
LICENSE	✓	19.12.2023 07:49	Dokument tekstowy	12 KB

Figure 2. iC Tool files

The extracted folders have the following functions:

- **Config:** A folder containing a record of user's individual settings regarding windows location and other iSMA Tool work settings, such as a language chosen for the iSMA Tool interface.
- **External:** A folder containing an API .dll file.

- **home:** A folder where all the data created by user are saved, i.e., device backups, applications, etc. It is also a folder where the kits library, available the in iSMA Tool, is located.
- **icons:** A folder with graphical files such as the iSMA Tool interface icons.
- **Localization:** A folder with the text files providing the iSMA Tool language sources.
- **log:** A folder, where the logs of the iSMA Tool, which also appeared in Console window, are saved. When contacting iSMA CONTROLLI technical support, it is advised to copy the last file with logs form that folder
- **de, es, fr, it, ja, pl, Resources, ru:** Folders with system libraries.

To properly install and work with the iC Tool the computer must meet the following minimal requirements:

- **Processor (CPU):** Intel Core i3-3xxx or equivalent;
- **Memory:** 4GB RAM;
- **Storage:** 50 GB internal hard driver;
- **Ethernet:** 100 Mbit or 1Gbit NIC;
- **OS:** MS Windows 7 (recommended MS Windows 10);
- **.NET Framework** 4.6.2 or higher.

WARNING! If the iC Tool is being run for the first time, it asks to accept the EULA license. The license must be accepted to run the program. Failure to do so closes the iC Tool.

Note: For the iC Tool to work properly it needs to be run periodically at least once a month, on a computer connected to the Internet for about an hour, depending on the data transfer rate. It enables the iC Tool to automatically download the latest data, such as kits and updates.

The iC Tool is a portable software. It is transferable and it can be installed on a portable data storage device, such as a USB memory stick. It allows the program to be run directly from a portable data storage device on any PC, including offline ones.

2.2 Importing Kits to iC Tool

The iC Tool allows to import kits into the program (the latest kits and kits bundle can be downloaded from the ismacontrolli.com website) and to expand the default kit base with external kits, for example, user's own kits. To add them, select the Import Framework Files option, available in the main menu under Import. Once selected, a dialog window opens to specify the location of the imported files.

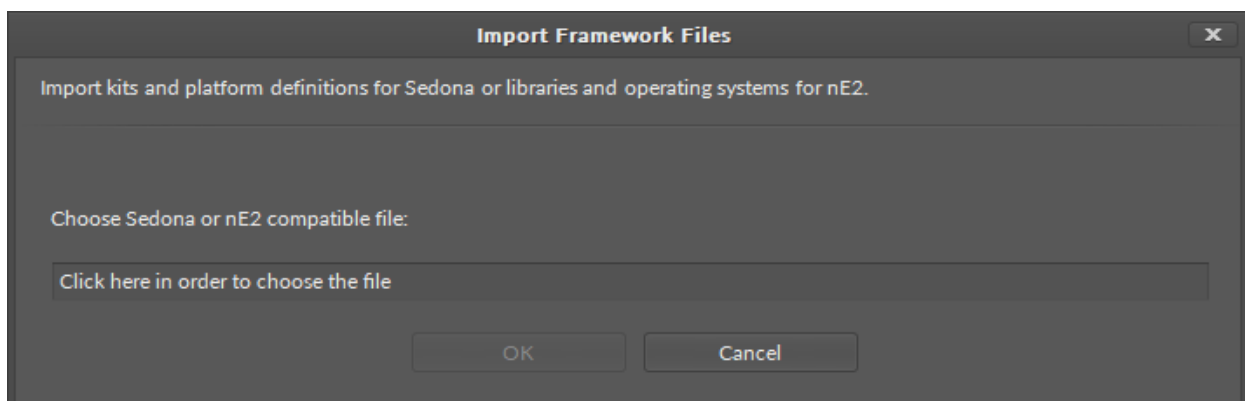


Figure 3. Dialog window for importing Sedona files

The iC Tool may import single files or compressed folders with .zip extension containing:

- a single kit; or
- whole packages containing kits, manifests, and .par files.

Note: The iC Tool can read the .zip files including other .zip files containing kits, manifests, and .par files, and can display a summary of how many files have been imported at the end of the process, in a window and in console, as shown on the figure below. If such package contains files which are not kits, manifests, or .par files, they are skipped.

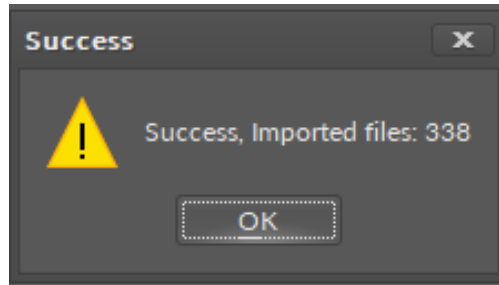


Figure 4. Summary of importing Sedona files

After a successful import, the added kits are uploaded to the Kit Manager. The list of kits, which can be installed on the selected device, is displayed in the bottom part of the Kit Manager view. It includes the kits, which were added manually. If the external kits do not appear on the list of available kits, it means they are not compatible with the selected device and cannot be uploaded. In this case, after deselecting the device, external kits will be displayed on the list in the lower part of the view, because in this mode the iC Tool shows all kits installed in it.

2.3 Connecting to Sedona Device

In order to connect to the device, right-click on the project folder and select an Add Device option.

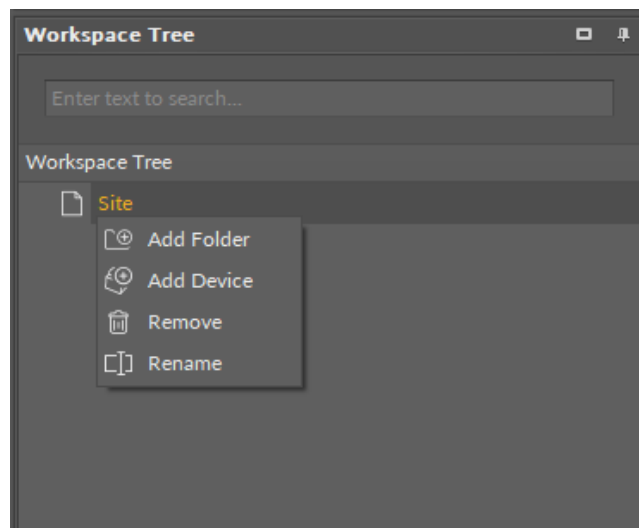


Figure 5. Adding a new device

After selecting the Add Device option, a device log in window pops up.

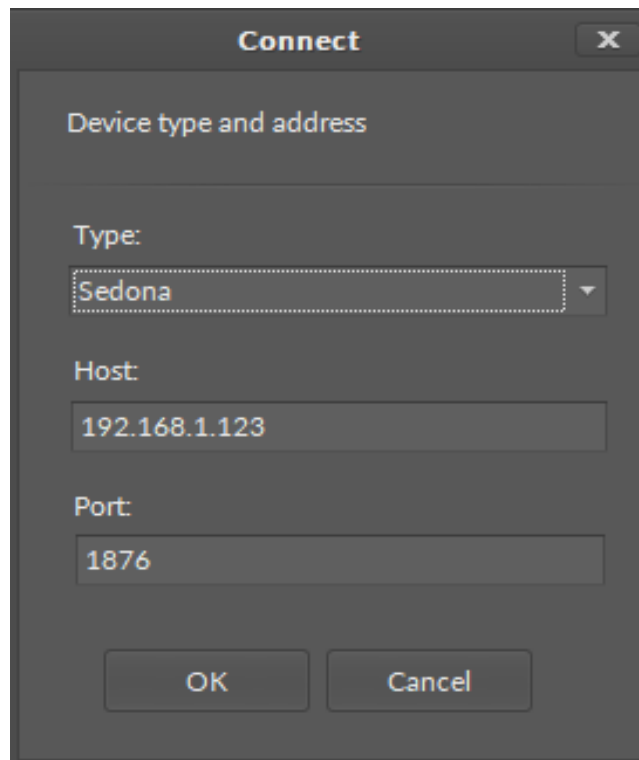


Figure 6. Sedona device connection window

By default, a new device has the following log in settings:

- **IP address:** 192.168.1.123;
- **Port:** 1876;
- **User:** admin;
- **Password:** empty box (no characters).

The connection parameters can be changed after connecting to the controller:

- **IP address:** in the plat component (see section Changing IP Address);
- **Sox Port:** in Sox service (see section Sox Protocol);
- **Admin password/new user:** in the user service (see section Users Service).

3 License

Sedona controllers have a built-in license mechanism. All out of the box controllers have a standard license version installed, and they are instantly ready to work. However, in case the license is changed (upgraded or downgraded) or it has to be re-uploaded after a memory clear it can be done either online, or offline.

The licenses are generated by the serial number or MAC address. The proper file name format is serialnumber.dat MACAddressWithoutDots.dat.

3.1 Updating License Online

The License Manager can upload the license to the device in online mode. The iC Tool connects to the license server, downloads, and saves the license to the controller. To open the License Manager, double click the device in the Workspace Tree window, and go to the Object Properties window, where the License Manager button is displayed. Once selected, it displays a pop-up window which enables to check the device license status and invoke one of two actions:

- Upload the license from the remote database;
- Upload the license from the file.

To download the license from the server, use the Get License from Remote Database option. While downloading, a pop-up window shows the progress. After completed download, use the Get License from Local Database option to send the license file to the controller. The new license will be implemented after the controller's reboot.

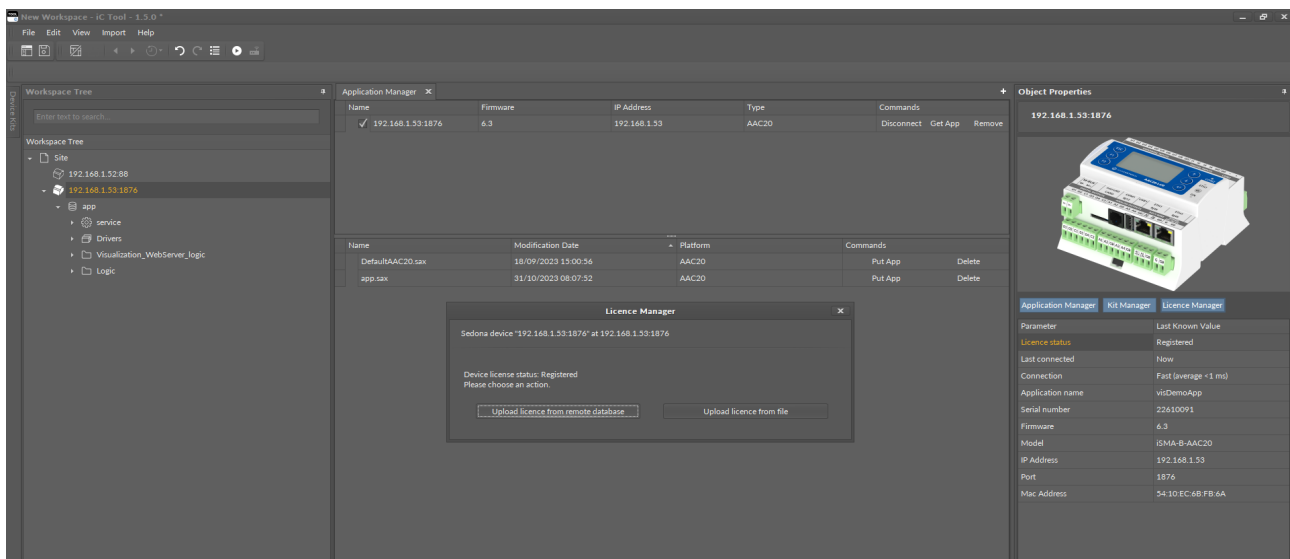


Figure 7. License Manager view

3.2 Updating License Offline

The license can be uploaded to the device offline using either the iC Tool's License Manager or AAC20 Updater. In order to use the offline License Manager, use the Upload License from File option. With the AAC20 Updater software the license file can be download using the ismaUI to local database (for Workbench AX users), copied from another PC, or e-mailed from the local iSMA distributor. Open the AAC20Updater.exe software and choose the license file (a proper file format is MACAddressWithoutDots.dat). After a successful upload, reboot the controller using the Reboot Device button. (For

more information about sending files, see Chapter [Sending Files to the Device](#)). The new license will be implemented after the controller's reboot.

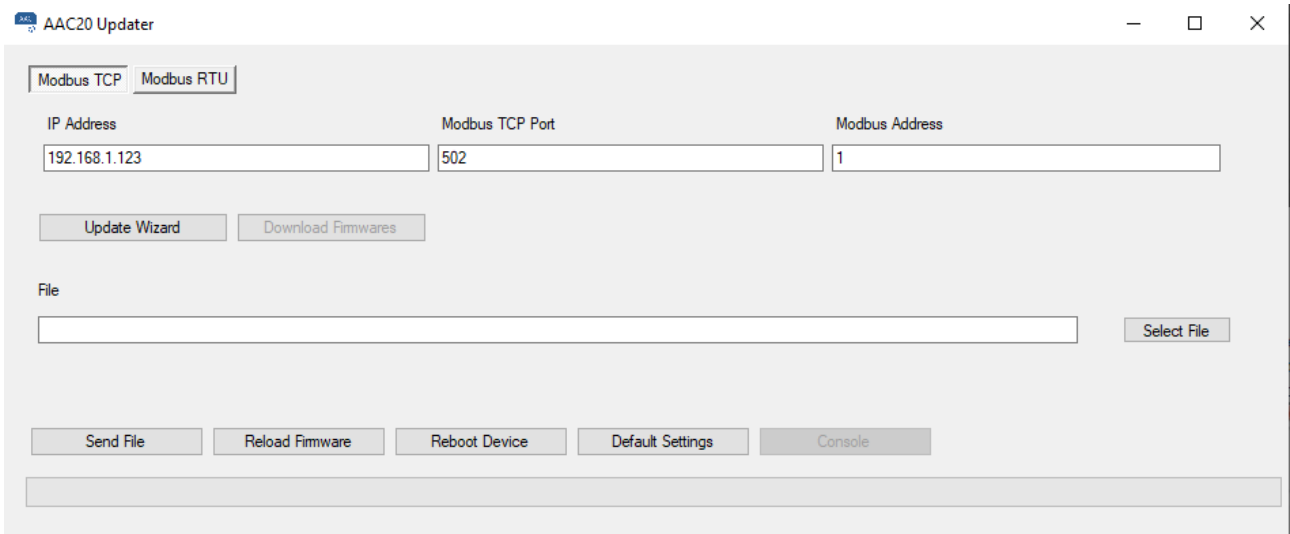


Figure 8. Sending Sedona Updater license file

4 Sedona Kits

The Sedona kits are sets of components for programming applications. The device is provided with a set of basic kits and, in order to develop applications, the set should be uploaded to the device. To add the new or update the kit, it must first be imported into the iC Tool (see Chapter [Importing Kits to iC Tool](#)). In the device the set of kits is stored in the kits.scode file. To manage controller's kits and compile the kits.scode, the iC Tool uses a Kit Manager tool.

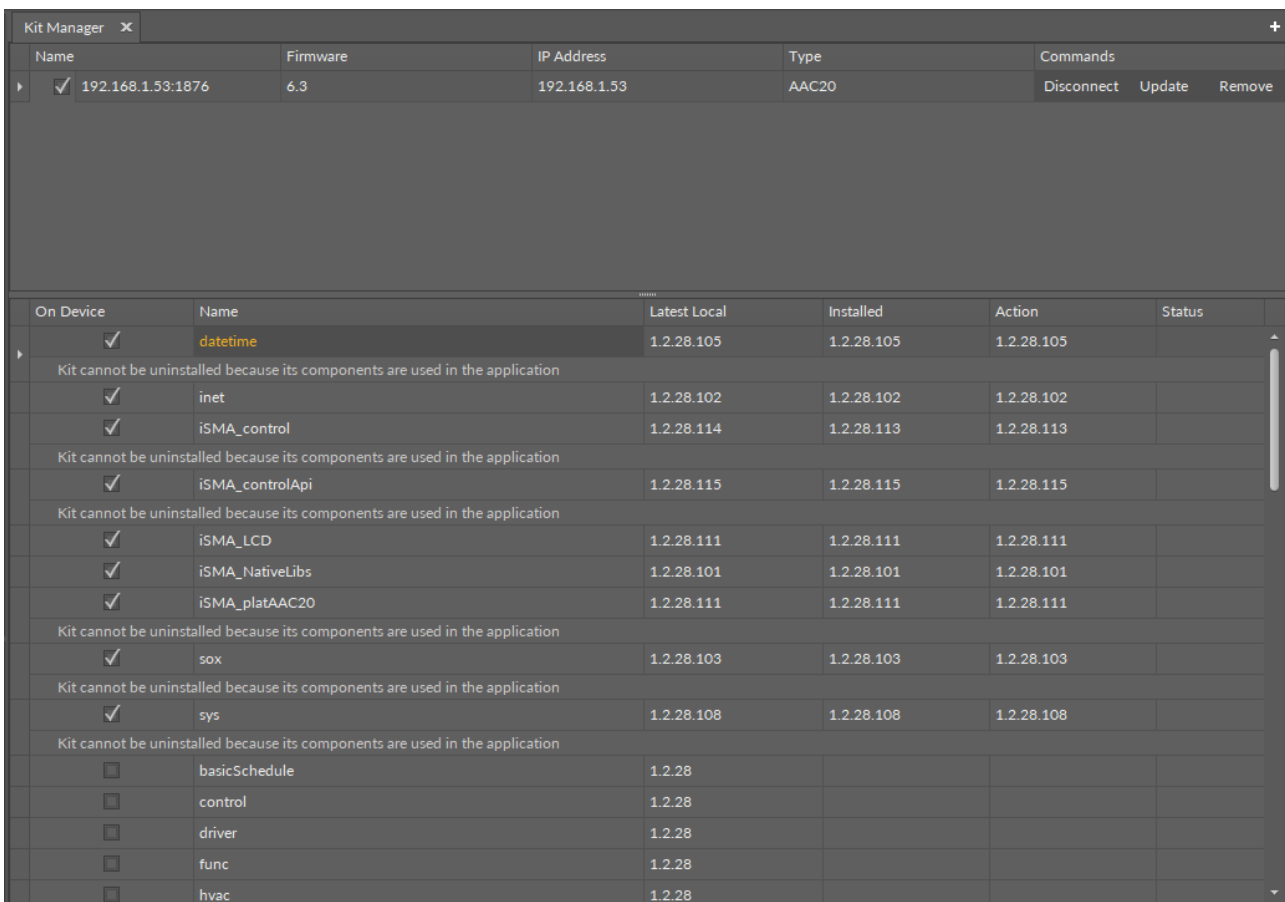
4.1 Kit Manager

Kit manager is a special tool that allows to manage Sedona kits in the controller. Its most important functions are adding/removing kits and upgrading/downgrading their versions. The Kit Manager's view consists of two windows - the upper window displays a list of devices and commands to run selected actions, the lower window displays a list of kits, their versions, and actions to be performed.

Warning!

Every time kits are edited in the device, it requires a restart of the device.

4.1.1 Using Kit Manager



On Device	Name	Latest Local	Installed	Action	Status
<input checked="" type="checkbox"/>	datetime	1.2.28.105	1.2.28.105	1.2.28.105	
Kit cannot be uninstalled because its components are used in the application					
<input checked="" type="checkbox"/>	inet	1.2.28.102	1.2.28.102	1.2.28.102	
<input checked="" type="checkbox"/>	iSMA_control	1.2.28.114	1.2.28.113	1.2.28.113	
Kit cannot be uninstalled because its components are used in the application					
<input checked="" type="checkbox"/>	iSMA_controlApi	1.2.28.115	1.2.28.115	1.2.28.115	
Kit cannot be uninstalled because its components are used in the application					
<input checked="" type="checkbox"/>	iSMA_LCD	1.2.28.111	1.2.28.111	1.2.28.111	
<input checked="" type="checkbox"/>	iSMA_NativeLibs	1.2.28.101	1.2.28.101	1.2.28.101	
<input checked="" type="checkbox"/>	iSMA_platAAC20	1.2.28.111	1.2.28.111	1.2.28.111	
Kit cannot be uninstalled because its components are used in the application					
<input checked="" type="checkbox"/>	sox	1.2.28.103	1.2.28.103	1.2.28.103	
Kit cannot be uninstalled because its components are used in the application					
<input checked="" type="checkbox"/>	sys	1.2.28.108	1.2.28.108	1.2.28.108	
Kit cannot be uninstalled because its components are used in the application					
<input type="checkbox"/>	basicSchedule	1.2.28			
<input type="checkbox"/>	control	1.2.28			
<input type="checkbox"/>	driver	1.2.28			
<input type="checkbox"/>	func	1.2.28			
<input type="checkbox"/>	hvac	1.2.28			

Figure 9. The Kit Manager

The columns in the above table are defined as follows:

- **On Device:** shows if a particular kit is already installed on the selected device;
- **Name:** the name of the kit;

- **Latest Local:** the number of the latest kit version installed locally in the iC Tool;
- **Installed:** the number of the kit version currently installed on the device;
- **Action:** the list of kits versions, which are available locally in the iC Tool and are ready to be installed on the device. A version number to be installed on the device is chosen from a drop-down menu (upgrade or downgrade process).
- **Status:** the column informs about an action that will be taken on the selected kit.

The user's role is limited to editing two columns: On Device and Action.

4.1.2 Operations on Kits

Below there are descriptions how to remove, replace, and add a kit to a device.

Adding Kits

In order to add the kit to the device, select its checkbox, and, if needed, define the version of added kit (by default, the newest version is selected). The Install command appears in the Status column, and the kit is added once the user confirms the Update command in the upper Device panel of the Kit Manager.

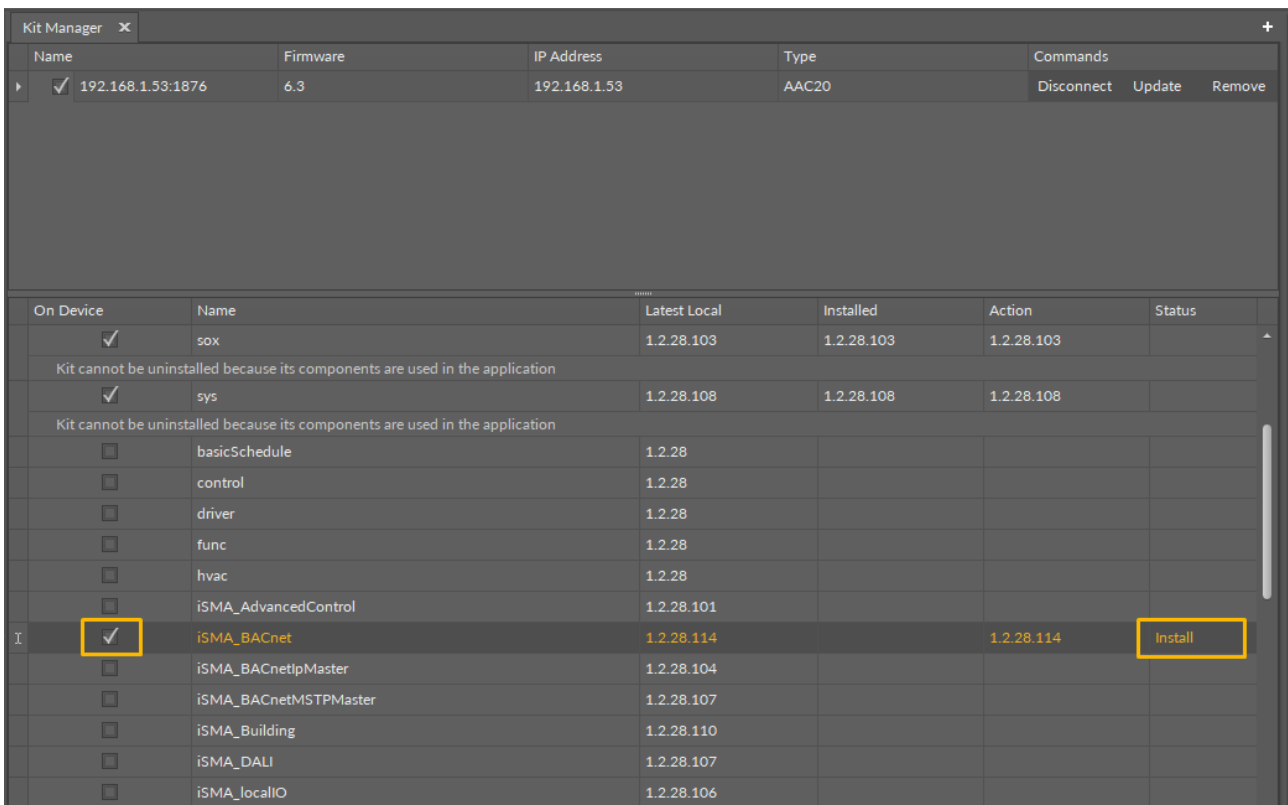


Figure 10. Adding a kit

Upgrading/Downgrading Kits

The kit installed on the device may be replaced by installing its newer or older version. Both upgrade and downgrade operations are done by choosing a proper number of the kit version in the Action column.

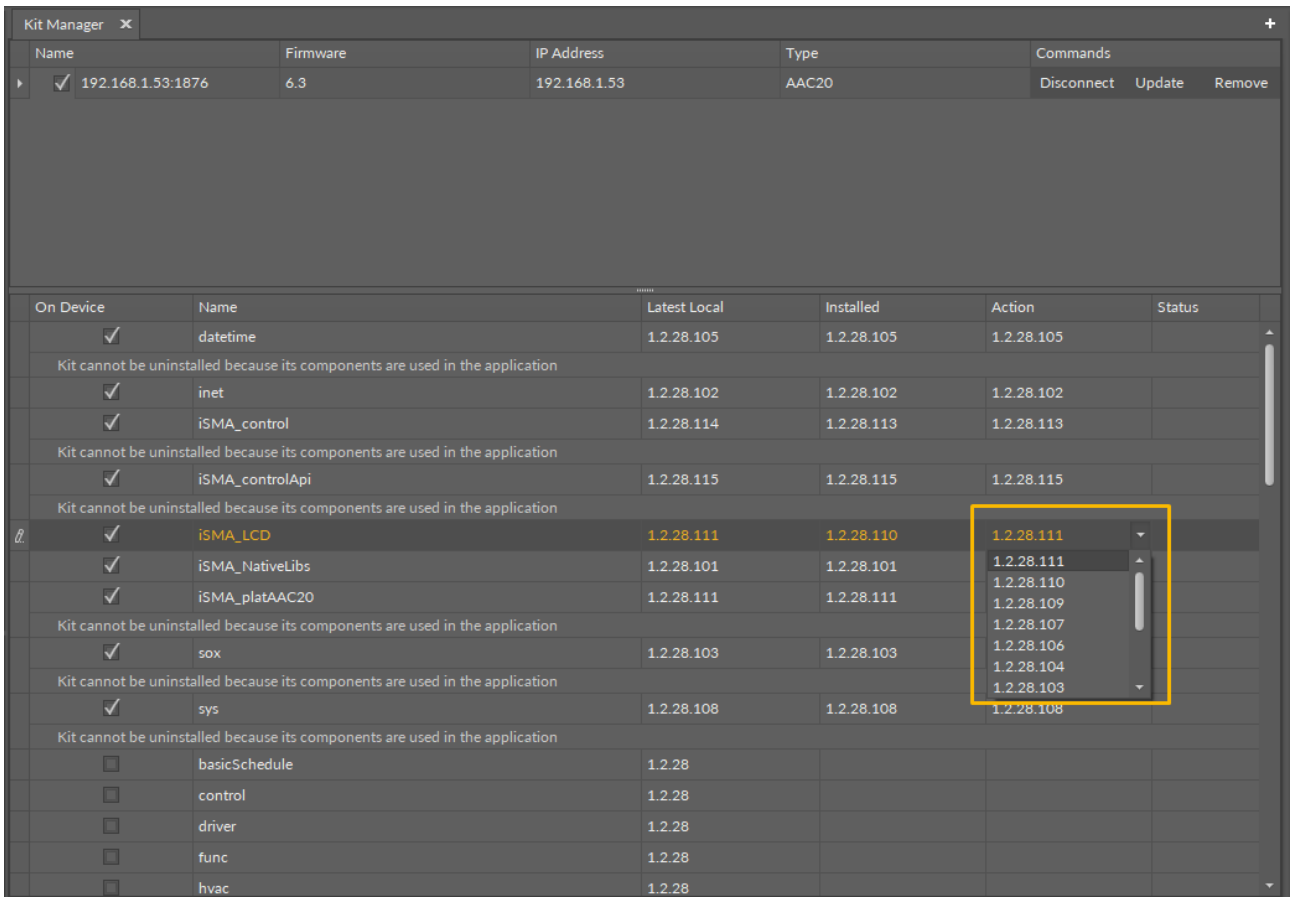


Figure 11. Selecting a version to upgrade or downgrade in the Action column

Depending if the current kit is replaced with a newer or older one, the iC Tool will display the planned action in the Status column, Upgrade or Downgrade.

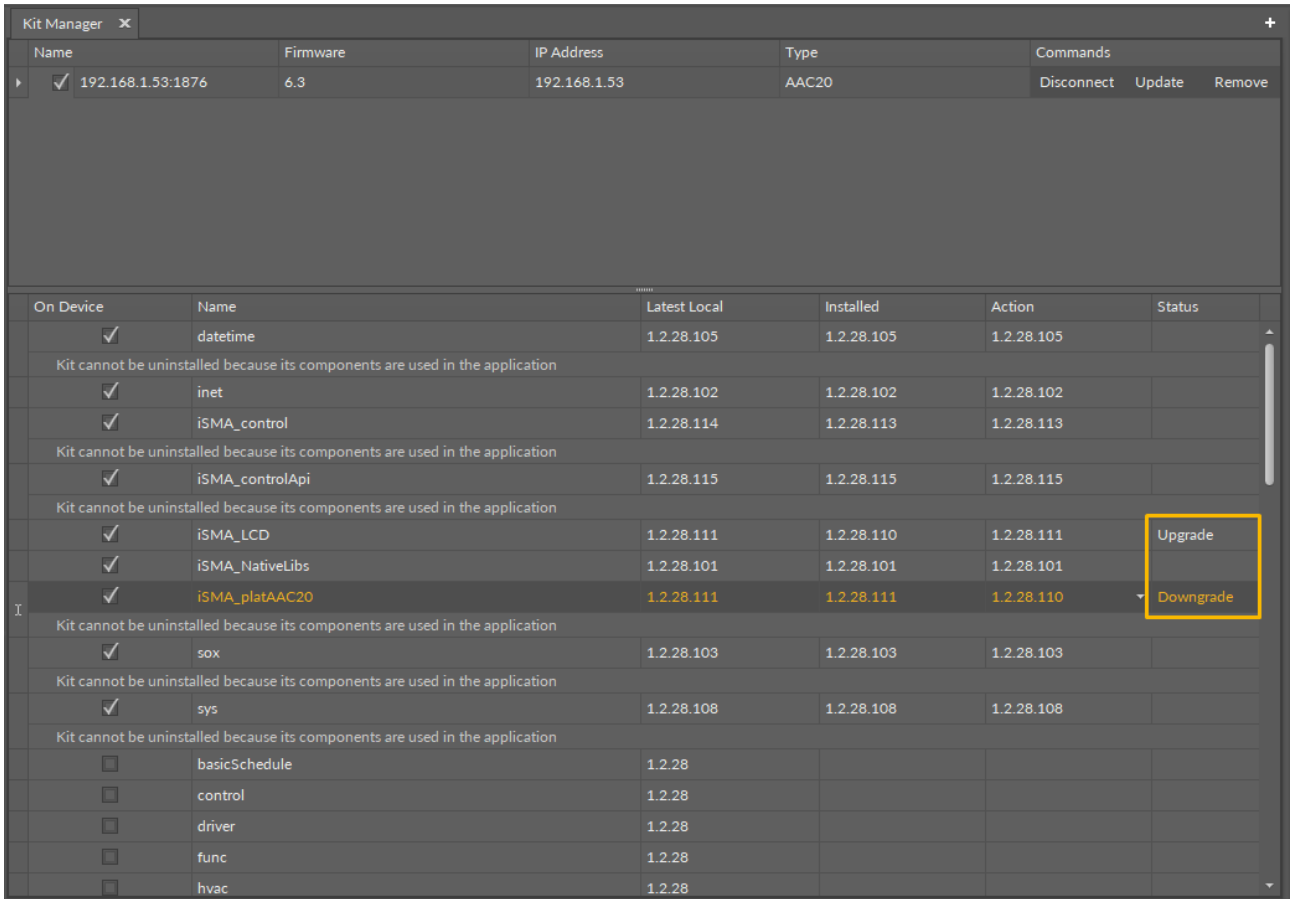


Figure 12. Upgrade or downgrade options

Removing/Uninstalling Kits

If a user deselects the checkbox, the selected kit is chosen to be removed. The Uninstall command appears in the Status column, and the kit is removed once the user confirms the Update command in the upper Device panel of the Kit Manager.

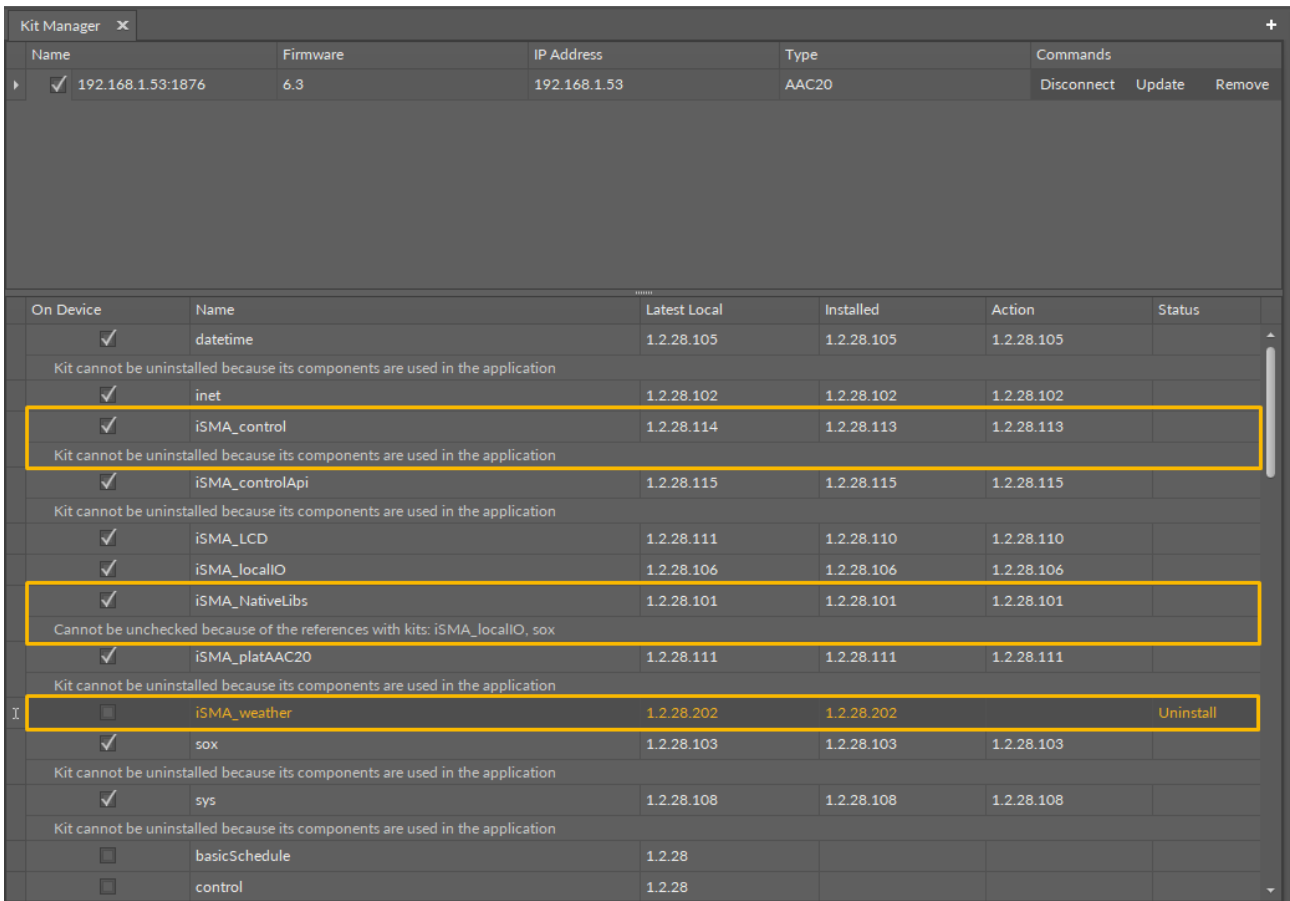


Figure 13. Removing kits

In order for the kit to be removed, the components used in the application shall be removed, or the kit needs to be left installed.

Completing Operations - Updating Kits

After defining the kits to be added, removed, or changed the Update command needs to be confirmed in the upper Device panel of the Kit Manager. The Update command is displayed in the Commands column.

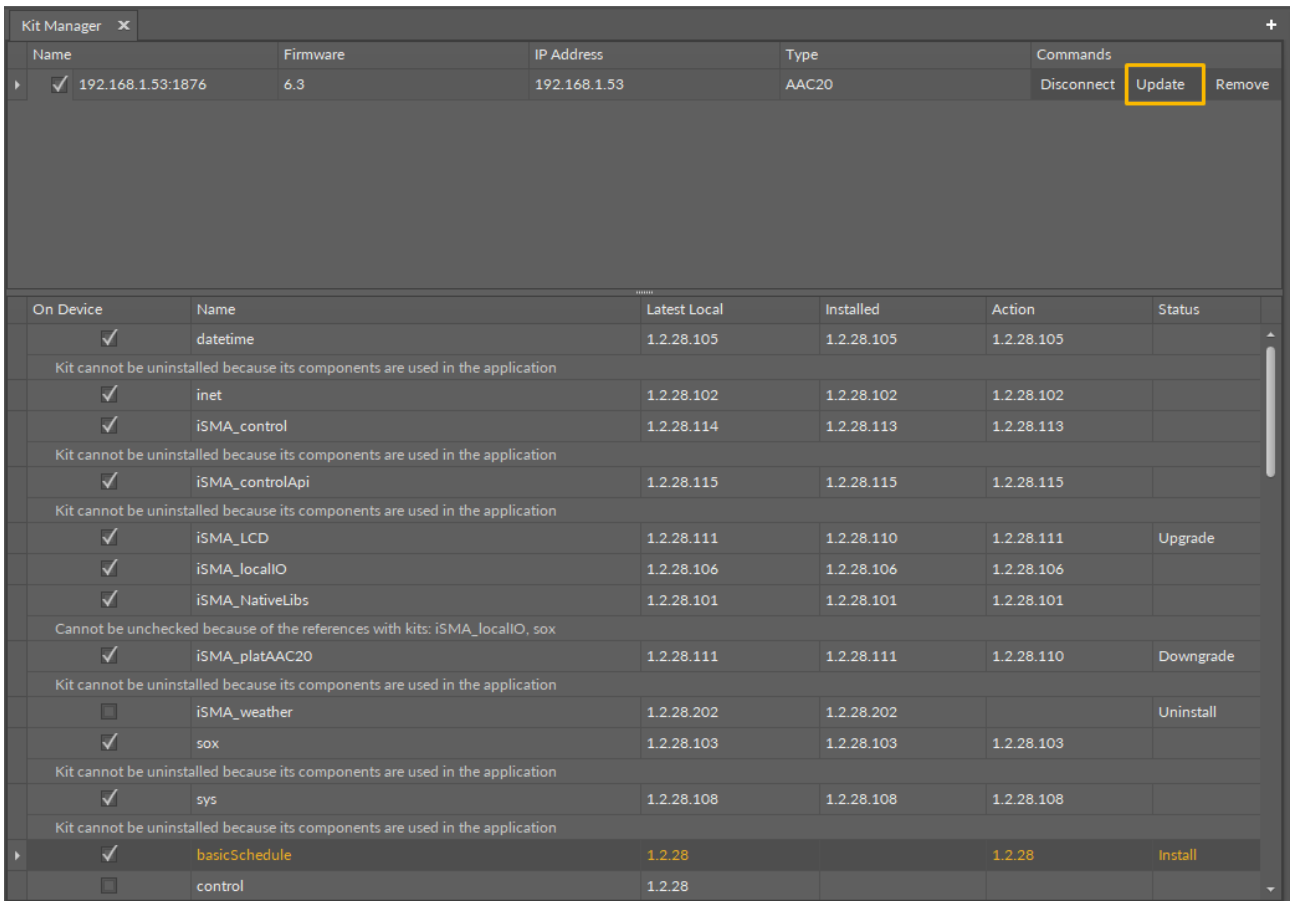


Figure 14. Updating kits

While updating the kits in the device, the iC Tool interface is blocked for the time.

Warning!

Finalizing the kits update requires restarting the device.

After restarting the device iC Tool reconnects with the device.

Note: The update process is monitored in detail in the Console window, which helps in analyzing the situation in case the update could not be completed.

4.2 Application Manager

The Application Manager is a tool that allows to manage applications of a specific device and the applications saved locally in the iC Tool.

The Application Manager allows to:

- manage applications saved locally in the iC Tool;
- load applications to the device;
- download applications from the device.

Warning!

Loading an application to the device overwrites the application already saved in the device.

Warning!

Each loading of application to the device requires restarting the device.

4.2.1 Using Application Manager

There are two ways to initiate the Application Manager:

- initiating the Application Manager from the context menu;
- initiating the Application Manager from the Object Properties window.

Name	Firmware	IP Address	Type	Commands
<input checked="" type="checkbox"/> 192.168.1.53:1876	6.3	192.168.1.53	AAC20	Disconnect Get App Remove

Name	Modification Date	Platform	Commands
DefaultAAC20.sax	18/09/2023 15:00:56	AAC20	Put App Delete

Figure 15. Application Manager

The columns in the above table are defined as follows:

- **On Device:** shows if a particular kit is already installed on the selected device;
- **Name:** the name of the kit;
- **Latest Local:** the number of the latest kit version installed locally in the iC Tool;
- **Installed:** the number of the kit version currently installed on the device;
- **Action:** the list of kits versions, which are available locally in the iC Tool and are ready to be installed on the device. A version number to be installed on the device is chosen from a drop-down menu (upgrade or downgrade process).
- **Status:** the column informs about an action that will be taken on the selected kit.

The user's role is limited to editing two columns: On Device and Action.

4.2.2 Operations on Applications

Below there are descriptions how to remove, replace, and add a kit to a device.

Downloading Application from Device (Get App)

After connecting with a chosen device its application can be downloaded by pressing the Get App button, see the figure below.

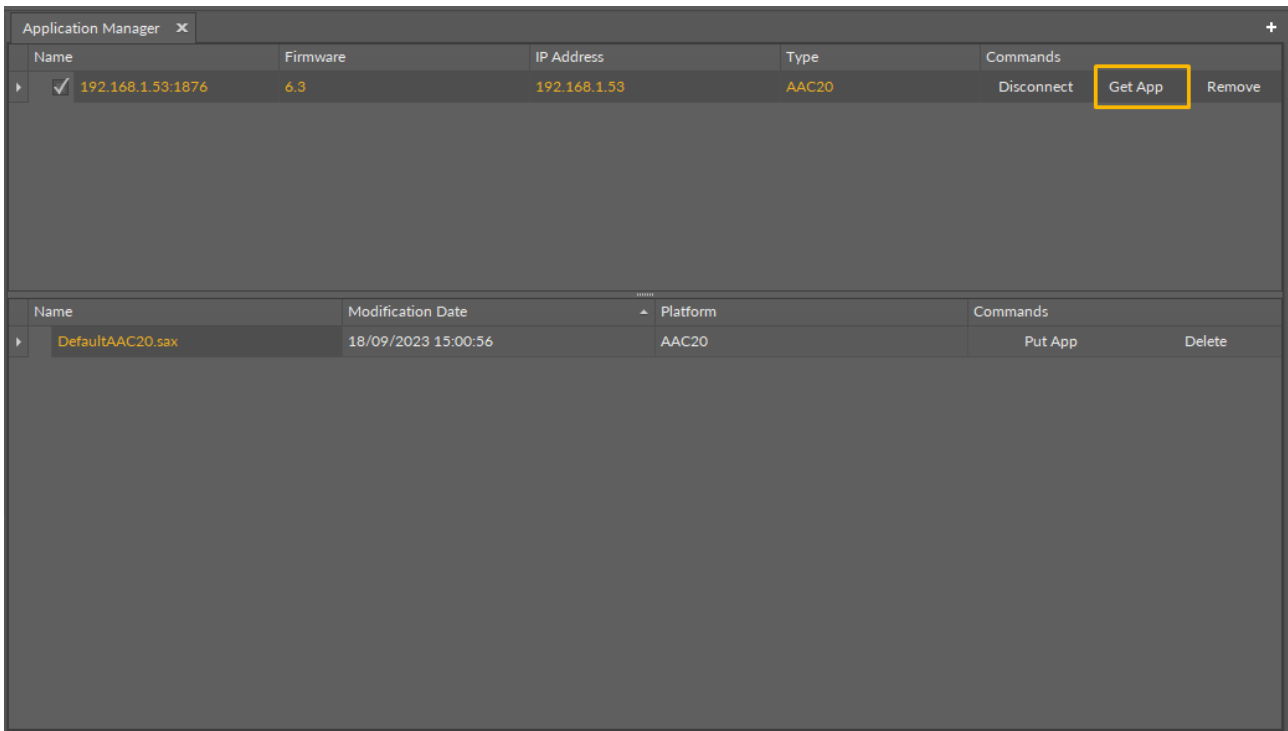


Figure 16. Download application

After initiating the command, a starting process reads an application from the device and creates the .sax file (with a name consisting of the application name and the device's IP address). This process blocks the iC Tool for the time of its operation. The newly created file will be added to the list of available applications saved in the iC Tool, and the physical .sax file will be located in the iC Tool main folder (/home/Applications). The default name of the application file may be changed by editing the Name column.

Note: Downloading an application from the device does not stop or restart the device. Downloading does not disrupt the device's work in any way other than pausing its interface for the time of downloading.

Note: Application downloading process is monitored in detail in the Console window, which helps to analyze the situation, in case downloading the application could not be completed.

Uploading Application to Device (Put App)

Application uploading procedure needs to begin with defining the device to work with Application Manager by selecting a checkbox for a particular device in the upper part of the Manager view.

Next, one of the available applications in the lower part of the Manager needs to be selected by pressing the Put App button in the Commands column.

Warning!

Pressing the Put App button will overwrite the application already installed in the device and reset the device in the final phase of the process.

The figure below shows a situation before beginning of the application upload into the iSMA-B-AAC20 controller, where the user is asked for confirmation due to overwriting of the application currently installed in controller.

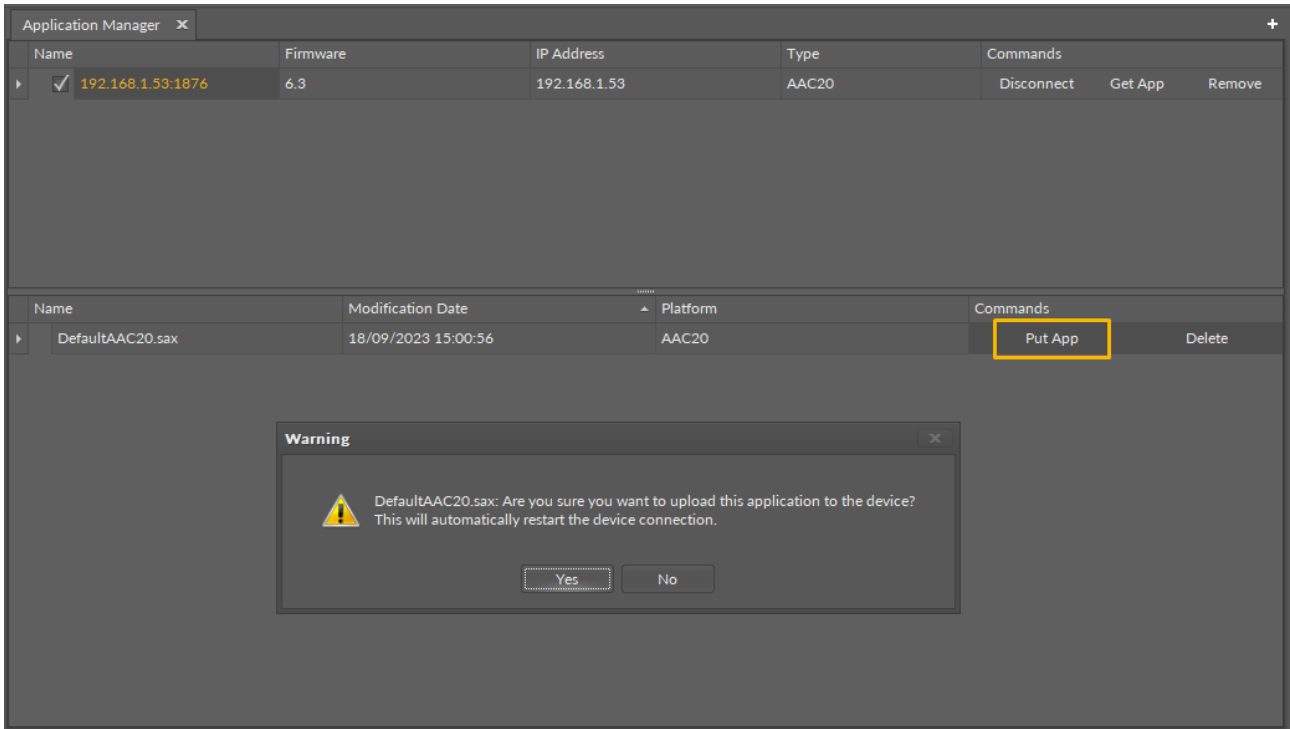


Figure 17. Put App notice

After approving the confirmation, a process blocking the iC Tool interface runs for a period of time, when the iC Tool uploads a chosen application and restarts and reconnects the device.

Note: Application uploading process is monitored in detail in the Console window, which helps to analyze the situation when application upload could not finish.

5 App

The app component is the fundamental component of the Sedona framework. It embraces all services, drivers, and components that allow to manage a Sedona device and create an application. The application consists of services and components available in the form of kits. Components within the application are processed in every working cycle of the device. Services include certain components enabling system functions such as user management. All these items must be placed under the main app component. When the application is modified, the app icon is displayed with a graphic notification that the application should be saved. There is a possibility to turn on the application autosave option.

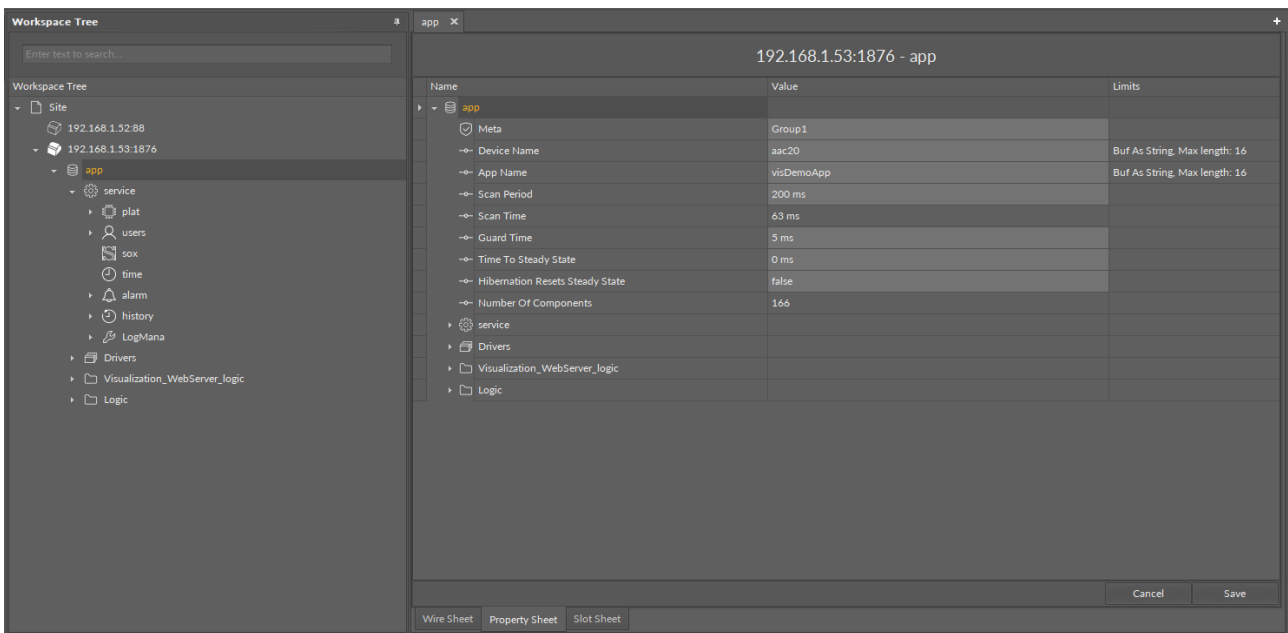


Figure 18. Sedona application structure view

The app component has the following slots:

- **Device Name:** shows the name of the device, may be edited manually by the user;
- **App Name:** shows the name of the application, may be edited manually by the user;
- **Scan Period:** allows to set one cycle execution time;
- **Scan Time:** shows the actual time of one cycle execution;
- **Guard Time:** allows to set time reserved to finish system tasks;
- **Time To Steady State:** allows to set the time from app start to steady state;
- **Hibernation Resets Steady State:** option not active for AAC20 controllers;
- **Number Of Components:** shows the number of components used in the application.

The app component has the following actions available at right-click or in the Object Properties window:

- **Save:** saves the application in the device's flash memory;
- **Restart:** restarts the application (Sedona Virtual Machine);
- **Reboot:** reboots the device;
- **Quit:** closes the SOX connection;
- **Hibernate:** deactivates the app component.

5.1 Plat

The Plat service is a component which shows device's main parameters. This component is placed under the Service folder and is associated with the device hardware. The component has to be placed under Service component in order to work properly.

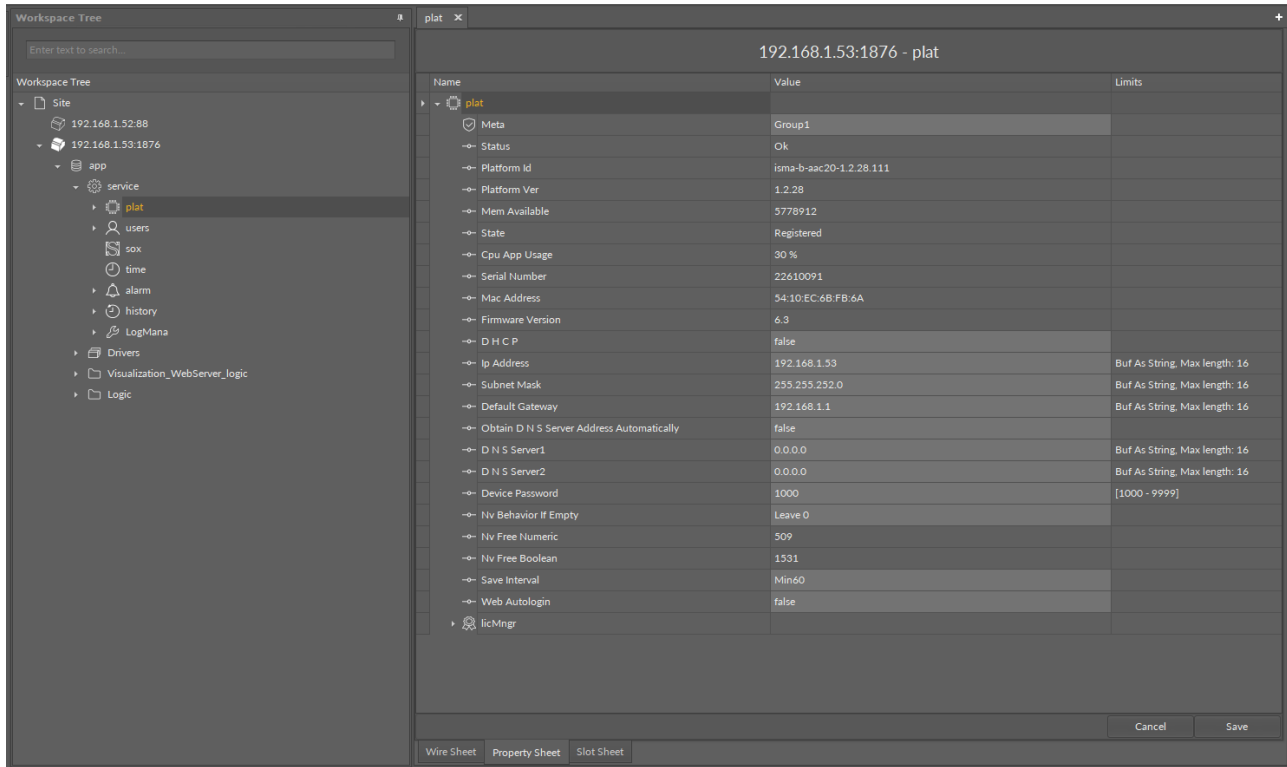


Figure 19. The Plat component view

The plat service has the following slots:

- **Status:** shows the component's status;
- **Platform Id:** shows the platform ID;
- **Platform Ver:** shows the platform's version;
- **Mem Available:** shows the RAM memory available in the controller;
- **State:** shows the license status;
- **Cpu App Usage:** shows the CPU app usage calculated using a formula: scan time/scan period * 100%;
- **Serial Number:** shows the serial number of the device;
- **Mac Address:** shows the MAC address of the device;
- **Firmware Version:** shows the firmware version installed in the controller;
- **DHCP:** enables address setting by the DHCP server;
- **IP Address:** allows to set the device's IP address (if the DHCP slot is enabled, this slot shows the current IP address);
- **Subnet Mask:** allows to set the device's subnet mask (if the DHCP slot is enabled, this slot shows the current IP mask);
- **Default Gateway:** allows to set the device's default gateway IP address (if the DHCP slot is enabled, this slot shows the current gateway IP address);
- **Obtain DNS Server Address Automatically:** allows to obtain the DNS IP address automatically from the DHCP server (if the DHCP slot is enabled), or set it manually;
- **DNS Server 1:** allows to set a first DNS server IP address;
- **DNS Server 2:** allows to set a second DNS server IP address;
- **Device Password:** allows to set a password for the platform user;

- **Nv Behavior If Empty:** allows to adjust the non-volatile components behavior after copying the output value (available options: Leave 0, Copy from Default);
- **Nv Free Numeric:** shows a number of available numeric non-volatile components;
- **NV Free Boolean:** shows a number of available Boolean non-volatile components;
- **Save Interval:** allows to set the application autosave interval;
- **Web Autologin:** allows to enable or disable the password protection to access the web page.

The plat service has the following actions:

- **Restart:** restarts the application (Sedona Virtual Machine);
- **Reboot:** reboots the device;
- **Reload Firmware:** reboots the device and updates the firmware if a proper firmware file has been successfully sent to the device before;
- **CopyFromNvToDefault:** copies values from the Out slot to the Default slot in all NV components (see section NV Components);
- **CopyFromNvToUser:** copies values from the Out slot to the User slot in all NV components (see section NV Components);
- **CopyFromDefaultToNv:** copies values from the Default slot to the Out slot in all NV components (see section NV Components);
- **CopyFromUserToNv:** copies values from the User slot to the Out slot in all NV components (see section NV Components);
- **SetAllNvInAuto:** sets all NV components in auto mode (see section NV Components).

5.1.1 Changing IP Address

iSMA-B-AAC20 controllers have two built-in Ethernet ports working in switch mode. By default, the new device's address is set to 192.168.1.123, subnet mask to 255.255.255.0, and default gateway to 192.168.1.1. The device offers the option to change the IP address at three levels:

- the iSMA Tool application;
- the website;
- the device display (available only for devices with a built-in display: iSMA-B-AAC20-LCD, iSMA-B-AAC20-LCD-M, iSMA-B-AAC20-LCD-D).

All these options allow to set a static IP address or enable an option of addressing by the DHCP server.

Changing IP Address in Application

To change the IP address in the application, log in to the device (admin or user with authorization to change the address in the plat component). Then go to the plat service (Device -> App -> Service -> plat) and make changes to the slots: IP Address, Subnet Mask, Default Gateway, and save the application (App Component -> Save action). To make changes, restart the device using the Reboot action in the App component. The device restarts with the new IP address.

Changing IP Address over Web Page

To change the IP address of the device, go to the device's website—enter the current IP address in the URL field of any browser, log in using a “platform” username and a numerical password from the Device password slot in the plat component (default: 1000).

Go to the IP Configuration tab and make appropriate changes in the following slots: IP Address, Subnet Mask, Gateway (Mac Address slot is read-only), click Submit to save the application, and restart the device using the Reboot button. The device will start up with the new IP address.

WARNING! Once changing of the IP address is implemented with a reboot, change the URL address in the browser accordingly in order to enter the device’s website.

Changing IP Address Using System Display (iSMA-B-AAC20-LCD, iSMA-B-AAC20-LCD-M, iSMA-B-AAC20-LCD-D)

To change the IP address from the system display, hold the F1 function key down until the screen shows "Enter password". Then enter a numeric password (plat component -> Device Password slot; the default value is 1000). In the system menu, use arrows on the right side to go to the Network Config line and press Enter. Then select the appropriate line and press Enter again. The changes make the arrow on the right side (+ / -) move to the next position by pressing the Enter key. To validate the change, hold Enter down until transferred to the system menu and then select Reboot. The device starts up with the new IP address.

5.2 Users Service

The users service allows to manage users of the controller. The service includes a User Manager view, which shows a list of registered users and allows to adjust their access rights to individual components (each Sedona application component has a Meta slot used to assign it to one or more groups of access rights; Sedona has 4 predefined groups, which are accessible in the User Manager). The User Manager also enables adding and removing users.

WARNING! In the users service only Sedona users can be modified. The “platform” user is the control operating system and can only be modified (password change) from the plat component, built-in display, or from the controller’s web page.

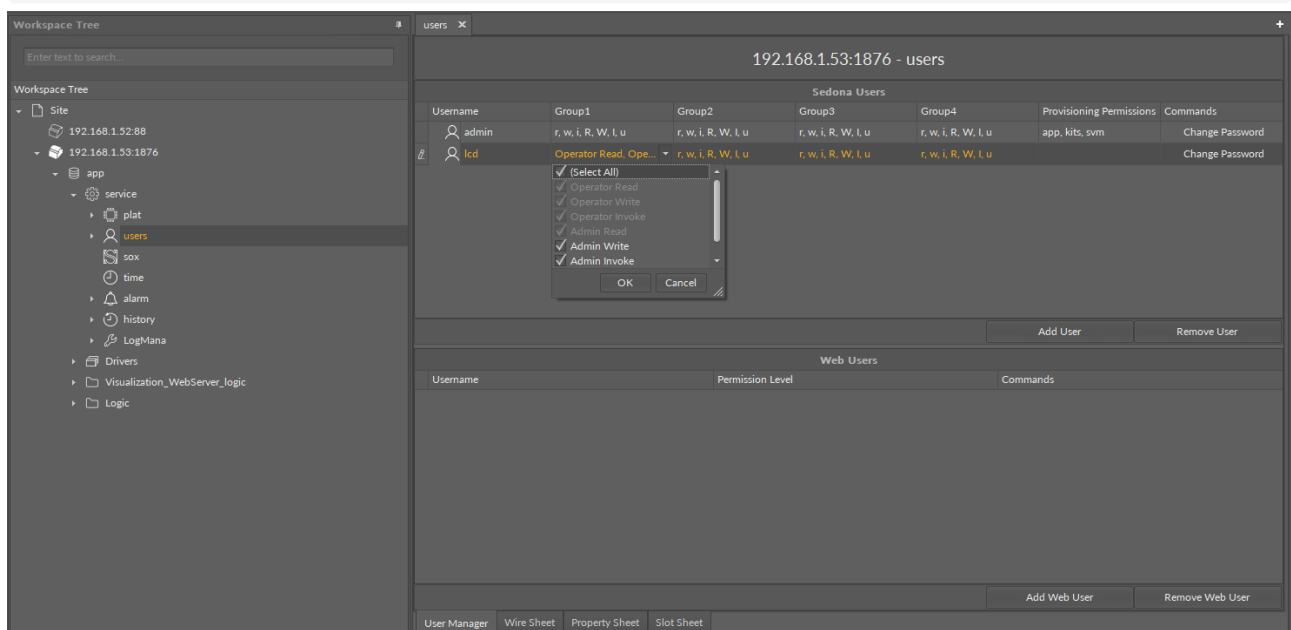


Figure 20. User Manager view

5.2.1 User Rights and Permissions

Users can be assigned the following types of rights:

- **Operator Read:** allows to read components, read values of operator properties;
- **Operator Write:** allows to change values of operator properties;
- **Operator Invoke:** allows to invoke operator actions;
- **Admin Read:** allows to read values of properties, read links, generate components links;
- **Admin Write:** allows to change values of properties, add components, sort dub components, rename components, generate links to components, delete links to components;
- **Admin Invoke:** allows to invoke admin actions of components;
- **Admin User:** allows to manage users (read, write, edit, delete).

Users can also be assigned the following provisioning permissions:

- Can provision app: can read/write app.sab file;
- Can provision app: can read/write kits.scode file;
- Can provision app: can read/write SVM files.

For devices with a built-in LCD, displaying components such as action editing/invoking is defined by assigning the given component to a group and defining access rights for the user from this group.

5.3 DateTime

The DateTimeService is a component, which operates the built-in real time clock (RTC). The clock can be synchronized with a local PC's clock, or by using the iSMASox driver, with the Niagara device. For more information regarding the DateTimeService see the iSMA Tool manual.

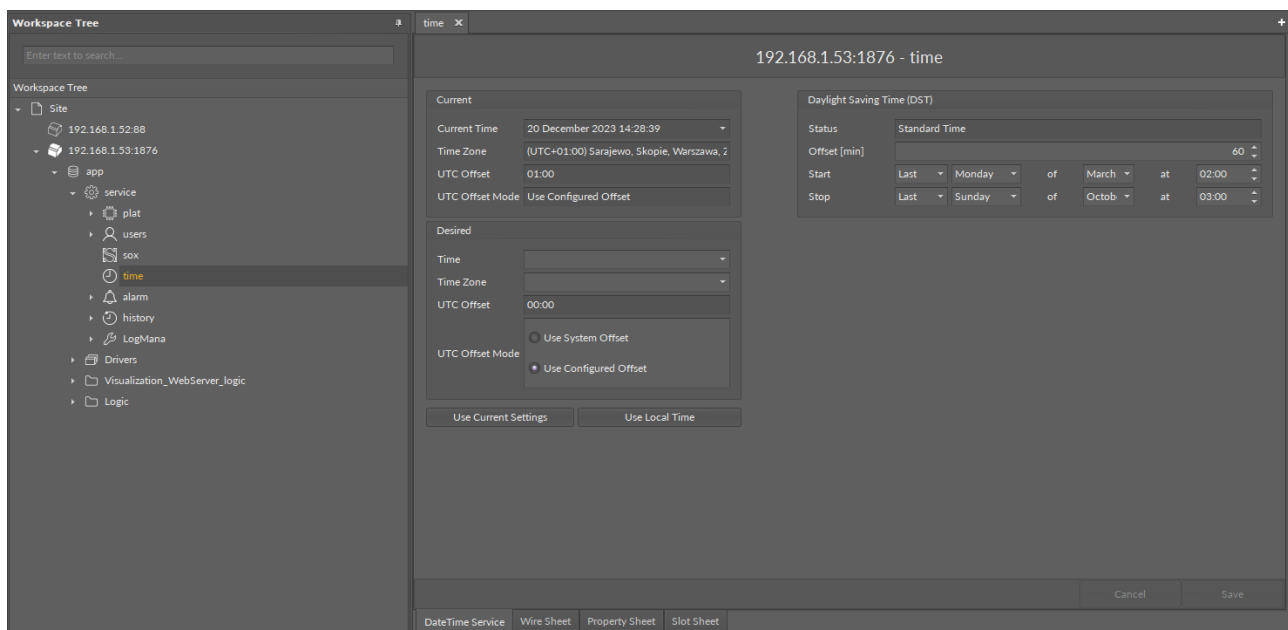


Figure 21. DateTimeService

6 NV Components

The NV components (non-volatile) are the components which values can be recorded in an EEPROM device's non-volatile memory. Whenever the device is restarted or the power is down, the values of NV components remain saved. The device has three types of NV components, broken down by the types of variables they support, which include:

- Boolean variables: NVBooleanWritable component;
- Integer variables: NVIntegerWritable component;
- Numeric (float) variables: NVNumericWritable component.

NV components can operate in the Auto mode (the In slot values are transferred to the Out slot) or in the Hand mode (the Out value is entered manually by the user and cannot be changed by the application).

There are available 512 memory cells for numeric (float/numeric and integer) values and 1536 memory cells for Boolean values in the iSMA-B-AAC20 controllers. NVNumeric and NVInteger components use one numeric cell memory for the Out value and one Boolean memory cell for the Auto/Hand switch mode. NVBoolean components use two Boolean memory cells, one for the Out value and one for the Auto/Hand switch mode.

Since the values of the components are not stored in the Sedona application but in the non-volatile memory of the device, when an application is copied between two devices, output values are not saved and will assume the values stored in the local EEPROM memory. To copy NV components to another device along with their values (e.g., setpoint), use global actions of the plat component:

Step 1: Use global actions CopyFromNvToDefault / CopyFromNvToUser;

Step 2: Save the application and copy it to another device;

Step 3: Use global actions on the target device CopyFromDefaultToNv / CopyFromUserToNv.

6.1 NVBooleanWritable

The NVBooleanWritable is the component that stores the output value in the non-volatile EEPROM memory of the device. After rebooting the device or in case of the power failure, the component value is restored from this memory. The iSMA-B-AAC20 controllers have 1536 Boolean memory cells. The occupied space meter for EEPROM is embedded in the plat component. The NVBooleanWritable component occupies two Boolean memory cells (value of the component and the position of the Auto/Hand switch).

The NVBooleanWritable component is also used to integrate Boolean variables from various sources. This is done using the 'reverse following the link' function. The Out slot is connected to the In slots of various protocols, for example, LCD or Modbus variable. When changing a value in one of the components, the device will perform the Set action on the NV component to synchronize the values in all the connected components.

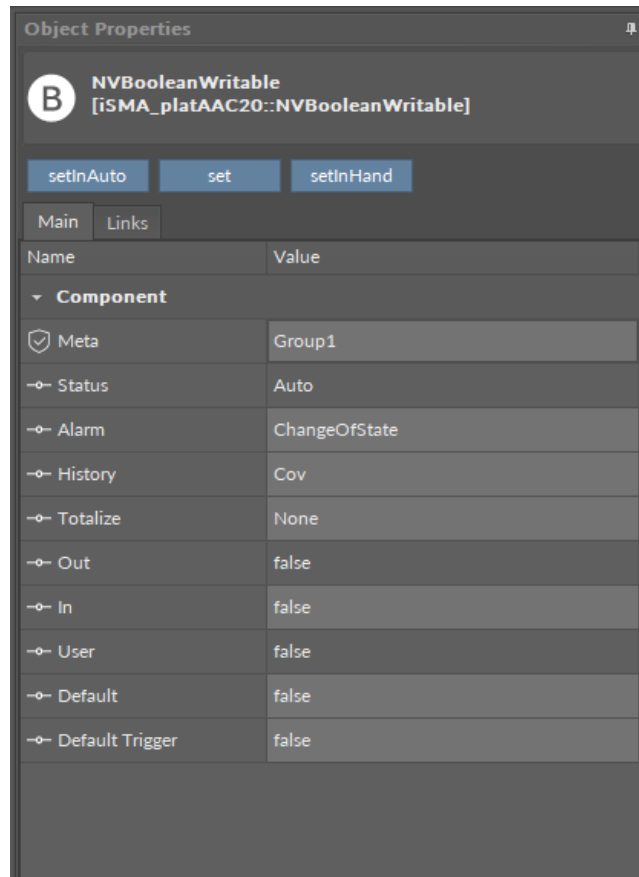


Figure 22. NVBooleanWritable component

The NVBooleanWritable component has the following slots:

- **Status:** shows the current status of the component (Auto/Hand);
- **Alarm:** alarm extension;
- **History:** history extension;
- **Totalize:** totalize extension;
- **Out:** the output slot,
- **In:** the input slot,
- **User:** the user value slot (introduced by the Set action);
- **Default:** the default value slot (set by global command from the plat component);
- **Default Trigger:** copies trigger from the Default slot to the Out slot.

The NVBooleanWritable component has the following actions:

- **Set:** allows to manually set the User slot and the In slot if there is no link to the In slot;
- **Set In Hand:** this option sets the value on Out slot and blocks changing from any other slots;
- **Set In Auto:** this option switches off the Hand mode and sets Out slot according to the In slot's value.

The NVBooleanWritable component allows adding extensions such as:

- **Alarm:** (change of state) generates alarms when the component changes state;
- **History:** records the component's value at a specific point in time over a defined period of time (interval), at a change of a value (cov), or in both of these cases (covInterval);
- **Totalize:** counts the time over which the value of a point is in a particular state.

6.2 NVIntegerWritable

The NVIntegerWritable is a component that stores the output value in non-volatile EEPROM memory of the device. After rebooting the device or in case of the power failure, the component value is restored from this memory. The iSMA-B-AAC20 controllers have 1536 Boolean memory cells and 512 numeric cells. Space meter of the occupied EEPROM's memory is located in the plat component. The NVIntegerWritable component occupies one memory cell of the numeric type (component value) and one memory cell of the Boolean type (switch position Auto/Hand).

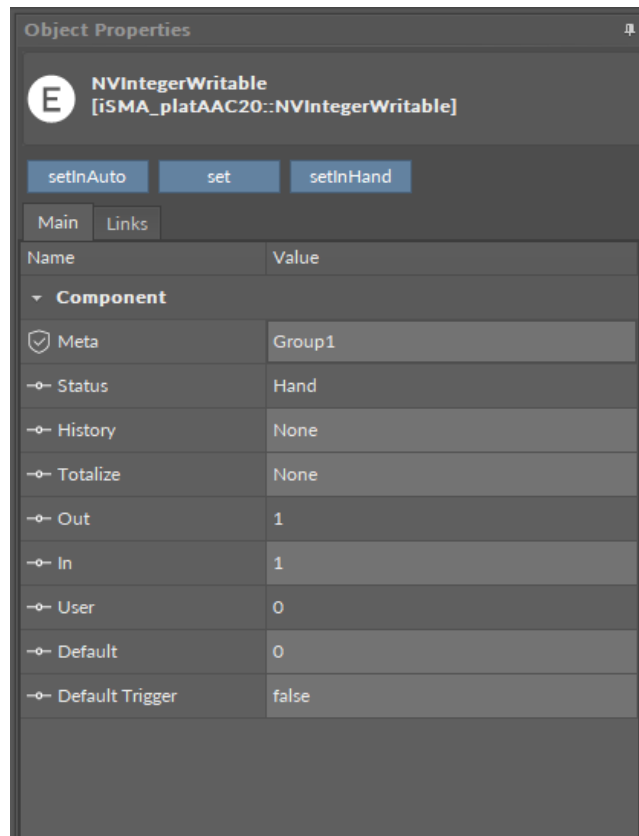


Figure 23. NVIntegerWritable component

The NVIntegerWritable component has the following slots:

- **Status:** shows the current status of the component (Auto/Hand);
- **History:** history extension;
- **Totalize:** totalize extension;
- **Out:** the output slot,
- **In:** the input slot,
- **User:** the user value slot (introduced by the Set action);
- **Default:** the default value slot (set by global command from the plat component);
- **Default Trigger:** copies trigger from the Default slot to the Out slot.

The NVIntegerWritable component has the following actions:

- **Set:** allows to manually set the User slot and the In slot if there is no link to the In slot;
- **Set In Hand:** this option sets the value on Out slot and blocks changing from any other slots,
- **Set In Auto:** this option switches off the Hand mode and sets Out slot according to the In slot's value.

The NVIntegerWritable component allows to add extensions such as:

- **History:** records the component's value at a specific point in time over a defined period of time (interval), at a change of a value (cov), or in both of these cases (covInterval);
- **Totalize:** counts the time when the point's value is not zero.

6.3 NVNumericWritable

The NVNumericWritable is the component that stores the output value in non-volatile EEPROM memory of device. After rebooting the device or in case of the power failure, the component value is restored from this memory. The iSMA-B-AAC20 controllers have 1536 Boolean memory cells and 512 numeric cells. Space meter of the occupied EEPROM's memory is located in the plat component. The NVNumericWritable component occupies one memory cell of the numeric type (component value) and one memory cell of the Boolean type (switch position Auto/Hand).

The NVNumericWritable component is also used to integrate numeric (float) variables from various sources. This is done using the "reverse following the link" function. The Out slot is connected to the In slots of various protocols, for example, LCD or Modbus variable. When changing the value in one of the components, the device will perform the Set action on the NV component to synchronize the values in all the connected components.

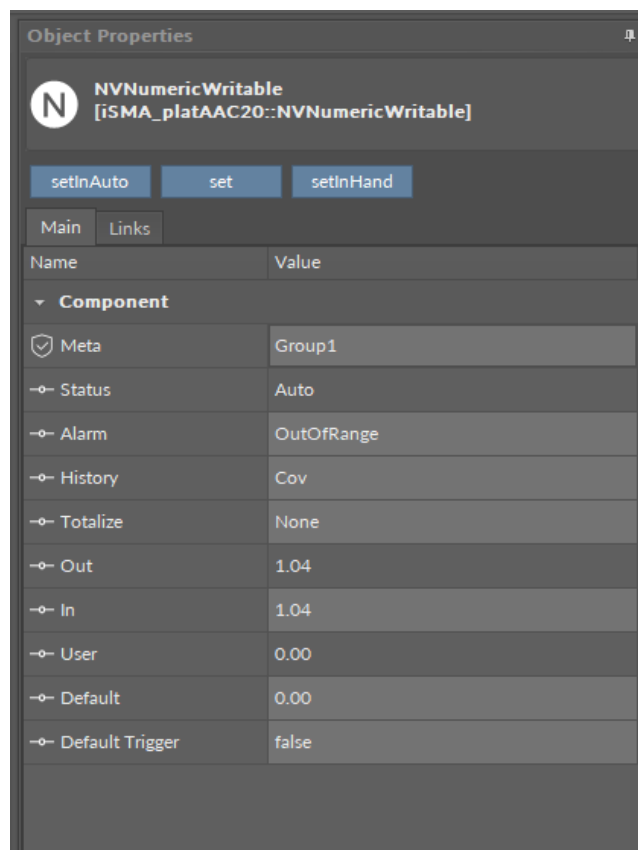


Figure 24. NVNumericWritable component

The NVNumericWritable component has the following slots:

- **Status:** shows the current status of the component (Auto/Hand);
- **Alarm:** alarm extension;
- **History:** history extension;
- **Totalize:** totalize extension;

- **Out:** the output slot,
- **In:** the input slot,
- **User:** the user value slot (introduced by the Set action);
- **Default:** the default value slot (set by global command from the plat component);
- **Default Trigger:** copies trigger from the Default slot to the Out slot.

The NVNumericWritable component has the following actions:

- **Set:** allows to manually set the User slot and the In slot if there is no link to the In slot;
- **Set In Hand:** this option sets the value on Out slot and blocks changing from any other slots,
- **Set In Auto:** this option switches off the Hand mode and sets Out slot according to the In slot's value.

The NVNumericWritable component allows to add extensions such as:

- **Alarm:** (OutOfRange) generates alarms when the component's value is out of a set range;
- **History:** records the component's value at a specific point in time over a defined period of time (interval), at a change of a value (cov), or in both of these cases (covInterval);
- **Totalize:** counts the time when the point's value is not zero.

6.4 NVMultiStateWritable

The NVMultiStateWritable is a component that stores the output value in non-volatile EEPROM memory of device. After rebooting the device or in case of the power failure, the component value is restored from this memory. The iSMA-B-AAC20 controllers have 1536 Boolean memory cells and 512 numeric cells. Space meter of the occupied EEPROM's memory is located in the plat component. The NVMultiStateWritable component occupies one memory cell of the numeric type (component value) and one memory cell of the Boolean type (switch position Auto/Hand).

In the NVMultiStateWritable component it is possible to define sixteen string outputs depending on the Integer value on the input of the component.

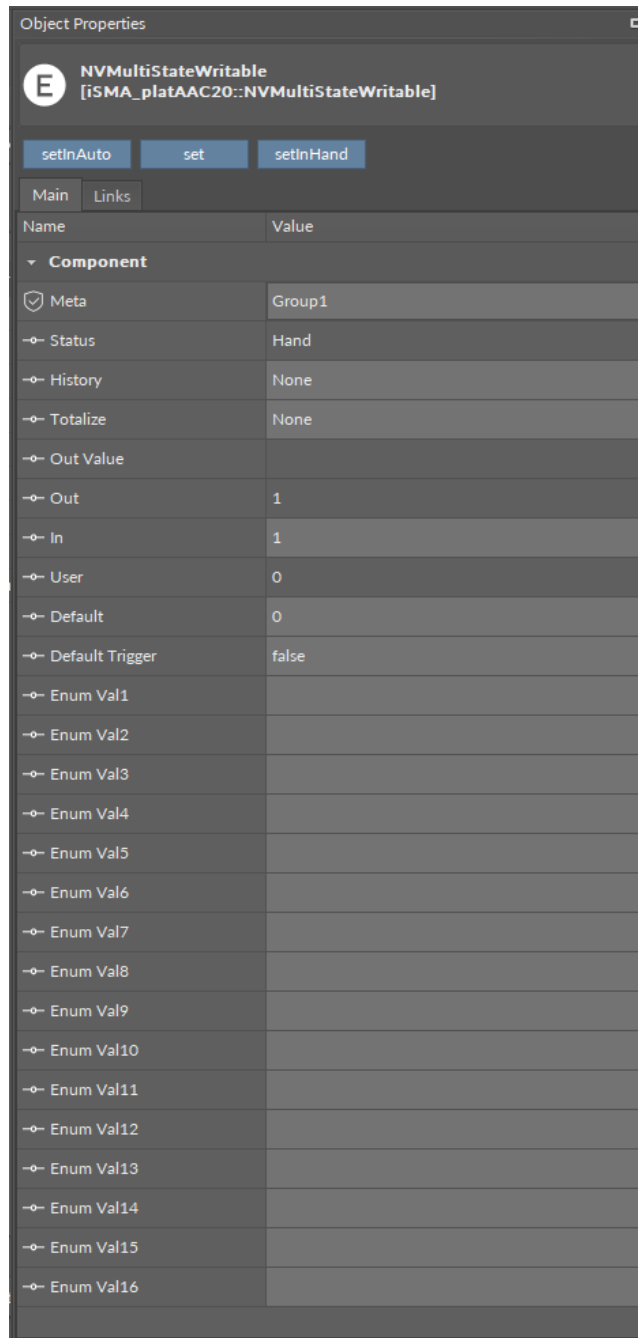


Figure 25. NVMultiStateWritable component

The NVMultiStateWritable component has the following slots:

- **Status:** shows the current status of the component (Auto/Hand);
- **History:** history extension;
- **Totalize:** totalize extension;
- **Out:** the output slot,
- **In:** the input slot,
- **User:** the user value slot (introduced by the Set action);
- **Default:** the default value slot (set by global command from the plat component);
- **Default Trigger:** copies trigger from the Default slot to the Out slot;
- **Enum Val1-Val16:** string values which will be shown on the output depending on the input value.

The NVMultiStateWritable component has the following actions:

- **Set:** allows to manually set the User slot and the In slot if there is no link to the In slot;
- **Set In Hand:** this option sets the value on Out slot and blocks changing from any other slots,
- **Set In Auto:** this option switches off the Hand mode and sets Out slot according to the In slot's value.

The NVMultiStateWritable component allows you to add extensions such as:

- **History:** records the component's value at a specific point in time over a defined period of time (interval), at a change of a value (cov), or in both of these cases (covInterval);
- **Totalize:** counts the time when the point's value is not zero.

7 History Extension

The iSMA-B-AAC20 controllers have an embedded history extension for writing trends in the microSD card memory. By default, the device is not equipped with an SD card, history service SD card should be installed when necessary. The history extension can be added only in the NVBooleanWritable, NVIntegerWritable, and NVNumericWritable components. To add the extension, choose one of three available types in the History slot components. Every historical type has predefined 2500 recordable samples. After reaching the limit, the oldest recordings are overwritten. While creating a historical record, the device opens a database file on the SD card. Any such file can serve up to 200 extensions. Once the limit is reached, the new database file is opened.

7.1 History Service

The history service is a component, which is responsible for writing historical values in the microSD memory. The history service is available in the iSMA_controlApi kit palette and the component must be placed under the service folder.

After any historical extension is added on the NV component, a historical database will be created under the history service component in order to store NV points of historical components.

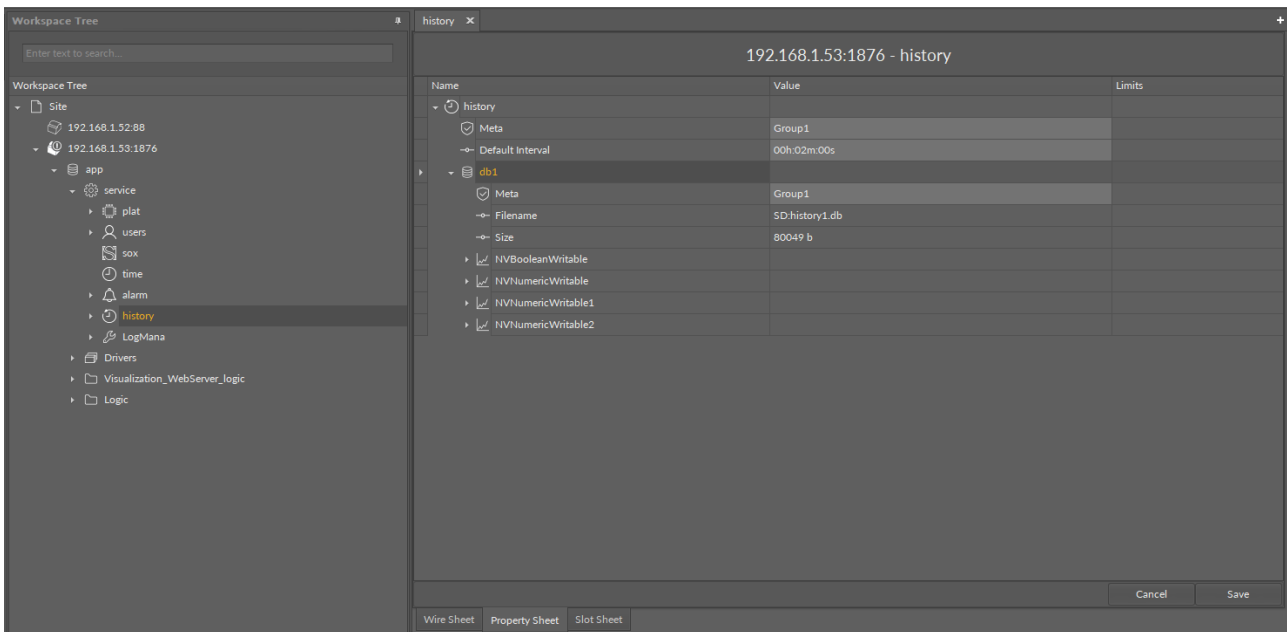


Figure 26. The History service in the Property Sheet view

7.2 COV

The COV (Change of Value) history extension is used to collect data whenever a specific change of value occurs. If the COV option is chosen in the NV component, a history COV component is created under the history service, and a link with NV point Out slot is made. For configuration settings please refer to historical components under the history service.

WARNING! To delete or change a historical extension type, do not remove it manually from the history service. Go to the NV component, choose the None option in the history slot, and save it (extension will be deleted by the system, along with all the database settings). In order to change the type, choose the None option first, save, and after that choose the required option.

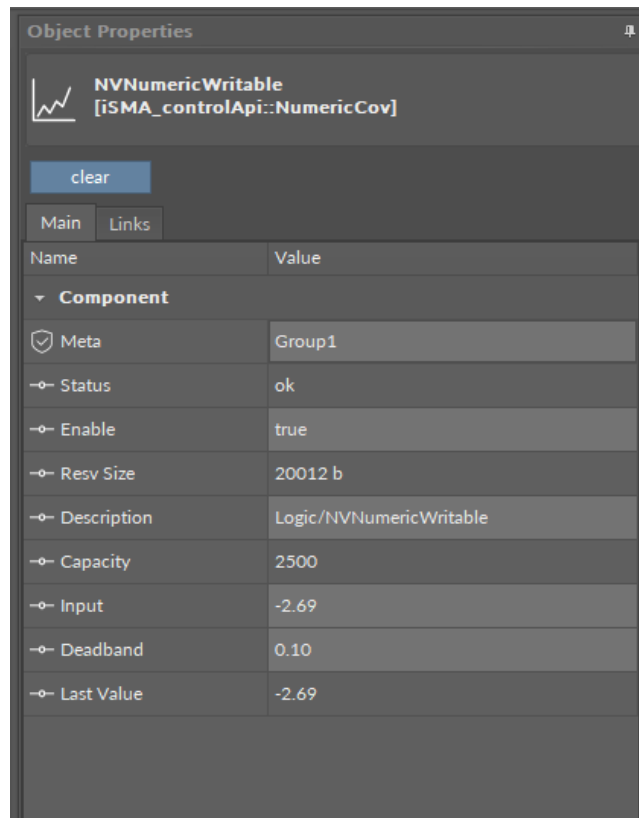


Figure 27. History COV

The component has the following slots:

- **Status:** shows the history component status;
- **Enable:** allows to enable or disable collecting the history logs;
- **Resv Size:** allows to set the history memory reserve size;
- **Description:** allows to enter a component's description;
- **Capacity:** shows the number of available history records;
- **Deadband:** allows to save hysteresis,
- **Input:** current value;
- **Last Value:** shows the last saved value.

7.3 Interval

Interval history extension is used to collect data at specified time intervals. When you choose interval option in the NV component, a history interval will be created under the history service and a link to the NV point Out slot will be created. For configuration and settings, please refer to historical components under the history service.

WARNING! To delete or change a historical extension type, do not remove it manually from the history service. Go to the NV component, choose the None option in the History slot, and save it (extension will be deleted by the system, along with all the database settings). In order to change the type, choose the None option first, save, and after that choose the required option.

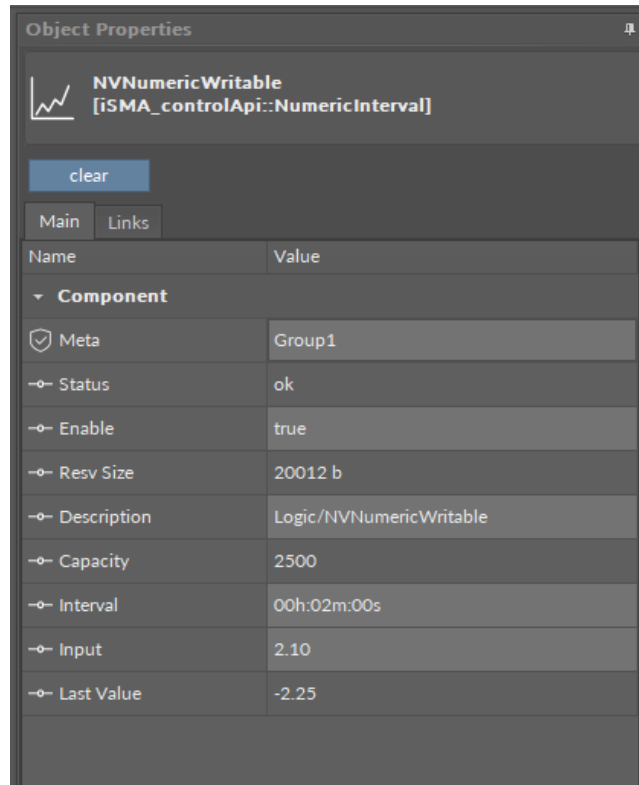


Figure 28. History interval

The component has the following slots:

- **Status:** shows the history component status;
- **Enable:** allows to enable or disable collecting the history logs;
- **Resv Size:** allows to set the history memory reserve size;
- **Description:** allows to enter a component's description;
- **Capacity:** shows the number of available history records;
- **Interval:** allows to set the saving time interval;
- **Input:** current value;
- **Last Value:** shows the last saved value.

7.4 COV_Interval

The COV_Interval history extension is used to collect data whenever required changes has been made or at specified time intervals (whatever happens first). When the COV_Interval option is chosen in the NV component, a history COVInterval component is created under the history service, and a link with NV point Out slot will be made. For configuration settings please refer to historical components under the history service.

WARNING! To delete or change a historical extension type, do not remove it manually from the history service. Go to the NV component, choose the None option in the history slot, and save it (extension will be deleted by the system, along with all the database settings). In order to change the type, choose the None option first, save, and after that choose the required option.

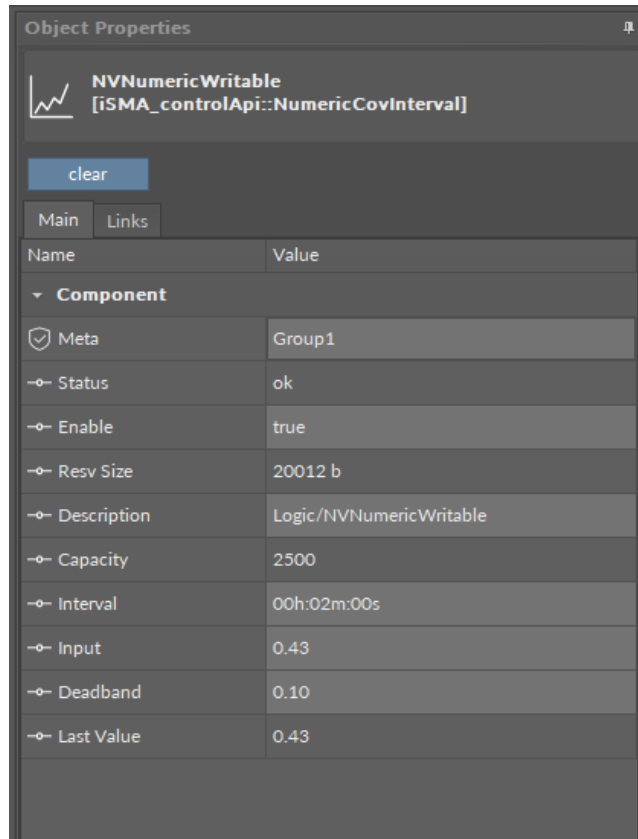


Figure 29. History_COV_Interval

The component has the following slots:

- **Status:** shows the history component status;
- **Enable:** allows to enable or disable collecting the history logs;
- **Resv Size:** allows to set the history memory reserve size;
- **Description:** allows to enter a component's description;
- **Capacity:** shows the number of available history records;
- **Deadband:** allows to save hysteresis;
- **Interval:** allows to set the saving time interval;
- **Input:** current value;
- **Last Value:** shows the last saved value.

8 Alarm Extension

The iSMA-B-AAC20 controllers have a service that allows recording events / alarms. Alarms can be generated off the NV-type components only. Alarms can be recorded only on the SD card. By default, the device is not equipped with an SD card. In order to use the alarm service, SD card should be installed. When an extension of the alarm is created, the database file is created on the SD card. The device stores up to 2500 alarm events. If this number is exceeded, the new event will overwrite the oldest records.

To add an extension to the component alarm, select an appropriate option in the Alarm slot (Change Of State for Boolean components, and Out Of Range for Numeric components). After changing and saving the settings, an alarm extension is created under the component, where all the settings associated with the process of generating an alarm event can be adjusted.

8.1 Alarm Service

The alarm service is a component, which is responsible for servicing alarms events. The alarm service component is available in the iSMA_controlApi kit and the component must be placed under the service folder.

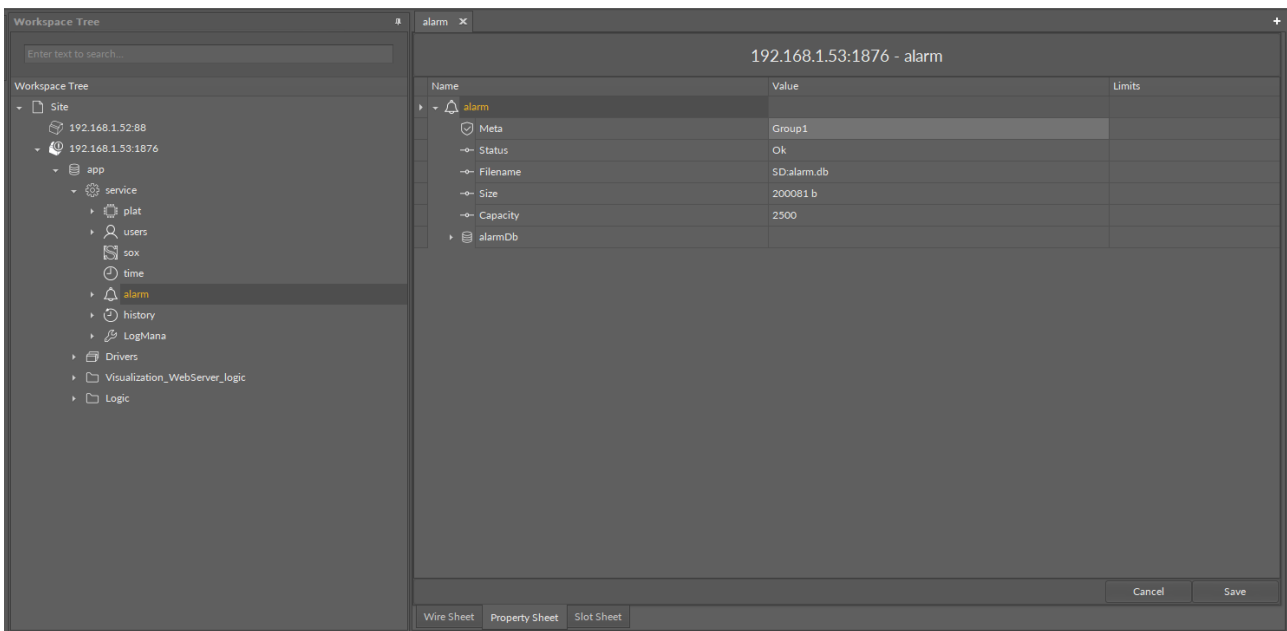


Figure 30. Alarm service in the Property Sheet view

8.2 Alarm Boolean Points

The Boolean alarm extension can be added only in the NVBooleanWritable components. To add the extension, choose the Change Of State option in the alarm slot, and an alarm extension will be created automatically under the NV component.

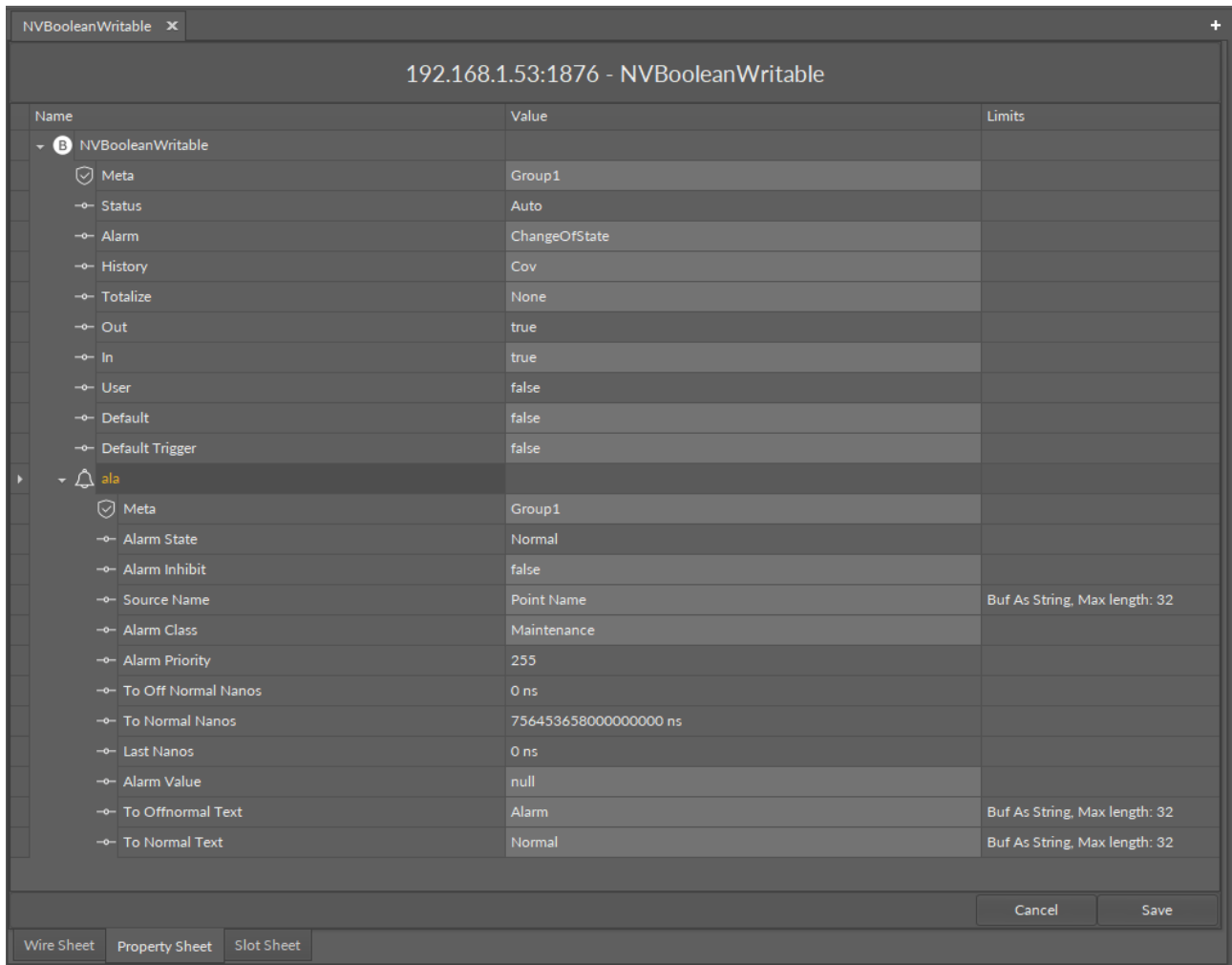


Figure 31. Alarm Boolean extension in the Property Sheet view

The alarm extension has the following slots:

- **Alarm Inhibit:** allows to enable or disable the alarm function;
- **Source Name:** allows to set the alarm name in the alarms database;
- **Alarm Class:** allows to set the alarm class and priority definition;
 - Available options: Live Safety (priority 15), Critical (priority 75), Maintenance (priority 155);
- **Alarm Priority:** shows the alarm priority number defined in the alarm class slot (default value is 255);
- **To Off Normal Nanos:** shows the time point of an alarm condition;
- **To Normal Nanos:** shows the time point of a normal condition;
- **Alarm State:** shows the actual alarm state;
- **Alarm Value:** allows to set an alarm generation value;
- **To Offnormal Text:** allows the enter a description in alarm state;
- **To Normal Text:** allows the enter a description in normal state.

8.3 Alarm Numeric Points

The numeric alarm extension can be added only in NVNumericWritable components. To add the extension, choose the Out Of Range option in the Alarm slot, and the alarm extension will be created automatically.

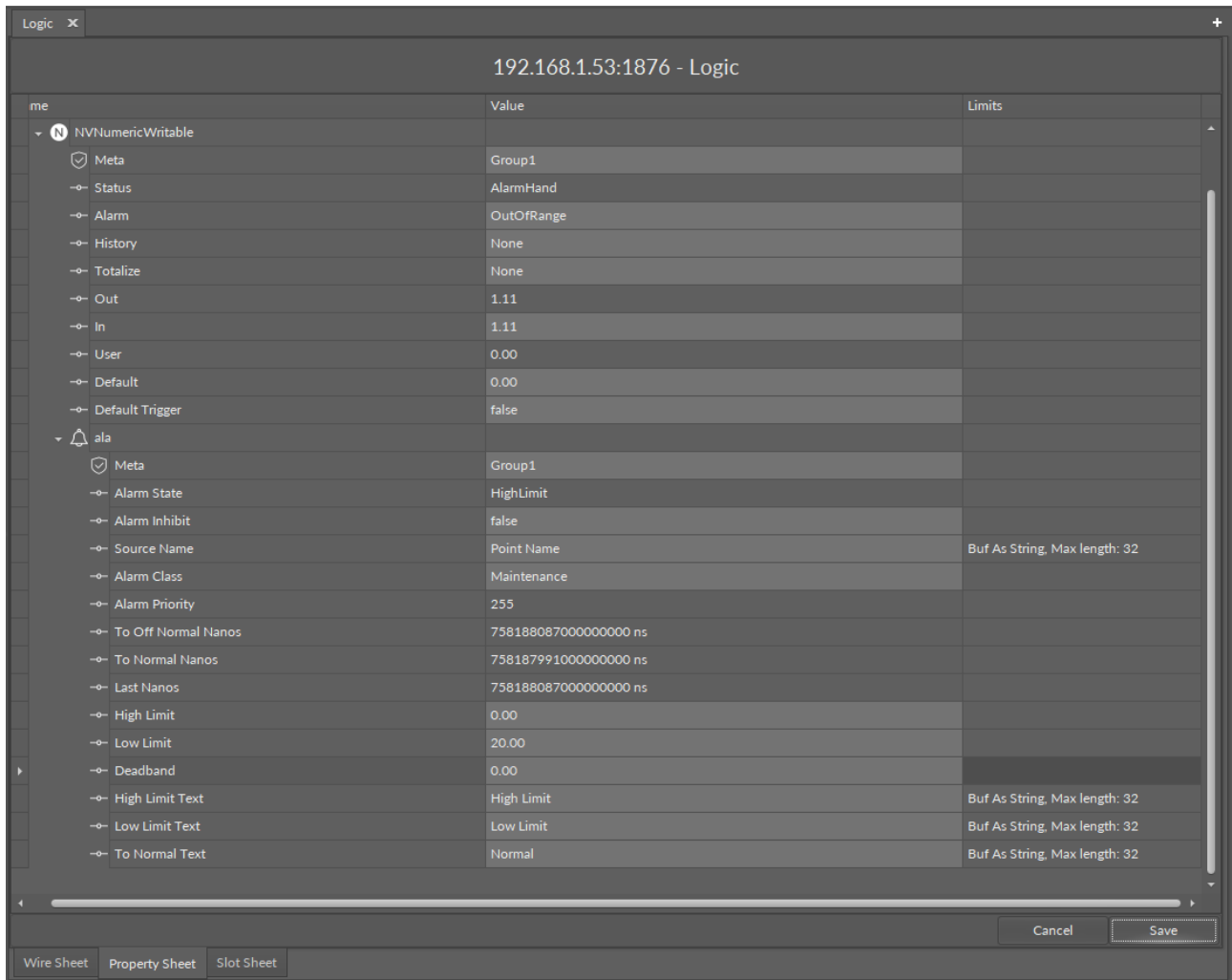


Figure 32. Alarm numeric extension in the Property Sheet view

The component has the following slots:

- **Alarm State:** indicates a current status of the alarm function;
- **Alarm Inhibit:** allows to enable or disable the alarm function;
- **Source Name:** allows to set the alarm name in the alarms database;
- **Alarm Class:** allows to set the alarm class and priority definition;
 - Available options: Live Safety (priority 15), Critical (priority 75), Maintenance (priority 155);
- **Alarm Priority:** shows the alarm priority number defined in the alarm class slot (default value is 255);
- **To Off Normal Nanos:** shows the time point of an alarm condition;
- **To Normal Nanos:** shows the time point of a normal condition;
- **Last Nanos:** shows the last time point;
- **High Limit:** allows to set a high limit alarm generation value;
- **Low Limit:** allows to set a low limit alarm generation value;
- **Deadband:** allows to set a high and low limit deadband value;
- **High Limit Text:** allows to enter a description in alarm high state;
- **Low Limit Text:** allows to enter a description in alarm low state;
- **To Normal Text:** allows to enter a description in normal state.

9 Scheduler

The schedules in iSMA-B-AAC20 controllers are being created using schedule components, as found in the iSMA_controlApi kit (kit is installed as part of a default application). There are two types of scheduling components available- BooleanScheduleWeekly for Boolean values and NumericScheduleWeekly for numeric values. For both types there are available max. 8 events for 7 week days. In a scheduling component the start time and value is defined. If value is null (for Boolean) or nan (for numeric), the event is discarding. For non-defined event scheduler will take value from the Def Val slot.

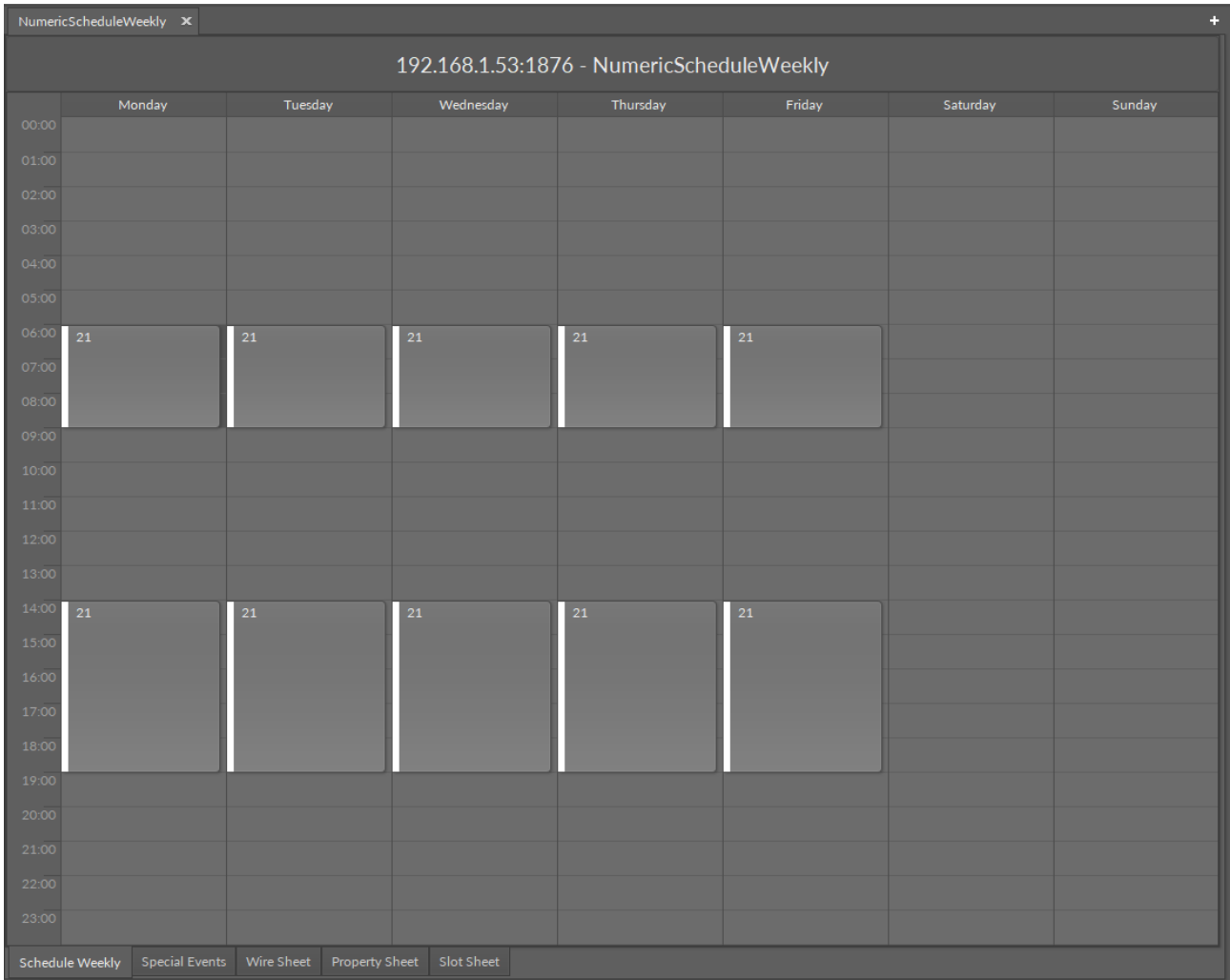


Figure 33. Schedule Weekly view

10 SOX Protocol

Sox is a standard protocol used to communicate with Sedona devices. The Sox protocol is also used by the iSMA Tool to program controllers. Note that Sox is of a service type and it is executed after application components. Therefore, if a difference between scan period and Scan Time (see App component) is minor, the services do not have enough time to be executed, and the programming interface can slow down. Sox is designed to be run over UDP/IP protocol (default port is 1876).

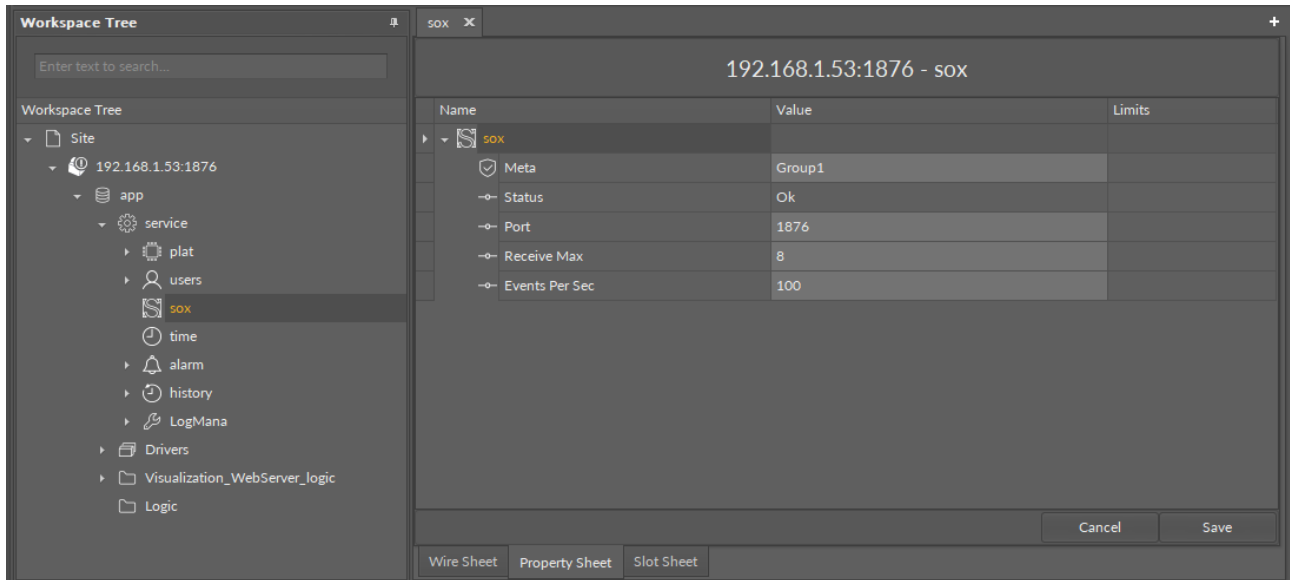


Figure 34. sox component

The sox component has the following slots:

- **Status:** shows the sox driver actual status;
- **Port:** allows to set the sox UDP port (default value is 1876);
- **Receive Max:** allows to set a maximum number of messages in receiving window;
- **Events Per Sec:** allows to set a maximum number of async events (telegrams) sent per second.

11 1-Wire

1-Wire is a special protocol for device which is made by the Dallas company. Temperature sensors use the DS18D20 chip. The 1-Wire bus uses 3 wire cable (+5 VCC, Data, GND) of maximum 100 m long (it is recommended to use shorter distance) and up to 32 sensors. All iSMA-B-AAC20 devices are equipped with a 1-Wire port which uses two analog outputs, AO5 and AO6.

WARNING! Before connecting a 1-Wire device to the controller, please add a 1-Wire network first. It will block the possibility of increasing voltage above 5 V and protect the 1-Wire sensor against damage.

11.1 Adding 1-Wire Network

To use the 1-Wire protocol, the iSMA_OneWire kit needs be installed on the controller using the Kit Manager (see chapter [Kit Manager](#)). The OneWireNetwork component needs to be placed under the Drivers folder.

Note: If a 1-Wire network has been added to the controller, the AO5 and AO6 are blocked for the 1-Wire type and cannot be used otherwise, for example, as regular outputs. This restriction cannot be changed manually.

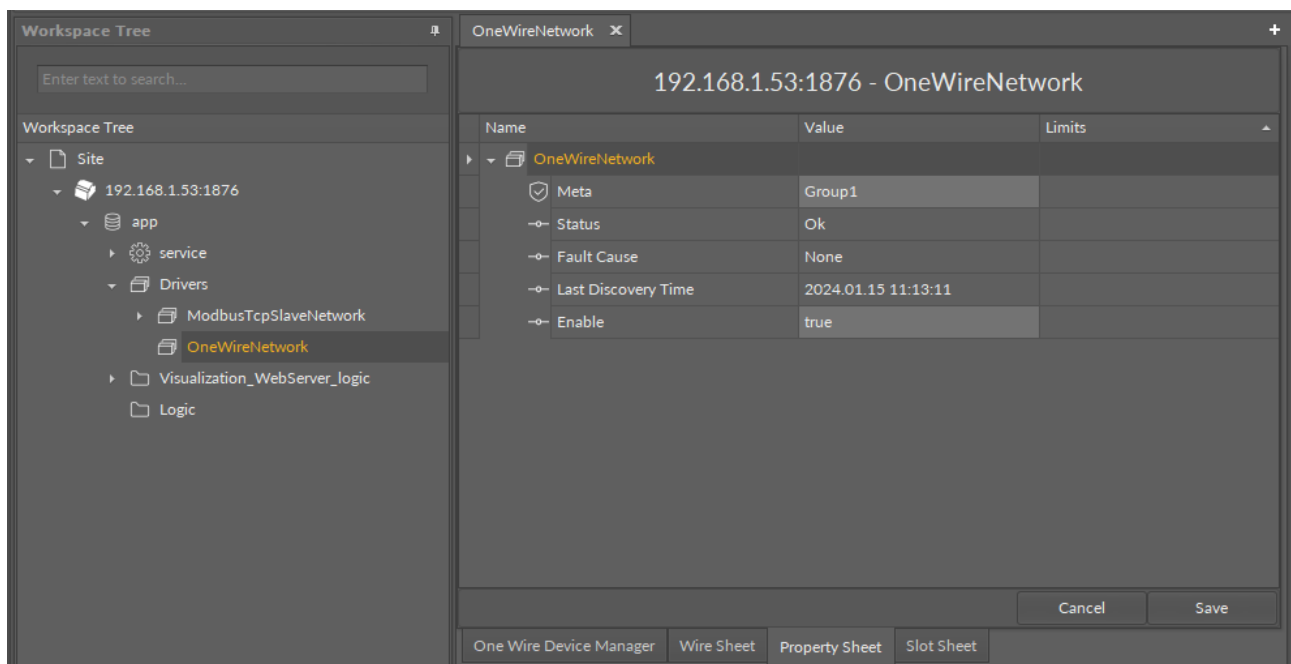


Figure 35. 1-Wire in the Property Sheet view

11.2 Discovering 1-Wire Sensors

The 1-Wire network has a discover action (in the context menu). This action will automatically search and detect all sensors connected to the bus. It will also create a discover folder and place all found sensors there. All sensors have a unique address assigned by the manufacturer (slot Address). There is also a Description slot to describe sensor function / place using max. 20 characters. All sensors components can be moved and grouped in OneWireFolders (folders must be also placed under the OneWireNetwork component).

11.3 Adding 1-Wire Sensors Manually

To add a sensor manually, the OneWireThermometer component needs to be moved from the iSMA_OneWire kit palette and be placed under the OneWireNetwork. Then sensor address needs to be entered and the application must be saved. If the connection and address are correct, the Status slot should have the Ok value, and the Out slot should show sensor temperature given in Celsius degrees.

12 LocalIO Driver

Local I/Os in the iSMA-B-AAC20 controller are supported by the iSMA_localIO kit. The iSMA_localIO kit consists of the following components:

- **localIO driver:** main component servicing physical I/Os;
- **LocalIOConfig:** component allowing to configure parameters of physical inputs and outputs of the device;
- **LocalIOFolder:** folder for grouping I/O components;
- 9 components servicing physical local I/Os:
 - **AODigital:** component for servicing analog output in digital mode (false: 0 V, true: 10 V);
 - **AOVoltage:** component for servicing analog output in voltage mode;
 - **DI:** component for servicing digital inputs;
 - **DICounter:** component reading high-speed counters of digital inputs;
 - **DO:** component for servicing digital outputs;
 - **UIDigital:** component for servicing universal inputs in dry contact mode;
 - **UIResistance:** component for servicing universal inputs in resistance mode;
 - **UITemperature:** component for servicing universal inputs in temperature mode (in order to get reliable readings, the appropriate temperature sensor type has to be selected in the LocalIOConfig component);
 - **UIVoltage:** component for servicing universal inputs in voltage mode (in order to improve readings, the LocalIOConfig component has to be activated for voltage reading).

In order to work properly components of this kit must be placed in a special LocalIOFolder under the localIO driver.

Note: It is recommended to pass the inputs and outputs signals through NV components.

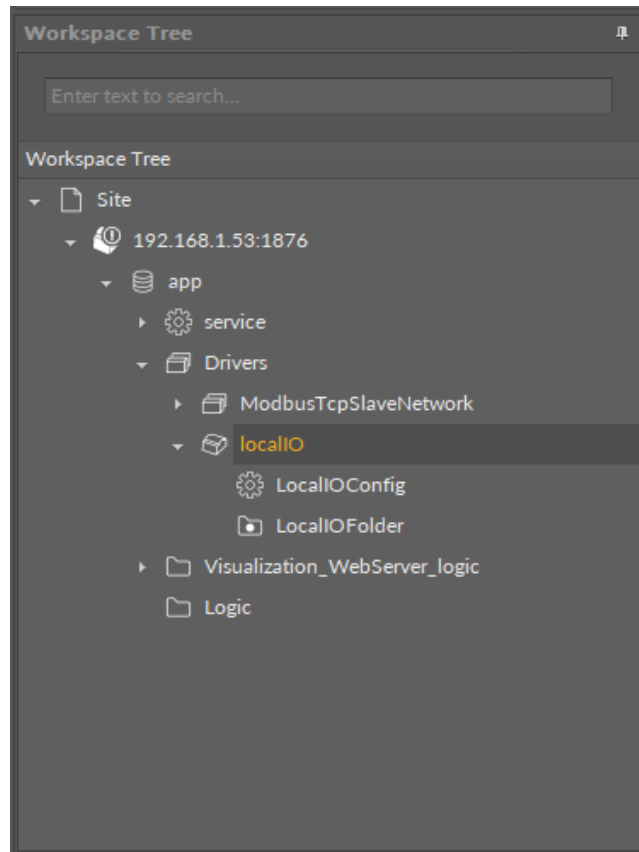


Figure 36. LocalIO driver

12.1 LocalIOConfig

The LocalIOConfig component is designed to configure the physical inputs and outputs of the device.

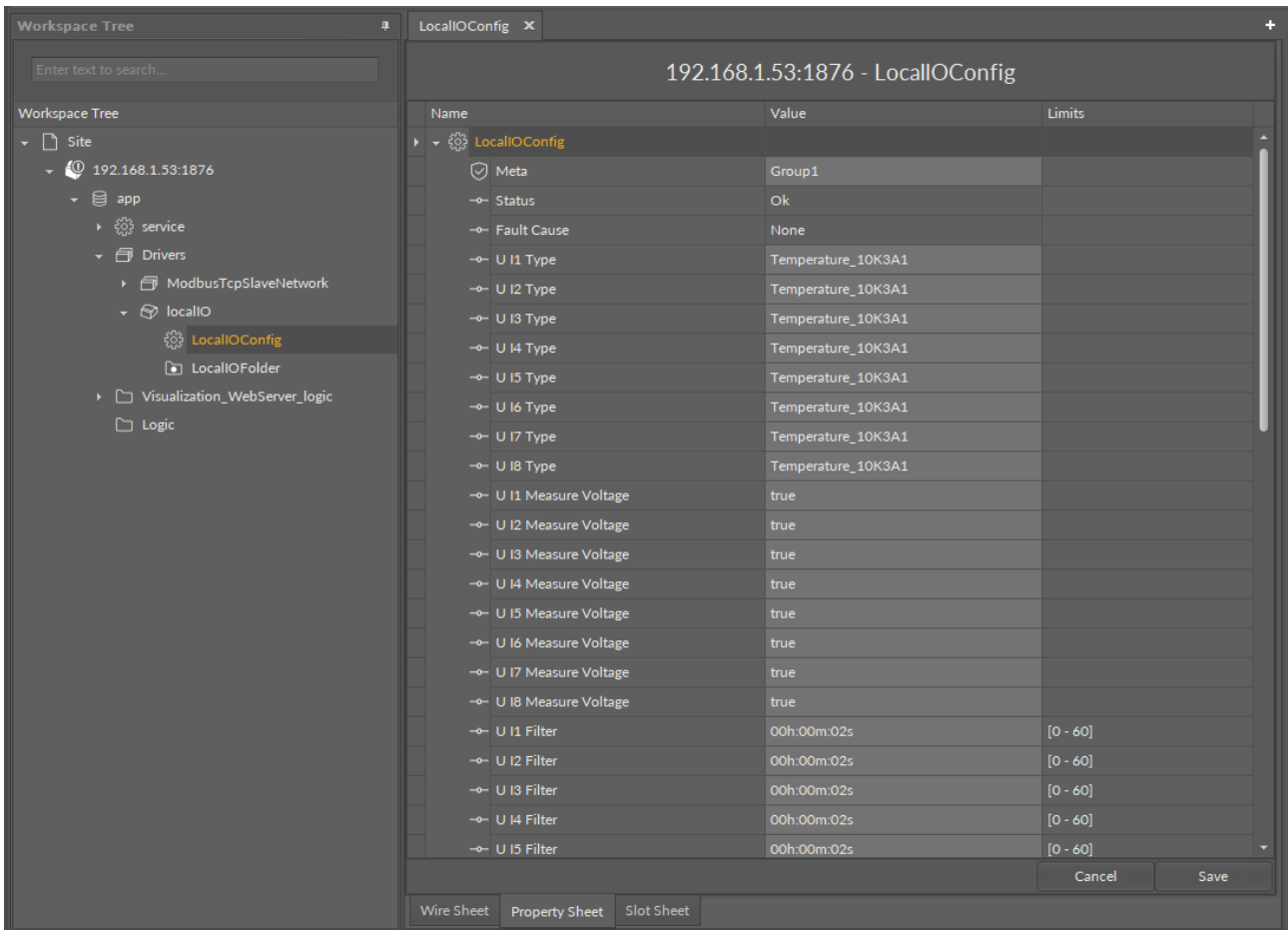


Figure 37. LocalIOConfig

The LocalIOConfig component has the following slots:

- **Status:** shows the component’s status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **U11-8 Type:** allows to select a temperature sensor for each universal input; table stored in the device allows to convert the sensor value from resistance to temperature;
 - Available options: Disable_Resistance, Temp_10K3A1, Temp_10K4A1, Temp_10K, Temp_20K6A1, Temp_2_2K3A1, Temp_3K3A1, Temp_30K6A1, Temp_SEI1, Temp_TAC1, Temp_SAT1, Temp_PT1000, Temp_NI1000, Temp_Ni1000_21C, Temp_Ni1000_LG;

Note: PT1000 and NI1000 sensors work only with 16-bit resolution.

- **U11-8 Measure Voltage:** allows to enable or disable voltage measurement for each universal input;
- **U11-8 Filter:** the time constant of the low pass filter (to eliminate signal noise);
- **U11-8 Resolution:** allows to set measurement resolution of each universal input;

WARNING! 16-bit resolution increases the time necessary to read a single entry. It should be used only when sensors PT1000 and NI1000 are connected.

- **AO1-6 Type:** allows to set an analog output mode:

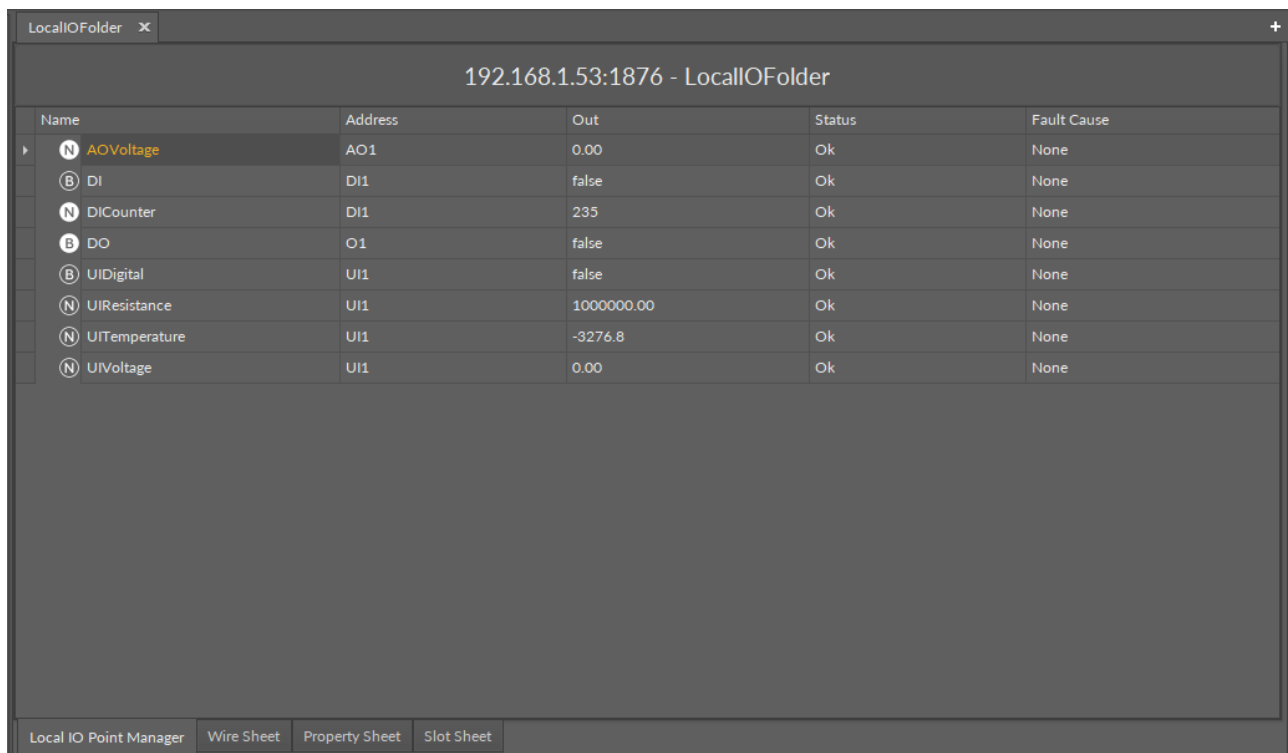
- Available options: voltage 0-10 V, PWM (option available only for AO1-4; AO5 and 6 do not support PWM mode);
- **Default State Of Analog Output:** allows to set a default value of the analog output after the controller has been rebooted (expressed in mV);
- **Default Digital State Of Analog Output:** allows to set a default value of the analog output working in digital mode after the controller has been rebooted;
- **Default State Of Digital Output:** allows to set a default value of the digital output after the device has been rebooted.

Note: Adding the LocalIOConfig component to the application disables the possibility of I/O configuration from the website and Modbus array.

Any settings of the LocalIOConfig component are stored in the component and can be transferred to other devices as part of a saved application (quick setup of multiple devices).

12.2 LocalIOFolder

I/O components service the physical inputs and outputs of the iSMA-B-AAC20 controller. The components are designed according to the input or output type and signal. In order to work properly, the I/O components have to be placed in the LocalIOFolder. Double-click on the LocalIOFolder opens the Local IO Point Manager, which is a view listing all added local I/Os and showing their basic data.



The screenshot shows a web interface window titled "LocalIOFolder" with a sub-header "192.168.1.53:1876 - LocalIOFolder". It contains a table with the following data:

Name	Address	Out	Status	Fault Cause
AOVoltage	AO1	0.00	Ok	None
DI	DI1	false	Ok	None
DICounter	DI1	235	Ok	None
DO	O1	false	Ok	None
UIDigital	UI1	false	Ok	None
UIResistance	UI1	1000000.00	Ok	None
UITemperature	UI1	-3276.8	Ok	None
UIVoltage	UI1	0.00	Ok	None

At the bottom of the window, there are navigation tabs: "Local IO Point Manager" (selected), "Wire Sheet", "Property Sheet", and "Slot Sheet".

Figure 38. LocalIOFolder

12.2.1 AODigital

The AODigital component allows to control the analog output in dry contact mode.

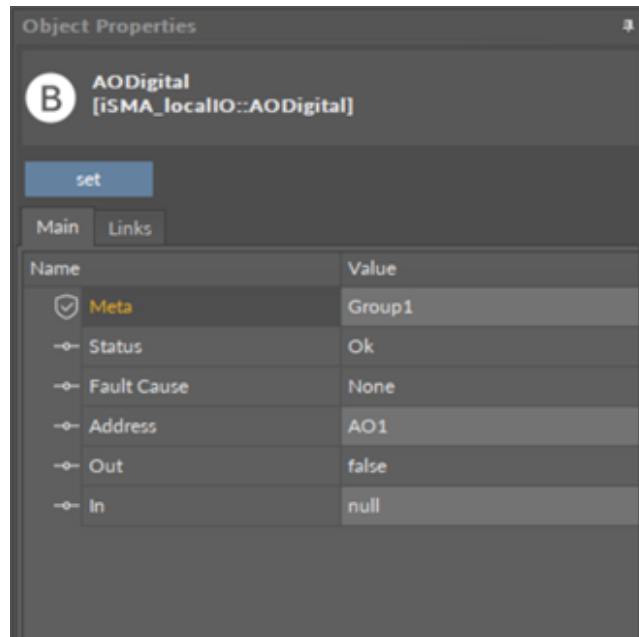


Figure 39. AODigital component

Slots

The AODigital component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical output of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot;
 - Available settings: AO1-6.
- **Out:** a value transferred from the In slot and then to the physical output;
- **In:** a value read from an application or network and then transferred to the Out slot.

Action

The AODigital component has one action available:

- **Set:** allows to manually set the In slot value.

12.2.2 AOVoltage

The AOVoltage component allows to control the analog output in voltage mode. The analog output in voltage mode transmits values from 0 to 10 V, which controls 0-100% of fan or pump operation.

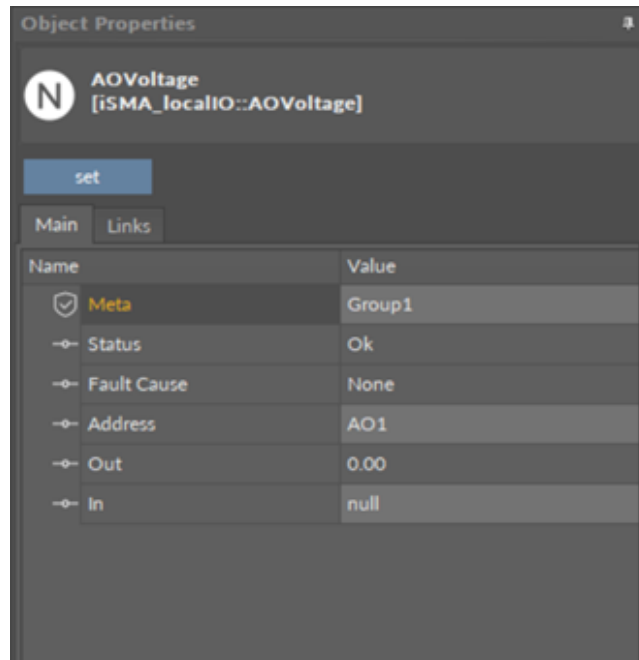


Figure 40. AOVoltage component

Slots

The AOVoltage component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical output of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: AO1-6.
- **Out:** a value transferred from the In slot and then to the physical output;
- **In:** a value read from an application or network and then transferred to the Out slot (expressed in mV, range 0-10000 mV).

Action

The AOVoltage component has one action available:

- **Set:** allows to manually set the In slot value.

12.2.3 DI

The DI component allows to control the digital input in the controller.

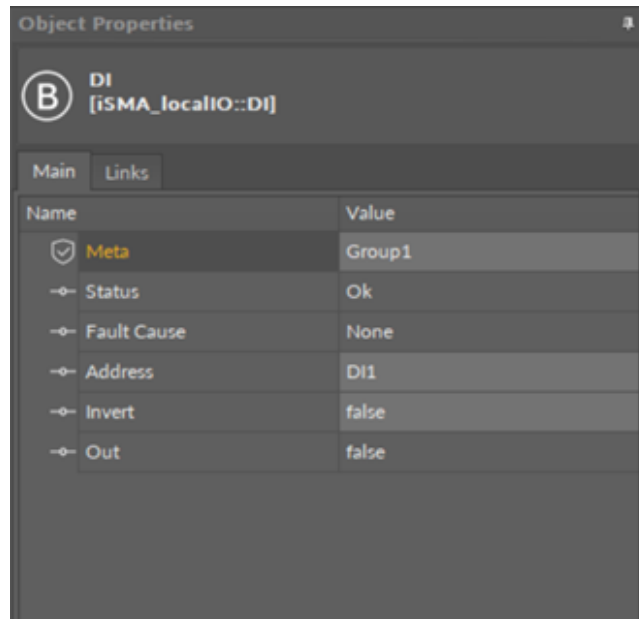


Figure 41. DI component

Slots

The DI component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical input of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
- - Available settings: DI1-4.
- **Invert:** a Boolean slot allowing to invert the value read from the digital input of the device;
- **Out:** a real value read from the digital input of the device.

12.2.4 DICounter

The DICounter component allows to reading high-speed counters of digital inputs in the controller.

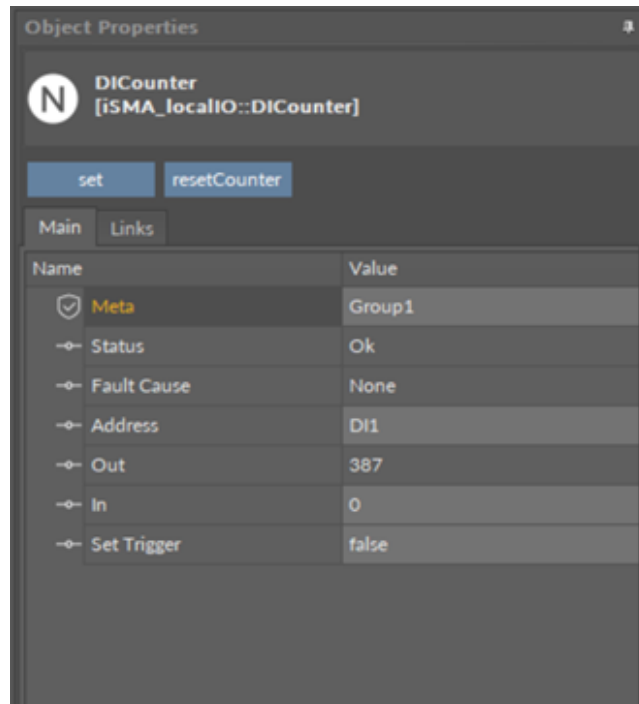


Figure 42. DICounter component

Slots

The DICounter component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical input of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: DI1-4.
- **Out:** a number of pulses on rising edge in a physical digital input of the device;
- **In:** the input slot receiving a value to be counted;
- **Set Trigger:** allows to activate or deactivate counting pulses on rising edge.

Actions

The DICounter component has actions available:

- **Set:** allows to manually set the In slot value;
- **resetCounter:** allows to reset the counter value to 0.

12.2.5 DO

The DO component allows to control the digital output in dry contact mode.

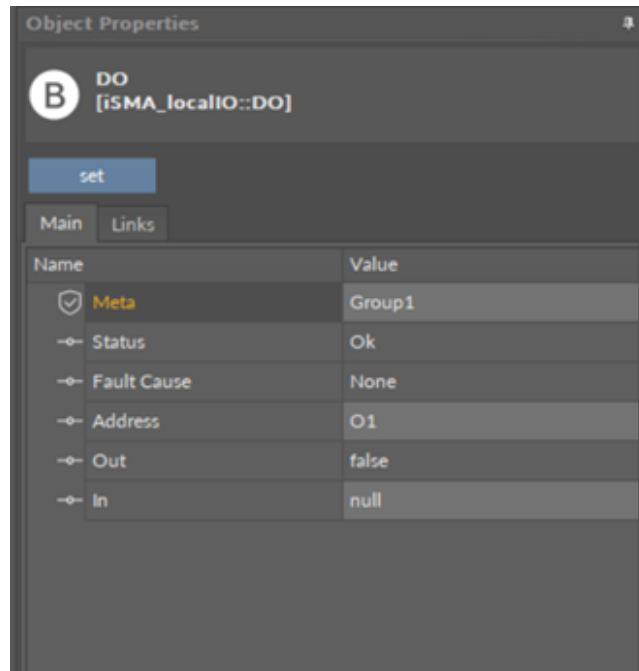


Figure 43. DO component

Slots

The DO component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical output of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: O1-4.
- **Out:** a value transferred from the In slot and then to the physical output;
- **In:** a value read from an application or network and then transferred to the Out slot.

Action

The DO component has one action available:

- **Set:** allows to manually set the In slot value.

12.2.6 UIDigital

The UIDigital component allows to control the universal input in dry contact mode.

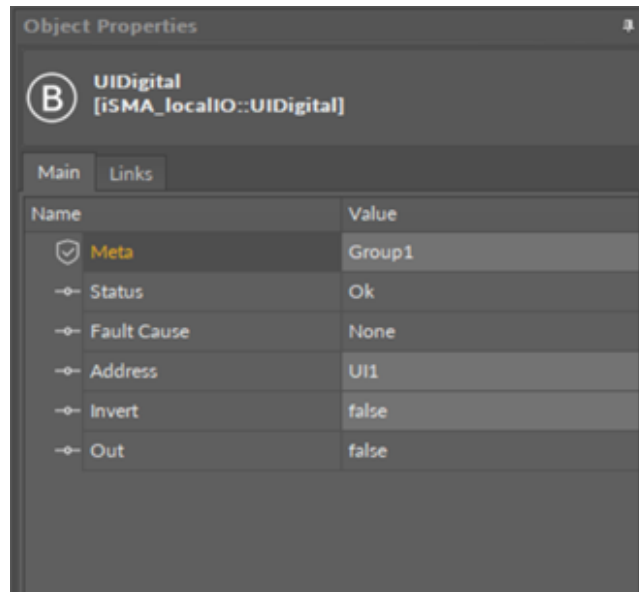


Figure 44. UIDigital component

Slots

The UIDigital component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical input of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: UI1-8.
- **Invert:** a Boolean slot allowing to invert the value read from the universal input of the device;
- **Out:** a real value read from the universal input of the device.

12.2.7 UIResistance

The UIResistance component allows to control the universal input in resistance mode.

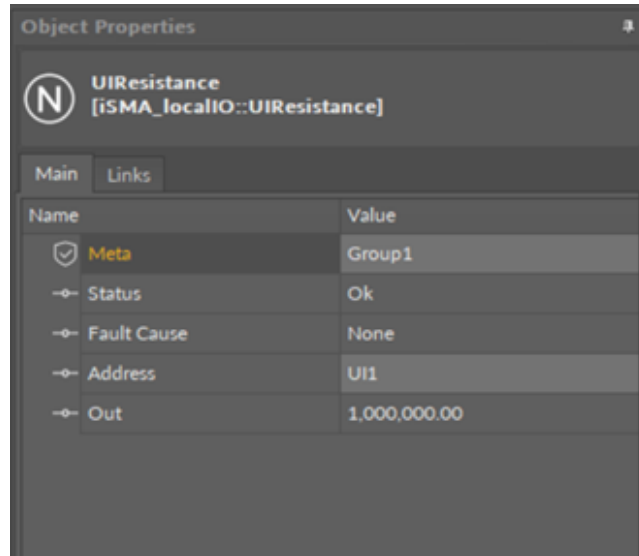


Figure 45. UIResistance component

Slots

The UIResistance component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical input of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: UI1-8.
- **Out:** a real value read from the universal input of the device.

12.2.8 UITemperature

The UITemperature component allows to control the universal input in temperature mode. The component's readings are based in the type of temperature sensor selected in the LocalIOConfig component.

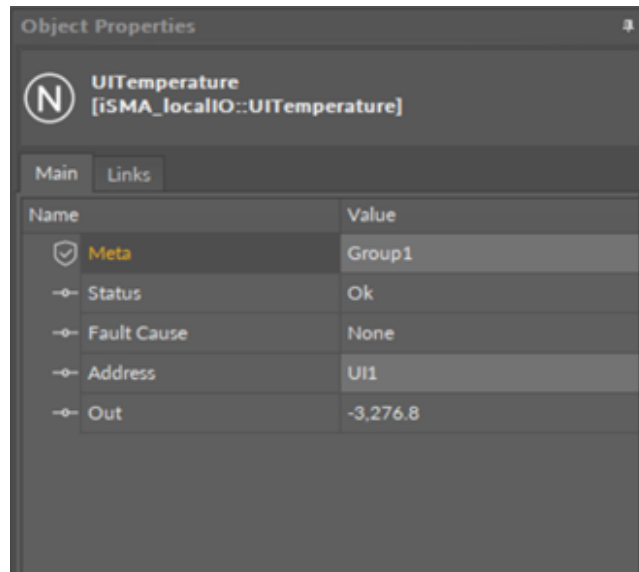


Figure 46. UITemperature component

Slots

The UITemperature component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical input of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: UI1-8.
- **Out:** a real value read from the universal input of the device.

12.2.9 UIVoltage

The UIVoltage component allows to control the universal input in voltage mode. The component receives values from 0 to 10 V.

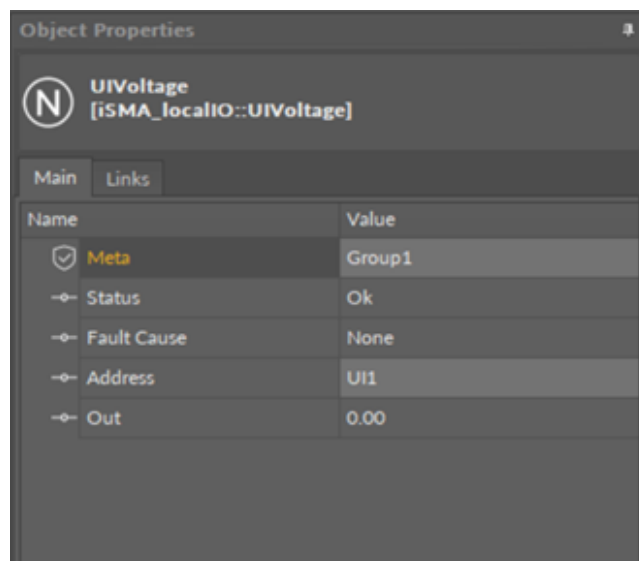


Figure 47. UIVoltage component

Slots

The UIVoltage component has the following slots:

- **Status:** shows the component's status;
 - Available information: OK (the component is working properly), Fault (the component is disabled);
- **Fault Cause:** shows the reason of the Fault status;
- **Address:** allows setting an address of a physical input of the device; once the component has been added to the folder, the slot's default value is 1—for the component to operate properly, the unique address value must be set in this slot.
 - Available settings: UI1-8.
- **Out:** a real value read from the universal input of the device.