User Manual

Control Software

March 8, 2022

User Manual Version 1.0

Applies for Control Software Version 2.0

support@bluefors.com
+358 9 5617 4800

BF1000-1234517327-71

BLUEFORS

IMPORTANT

Read this user manual carefully before use. Keep the manual for future reference.
For reading digital copies of this document, use a PDF reader.

IMPORTANT

The information contained within this document is effective as of the publication date. Bluefors Oy reserves the right to make changes to the product and information contained within this document relative to the specifications, features, and design of the product.

IMPORTANT

The information contained within this publication covers a wide range of applications and may not specifically apply to your equipment layout or custom setup. Contact us directly (support@bluefors.com) if you have any question about the specifications or any other content contained within this publication.

BF1000-1234517327-71

BLUEFORS

Table of Contents

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

Table of Figures

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

List of Tables

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

1 Introduction

Control Software is a software for controlling the Bluefors dilution refrigerator measurement systems.

2 Getting started

2.1 System requirements

The Control Software needs a computer with a 64-bit Windows operating system and at least 8 gigabytes of memory.

The software also needs the National Instruments DAQmx framework to be installed with support libraries for C language. This is needed for communicating with the Bluefors Control Card handling flowmeter reading and switching valves on and off.

Many of the devices installed in the system are connected to the computer via serial bus or have an internal virtual serial port. To work with these, the drivers for them must be installed as well. If the software is just being updated and the existing drivers worked with a previous version, they will also work with the new software.

Depending on the system configuration, some of the following drivers may be needed:

- **MaxiGauge TPG366**: MaxiGauge has its own virtual serial port, which may need a driver to be installed. The driver has been delivered together with the system.
- **Lake Shore 372 resistance bridge**: This device may require a driver for its virtual serial port, which can be downloaded from the Lake Shore website[1] if the device is not detected.
- **Moxa UPort 1600-8**: Most recent Bluefors systems have a USB serial port device from Moxa. This requires drivers to be installed, which can be found on the Moxa website[2].
- **FT2232**: Older Bluefors systems may use serial ports based on the FT2232 chip, drivers for which can be found on the FTDI Chip website[3].
- **PL-2302**: There are a few old systems with Belkin-branded USB-RS232 cables, which use a Prolific chip that will not work with Windows 10 and is known to cause stability issues with older Windows versions. If they are still working, they can be safely used, but as they are not supported, no drivers

[1] https://www.lakeshore.com/resources/software
[2] https://www.moxa.com/en/products/industrial-edge-connectivity/usb-to-serial-converters-usb-hubs/usb-to-serial-converters/uport-1200-1400-1600-series#resources
[3] VCP driver: https://ftdichip.com/drivers/vcp-drivers/

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

can be provided for them. It is highly recommended to contact Bluefors support to replace them if the system needs to be upgraded.

2.2 Installing the software

The Bluefors Control Software installer is available as a smaller installer containing only the software itself, and as a bigger, bundled package with the National Instruments DAQmx driver framework. The framework is needed for controlling the Bluefors hardware, so it must be installed. However, if it is already installed, e.g., by previous version, the smaller package will be sufficient.

2.2.1 Installation procedure

The installation procedure is described in steps 1–5.

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

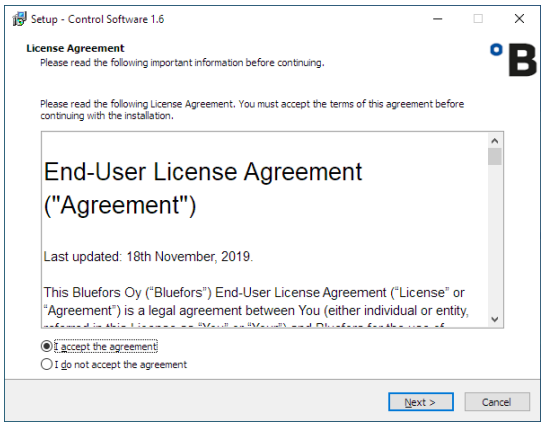


Figure 1: Software license agreement

STEP 1

Read the license agreement and choose to agree if you agree with the terms.

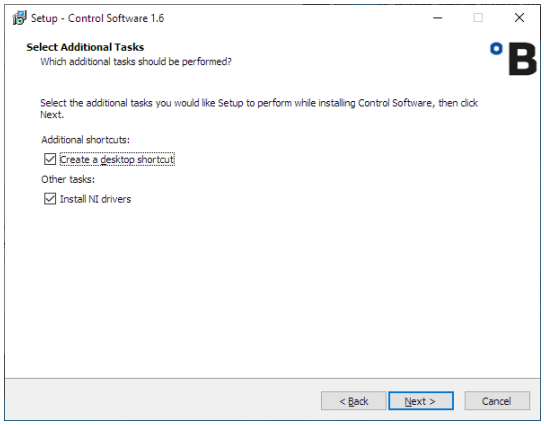


Figure 2: Select additional options

STEP 2

Select the additional options. The **Install NI drivers** option only exists with the bundled installation. The framework is required for Control Software to work correctly with Bluefors hardware.

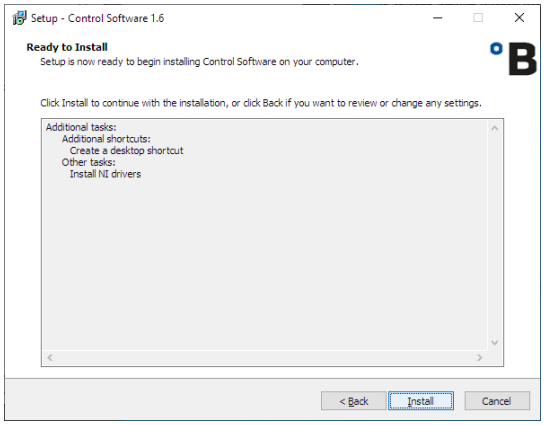


Figure 3: Verify actions to be taken

STEP 3

Verify the actions to be taken. Click **Install** to continue with the installation.

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

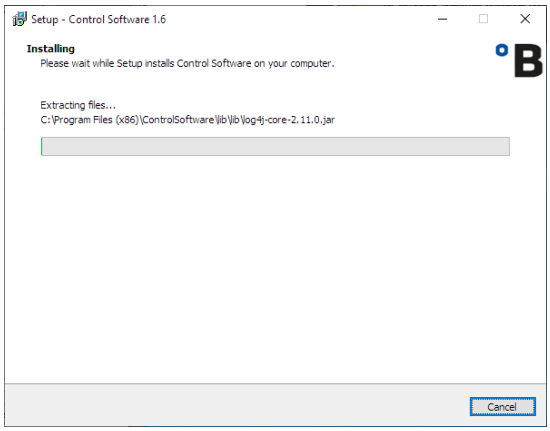


Figure 4: Wait for the software to be installed

STEP 4

Wait for the software to be installed.

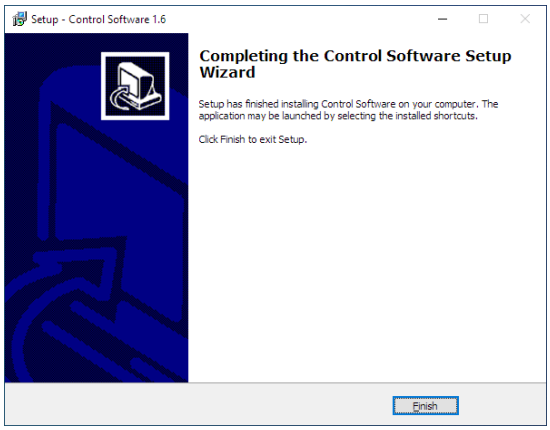


Figure 5: The installation is complete

STEP 5

The installation is complete. Click **Finish**.

2.2.2 National Instruments DAQmx framework installation procedure

If “Install NI drivers” is selected, after installing the software, a National Instruments DAQmx framework installer will be launched. This setup has multiple packages to choose from to install, but the default choices will be sufficient.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

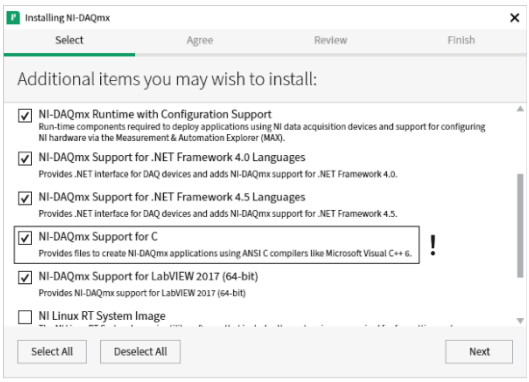


Figure 6: Choose additional items to install

STEP 1

Choose the items to be installed (the defaults will be sufficient). Make sure that **NI-DAQmx Support for C** is selected.

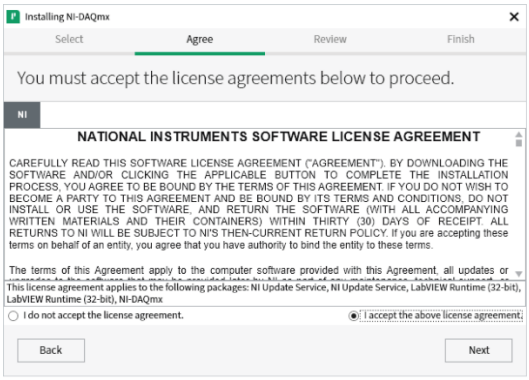


Figure 7: NI-DAQmx software license agreement

STEP 2

Read the license agreement, choose “I accept the above license agreement”, and click **Next**, if you agree with the terms.

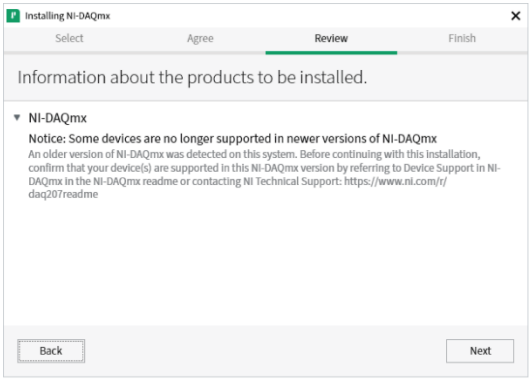


Figure 8: Device support warning

STEP 3

The installation may show a notice that this version may not support some older devices. If you have other National Instruments devices connected to this system, verify before installing the update that they work with this version of NI-DAQmx. The Bluefors hardware is known to work with this version.

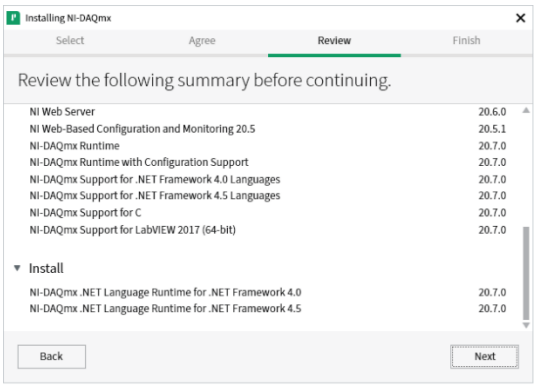


Figure 9: Review components to be installed

STEP 4

Verify that the list contains at least **NI-DAQmx** and **NI-DAQmx Support for C**.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

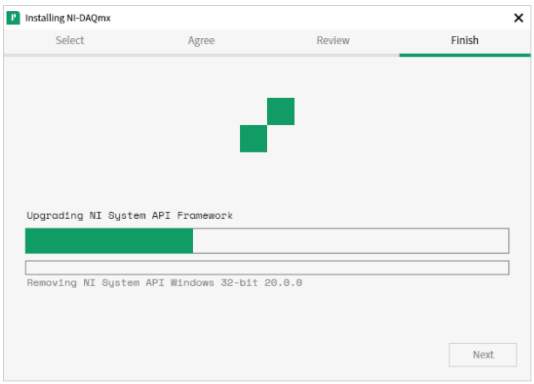


Figure 10: NI-DAQmx installation in progress

STEP 5

Wait for the software to be installed.

The NI-DAQmx installation may force the system to restart after the installation. Even though the restart may interrupt the final step of the Control Software installation, the installer has been designed in such a way that it is safe to restart the computer at this point. The software is already fully installed, and the last dialog is only for notifying about the successfully completed installation process. See Figure 10.

NOTE: The NI-DAQmx may force a restart before the Control Software installation is finished. The computer can be restarted as the software is already fully installed.

2.3 Starting up the software

When the installation has been completed, there should be two shortcuts visible: the **core server** and the **frontend**. The core server is the actual Control Software taking care of communicating with the devices and running the control scripts. The frontend is an interface for controlling it. There can be any number of user interfaces running but only one core server.

To run the software, start the core server. The core server has its own minimal user interface containing only the tray icon and status window. The minimal UI can be disabled with the –nogui parameter. Once the core is running, start the frontend to control it. The frontend can be freely started and shut down without influencing the functionality of the program.

2.3.1 Starting up the core

Start the core via the **Control Software Core** shortcut. See Figure 11.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

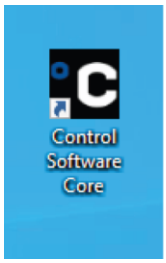


Figure 11: Shortcut to core server

Once started, the core will be located in a system tray area and its user interface can be accessed by right-clicking it. Use this menu to open the status window or shut down the server. See Figure 12.

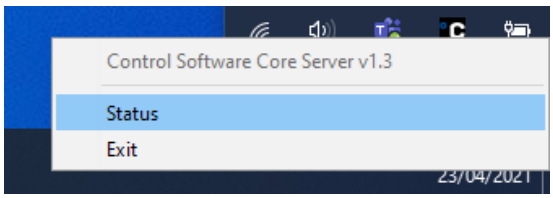


Figure 12: Core icon pop-up menu

The core status window shows the latest log messages produced by the core. The log events shown in the window are also saved to log files in the following directory: C:\Users\<username>\.bfcontrolsoftware\logs.

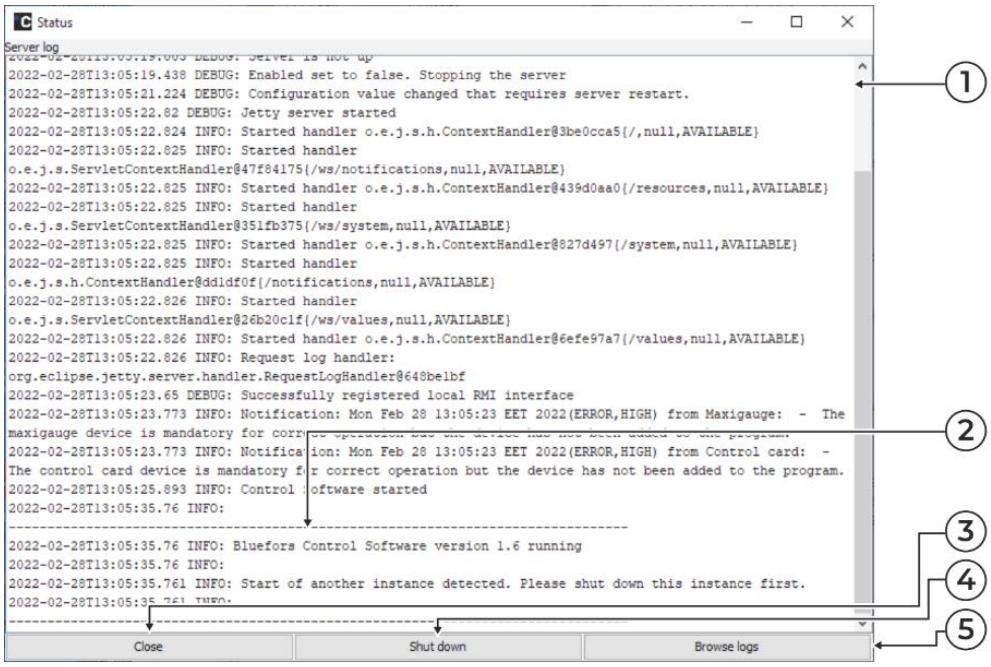
The status window contains three buttons (see Figure 13):

- **Close**: Closes the window
- **Shut down**: Shuts down the core server
- **Browse logs**: Opens a file browser in the program log directory.

Only one core server can be running at a time. If another instance is started, the status window of the core that is running will show a notification stating that an attempt to run a second instance of the server was detected.

Whenever updating the server, it is advisable to double-check that the old instance of the core server has been shut down first.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Status log
2	Message shown when the second core instance is being launched
3	Close the status window
4	Shut down the core server
5	Browse the program logs

Figure 13: Core user interface

2.3.2 Starting up the user interface

Start the frontend via the **Control Software Frontend** shortcut. See Figure 14.

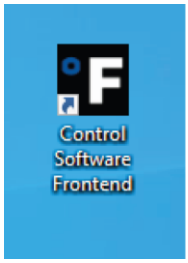


Figure 14: Frontend shortcut

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

When the frontend is started, it must first establish a connection to the core. See Figure 15.

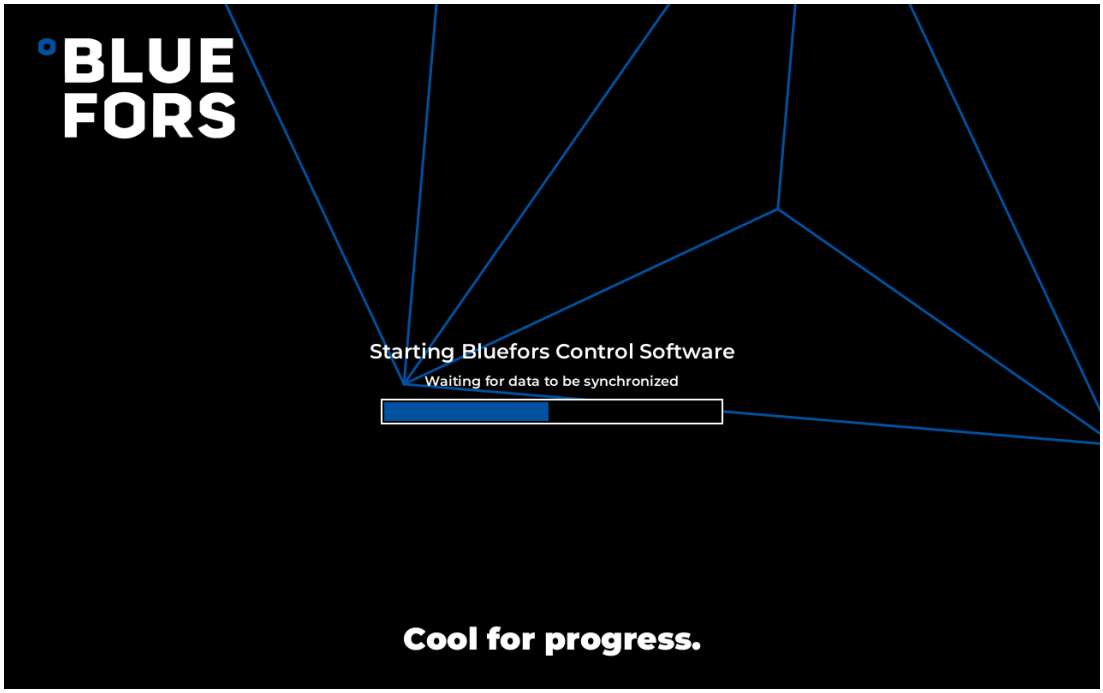


Figure 15: Frontend starting up and synchronizing the data with the core

If the core is not started, or the frontend fails to connect to it, the program will wait in this state until it detects that the correct version of core server is running. The frontend will also return to this state if the connection to the core is lost.

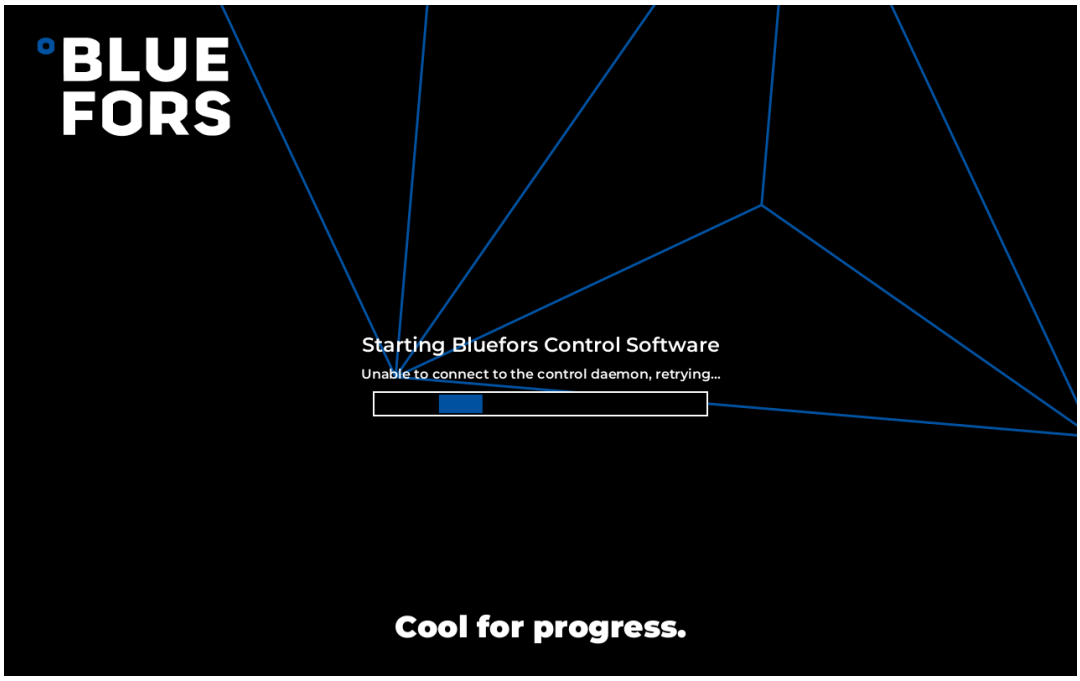


Figure 16: Frontend trying to establish connection when core is not running

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

NOTE: The frontend and core versions must always match. If a mismatch is detected, a notification will be shown to indicate the version mismatch and the frontend will terminate. See Figure 17. If such an event occurs, verify that the core has been restarted after the update.

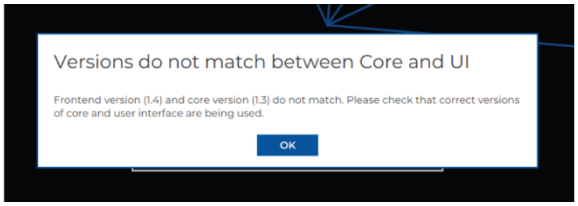


Figure 17: Frontend and core server version mismatch detected

3 First-time setup

When the program is started for the first time, it needs to be configured for use. For the most part, the program can be configured in the **Configuration** tab. Some systems are configured with the subsystem they are related to, such as temperature process control parameters.

NOTE 1: Leaving the system in an unconfigured or incorrectly configured state may cause the system to malfunction, but it is unlikely to cause serious consequences.

NOTE 2: It is always possible to start over by resetting the system to a completely unconfigured state. To reset the configuration, navigate to the **General** tab in the **Configuration** tab and click **Reset configuration to default**

3.1 Connecting the hardware

All the details related to the hardware connection can be set from the **Devices** tab. The hardware can be added automatically or manually. Automatic detection is preferred, which can be initiated by clicking **Autodetect**. See Figure 18.

If automatic detection fails to find one or more devices, they can be also added manually, if the target addresses, e.g., serial ports or IP-addresses, are known.

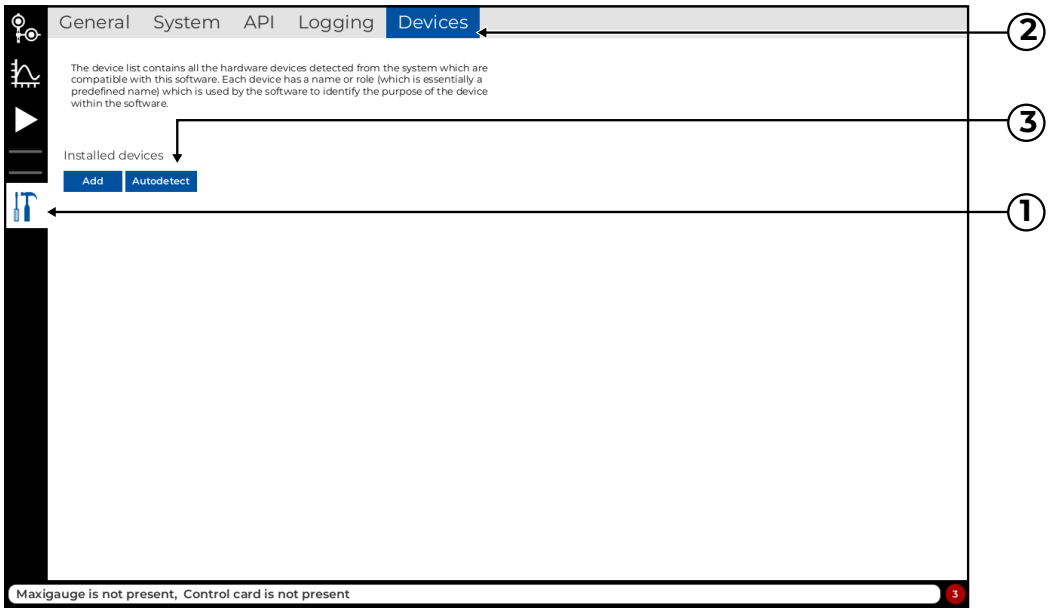
Many of the used devices use a serial port for communication. However, there is no unified way of identifying the devices connected to the serial ports. In standard Bluefors systems, the devices are usually connected to ports with known indices, i.e., to a specific port in the Moxa USB-RS232 hub, but there are exceptions. For this reason, the automatic detection of serial devices just tries to communicate with each type of device from each port until it finds a device that responds correctly.

All devices that Bluefors uses work well with this probing approach. However, there may be serial devices from other manufacturers that may show unexpected behavior when they are being probed. For this reason, if there are any other serial devices connected to the computer, including internal USB-RS232 converters, care

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

should be taken to verify that Bluefors Control Software does not interfere with them.

NOTE: Automatic detection probes the devices from all serial ports. Proceed with caution if there are devices from other manufacturers connected to the system.



Item	Description
1	Select **Configuration** tab
2	Select **Devices** tab
3	Click **Autodetect**

Figure 18: Automatic detection of hardware

In a standard system, the program should detect the Bluefors valve control card, Pfeiffer MaxiGauge, EL302P (4K Heater, only with an external 4K Heater source), Temperature Controller (Lake Shore 370/372 or Bluefors Temperature Controller), and FSE Control Unit (Fast Sample Exchange systems only). If these are found, the system can be fully operated.

The automatic detection may also find pumps, compressors, etc., but the system can be fully operated without them. They are controlled via the Bluefors control card. The serial connection is used only for retrieving statistical information, e.g., running hours, operating temperature, and error codes.

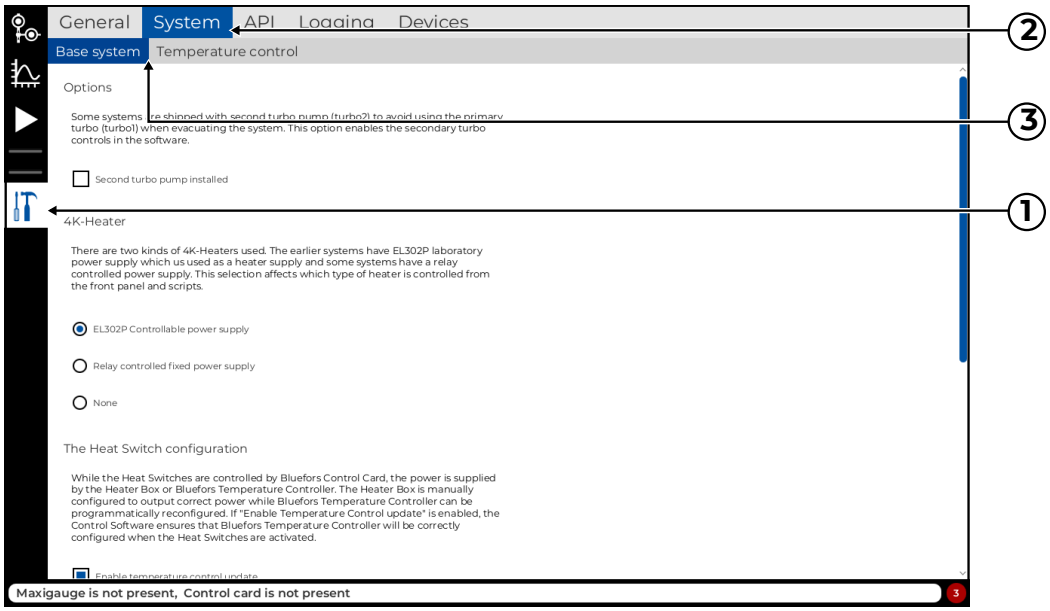
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Table 1: List of devices that may be detected

Device name	Essential	Purpose
Bluefors Control Card	Yes	Operates pumps and valves, reads the flowmeter
Pfeiffer MaxiGauge TPG256/TPG366	Yes	Reads pressures in the system
TTi EL302P	Option	4K Heater for speeding up the warm-up process
Bluefors FSE	Option	Controls the Fast Sample Exchange subsystem
Lake Shore LS370/372	Option	Reading and precise controlling of temperatures
Bluefors Temperature Controller	Option	Reading and precise controlling of temperatures
Cryomech CPA-series compressor	No	Shows status of Cryomech compressor
Cryomech CP2800/CP100 compressor	No	Shows status of Cryomech compressor
Pfeiffer Hi-Pace series pump	No	Shows status of Pfeiffer scroll and turbo pumps
Edwards nXDS series pump	No	Shows status of Edwards nXDS pump
Agilent Turbo-V 750/850	No	Shows status of Agilent turbo pump

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

3.2 Configuring the system parameters



Item	Description
1	Select **Configuration** tab
2	Select **System** tab
3	Select **Base system** or **Temperature control** tab

Figure 19: System configuration

Next, do the system-specific configuration in the **System** tab. The system tab contains base system configuration and temperature control configuration.

The base system configuration contains four settings.

Auxiliar turbo

If your system is supplied with a second turbo pump, shown in the bottom-left corner of the physical front panel and like in Figure 20, select the **Second turbo installed** option in the **Options** section (see Figure 19).

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

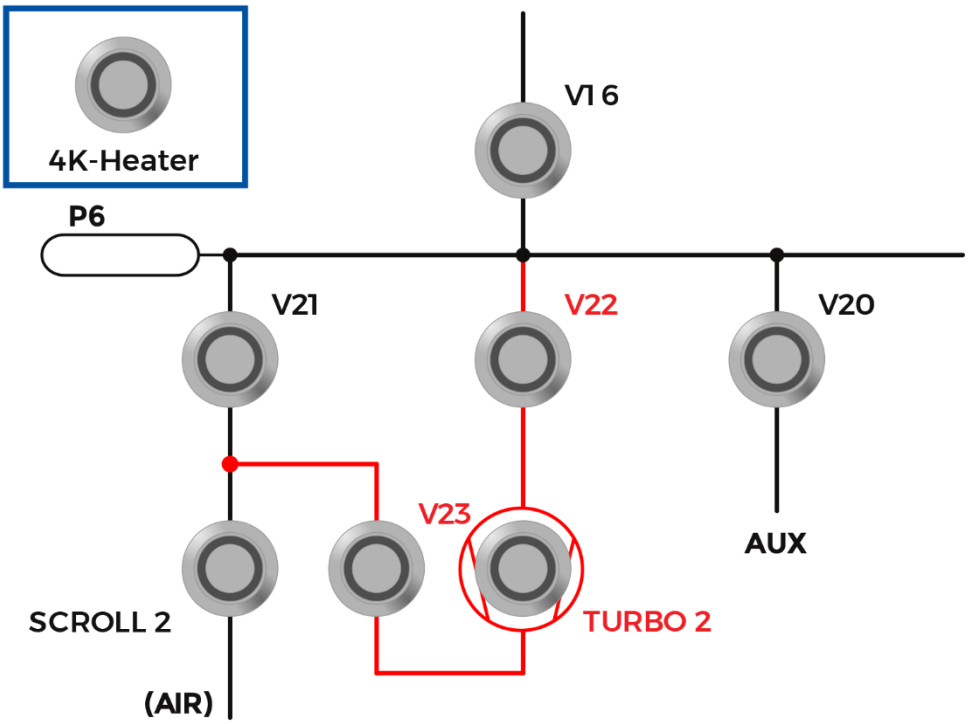


Figure 20: Optional turbo configuration in the front panel

If left unconfigured, the extra turbo is hidden from the user interface. However, it will work from the scripts. If the extra turbo is configured and does not exist, its controls just do nothing and the scripts using it just fail to evacuate the system.

NOTE: If your system has the extra turbo and it is left unconfigured, there is a possibility that the valves may have been left open and the pump running but it is not visible in the user interface.

3.2.1 4K Heater

The Bluefors systems are commonly supplied with a 4K Heater. The older systems have the TTi EL302P PSU used to drive it. The newer systems may also be supplied with a simpler relay-driven output which is fully controlled by the Bluefors valve control card.

If the EL302P is detected or known to exist, choose **EL302P Controllable power supply** (see

Item	Description
1	Select **Configuration** tab
2	Select **System** tab
3	Select **Base system** or **Temperature control** tab

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Figure 19). If a 4K Heater exists, but it is not EL302P, choose **Relay controlled fixed power supply**.

The following features control the settings:

- 4K Heater button in front panel
- *heater* script variables
- Heater safety timer

If no 4K Heater is present for heating up the system, choose **None** to remove it from the scripts and front panel. If this is selected even if the heater exists, the heater cannot be controlled from the front panel or by using heater variables in the script.

If EL302P is used as a heater, selecting it as a 4K Heater will link it to the features defined above. Note that the PSU itself can always be controlled from its dedicated heater user interface tab despite the choice in this configuration, but if it is not selected here, it will not be linked to the heater variables and will not use the safety timer. Turning this on from the front panel button, or from the *heater* variable, sets the PSU output on, voltage to 30, and current limit to 2 amperes.

If relay control is used for the heater, one of the control card channels is configured to act as a heater. If this option is selected on systems without this extension, the heater may appear to be constantly active without the ability to turn it off.

> NOTE: Some systems have two 4K Heaters. In that case, both may have their own dedicated manual control device views, but if used as 4K Heaters, they are both used simultaneously when operated from the scripts and front panel.

3.2.2 Heat Switch configuration

While the Heat Switches are controlled by the Bluefors Control Card, the power is supplied by the Heater Box, or Bluefors Temperature Controller. The Heater Box is manually configured to output correct power while Bluefors Temperature Controller can be programmatically reconfigured. If **Enable Temperature Control update** is enabled, the Control Software ensures that Bluefors Temperature Controller will be correctly configured when the Heat Switches are activated.

The **System type** options are used for determining correct power to use for the Heat Switches if updating is enabled.

3.2.3 Flowmeter calibration

The flowmeter used for showing the circulation flow rate must be calibrated to show the correct values. The configuration values can be found from the **Flowmeter** tab of the old ValveControl software, as shown in Figure 22.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Figure 21: EL302P PSU used as a 4K Heater

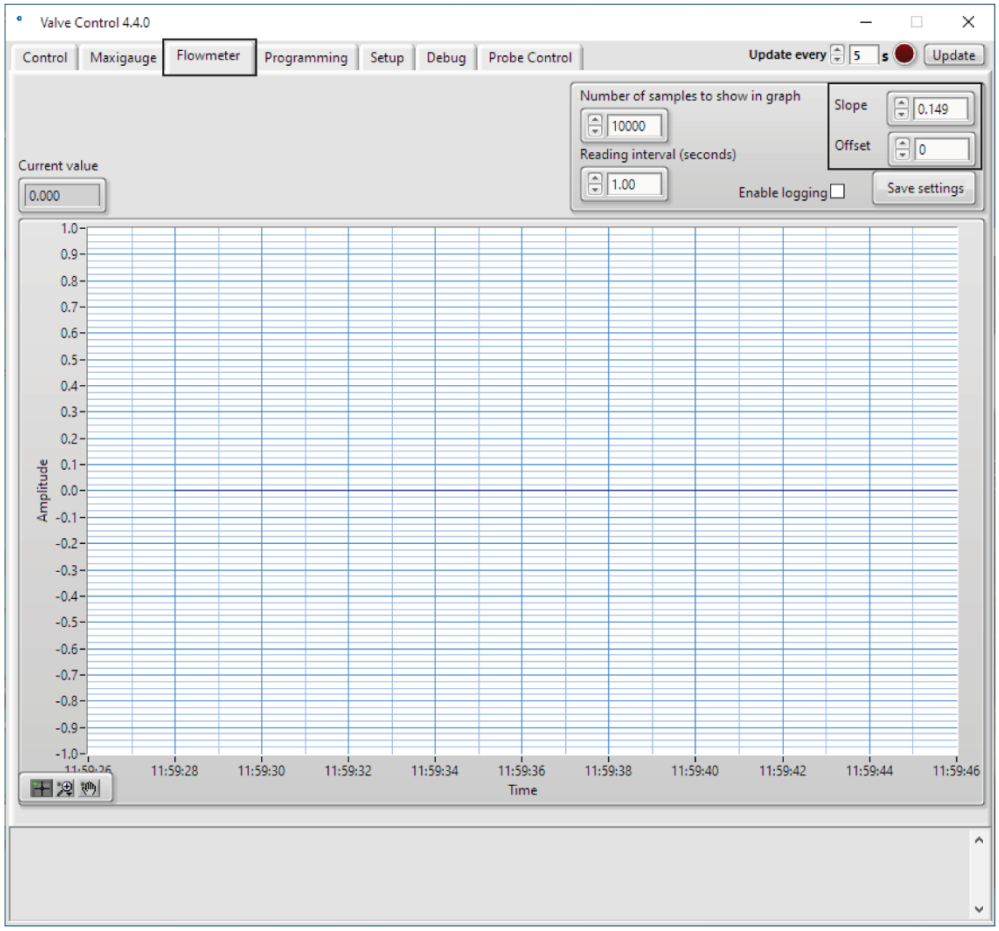


Figure 22: Calibration settings of the flowmeter in the ValveControl software

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

3.2.4 Preferred temperature device

The first setting in the **Temperature control** tab is **Preferred temperature device**. If the system has both Lake Shore and Bluefors temperature controllers, use this setting for choosing which one to use as a main temperature controller in the system.

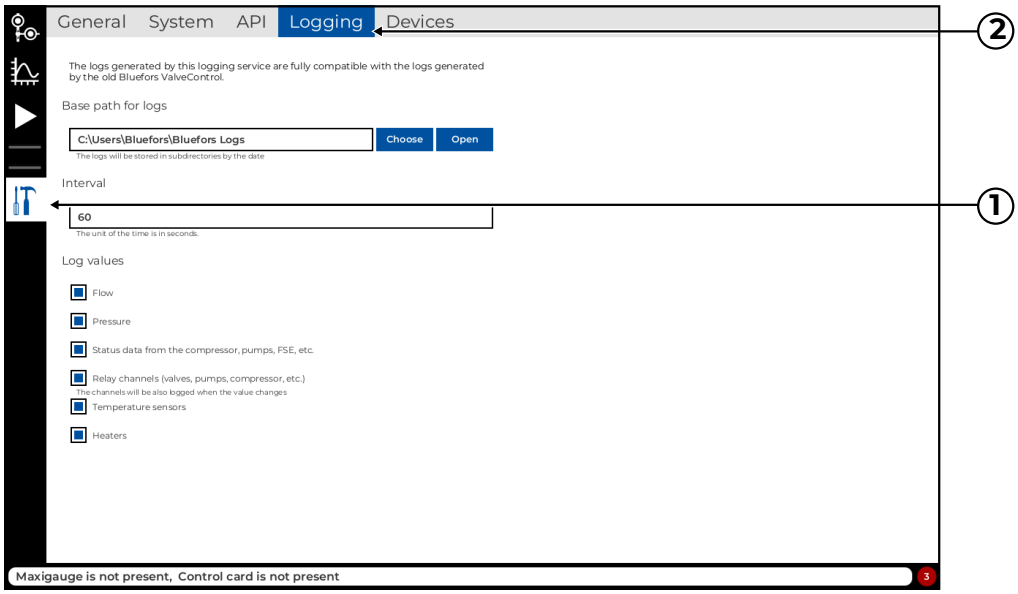
3.2.5 Sensors

Sensors section contains a list of the temperature sensors needed for controlling the system. To use the temperature controlling features and to show correct temperatures in the front panel, the mappings must be adjusted to match the sensors and the temperature controlling device channels. If the system has two Bluefors Temperature Controllers, the second controller channels are numbered from T9 to T16.

3.2.6 Heaters

To drive the heaters with correct power, the heater resistances must be known. Both heaters have two resistances: heater resistance and lead resistance. The heater resistance is needed for both. If the Lake Shore device is being used, the lead resistance must also be set for the Still Heater because the Lake Shore Still output is voltage-controlled output and, therefore, both resistances must be known for calculating the correct output power.

3.3 Logging



Item	Description
1	Select **Configuration** tab
2	Select **Logging** tab

Figure 23: Log configuration

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

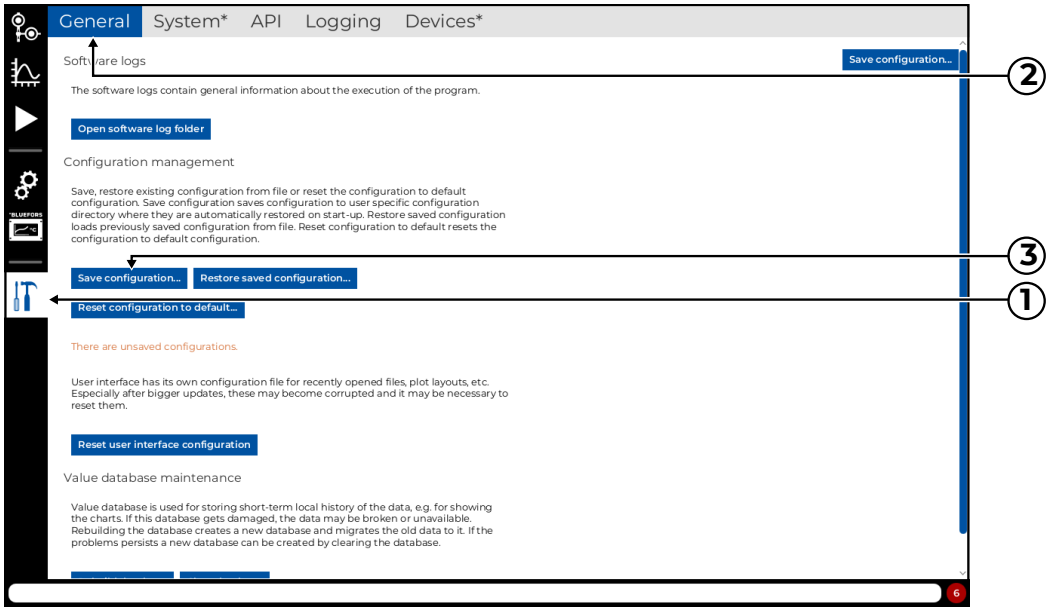
If logging is required, the program currently creates backward compatible logs. The log configuration can be done in the **Logging** tab. The following can be configured: choosing where to store logs, how often, and what values to log.

3.4 API

If remote controlling is needed, it can be configured in the **API** tab. This is disabled by default and can be left unconfigured if it is not used.

More detailed instructions on how to set up and use the API can be found in Section 6.

3.5 Saving configuration



Item	Description
1	Select **Configuration** tab
2	Select **General** tab
3	Click **Save configuration**

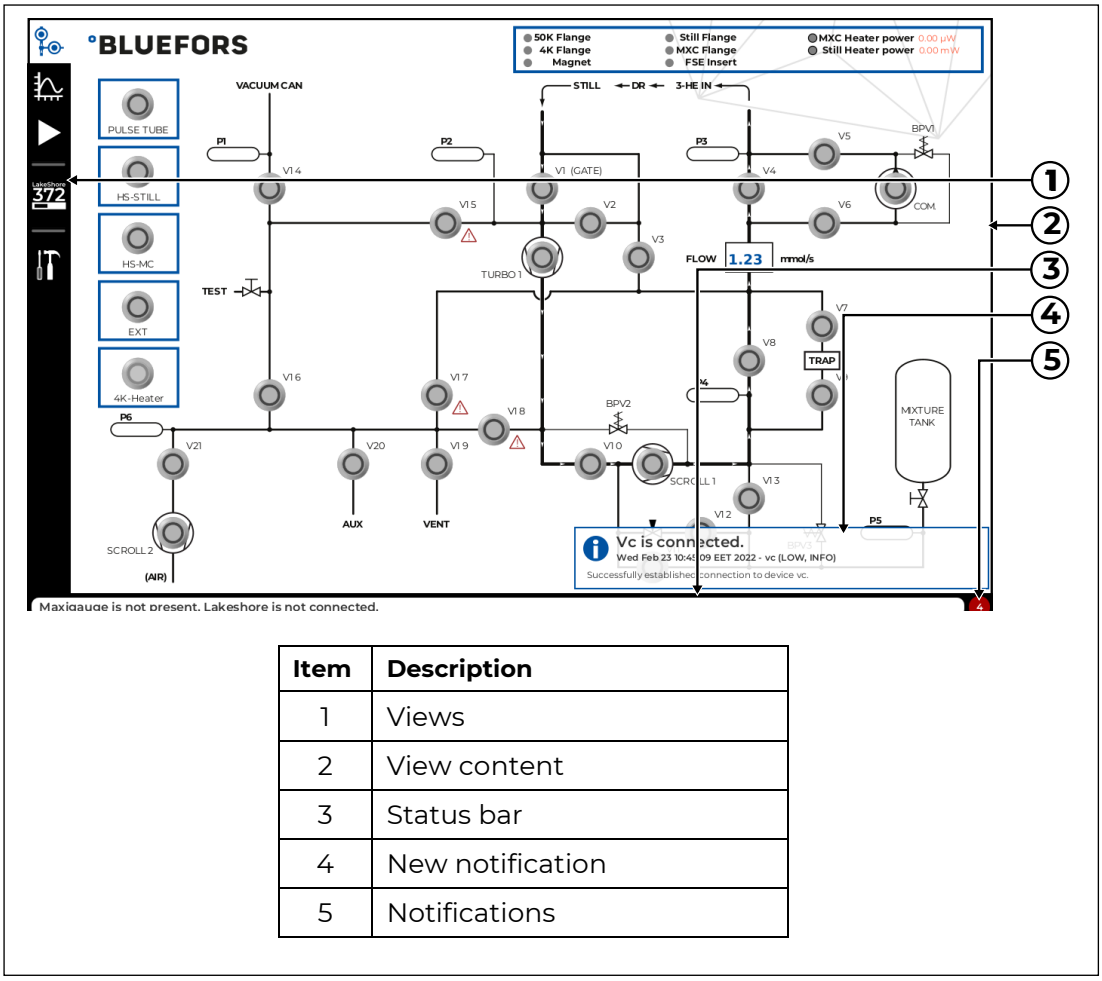
Figure 24: Saving the changed configuration in the General tab

Once you have configured what is needed, save the configuration to disk by clicking **Save configuration** in the **General** tab. The configuration will be automatically restored on restart or if explicitly invoked by clicking **Restore saved configuration**.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

4 User interface

The user interface has been designed to be simple and intuitive. It is composed of two principal elements: **views** and **notifications**. The main content is divided into distinct views. The selected view occupies a major part of the window, and you can choose the view that is shown from the icon list on the left. The bottom of the screen is reserved for notifications, which are asynchronous messages from the program to the user. On the bottom, there is a status line showing current persistent notifications, which represent ongoing conditions, and a button in the right corner to show all notifications. When a new message arrives, it is also shown briefly in the right corner of the content view.



Item	Description
1	Views
2	View content
3	Status bar
4	New notification
5	Notifications

Figure 25: User interface elements

The user interface views are layered. The functionality is divided into three groups: **Dilution Refrigerator control**, **Device views**, and **Maintenance**, as shown in Figure 26. Each view may be further divided into tabs and subtabs for specific tasks and features. For example, the Lake Shore resistance bridge device view is divided into temperature plot, resistance plot, status information, and configuration view. The configuration view is also further divided into distinct groups of settings, e.g., measurement input configuration or display configuration.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

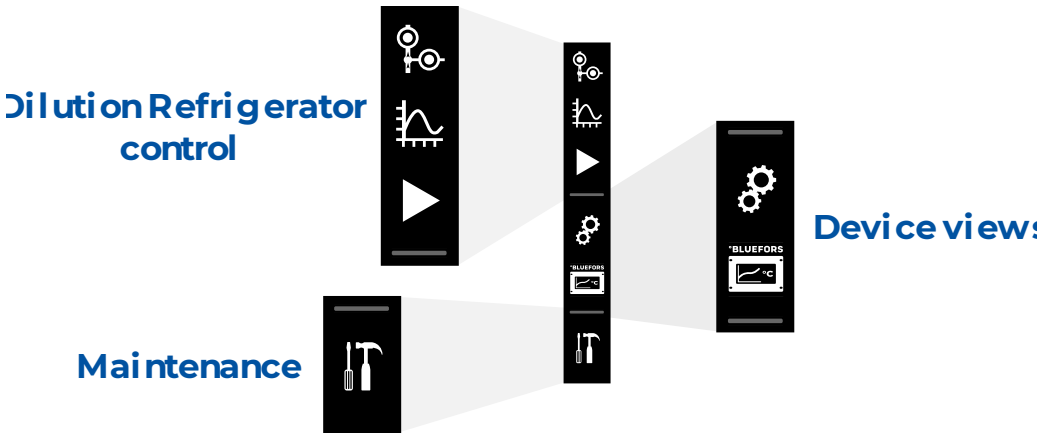
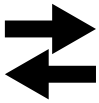


Figure 26: Functional control groups / views in Control Software

Table 2 explains the icons that are used in the functional control groups / views.

Table 2: Icons used in the functional control groups / views

Icon	Group / View	Description
	Dilution Refrigerator control	Front panel
	Dilution Refrigerator control	Plots
	Dilution Refrigerator control	Script engine
	Device views	Status view
	Device views	Bluefors Temperature Controller
	Device views	Lake Shore 372/370 AC Resistance Bridge
	Device views	Fast Sample Exchange
	Device views	4K Heater
	Maintenance	Program configuration

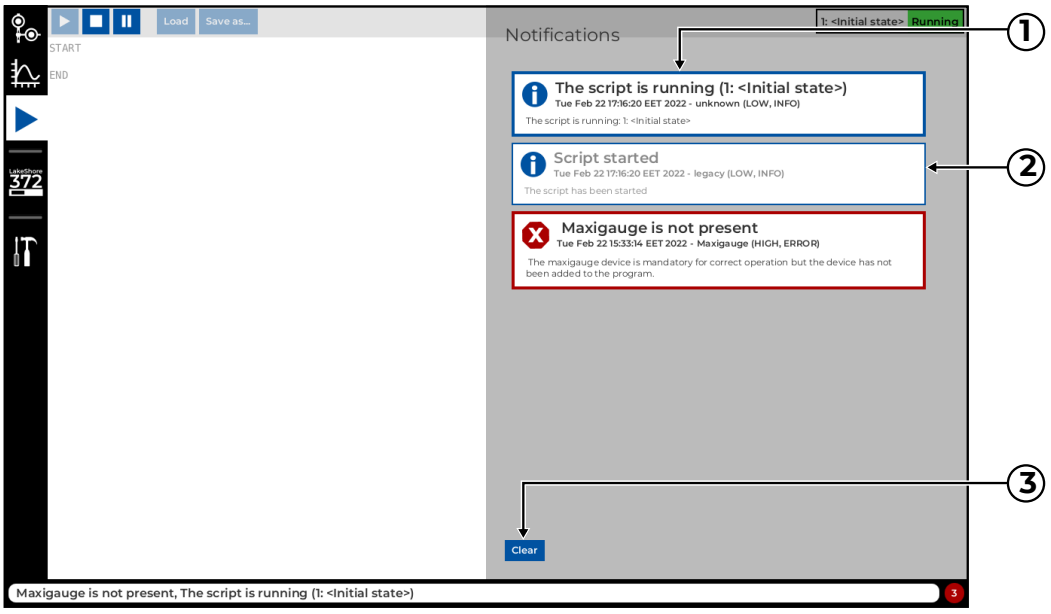
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Section 4.1 describes the notification system of the user interface. The different functional control groups are described in Sections 4.2, 4.3, and 4.4.

4.1 Notification

Notifications are asynchronous messages from any part of the program relayed to the user. Persistent notifications can be seen in the status line and the rest of them can be seen by opening the notification list. The notification list can be opened by clicking the notifications button in the lower right corner of the screen.

The button number indicates the number of notifications, the color of the button indicates the type of the most critical notification, and the border color of the button indicates the severity of the most severe notification on the list.



Item	Description
1	List of notifications
2	Notification
3	Clear the seen notifications

Figure 27: Notification user interface

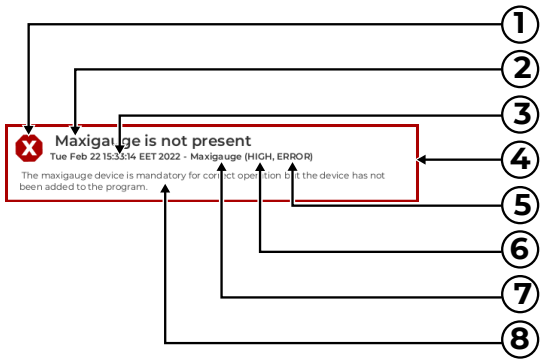
A notification contains many kinds of information describing the condition it presents.

- **Title**: The title of the message
- **Message**: The message content describing the condition
- **Time**: Time when the condition occurred or started
- **Source**: What part of the program generated the notifications
- **Type**: The notifications are divided into errors, warnings, and informational messages

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- **Severity**: Severity level can be low, medium, or high. It defines how serious the condition is.
- **Seen**: The notification can be seen or unseen. Once user has seen it in the notification list, it is seen.
- **Persistent**: Persistent notifications cannot be cleared until the condition has been resolved. Persistent notifications are also shown in status bar.

All these features are also shown in the notification box.



Item	Description
1	Type symbol
2	Title
3	Time
4	Border (persistence, severity)

Item	Description
5	Type text
6	Severity
7	Source
8	Message

Figure 28: Notification structure

The type of the notification appears as symbol and in written form.

Table 3: Notification symbols

Symbol	Notification	Description
	Error	Some part of the program has failed to work as expected, e.g., lost connection to hardware.
	Warning	Something, which is not an error but needs attention, has occurred, e.g., elevated coolant temperature which is still in range.
	Informational	Informational messages are “good to know” conditions, such as notification that the script has just been started.

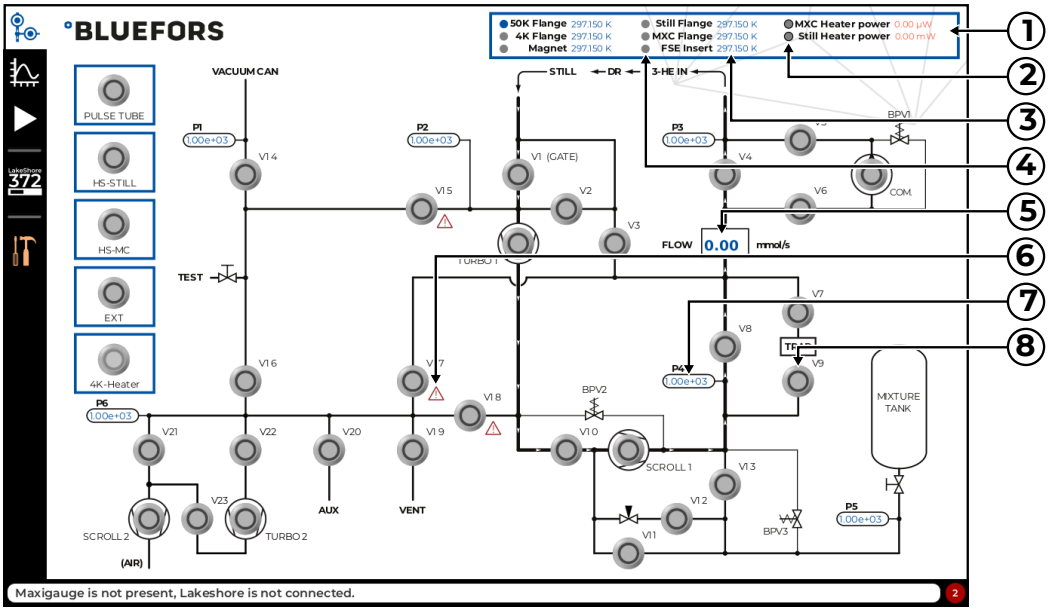
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

The border style defines the severity and persistence status. The color denotes the severity level (low=blue, medium=yellow, high=red). Persistent notifications have thicker border to indicate that they cannot be cleared until the condition becomes inactive. All seen notifications have a paler text color that the other text color.

4.2 Dilution Refrigerator control

Dilution Refrigerator control tabs are used for controlling the Bluefors system. These are all present in every Bluefors system and are essential for a correct functionality.

4.2.1 The front panel



Item	Description
1	Set heater powers
2	Heater enable buttons
3	Temperature readings
4	Scan indicator

Item	Description
5	Flowmeter
6	Potentially hazardous operation indicator
7	Pressure sensor
8	Valve / Pump / Heater button

Figure 29: The front panel view

The front panel view shows the valve diagram of the system. It can be used to get a quick overview of the system status as well as to manually operate most of the system controls.

Some parts of the diagram may not be present in every system. For example, auxiliary turbo is only visible if it is present in the system. This can be changed from the **Configuration** tab.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

4.2.1.1 Diagram elements

The front panel diagram has both active and passive elements. Passive elements are for a reference showing the system structure, such as tanks, pipes, and manually operated valves.

Buttons

The circular buttons in the front panel interface represent their counterparts in the physical front panel, and they operate similarly. The circular indicator shows whether the corresponding element, e.g., pump or valve, is activated, and its status can be changed by clicking the button.

Plain buttons in the diagram that have no decorations represent valves in the system. Each valve also has a name which is located next to the button, often diagonally on its top-left corner.

Buttons that are on top of a pump symbol (surrounded by a black ring), represent the pumps in the system.

On the left, there are also boxed buttons, which control other devices that are not part of the diagram, e.g., a Pulse Tube and Heat Switches that are connected to cryostat flanges not visible in the diagram.

Some buttons have a red exclamation mark inside a triangle (item 6 in Figure 29). This indicates that the operation requires additional confirmation before the operation is executed. This precaution has been added to operations that could have critical immediate consequences. The operations that need confirmation are opening valves V15, V17, and V18 or turning on the 4K Heater. An example of a confirmation dialog is shown in Figure 30.

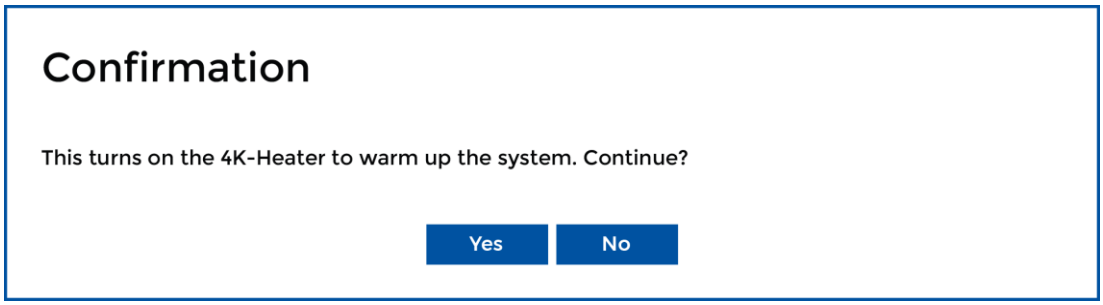


Figure 30: Verification dialog shown when turning on the 4K Heater

For example, opening any of the valves V15, V17, or V18, which connects circulation and service manifold, could cause loss of mixture if the system is running cold and any of the valves on service manifold side would be temporarily open to air.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

CAUTION

You are responsible for the correct performance of any operation. Only certain critical errors have been protected from accidental clicks. Always read and follow the instructions, safety information, and warnings stated in the system user manual. Incorrect action or operation may cause critical malfunction or system breakdown.

Sensor readings

Sensor readings are actively updated values that shows the status in certain parts of the system, e.g., pressure sensors attached to various parts of the system.

Pressure sensors are located in rounded boxes in the diagram and are labelled from P1 to P6. They show the system pressures in millibars in exponential notation.

Flowmeter shows the mixture circulation flow in mmol/s.

Temperature sensor readings shows the temperature within the system. The temperature control devices commonly have only one channel that can be measured at a time. The circular indicator next to the sensor name indicates whether the corresponding channel is currently being scanned.

Mixing Chamber and Still Heater outputs have also similar looking indicator and reading. However, the heater values are settable. The heater can be switched on and off by clicking the circular indicator and heater power can be set by clicking the power value.

4.2.2 Plots

The plot view has multiple charts showing different variables in the program, e.g., pressures, flow, and temperatures. See Figure 31.

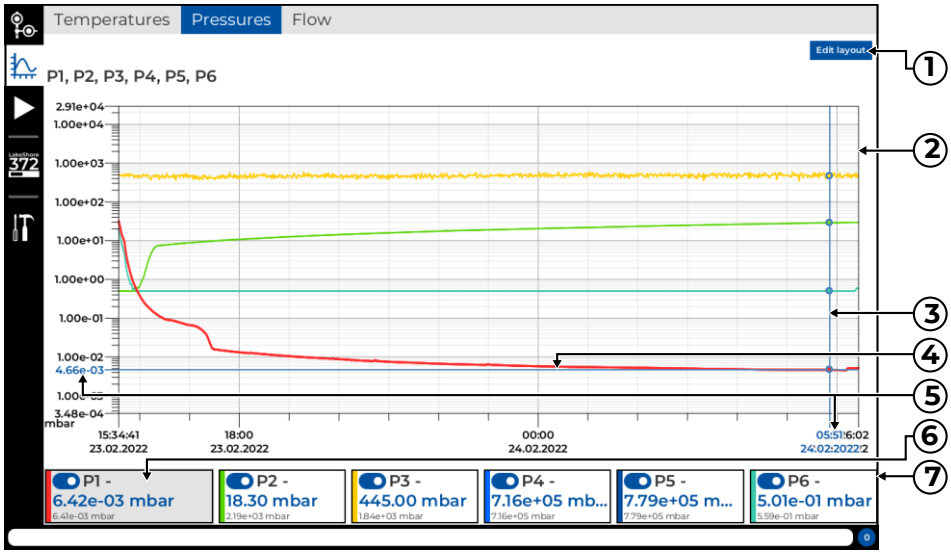
This view shows one or more charts at the top and a list of all plots at the bottom of the window. The plot list contains the plot color, name, current value, value pointed by the cursor, and, optionally, status information or additional controls, such as the enable switch for pressure sensors. The plot information box may also be hidden if the corresponding sensor is not present in the system. In edit layout mode, all the plot information boxes will always be visible to allow you to add or remove them from charts. By default, the plot will be scaled automatically. If zoomed or moved, the automatic scaling and positioning will switch off for affected axes, but it can be resumed by clicking the right mouse button.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

The chart view actions are as follows:

- Whenever a plot is being pointed at using a mouse cursor (either the box or the plotline itself), the corresponding line and box will be highlighted.
- When moving the mouse cursor over the chart, a cursor will be shown on the chart and its position is shown in bottom-left corner. All the plot information boxes also show the corresponding value at the time pointed by the cursor.
- You can zoom in to and out of the plot by using the mouse wheel on the chart. Performing these operations over an axis label will zoom in and out in relation to one axis only.
- The position shown in the chart can be moved by holding the left mouse button down and dragging around. Similar to zooming, doing this while pointing at an axis marker will perform this on one axis only.
- Right-clicking on the chart or either of the axes resumes the automatic scaling and positioning to one or both axes.

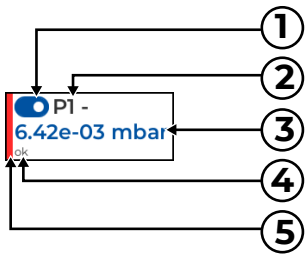


Item	Description
1	Layout editor
2	Chart space
3	Cursor
4	Highlighted plot

Item	Description
5	Cursor position
6	Highlighted plot information
7	Plot list

Figure 31: Plot view structure

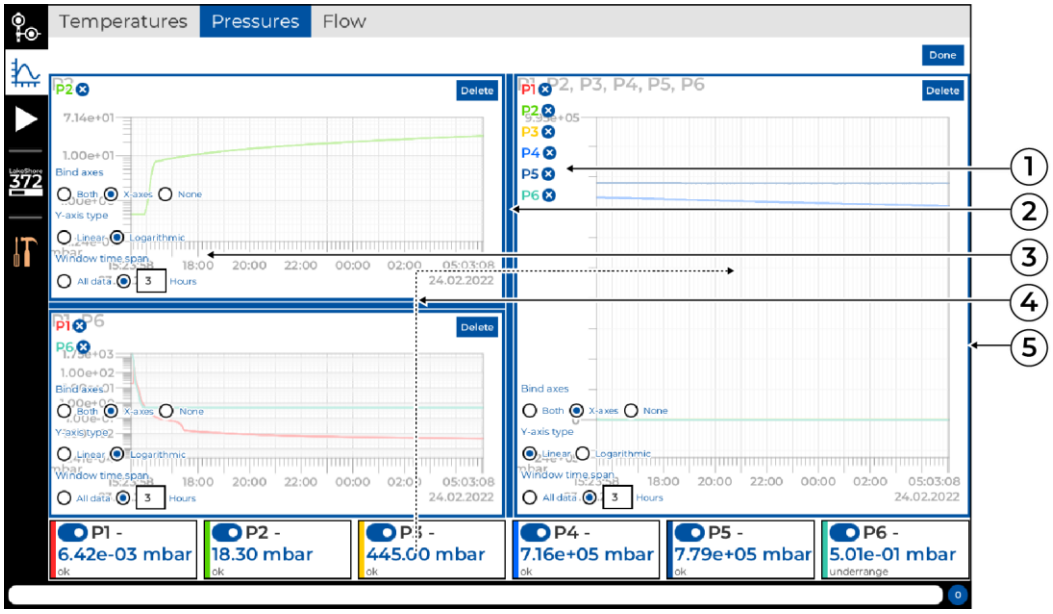
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Optional enable switch
2	Name
3	Current value
4	Status / Value at cursor
5	Plot color

Figure 32: Plot information

4.2.2.1 Editing the layout



Item	Description
1	Chart plots
2	Chart separator line
3	Chart settings

Item	Description
4	Drag and drop
5	Drop target

Figure 33: Plot view editor

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

You can customize the plot view to meet your needs.

- Activate the edit mode: Click **Edit layout** at the top-right corner of the window.
- Resume the normal functionality: Click the **Done** button at the top-right corner of the window. The configuration will be automatically saved, and they will be restored when the frontend is started.
- Reset all user interface layouts and other configuration to defaults: Click **Reset UI configuration** in the **Configuration** tab.

When the edit mode is active, the plot view controls are different. Each plot view contains controls that can be used to adjust its functionality:

- **Bind axes**: This can be used for binding together the fitting of charts by either the X-axis only, or by both axes. This can be used for maintaining the same location in multiple charts while scrolling. By default, only the X-axis is bound in every chart so that the time span is the same in every window when scrolled.
- **Y-axis type**: Here, you can select whether the Y-axis should be linear or logarithmic.
- **Window time span**: This defines how many hours of data you want to show in the chart, starting from the latest measured value.

It is also possible to add, remove, copy, and move plots from other views or plot information boxes.

> NOTE: When edit layout mode is activated, all information boxes become visible to make it possible to add and remove data that may be currently unavailable, e.g., temporarily disabled temperature sensor channels.

The plots are added and moved simply by dragging and dropping them from a plot information box, or another chart, to the chart area. Depending on the exact location, the plot will be either added to the chart under the drop location or the chart will be split, and a new chart will be created. The action taken will be shown as a wireframe representing the action that will be taken. Different drop zones can be seen in the following figure.

- When dropping on top of another chart, the plot is copied (or moved from another plot if shift is being held down) to the plot.
- When dropping between the charts, a new chart will be created in between, dividing the space equally between all plots in the chain.
- When dropped at the edge of another chart, the chart will be split so that the new chart will get half of the space of the old chart.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Item	Description
1	Add plot to chart
2	Add chart between charts
3	Split the lower chart half

Figure 34: Actions that will be taken when the plot is dropped to different locations in the chart space

It is also possible to adjust the sizes of the charts by dragging the spacers between the charts. Charts can be also deleted either by clicking **Delete** in the chart or removing the last plot from the chart. If the last chart will be deleted, a placeholder will be shown instead.

4.2.3 Script engine

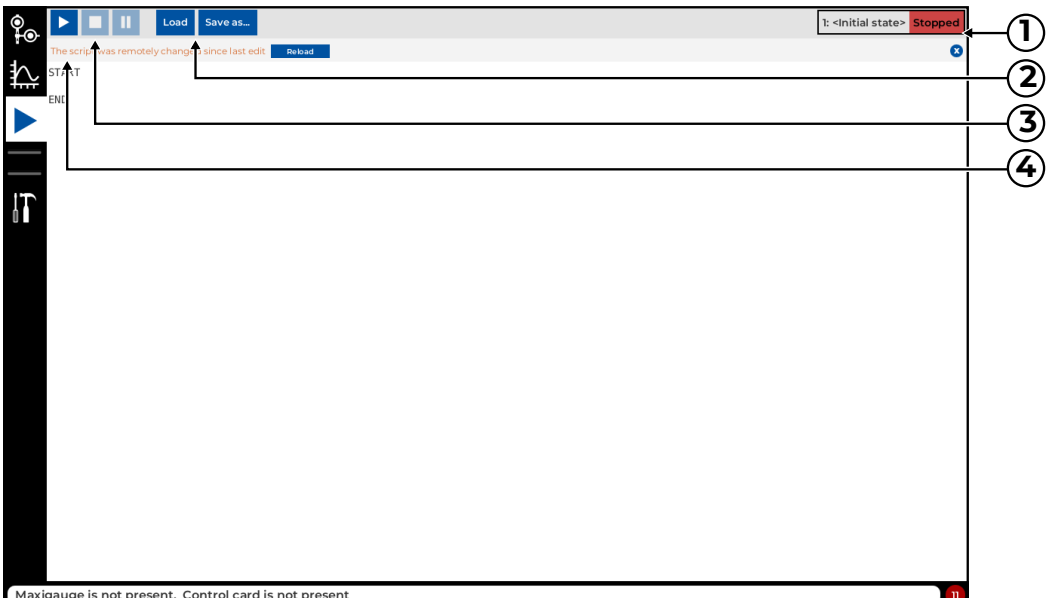
The **Script engine** tab can be used for performing automated tasks for the system, for example, cooling down the system. The program is shipped with a set of built-in scripts that can be used for automating normal cooldown and warm-up related tasks for standard systems.

See Appendix II for the language definition.

4.2.3.1 Controls

The script view has controls for controlling the script status and loading and saving scripts.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Script engine status
2	Loading and saving
3	Script execution control
4	Notification area (normally hidden)

Figure 35: Script view controls

The script view has four groups of controls.

Execution control

Script execution controls are used for starting, pausing, and stopping the scripts. Only one script can be running at a time. When a script is running, it cannot be changed.

However, a script that is running can be paused or stopped.

> NOTE: Pausing a script just halts the execution temporarily and does not have the functionality to ensure that the paused script can be correctly resumed.

Script engine status

The script engine status box shows the current status of the script engine. It has two parts: the status message and execution status. The message shows general information within the current state, if any, e.g., waiting time.

The engine status is shown on a colored background indicating the status criticality:

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

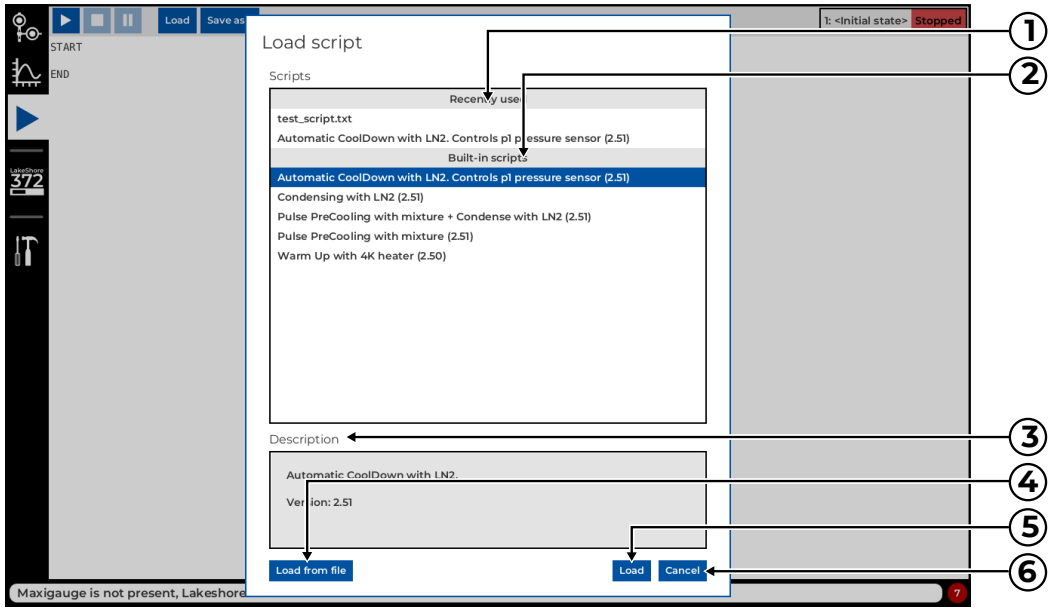
- **Stopped (Red)**: The script engine is not currently executing anything. In this state, a new script execution can be started.
- **Error (Red)**: The script execution has ended in an error. Functionally, this is similar to the stopped state, so script execution can be (re)started.
- **Paused (Yellow)**: The script execution is temporarily paused by the user. The script execution can be either resumed or stopped
- **Running (Green)**: The script is currently executing. The script can be either paused or stopped in this state.

File operations

Scripts can be saved and loaded. Saving a script opens a standard file saving dialog for saving the script to a text file.

Load script can be used for loading either one of the built-in standard scripts for common operations, or for loading one from a file.

4.2.3.2 Loading a script



Item	Description
1	List of recently loaded scripts
2	List of built-in scripts
3	Additional information about selected script

Item	Description
4	Load script from a file
5	Load the selected script
6	Continue without loading a script

Figure 36: Load script dialog

When the **Load** button is clicked, a dialog is opened for choosing a script to load. The dialog contains list of built-in scripts for common operations and scripts that have been recently opened. Optionally, a script can be loaded from a file via this

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

dialog by clicking **Load from file**, which will open a standard open file dialog for choosing a script file to load.

4.2.3.3 Notifications

The script engine may also show notifications in certain conditions that need your attention.

Currently, the only notable notification is *The script was remotely changed since last edit*, which is shown when the script has been loaded or edited in the frontend since the last run and at the same time the script has been changed remotely, e.g., via an API or from another UI instance. In this condition, you may either reload the updated script from the core by clicking **Reload** or just close the notification and continue editing the current copy.

4.3 Device views

Device views are dedicated control views for auxiliary devices that the system may have, e.g., the Bluefors Fast Sample Exchange Unit or Bluefors Temperature Controller. Some functionality may overlap with the control functionality of the dilution refrigerator measurement system, because in addition of having manual control interface, some devices are also used as part of the dilution refrigerator control logic.

CAUTION

Always read and follow the instruction, safety information, and warnings stated in the system user manual. Incorrect action or operation may cause critical malfunction or system breakdown.

As an example, Bluefors Temperature Controller is used as a part of the cooldown process via temperature control view, but also has its own dedicated device view. While the temperature control view provides unified controls over all temperature control devices, it exposes only small subset of its features. For this reason, full control over the device will be provided also through the dedicated device view, which allows the use of the full range of the device features. However, if you use the dedicated device controls during the cooldown process, make sure that you are fully aware of the consequences the operations may have, such as interfering the standard dilution refrigerator control process when it is running. If the device configuration is changed, the software will reconfigure it back when it needs it for the dilution refrigerator control.

NOTE: Some features are overlapping as some devices are used as part of the dilution refrigerator control. Do not use the dedicated device controls during

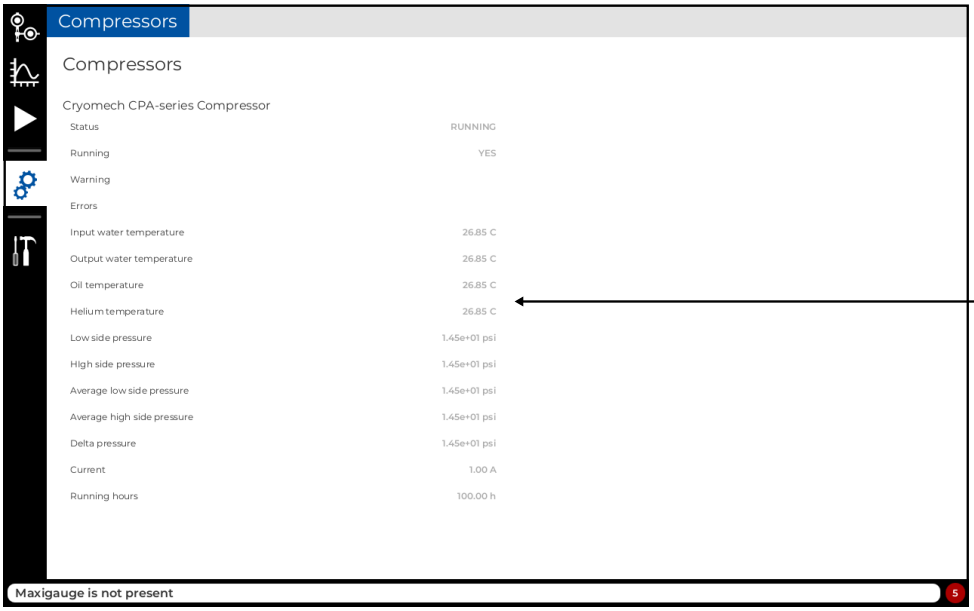
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

the cooldown process if you are not fully aware of the consequences the operations may have.

4.3.1 Status view

The **Status** view shows statistics from various devices that do not have dedicated views of their own, e.g., pumps that provides only statistical information.

This view also contains a list of errors and warnings that may be present in the system.



Item	Description
1	Status information from devices

Figure 37: Status and error tabs in status view

4.3.1.1 Compressors / Pumps

The **Compressors** and **Pumps** tabs show general status information from pumps and compressors, such as running hours and temperatures. What is being shown depends on what information the device can provide.

4.3.2 Bluefors Temperature Controller

The Bluefors Temperature Controller view shows all details related to Bluefors Temperature Controllers. The view is split into subviews by function:

- Resistance plots
- Temperature plots

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- Status
- Device configuration.

4.3.2.1 Resistance and temperature plots

The resistance and temperature plots show the measured values from the device. For more detailed information about chart functionality in general, refer to Section 4.2.2.

4.3.2.2 Status

The **Status** tab shows controls for the scanner and buttons to read and write all values.

There are three settings:

- **Update from device**: Reads all values (measurements and settings) with one button. This is a long processing operation.
- **Update to device**: Write all edited values to the device with one button.
- **Active channel**: Shows the currently scanned channel. This is a read-only value.

4.3.2.3 Configuration

The **Configuration** tab contains most of the Lake Shore bridge configuration. When the controller is connected, the configuration is initially synchronized with the device. Thereafter, device values are listened and updated whenever the device sends an update. When the values are edited, they are no longer updated by events sent by the device. Rather, they stay in an edited condition until you update the values to the device or reads new values manually from the device.

For this purpose, every configuration set has **Update from device** and **Write to device** buttons.

Channels

This tab is used to configure channels. Each channel has three different excitation modes.

The channels may have the following configuration:

- **Active**: On / Off value showing if the channel is on
- **Name**: Custom name for ease of identification for the channel
- **Coupled Heater**: Heater used by the channel
- **Excitation Mode**: The excitation mode can have following values:
    - **Current excitation**
    - **VMAX**
    - **CMN**
- **Excitation Current Range**: Shown if the Excitation Mode is “Current excitation.” Select one of the current excitation amplitudes from the drop-down menu. See the table in “Resistance measurement ranges” in the

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Bluefors Temperature Controller System Description for the current excitation ranges.

- **Excitation CMN Range**: Shown if the Excitation Mode is “CMN.” Choose from two current excitations, 50 μA or 150 μA, both of which have the same excitation frequency of 64 Hz.
- **Excitation VMAX Range**: Shown if the Excitation Mode is “VMX.” This is the highest possible voltage over sample without exceeding a given value. It is used for selecting the current excitation amplitude.
- **Custom time constants in use**: The configuration to override the default timers’ values
- **Wait Time (s)**: Amount of time waited after switching channel before doing the actual measurement in the channel
- **Measurement Time (s)**: Amount of time used to do an actual measurement of a single value in the channel
- **Calibration Curve Number**: Calibration curve used by this channel.

Heaters

The heaters may have following configuration:

- **Active**: Active state of the heater
- **Name**: Custom name for ease of identification for the heater
- **Mode**: Heater mode:
  - **Manual mode**: In manual mode the heater power is set to the desired current (amperes) or power (watts) value.
  - **PID mode**: The closed-loop mode is implemented as a Proportional Integral Derivative (PID) controller.
- **Resistance (Ω)**: Heater resistance in ohms
- **Power (W)**: Applied manual power in watts
- **Maximum Power (W)**: Hard safety limit for power in watts
- **P**: Proportional value for PID controller
- **I**: Integral value for PID controller
- **D**: Derivative value for PID controller
- **Setpoint (K)**: Setpoint for control algorithm
- **Relay Mode**: Sets which mode of external loop (shorted or open) enables the heater
  - **Shorted**: Set loop to be shorted
  - **Open**: Set loop to be open
- **Relay Status**: Real status of external relay (shorted or open).

Curves

Curves is a collection of 100 curves that can be used by channels. Like all other tabs, this has update and write buttons. In addition, this tab has Load from file and Clear curve buttons. The Clear curve button writes the current curve off and if write to device is called after, it will also write it off from the connected device. The Load from file button is for supporting curve files that were used for Lake Shore.

The curves may have the following configuration:

- **Name**: Name of the curve
- **Sensor Model**: Sensor model

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- **Curve**: Curve shown as a plot.

System

The **System** tab contains read-only values that describe currently connected Bluefors Temperature Controller: versions, labels, etc.

The system tab may have the following configuration:

- **API Version**: Public API version that the Bluefors Temperature Controller provides
- **Sensor Model**:
- **Software Version**: Software version
- **Device ID**: Device ID
- **Device Firmware**: Device firmware version
- **Type**: System type
- **Serial**: Device serial number
- **Label**: System label.

4.3.3 Lake Shore 372/370 AC Resistance Bridge

The Lake Shore 372/370 view shows the following data and configuration for the Lake Shore resistance bridges:

- Resistance plots
- Temperature plots
- General
- Device configuration.

4.3.3.1 Resistance and temperature plots

The resistance and temperature plots show the measured values from the device. For more detailed information about chart functionality in general, refer to Section 4.2.2.

4.3.3.2 General

The **General** tab shows the scanner status and control for updating all configuration to or from the bridge at once.

Scanner configuration:

- **Autoscan on**: Controls whether autoscan is enabled
- **Scanner channel**: Shows and controls the currently scanned channel. It is possible to change the scanned channel even when autoscan is enabled.

4.3.3.3 Configuration

The **Configuration** tab contains most of the Lake Shore bridge configuration. Unlike many other values in the program, the Lake Shore bridge configuration is

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

not automatically synchronized to the device when changed. When the bridge is connected, the configuration is initially synchronized with the bridge. After changing the configuration, the program must be explicitly invoked to update the changed configuration to the device. It is also possible to reload the configuration from the bridge in case it was changed from the device itself after the initial synchronization sequence.

For this purpose, every configuration set has **Update from bridge** and **Write to bridge** buttons. The **General** tab also contains buttons for reading or writing all configuration at once.

> NOTE: The actual commands that read and write data to and from the device are done in batches. A single command is used for accessing multiple settings. For this reason, this also updates the data in batches to the device. Many of the settings must be updated this way to make sure the written values are in a consistent state, e.g., updating PID parameters one by one when the heater is on could cause the control loop to be temporarily in an inconsistent state and therefore potentially interfere with the control process.

This manual contains very brief explanations about what the configuration is. More detailed and accurate descriptions can be found in the Lake Shore Model 372 Resistance Bridge and Temperature Controller user manual.

Inputs

This tab contains configurations for all 16 + 1 inputs.

This device has only two inputs. The control input (A, 372 only) is constantly active and inputs from 1 to 16 are temporally multiplexed into a single measurement channel by a separate scanner unit. Therefore, only one channel is read at a time.

Despite multiplexing, each input has individual settings that are applied by the device when the channel is selected, i.e., being scanned.

The inputs have the following configuration (some configuration may be hidden if they are not applicable):

- **Enabled**: Shows whether this channel is in use. If disabled, the channel will be excluded from the autoscan loop and hidden from the plots, and the readings in the bridge display will show ”DISABLED.”
- **Name**: The name of the channel that can be shown in the bridge display and in these program plots
- **Autorange**: If enabled, the bridge automatically tries to determine the most suitable resistance range for the measurement.
- **Range**: The resistance range the bridge uses
- **Shunted**: Shunts the output of the channel
- **Excitation mode**: The bridge uses the current excitation for the resistance measurements but for measurement inputs can simulate voltage excitation.
- **Excitation range**: Determines the maximum excitation current or voltage the bridge uses

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- **Dwell time**: The time this input is being measured for when autoscanned (only measurement input)
- **Pause time**: When the scanner switches to this channel, this is how long the scanner should wait until the measurement is settled.
- **Temperature curve**: The curve used for calculating the temperature from resistance in this channel
- **Enable filter**: Enables filtering for the channel reading
- **Settle time**: Time in seconds for the filter settle time
- **Filter window**: Window of the filter as a percentage
- **Enable Alarm**: Enables the alarm for this channel
- **Alarm low setpoint**: Activate the alarm if the value goes below this value.
- **Alarm high setpoint**: Activate the alarm if the value goes above this value.
- **Latching alarm**: Keeps the alarm on after the condition has been restored
- **Audible alarm**: Enable an audible alarm.
- **Visual alarm**: Enable a visual alarm (alarm LED).

Outputs

This tab controls the configuration for the Lake Shore outputs.

The Lake Shore has three outputs: **sample**, **warm-up**, and **still** outputs. They all have different characteristics depending on the type and mode that are being used. Only configuration relevant to the mode being used will be shown in the user interface.

Only the sample heater, which is also called the mixing chamber heater in Bluefors systems, is the only current-driven heater. The others have voltage sources. For this reason, the unified temperature control module needs to know the lead and sensor resistances to correctly calculate the output power. For the same reason, whenever the still heater value is set from the temperature control view, the set output value does not directly match any of the settings shown in here as the actual output voltage is calculated within the software.

The outputs may have following configuration:

- **Mode**: The output mode which can be either:
  - **Off**: The output is inactive
  - **Monitor out**: (Still) Configures the still heater to output one of the monitor output values
  - **Manual**: The output is controlled by the user-supplied manual value
  - **Zone**: (Sample, Warm-up) Configure the heater based on zones
  - **Still**: (Still) Configure the output as the still heater. This is like manual mode, but the bridge also shows in the front panel that the still heater is active.
  - **PID**: (Sample, Warm-up) Use the PID controller to set this heater value based on the input temperature.
  - **Warm-up**: (Warm-up) Set this output to work as a warm-up heater.
- **Input**: Input to follow, e.g., with the PID controller
- **Enable at startup**: Determines whether to enable when the bridge is turned on
- **Autoscan pause**: Determines the autoscan pause time when this heater is used in the PID

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- **Polarity**: Determines whether the heater should be operated in bipolar mode
- **Heater range**: Determines the range used for the heater
- **Heater resistance**: Determines the resistance of the heater connected to this output
- **Display units**: The units used for configuring and displaying the output value
- **Maximum current**: The maximum current the warm-up heater may deliver
- **Maximum user current**: The maximum user-specified current for the warm-up heater
- **Still value**: Output value when the still output is in still mode
- **PID P**: Proportional value of the PID controller
- **PID I**: Integral value of the PID controller
- **PID D**: Derivative value of the PID controller
- **PID Setpoint**: Target temperature for the PID controller
- **PID Enable ramping**: Use ramping with the PID controller
- **PID Ramping rate**: Ramping rate for the PID controller
- **Manual output**: The output value if set to manual mode
- **Warm-up control**: Determines whether to automatically turn off the warm-up heater
- **Warm-up percentage**: Warm-up heater value in percentage of maximum value
- **Monitor source**: Source of the monitor value
- **Source**: Determines whether to use the resistance or temperature value in control.

Display

The configuration in this tab configures what to show on the local display in the device.

There are a few general settings affecting the display functionality in general.

- **Display function**: Defines the general functionality of display. There is a predefined mode for showing the control input or measurement input and custom mode.
- **Number of locations**: The number of locations to show in the display
- **Displayed info**: On the lower half of the display, more detailed information can be shown from data specified by this configuration
- **Brightness**: Display brightness.

The display can be also configured to contain 2, 4, or 8 display locations which can be configured to show different values from different inputs. Each location has two settings:

- **Input**: The input number to show value from
- **Show value**: What value to show.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Relays

The bridge also has two configurable relay outputs. The relays can be configured to follow an alarm in some channel or based on the configuration defined in zone configuration.

- **Mode**: Determines whether this relay is off, on, controlled by alarm, or controlled by heater zone configuration
- **Input**: Used to input which alarm to follow
- **Alarm to follow**: Determines whether this should follow the lower limit, upper limit, or both alarms.

Instrument

The configuration in this tab controls the general configuration of the bridge.

- **Control input excitation frequency**: Excitation frequency for the control input
- **Measurement input excitation frequency**: Excitation frequency for the measurement input
- **Enable common mode reduction**: Enables common mode reduction between the scanner and the bridge
- **Monitor output**: Defines what output to show in the monitor output.

Curves

This tab can be used for managing the calibration curves in the device. There are 59 curves in the device. This tab only shows curves 21–59. Curves 1–20 are predefined by Lake Shore and cannot be changed.

Each curve has the following settings:

- **Name**: Name of the curve
- **Serial**: Serial number of the curve
- **Limit**: Temperature limit for the curve
- **Format**: Defines whether the defined data is in ohms/Kelvin or Log ohms/Kelvin.

Each curve holds up to 200 points of calibration curve data. The curve itself can be shown in here but it can be only set by loading a curve from a file. The curve can be loaded using the **Load curve** button.

> NOTE: The curve limit and format shouldn't be changed manually as the parameters only define how the input data should be interpreted, without making any changes to the data itself.

4.3.4 Fast Sample Exchange

The **Fast Sample Exchange** tab shows the controls related to controlling the sample exchange subsystem.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

The controls are divided into three groups: general controller status, movement, and valves.

4.3.4.1 Controller status

The controller status section contains the general status of the controller and a remote control configuration.

The device can be controlled from the software when remote control is active. The remote will be automatically switched off if the controller is in maintenance mode.

If there is a device level error or warning present it will be shown in the status textbox. The **INFO** button can be used to show a more detailed explanation about the current state.

If a critical error is encountered in the device, it stops operating until the error state is cleared. The **CLEAR ERRORS** button can be used to recheck the status after the actual error condition has been cleared off.

4.3.4.2 Movement

The movement section contains the controls and status information related to the movement and position of the sample exchanger. From the movement controls, it is possible switch the motor power on and off, stop movement, set the exact position, or drive it fully in or out.

When the device is initially switched on, it does not know the exact position of the sample exchanger. Therefore, for most of the controls to fully work, the sample exchanger must be driven fully out.

> NOTE: Only the controls allowed in the current state are enabled, e.g., it is not possible to drive the sample exchanger in if the FSE Gate Valve is closed or drive it to a certain position if the zero point is not known.

The status line contains the status of the driver that drives the motor. It can be a combination of the following messages:

- **Motor off**: The motor is off
- **Motor on**: The motor is on and idle
- **Motor current off**: The motor driver is on but the driving current to the motor is off
- **Driver in error state**: The motor driver is signaling an error condition. This is usually due to an inability to achieve a target position.
- **Driver in warning state**: The motor driver is signaling a warning condition
- **Torque limited**: The sample exchanger is moving at the maximum driving current which usually means that it is stuck or moving is requiring too much force. If the controller is in this state for too long it gets into an error state.
- **Driving to position**: The sample exchanger is moving to a defined position.

There is also a position status line which can show a combination of the following status messages:

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- **<position> mm**: The position measured in millimeters
- **??? mm**: The position is unknown
- **going to <destination>**: The sample exchanger is moving to a destination position
- **IN**: The sample exchanger is fully in at the limit switch
- **OUT**: The sample exchanger is fully out at the limit switch
- **force sensor active IN**: The sample exchanger's inwards force sensor has been activated to prevent damage to the hardware
- **force sensor active OUT**: The sample exchanger's outwards force sensor has been activated to prevent damage to the hardware
- **Going to home position**: The sample exchanger is driving to either a fully in or out position.

4.3.4.3 Valves

The valves section allows control of the valves related to the sample exchanger. In this section, there is a general status line showing the status from the valve-related sensors and the status and controls for the FSE Pump and Gate Valves.

The status line may contain any combination of following values:

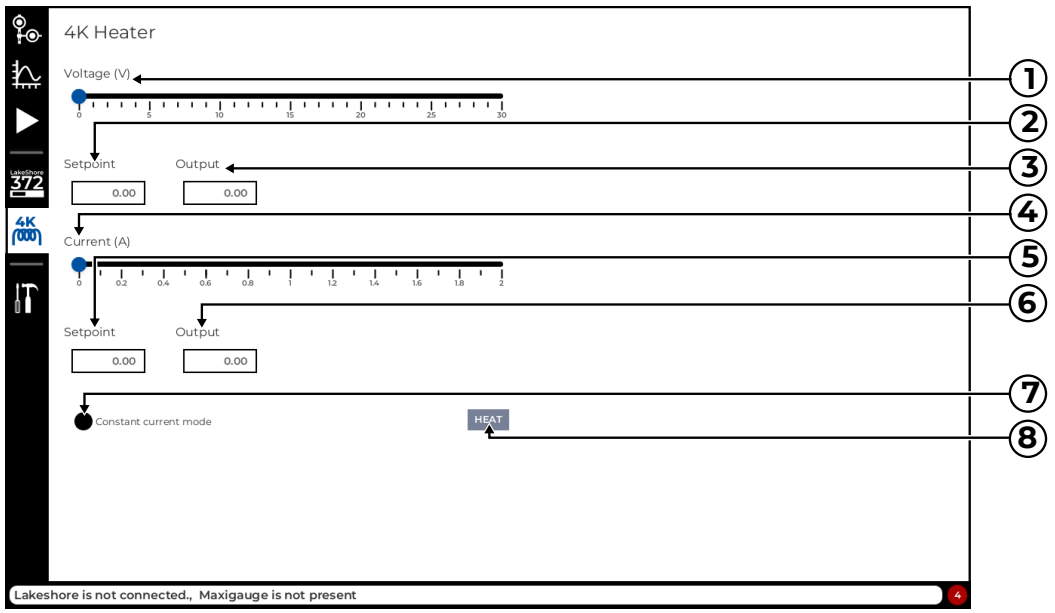
- **Check control pressure**: The control pressure driving the valves is absent and the device is unable to function.
- **FSE (Fast Sample Exchange) Insert mounted**: The sample exchanger FSE Insert is mounted to the cryostat.
- **Vacuum ok**: There is a vacuum inside the bellow so that the gate can be opened.

The valves can be controlled here if allowed by the state of the system. The valves can be set to either open or closed based on their current state. If the operation has failed and it is not known whether they are actually open or closed, they can be manually set. The status of the valve can be one of the following:

- **Open**: The valve is open
- **Closed**: The valve is closed
- **Opening**: The valve is opening
- **Closing**: The valve is closing
- **Open error**: The valve is detected to be open but the operation to change the valve state has failed
- **Closed error**: The valve is detected to be closed but the operation to change the valve state has failed.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

4.3.5 4K Heater



Item	Description
1	Voltage adjustment
2	Set voltage
3	Actual output voltage
4	Current adjustment

Item	Description
5	Set current
6	Actual current output
7	Constant current mode indicator
8	Enable heater

Figure 38: 4K Heater view controls

The 4K Heater panel can be used for controlling the TTi EL302P controlled 4K Heater.

The Control Software supports up to two units to be used as a 4K Heater. If two devices are added, both will have dedicated control views. However, when used from the scripts or front panel, they are operated in parallel.

NOTE 1: This controls only the TTi EL302P laboratory power supply, if one is installed, regardless of whether it is configured to work as a 4K Heater. Therefore, this cannot be used for controlling relay-based 4K Heaters.

When configured to be used as a 4K Heater and activated from a script or a front panel, the power supply is automatically set to output 30 V and 2 A, and the values set in this panel will be ignored.

The view contains the same controls as the physical interface of the laboratory power supply.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

The voltage slider controls the output voltage, and the current slider controls the output current. The values show the voltage set by the slider and the actual output value being applied.

The actual output value differs from the setpoint value if the power supply is unable to deliver the setpoint value, e.g., the voltage will drop if driving a low current to a low resistance load.

The indicator light shows whether the device is currently in constant voltage or constant current mode.

The enable button can be used for activating the output.

NOTE 2: Never activate the 4K Heater during normal operation in a circulation state. The heater is powerful enough to disrupt the normal behavior, and the whole warm-up and cooldown cycle must be executed to restore normal functionality.

4.4 Maintenance

Maintenance controls provide tools to configure the system and perform other maintenance tasks.

4.4.1 Program configuration

The configuration view contains controls for configuring the software for correct operation.

Whenever some configuration is unsaved, the configuration icon will be colored to indicate the presence of an unsaved program configuration. None of the core settings will be saved automatically. You must save them from the **General** tab.

The configurations are divided into five categories:

- **General**: Maintenance tasks, such as saving and loading the configuration to disk or rebuilding the internal value cache database
- **System**: Bluefors system structure specific configuration
- **API**: Control API configuration for allowing remote controlling of the system
- **Logging**: Measurement data logs locations, interval, and what to log
- **Devices**: Hardware devices controlled by this software.

4.4.1.1 General

The **General** tab contains maintenance activities for the program. The activities are divided into three groups: System logs, Configuration management, and Value database maintenance.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Software logs

The Software logs section is related to operating with the logs related to the execution of the program.

Currently, the only action is to open the software log folder, which opens Explorer in the location where the logs are stored. These logs are useful when troubleshooting problems with the program and so are often requested by Bluefors in such cases.

The logs are human-readable, text-based logs split into multiple files by age. The most recent one is uncompressed while the rest are compressed and numbered based on their age. The oldest ones will be automatically deleted to avoid filling up the disk.

Configuration management

Configuration management actions handle the loading, saving, and reconfiguration of the program configuration.

These actions allow saved configuration to be loaded the next time the Control Software Core is started, restored without restarting the software, and reset to defaults to start over.

The following actions are defined:

- **Save configuration...**: Saves the current configuration to disk to be restored when the core is restarted
- **Restore saved configuration...**: Restores configuration from the disk to currently running configuration
- **Reset configuration to default...**: Resets the configuration currently in the program to factory defaults, i.e., an unconfigured state. This will not overwrite configuration saved to the disk.
- **Reset user interface configuration**: Resets the user interface configuration, e.g., chart layouts and recently opened script files. This may be necessary if the user interface does not seem to work properly.

Value database maintenance

The value database is internal short-term storage containing all changes in the internal value tree structure, i.e., short-term measurement data log. The data is used, for example, to show plotlines in user interface charts.

Sometimes the database may become corrupted and reading or writing to it may fail. To recover from this state, it is possible to first try to rebuild the database, which will attempt recover all existing data from the value database. If this does not work, the database can be recreated. It is good to note that recreation of the database does not guarantee recovery of the data, it may still result an empty database, like with clearing.

The database is not used for the actual control process, so it is safe to clear it. However, all historical data will be lost, e.g., data from charts. The effect may not be immediate as some of the data is only loaded when the frontend is restarted.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

There are two actions:

- **Rebuild database**: Creates new database and tries to recover data from old database to it
- **Clear database**: Creates a completely new database.

4.4.1.2 System

The **System** tab contains settings for configuring the Bluefors system logical structure, e.g., what kind of hardware is being used for the normal control process.

The System tab is divided into base system configuration and temperature control configuration.

The base system configuration contains four features to set: installed options, 4K Heater device type, Heat Switch configuration and flowmeter calibration.

The temperature control configuration contains configuration for setting what device is being used for the temperature control operations and how. The tab contains configurations for preferred temperature control device, temperature sensor mappings, and heater resistances.

Options

The **Options** section contains all the options that may be installed and cannot be automatically detected.

Currently, only one is listed: **Second turbo installed**.

The second turbo is an additional turbo pump that can be used for evacuating the system, i.e., **Turbo 2** in the diagram. If the option is selected, it shows the second turbo along with related valves in the front panel.

In general, it is not likely to cause malfunction even if this is configured incorrectly. If it does not exist but is enabled, the controls just do nothing and the scripts that use it will fail. If it exists, but is not enabled, it is just simply ignored with an evident exception that if the corresponding valves have been left open, or the pump is running, ignoring them may cause damage to the pump and the system may fail to operate correctly.

Although, it is good to note, that this only affects what is being shown in the front panel. If the scripts use the option, they will try to operate it regardless of this configuration.

> NOTE: In general, it is not likely to cause malfunction even if the turbo option is configured incorrectly, except for in the evident case of the corresponding valves or pump being left active when disabling the option.

4K Heater

The 4K Heater configuration affects the device selected to act as a 4K Heater.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Currently, there are two different kinds of devices used for this purpose: an EL302P Laboratory power supply or a relay controlled fixed voltage output. This setting controls which device will be used for controlling the 4K Heater from the front panel button, from corresponding script variables, and subject to the safety timer that turns the heater off after a certain period.

When an EL302P is selected as a 4K Heater, the front panel button and corresponding script variable operate the laboratory power supply. The supply can be controlled manually from its dedicated control view, but when switched on or off from the 4K Heater front panel button, or from the script, it will automatically configure the device to reflect the default 4K Heater behavior, regardless of what is being manually set. When switched on, output is enabled and the EL302P supply is set to output 30 volts and 2 amperes. When the output is switched off, it disables the output.

> NOTE 1: If an EL302P device is present, the heater device view will be visible and usable regardless of this configuration. It provides manual control over the heater device and should not be confused with system-level 4K Heater configuration.

If an EL302P is not present and this option is chosen, the heater button will be disabled.

> NOTE 2: In some systems, there can be two EL302P devices used as a 4K Heater. If two are detected, both will work in parallel when operated as a 4K Heater. Otherwise, both have their own dedicated control views.

If configured as a relay-controlled device, one of the Bluefors Control Card channels controls a relay that either connects the heater to a fixed power supply or shunts the output.

> NOTE 3: If a relay-controlled 4K Heater is being used, it is still possible to connect an EL302P power supply, control it via its dedicated control view, and use it as an independent general-purpose output.

If a relay-based 4K Heater is selected when the system does not have a relay-based heater, the heater appears to be constantly active and there is no way of turning it off. This does not mean that the heater would be accidentally activated. The Bluefors Control Card just reports all non-existent channels to be active. So, if no extension card for the 4K Heater is installed, the output just appears to be constantly active while in reality it just does not exist.

It is also safe to change this configuration regardless of its current state as changing the heater type will not cause any of them to be turned on or off. The system updates the heater button state to match the state of the hardware, not the other way round.

> NOTE 4: Changing the heater type does not change the state of the hardware. When changed, it updates the program internal state to match the hardware state.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

The only actual concern in this configuration is, like with turbo option, that if the heater is left on and its type is changed to something else, the heater will remain on and so an incorrect configuration may hide its true state.

Heat Switches

The Heat Switch configuration defines how the Heat Switches are controlled. The Heat Switches are, in any case, turned on and off via the Bluefors Control Card. In the older systems, the Heat Switches are connected to the Heater Box with a fixed output when activated. New systems with a Bluefors Temperature Controller also use the same relay channels to activate heaters, but instead of a dedicated Heater Box, the power is supplied by the Temperature Controller, which has controllable heater outputs.

Both will work if the controller type is set to Bluefors Control Card. However, if a Bluefors Temperature Controller type heater is used and the corresponding source is selected, the Control Software will also make sure that the Bluefors Temperature Controller will be correctly configured when activated. The type is used for determining what output power the Temperature Controller should be configured to use.

So, choosing Bluefors Control Card will always work, but configuring the Bluefors Temperature Controller correctly provides more robust functionality. If an incorrect system type is chosen, the Heat Switches may just fail to operate correctly.

Flowmeter calibration

The flowmeter, showing the gas flow in circulation, provides analog output that is connected to the Bluefors Control Card. The raw input voltage is converted to flow inside the Control Software. The parameters entered here depend on the system type and can be read in the system manual.

Unless otherwise specified, the standard values are:

- **SD/LD**: Offset: X.XX, Multiplier: Y.YY
- **XLD**: Offset: X.XX, Multiplier: Y.YY.

Temperature control device preference

The Control Software automatically chooses the device that will be used for the temperature controlling functionality. The device preference can be used for controlling what device to use if the setup contains both Bluefors and Lake Shore devices.

If the system has two Bluefors Temperature Control devices, the first one will be used for channels 1–8 and second for channels 9–16.

Sensors

The **Sensors** section contains list of temperature sensor mappings that needs to be set to use the temperature controller in scripts and to show correct temperatures in front panel.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Each sensor has its own field where the corresponding temperature controller channel can be selected. If two Bluefors Temperature Controllers are used, the second controller channels are assigned to channels from T9 to T16.

Mixing Chamber and Still Heater resistances

To be able to reliably determine the output power, the heater resistance must be known. In addition, for some heaters the lead resistances may be also required.

The reason for this is that while some heater outputs are current controllers, it is possible to calculate the actual power dissipated by the heater just by knowing the heater resistance, while some of the heaters are driven by voltage sources. In this case, to calculate the actual power dissipated by the heater, the lead and heater resistances must be known.

For both devices, the sample heater is driven by current, and the still heater is driven by voltage. So, lead resistances are currently needed only for the still heater. However, in case this changes in the future, the lead resistance should be defined for both.

Both devices are also capable of doing the power calculations automatically within the device for the sample heater. For the still heater, the calculations are done in the Control Software. However, this is done with full transparency within the software so that the controls will be exactly the same from the user interface and via API regardless of the actual heater type.

The resistances are stored in Control Software configuration and are updated to the device automatically as part of the configuration.

The equation used for calculating voltage from power: $U = (R_h + R_l)\sqrt{\frac{P_h}{R_h}}$

and for calculating power from voltage: $P_h = \frac{U_h^2 R_h}{(R_l + R_h)^2}$

where:

- R_h = *heater resistance*
- R_l = *lead resistance*
- P_h = *heater power*
- U = *voltage output value.*

4.4.1.3 API

The **API** tab contains configuration for the Control API used for remote control.

> NOTE: More detailed information about API configuration and exact functionality can be found in Section 6.

The Control API provides both a secure encrypted connection and unencrypted connection for local use. Access keys can also be defined to restrict what different

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

groups of API users can do by using the API. A secure connection also needs a certificate for operating correctly.

The API view provides settings for configuring these details.

General

The only general configuration is:

- **Enable API**: Enables or disables the API completely
- **Enable access log**: If enabled, the software will write an entry to the access log in the software log directory (see Section 4.4.1.1) for every connection made to it.

HTTP(S)/(Secure) WebSocket

HTTP/WebSocket servers and HTTPS/Secure WebSocket servers have identical settings:

- **Enable server**: Enables the corresponding server
- **Port**: The TCP port that is used for connecting to the API.

Certificate configuration

Certificate configuration shows general information about the certificate used for secure connections. By default, the program uses a self-signed certificate that can be exported from here to be used for verification. You can also generate a new self-signed certificate or load a custom one from a file.

> NOTE: Control Software does not yet support password-protected certificates. Therefore, anyone with access to the user account also has access to both public and private parts of the certificate.

Access control keys

The Control API uses keys to authenticate the API users.

Each key can have an optional name, which can be used for stating its purpose.

Each key has **Read** and **Write** fields for every endpoint, which indicates whether the key grants the permission to the corresponding endpoint.

4.4.1.4 Logging

The Logging configuration defines what values to log. The logging has been designed to be fully backward compatible with the old ValveControl software.

The logging configuration is self-explanatory. You can choose what to log, where, and how often.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

The logs will be written to the selected path and separated into directories by date in the format *YY_MM_DD* to automatically sort them conveniently.

The following values can be logged:

- **Flow**: Writes the flowmeter reading to a file
- **Pressure**: Writes pressure channel values P1–P6 to a file
- **Status data**: A status log containing general statistical information, similar to status view content, such as pump temperatures and running hours
- **Relay channels**: The statuses of the relay-controlled features, e.g., the activation status of the valves, pumps, and compressor. Unlike other logs, this is also written whenever any of values changes to preserve the exact control flow of the process.
- **Temperature sensors**: The temperatures, resistances, and power (only with Lake Shore devices) read from the main temperature controller
- **Heaters**: The power levels applied to the heaters via the temperature controls for the heaters.

4.4.1.5 Devices

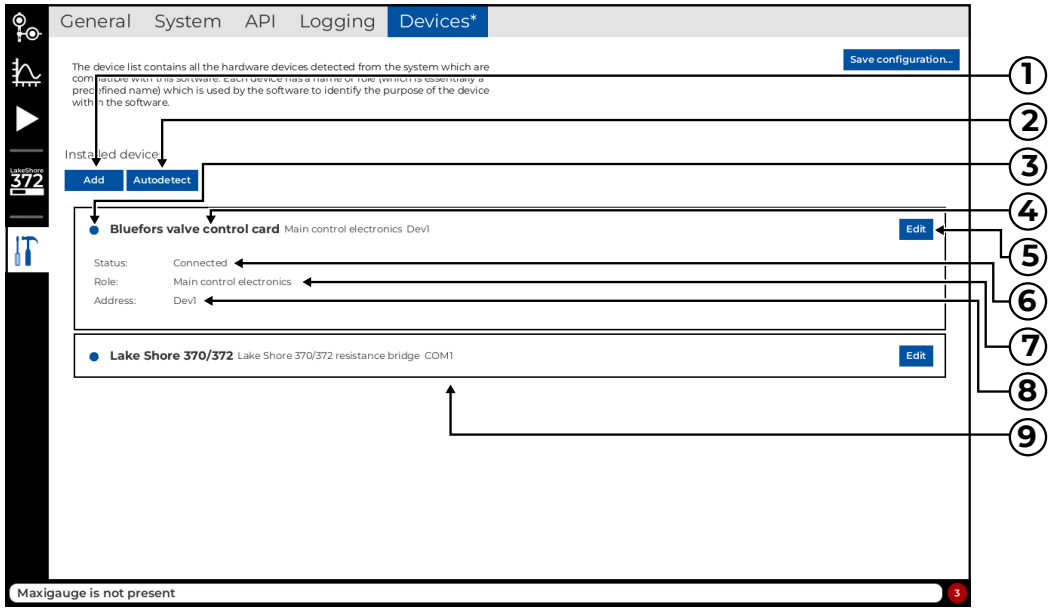
The **Devices** view defines connections to the hardware devices. Each individual controllable device in the system appears as a separate device in the Control Software. Although, the mapping between logical devices and hardware devices is not direct. For example, most of the pumps are controlled via Bluefors Control Card but statistics are read directly from them.

More detailed information about what devices there are and what should be found, can be found in Section 3 about setting up the system for the first time.

From this view, you can add devices, autodetect devices, or change existing device details.

For each device, the view contains a device box showing all the device details, as shown in Figure 39. The **Devices** tab has controls for adding and automatically detecting devices and showing all current devices.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Add new device
2	Automatically detect devices
3	Connection status of the device
4	Device name
5	Edit device connection configuration

Item	Description
6	Connection status
7	Role of the device
8	Address or port of the device
9	Unexpanded device box

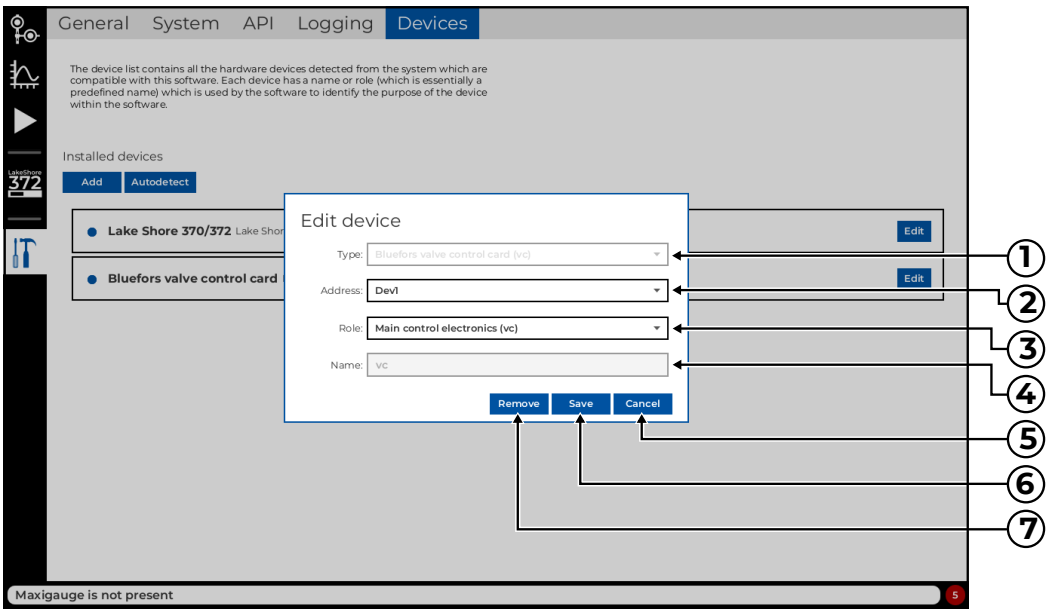
Figure 39: Device configuration tab

Adding or editing devices

The devices can be added manually by clicking the **Add** button in the **Devices** tab. The details can be edited later by clicking the device box in the device list.

Both open a similar dialog. The device type can be only changed when adding the device, and devices can be only removed when they have already been added.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Type of the device to add
2	Location of the device, e.g., address or port
3	Role of the device in the system, i.e., name preset
4	Name of the device when role is set to **Other**

Item	Description
5	Cancels current edit or add action
6	Saves or adds the device
7	Removes the device when editing

Figure 40: Edit device dialog

The **Edit device** dialog allows you to edit the device configuration. The **Add device** dialog is similar but allows you to change the device type and does not have the option to remove the device as it does not exist yet.

Autodetection

Start the automatic device detection by clicking the **Autodetect** button in the **Devices** tab.

When automatic detection is started, all the devices will be disconnected to allow probing of the devices in all ports. Once the automatic detection is finished, you can either discard the results and restore the old connections or add new devices from the list and discard all the old devices.

Most of the devices that are being probed are serial devices. The serial protocol itself does not define what kind of traffic is in the bus. Because of this, there is no reliable way of identifying the devices any other way than just trying to communicate with them. The automatic device detection also works this way. It tries to communicate with every type of known serial device to every serial port. If

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

the device responds back correctly, it is very likely to be the correct device. However, all incorrect probing attempts are nonsense from the device's point of view. It is known that all devices that Bluefors uses tolerates this well, but there may be third-party devices that do not tolerate it or may even break. For this reason, care should be taken if such devices are connected when running automatic detection. Any third-party devices should be disconnected when automatic detection is being run.

NOTE 1: Devices will be temporarily disconnected while detecting devices. Connections will be restored once the process has been completed.

NOTE 2: If detection results are accepted, the new devices will replace all old devices.

DISCLAIMER: USE OF THE AUTOMATIC DEVICE DETECTION FEATURE WITH ANY THIRD-PARTY DEVICE IS PROVIDED "AS-IS" WITHOUT WARRANTY OF ANY KIND (EITHER EXPRESSED OR IMPLIED) AND DONE AT YOUR OWN DISCRETION AND RISK WITH THE AGREEMENT THAT YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGE, LOSS OR ERROR (INCLUDING LOSS OF DATA) THAT MAY RESULT FROM SUCH USE. IN NO EVENT SHALL BLUEFORS BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DIRECT OR INDIRECT DAMAGE OR LOSS RESULTING FROM SUCH ACTIVITIES WHETHER IN CONTRACT, TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), OR OTHER THEORY AND REGARDLESS OF WHETHER BLUEFORS WAS ADVISED OF OR WAS AWARE OF THE POSSIBILITY OF SUCH DAMAGES OR LOSSES.

When automatic detection has been completed, a list of detection results is shown. The list contains all detected devices and their roles. The roles are assigned in the order they were detected. It is possible that the natural port ordering does not match the device ordering, so the roles can be adjusted here if the exact device locations are known. You can also select which devices will be added.

Autodetection does not find all devices. Currently, only Bluefors Temperature Controllers must be added manually as the device does not support any feasible way of automatic detection.

Bluefors Control Card uses National Instruments USB-6008 data acquisition hardware as a frontend to the computer. If other devices of the same type exist in the system, they may be detected as a Bluefors Control Card.

NOTE 3: Currently, automatic detection does not detect Bluefors Temperature Controllers.

NOTE 4: Bluefors uses NI USB-6008 as a hardware frontend and other devices of the same type may be detected as a Bluefors Control Card.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

5 Program structure

The program has been designed to be a replacement for the old Bluefors ValveControl software. It provides a simple manual interface for controlling the system with scripting support for automating tasks.

Internally, the program is a modular measurement framework, and the simple user interface logic is designed to hide unnecessary complexity. However, when using the API, it is essential to understand internal details.

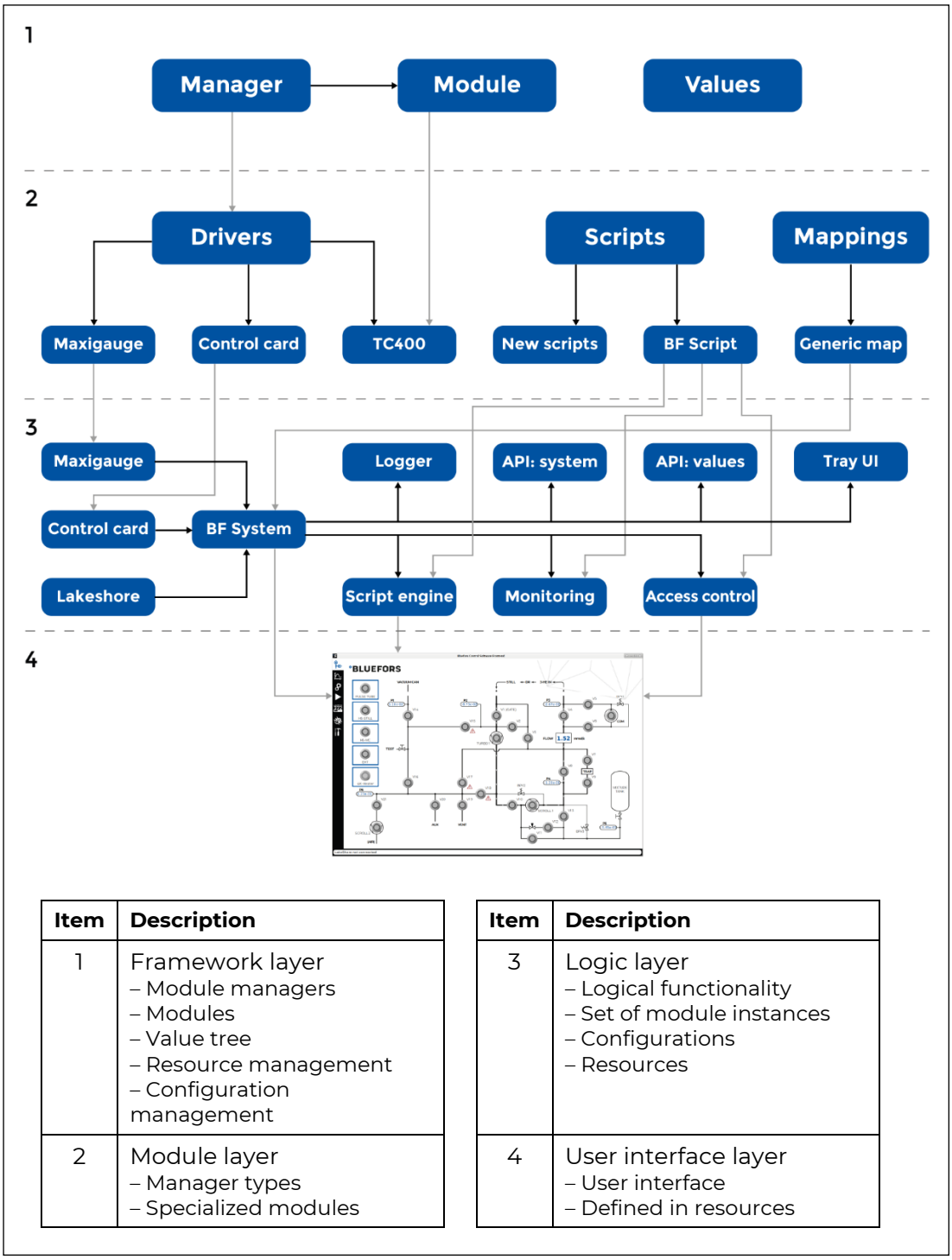
5.1 High-level design

The relationships between managers, modules, control logic, and user interface can be seen in Figure 41.

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS



Item	Description
1	Framework layer – Module managers – Modules – Value tree – Resource management – Configuration management
2	Module layer – Manager types – Specialized modules

Item	Description
3	Logic layer – Logical functionality – Set of module instances – Configurations – Resources
4	User interface layer – User interface – Defined in resources

Figure 41: The layered structure of the control software

5.1.1 Managers and modules

The core functionality of the program is constructed from **modules** which all take care of certain tasks within the program, such as communicating with Fast Sample Exchange (FSE), writing valve logs or managing the core user interface. The

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

modules are managed by **module managers**. Each manager takes care of one specific type of module, like device communication drivers or system mappings.

The following managers exist in the standard Bluefors system, which have roles in the control process:

- **driver**: Driver modules handle communication with the devices and provide interfaces to operate them.
- **mapper**: Mapper modules are used for “mapping” driver data to logical names, for example mapping raw control card relay channels to logical valve names from v1 to v23.
- **script**: Script modules provide modules for running scripted tasks within the program. These may be either fully user-configurable scripts, such as a Bluefors script available via user interface, or automatic monitoring of tasks running in the background.
- **general**: General modules provide general purpose features, such as logging modules or a core system tray user interface.

5.1.2 The value tree

The central structure of the program is the value tree. It contains all run-time data in the program, such as measured values, controls, and device configuration. By far most of the logic in the program involves interaction with other value tree nodes.

The value tree follows the same hierarchical structure as modules and managers. Each manager has a branch starting from the root of the tree. Each module may have a branch under the corresponding manager. A branch only exists for a module that needs it. For example, logger modules do not have them because those modules only receive data and do not provide any. The hierarchical structure is illustrated in Figure 42. The exact structure of the branch is specific to the module. However, certain types of modules may have fixed common characteristics.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

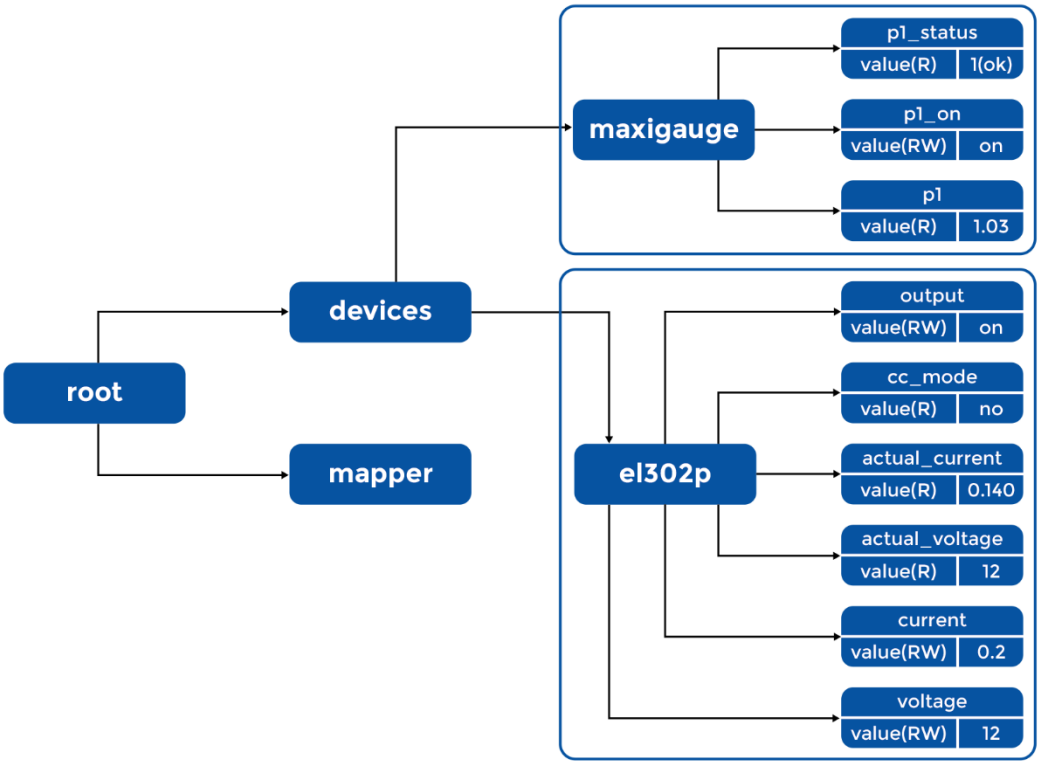


Figure 42: Example value tree

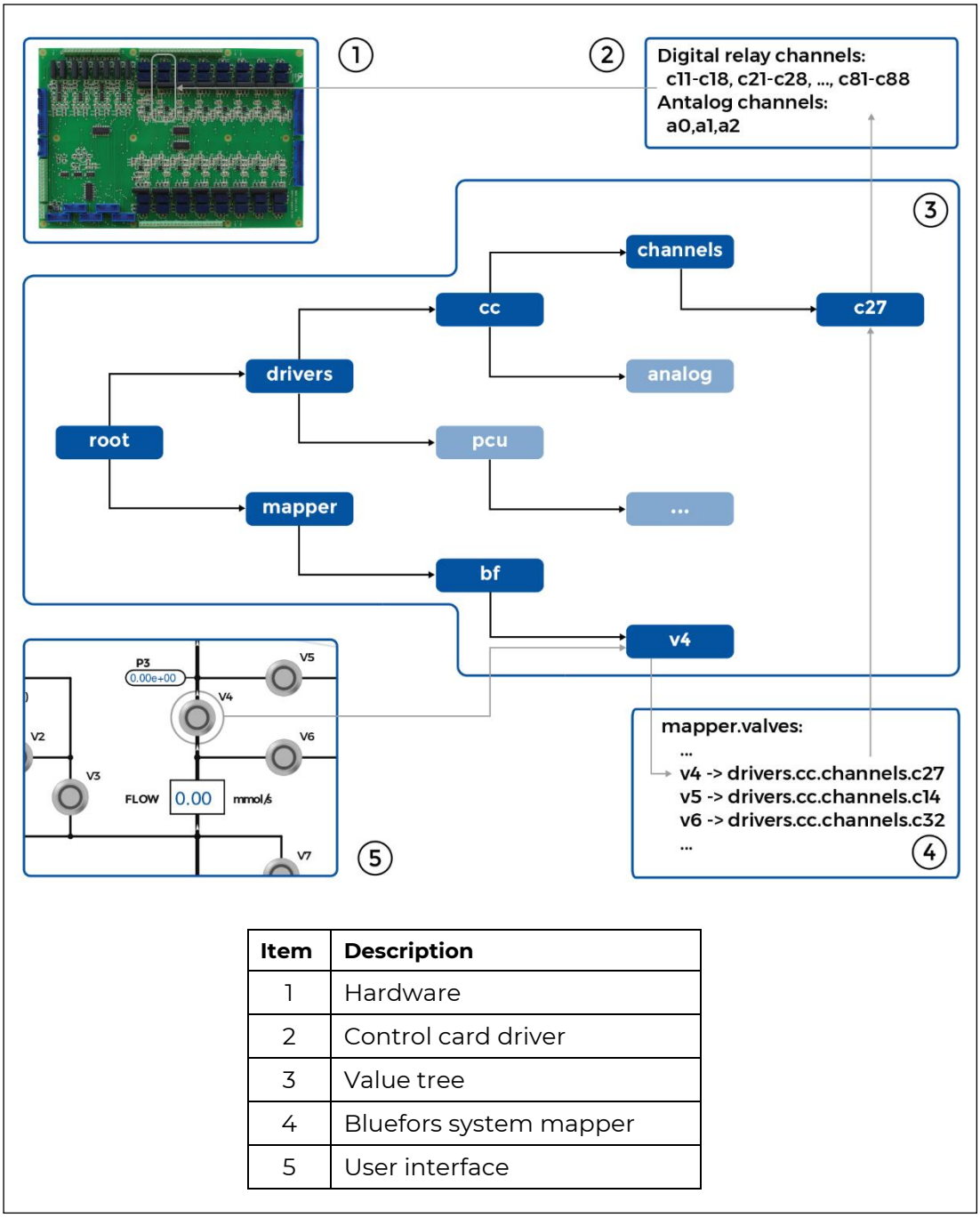
5.2 Dataflow

All interaction with the measurement and control data is done via the value tree. The whole tree is available to API and user interface logic. Therefore, it is possible to interact with any values provided by the modules. However, there are common patterns about how these are generally handled.

5.2.1 Standard dataflow

The data flow from the device to the user is shown in Figure 43. In the typical data flow, when you click a button, the program updates the value to the corresponding node in the mapper module in the value tree. The update request to the mapper module is relayed to the device node, and the underlying device driver module performs the actual write operation.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Hardware
2	Control card driver
3	Value tree
4	Bluefors system mapper
5	User interface

Figure 43: The typical data flow

The physical device provides variables that can be read or controlled, e.g., measurement data or device configuration. Each device driver module provides value tree nodes for interacting with these from the program. The device drivers provide a low-level interface to the devices. The point is to provide at least part, but usually all, of the device functionality to the program.

The data provided by a device may not represent the logical purpose of it in the program. For example, Bluefors Control Card is just a relay control card with up to eight sets of eight outputs. These are numbered from 1.1 to 8.8. The driver module

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

provides the channels as values from ch11 to ch88 via the value tree. These can be used for switching the corresponding relays.

However, the logical functionality of the Bluefors system is composed of valves and pumps that the relays control, so it would be impractical to directly control the relays that control them. To mitigate this, the Control Software also has mapper modules that map the raw hardware interfaces to logical structures of the system, e.g., from a relay channel to the valve it controls. The mappings can be simple links or complex modules with intelligence. The value mappings are an example of a simple mapping where each relay channel directly maps from one relay to one valve.

As an example of a complex mapping, the temperature controller module provides two heaters: the Mixing Chamber Heater and Still Heater. These are both controlled by two variables: enabled and power. The actual underlying hardware can be either Lake Shore 372 (or 370) or Bluefors Temperature Controller. All of these have more refined control over heaters providing different operation modes, range settings, etc. The Lake Shore devices also only have a sample heater that can be controlled by power, so the actual power for the second heater must be calculated. It is evident that mapping this functionality to one power value requires additional logic to work.

5.2.2 Mapped vs. driver values

The use of mapped values to control the drivers is the most common dataflow in the Control Software. However, the devices can also be controlled directly. The mapped interface has been designed with simplicity and compatibility over many configurations in mind. It provides only a small subset of the underlying functionality of the hardware. For example, all supported temperature control devices provide a rich set of features and fine-grained control over them.

This functionality is accessible directly from the driver modules and can be used if care is taken to avoid interfering with the control process, e.g., if the temperature control device configuration is being changed during the cooldown process.

The device-level configuration and values that may provide value to the end user are also exposed to the user interface in dedicated device views so you can take full advantage out of them.

NOTE 1: The device drivers provide direct access to all device features, but care must be taken to avoid interfering with the normal functionality of the dilution refrigerator.

NOTE 2: When controlling the system via Control API, it is good to keep in mind that mapped functionality remains constant over systems with different configurations while underlying devices and the modules used for controlling them may change.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

5.2.3 Other values

While most of the data is accessible directly from drivers or via mapper modules, there are also other modules with accessible data. For example, the Bluefors script engine provides variables for updating, starting, stopping, and pausing the script as well as reading its current status.

The way these are used is specific to the module, or module type.

5.3 Value tree content

The value tree is the central data structure that contains all the data and functionality required for normal operation.

Each tree node may contain any number of child nodes. Each node may also have a content type. The content type is the actual value or function that the node represents. Two types of nodes are commonly being used:

- **Value**: Represents any measured or controllable value, e.g., a still temperature
- **Call**: A function that can be called, e.g., start the script.

The type system is hierarchical. Value and Call are the two main types of value, but they are often refined into subtypes. Each subtype layer defines more specific control over it.

The type is defined as a dot separated string of type specifiers. For example, *Value.Number.Float.Unit* defines that the content type is Value with a subtype of a numeric value with a subtype of a floating point value and with unit information.

For example, in the example above, any content with the Value type can be set and read as a string. If the type is Number, it can be also set and read as an integer or a floating point number. The floating point subtype defines that the value is a floating point value and should be treated as such. The unit type also contains information about what unit it is and provides functions to present and convert it between different units and magnitude systems.

5.3.1 Value content

Value is a communication channel between the modules and the value tree.

The content of the **Value** type is a variable that can be read or written to. Instead of handling single values, it uses samples. In addition to value, the sample contains the time of the event, status of the operation, and other metadata. The sample structure is described in Section 5.3.1.1.

The value will hold two samples: the latest and the latest valid sample, if any. This is an important feature when dealing with unreliable data sources. For example, the device communication channels may have short interruptions. Whenever an interruption is perceived, the values will be updated accordingly. However, when

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

showing the values in the user interface, or when running scripts, for example, it is sufficient that there is a recent enough valid reading that can be shown.

If the latest valid value becomes too old, the latest one can be used instead to show the actual state. For example, if the serial line has been disconnected long enough so that the latest value is not reliable enough, the device can be considered to be disconnected instead of there being an insignificant interference in communication.

You can read both values. When you write or update the value, instead of providing full sample, you provide only a value itself. The value is relayed to the write function in the underlying module that executes an operation with it and creates a new sample as a result and stores it in the value. The operation may be internal buffering, or the device driver may write it directly to the device.

As the only exception, direct one-to-one links in mapping modules appear as normal values, but instead of implementing read and write logic, they transparently relay the requests to the target node in the value tree.

The value update data flow can be seen in Figure 44. When you read the value, it reads the value (via mapper link) from the sample stored in the driver value node. When you decide to write the value, it invokes a write operation in the value node, which is passed to the hardware driver and eventually to the hardware. When the value has been written, the driver will read back the value and write the new updated value to the value node, which in turn notifies you about the updated value.

Commonly, drivers also update the values when new data becomes available or at a fixed interval.

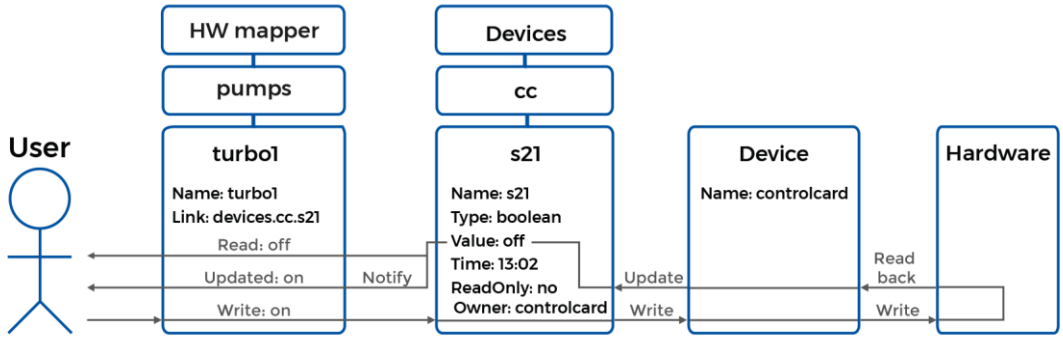


Figure 44: How values are being read and written

5.3.1.1 Samples

The data in values is encapsulated in **value samples**, which also contain metadata about the data. Each value keeps track of the latest sample and the latest valid sample.

The samples can be considered to be events rather than plain representations of the values. Each change in value, whether it was a disruption in communication or value acquisition, will result in a sample to be created.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Each sample has the following content:

- **Value**: The data stored in the value (the data may not exist if the operation has failed)
- **Timestamp**: The time the value was acquired
- **Status**: The status code which defines whether the data was valid (these are shown in Table 4)
- **Exception**: If the operation failed, the value may have Java exception data available for further analysis.

Table 4: Value sample status codes

Status code	Valid	Description
SYNCHRONIZED	Yes	The data is valid and synchronized with the target, usually device.
INVALID	No	The data is not valid, for example if read failed or the value has never been read from the device.
CHANGED	Yes	The value is valid and changed on the module side but not updated to the device yet. This is used, for example, with settings that can be changed by the user but are updated to the device only via an explicit request.
DISCONNECTED	No	The device is disconnected, and no data is available.
INDEPENDENT	Yes	The data is not bound to any physical device and the update is handled completely programmatically.
QUEUED	Yes	The value is waiting to be updated back to the device. Changed values will be transferred to this state when the update sequence is initiated and before the values are actually updated.

5.3.1.2 Reading and writing

When a value is being written, the request is relayed to an underlying module that performs the update. When the operation is finished, it is guaranteed that the value is updated. The updated value represents the state after the update. In some cases, this may not be what was written to it due to an error, rounding, or value being out of range.

For reading, there are two different functions to execute the task. It is possible to request either the latest value or latest valid value. The latest value is always the most recent sample provided by the module, which may be also invalid, e.g., due to disconnection. The latest valid value returns the latest value, which is still considered to be valid, if any. The validity is checked from the sample status. The value also has an attribute maximum age, which defines how old the sample can be to be still considered valid.

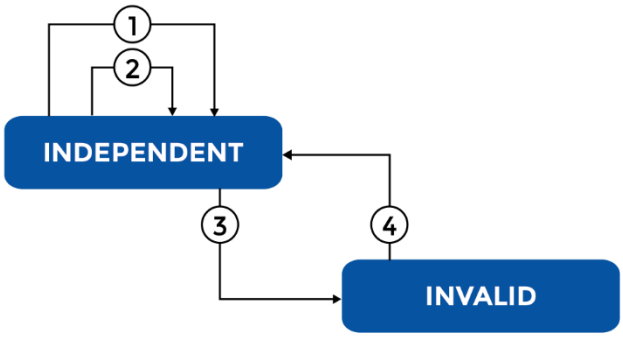
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

The rationale behind this is that there may be communication errors which could cause values to be unavailable for a short period of time. However, for the control process, it is usually sufficient that the latest value is recent enough to be usable, e.g., 5 seconds. The use of the latest valid value makes it possible to mask out minor issues, e.g., in communication channels. However, sometimes the absolute status is needed. For example, when a value is being written and updated, it is important to know what the real result of the operation was to know whether it was really executed at all.

5.3.1.3 Use patterns

There are three ways the values are commonly used in the program. These are shown in Figure 45, Figure 46, and Figure 47.

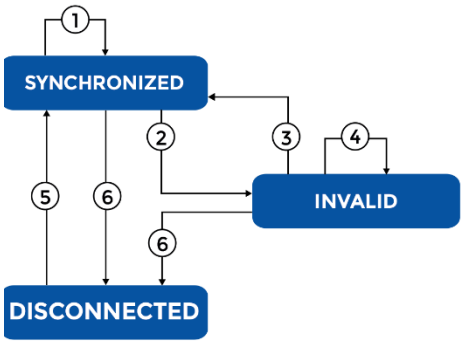
There are three common patterns about how the values are being updated and how the status is updated.



Item	Description
1	Update value
2	Set value
3	Unexpected error
4	Resolve error

Figure 45: Local value state transitions

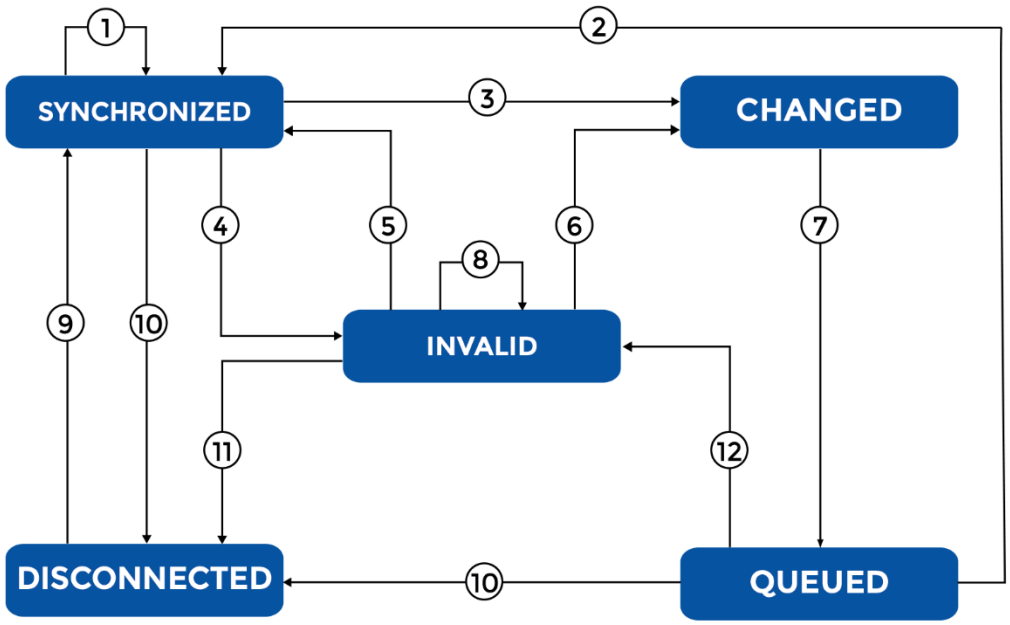
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Read / write value from / to device
2	Read / write returned unexpected value, other error
3	Read / write successful
4	Read / write still unexpected value, other error
5	Connection ok, value updated from the device
6	Connection lost / error

Figure 46: Immediate value state transitions

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	Value update from device invoked
2	Value update completed
3	New value written
4	Read returned unexpected value, other error
5	Read successful
6	Wrote new valid value

Item	Description
7	Value update executed
8	Read unexpected value, other error
9	Connection ok, value updated from the device
10	Connection lost
11	Connection lost/error
12	Write failed, invalid response

Figure 47: Delayed value state transitions

Local values

Local values are bound to variables within the module itself. They are used for values that are handled directly by the module, so there is no need to think about synchronization status. Therefore, the only sample type used is *independent*.

Immediate device values

Immediate device values are values with a target that is outside of the program, for example, in a device connected via a serial cable. The value itself is handled by the module, but the target value is in the device. To reach the value, a connection must be established. The connection may also be unreliable. Taking this into account,

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

the values use the synchronized, disconnected, and invalid status values. When writing to the value, the module relays the value to the device and reads the updated value back. Depending on the device, it may keep the value updated by polling it with either regular interval or asynchronous notifications. Either way, the value is actively kept up to date with the device.

When a sample with a synchronized status is received, it is considered to match the corresponding value within the device reasonably well.

When a sample with a disconnected status is received, the module has lost connection to the target device and no value can be read.

When a sample with an invalid status is received, the value could not be read for some other reason. It is possible that the connection was established, and the read command was successful, but the response was unsuccessful.

Unlike local values, immediate device values also have a maximum age. This is used for masking out brief disconnections and invalid values. When the latest valid value is requested, a latest sample with a synchronized status and recent enough timestamp is returned even if there would have been other samples in between.

Delayed device values

Delayed device values share many characteristics with immediate values. The difference is the way the value is updated. While immediate values are written directly to the device and actively updated from the device, the delayed value is only updated from and to the device explicitly. One common reason for this is that the settings are read and written in batches.

Delayed values also share the same statuses as immediate values. However, since delayed values are not updated with a known interval, the maximum age is not used. When writing to the value it cannot be immediately updated to the device. Instead, the temporary value in the module is updated and a new sample with the status *changed* is created. This denotes that the value is no longer synchronized with the device as the user has changed it, but it is still valid. When a write operation is invoked[4] the value status changes to *queued*, which means that the value is waiting to be updated to the device.

Once the update is done, a new sample is read, and the value is set back to a *synchronized* state. If the value is updated from the device when it is in a *changed* state, it may be updated back to the value read from the device. This is a partially device-specific functionality. For example, upon reconnection, the module may keep the changed values as is, even if the values would be read from the device. This is to avoid the user losing an in-progress configuration in case of a reconnection. However, the values will be reverted if the user explicitly requested the update.

[4] This does not specify how exactly this is done. It is module specific functionality.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

5.3.1.4 Value types

There are currently string and numeric values. Numeric values also have subtypes for integers and floating point values. Integers also have a subtype for enumeration value, and float point values have a subtype for values with units.

Value.String

The String type is used for textual content. The only supported operations are reading and writing the content.

Value.Number

Number is the main type for all numeric values. It is not used by itself. It just provides the compatibility and conversion support between different types of numbers.

Value.Number.Integer

The Integer subtype represents any integer. Integers are only used explicitly when integer numbers are expected, e.g., when a channel index is being scanned.

Value.Number.Integer.Enumeration

Enumeration is a subtype of Integer that inherits all the features of Integer but also adds a list of possible names to the values. The names can be used when reading or writing the values. These are commonly used with devices which have status values where each numeric value has a special meaning, e.g., 0 = O, 1 = On, 2 = Error.

The list of possible values can be requested from the value.

Value.Number.Float

The Float subtype represents any floating point number. Internally, this is a double precision floating point number. These are commonly used for measured data and calibration values which do not have unit.

Value.Number.Float.Unit

Unit is a subtype of floating point number that represents a value with unit information. The value with a unit can be converted from and to strings with different similar units and magnitude prefixes, e.g., you may request or set the value in millikelvin or Fahrenheit. Internally, the values are always stored in its base type as a float number.

5.3.2 Call

The **Call** content type is used for implementing function calls, i.e., executing tasks, from the value tree, such as starting or stopping the scripts.

When a call is being invoked, it executes the corresponding task. When the task is finished, you will be notified about it. The response may either be success or an error response if the operation failed.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

It is also possible to pass parameters to function calls, and they may return a value.

6 Control API

The Control API provides remote access to the software. It supports HTTP and WebSocket based APIs with and without encryption.

All the requests and commands in the API use JSON[5] as a format for the content.

6.1 Structure and terminology

6.1.1 Addressing

The resources in the API are referred by URIs, which are defined in RFC 3986[6]. This manual follows the same terminology when referring to URI components:

http	://	localhost:1234	/	values	?	filter=value	#	something
ws	://	localhost:1234	/	ws/values				
scheme		authority		path		query		

- **Scheme**: The scheme defines the used protocol. In this case it is either http, https (secure HTTP), ws (WebSocket), or wss (secure WebSocket).
- **Authority**: The authority defines the host name and protocol number in form *<hostname>:<port>*.
- **Path**: The path defines the location of the resource in the server.
- **Query**: The query part contains additional data which will be passed to the resource found on the path. In this case, this is a list of parameters separated into key-value pairs that are separated by commas, e.g., *<param1>=<value1>,<param2>=value2 ..., <paramN>=<valueN>*.
- **Fragment:** The fragment part of a http URI is not used in this API.

6.1.2 Services

The Control API provides a variety of functionalities for different purposes. These functionalities are divided into services by function, e.g., general system information or value data access.

The services are called **endpoint**s and are identified by the first part of a URI path.

All HTTP endpoints are identified by the first part of the path, e.g., *http://localhost:1234/**values**/driver/*. The first part of the path that identifies the

[5] https://www.json.org/

[6] https://datatracker.ietf.org/doc/html/rfc3986

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

endpoint is called the **endpoint name**. The rest of the path is called the **endpoint path** and is used for identifying a resource within the endpoint.

All WebSocket endpoints are identified by the first two parts of the path. The first part is always *ws* followed by the WebSocket endpoint name, e.g., *http://localhost:1234/**ws/values***. As with HTTP, the rest of the path is the **endpoint path**. However, an endpoint path is rarely used with WebSocket endpoints because the connection is bidirectional and it allows interaction with multiple resources, e.g., subscribing to listening changes in multiple values.

6.2 Access protocols

The API supports HTTP and WebSocket with[7] and without encryption. The HTTP-based API is easy to use. However, it is a request-based protocol, and the server is unable to notify the client when new data is available, so you must ask for new data, i.e., poll the data. The WebSocket protocol establishes a continuous bidirectional connection, allowing you to subscribe for listening changes in the data so, instead of asking for changes, the server automatically sends new data when it is available.

> NOTE: Both protocols provide access to the same data. If a simpler HTTP connection is sufficient, there is no need for implementing support for WebSocket.

6.3 Hyper Text Transfer Protocol – HTTP

Hyper Text Transfer Protocol, or just HTTP, is very well known to us all from delivering web pages, but it is also commonly used in machine-to-machine communication. Part of it can also be easily experimented with using any web browser, which makes it an easy choice.

The HTTP protocol is defined in standard RFC 2616[8]. The secured connection (HTTPS) is defined in RFC 2818[9].

6.3.1 Quick start

The simplest way of communicating with the API is to use the HTTP or HTTPS protocol for communicating.

This part of the API uses simple client-initiated request-response communication.

As defined in Section 6.1.1, HTTP endpoints use a URI path to identify the endpoint and resource in it.

[7] HTTPS and Secure WebSocket
[8] https://datatracker.ietf.org/doc/html/rfc2616
[9] https://datatracker.ietf.org/doc/html/rfc2818

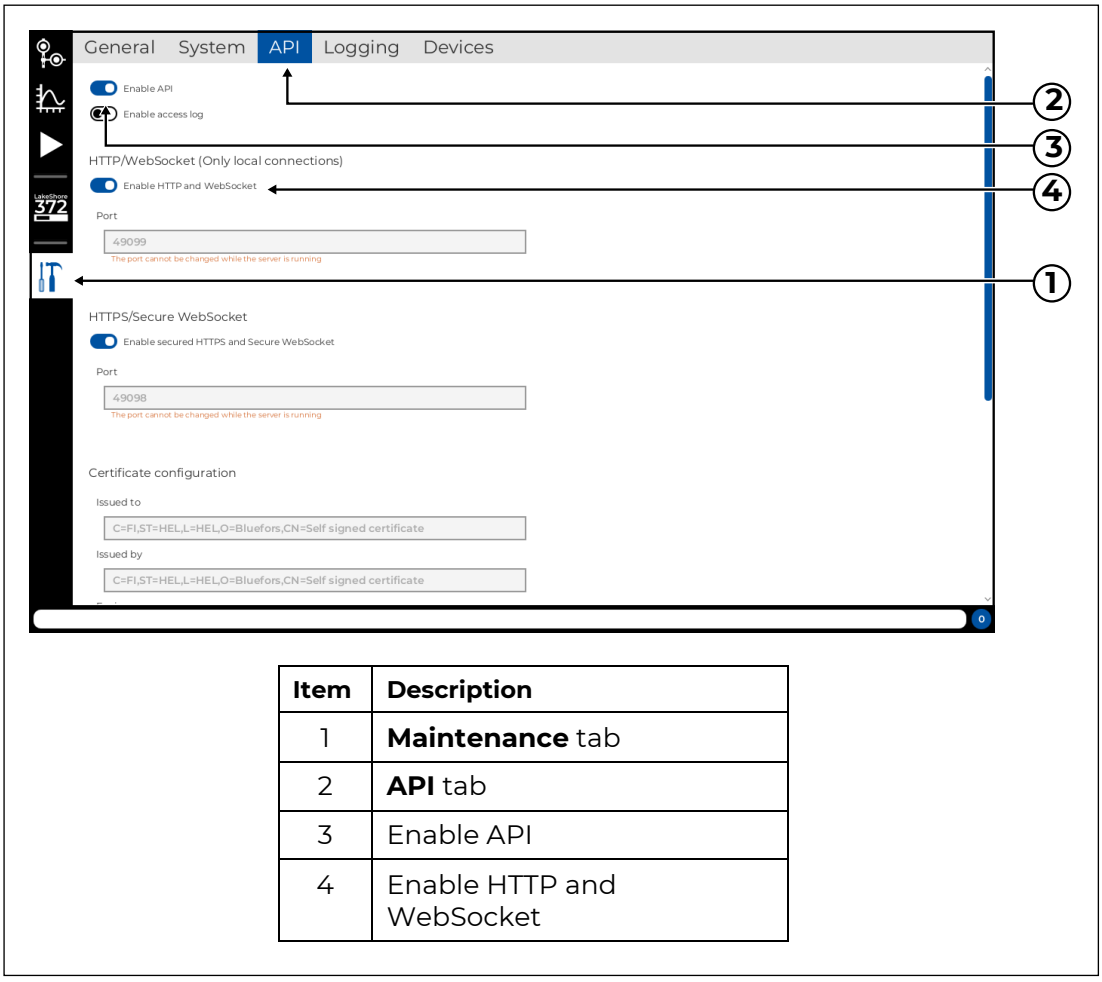
Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

6.3.1.1 Example

One of the great advantages with the HTTP API is that it can be tested with any web browser.

For example, the flow sensor can be read in the following way:

1. Make sure that you have the Bluefors control card device added and connected to either real or simulated hardware. This works even without one, but it makes more sense to get the actual reading out. Navigate to the **Maintenance** tab.
2. Select the **API** tab.
3. Make sure that the HTTP/WebSocket port is 49099. Enable the API.
4. Enable the HTTP and WebSocket.

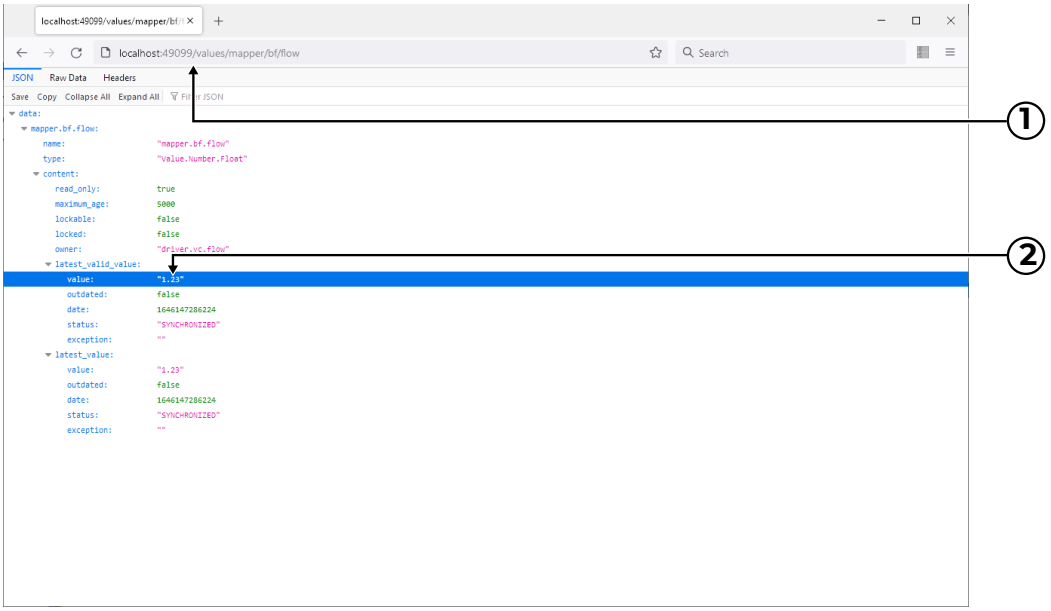


Item	Description
1	**Maintenance** tab
2	**API** tab
3	Enable API
4	Enable HTTP and WebSocket

Figure 48: Enabling the API and HTTP/WebSocket

5. Open a web browser and navigate to the following URL: http://localhost:49099/values/mapper/bf/flow.
6. The browser should now show a JSON response containing the flow with some additional data:

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**



Item	Description
1	API URL
2	Current value of the flowmeter

Figure 49: Accessing flowmeter value in Control API by using a web browser

The only limitation is that web browsers normally only allow the getting of the pages, using the *GET* command. However, the standard also defines the POST, *PUT*, and *DELETE* commands, which are also used by the API[10].

6.3.2 HTTP protocol

The protocol description in here has been simplified to contain only the relevant details that matter when using a library to communicate with the program. For example, the packets may also contain cookies, headers, etc., and the content may be split into multiple packets, which are ignored by the Control API or are transparent to the user.

The communication is request-response based, which means that you send a request, and the server sends a response. The actual data transferred is the request content, while the response content is the path in the URL and the query parameters defined in the URL. The protocol uses four different commands: **GET**, **POST**, **PUT**, and **DELETE**. Each of these commands has a specific purpose, and

[10] POST is used by web browser when sending data to server, e.g., filling a form in a web page or uploading a file, but it cannot be directly accessed without additional tools or extensions.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

while these are used differently in different APIs, this program follows the standard convention. The command type also defines what data it accepts, e.g., the **GET** command only has content in response while **POST** and **PUT** allow content in both request and response.

- **GET**: Returns the resource or data, e.g., static resource or value in value tree
- **POST**: Updates the data (this is only supported on data that can be changed, e.g., writeable values)
- **PUT**: Adds data (this is currently not used, but may be used e.g., for adding a device)
- **DELETE**: Deletes data (like PUT, this is currently not used).

With all endpoints, the endpoint path is used for identifying the resource in the endpoint, for example, a specific value, or a branch with multiple values in the value tree.

The query parameters contain additional information, e.g., the API key for identification or formatting the data to be returned.

Requests contain content only with **POST** and **PUT** commands. The request content for the **POST** command contains an updated version of the data. The format is the same as what was received with the **GET** command. **PUT** works similarly, but it can be used for adding new data, for example device nodes to a device list.

The response content always contains the data being requested or updated. If the data is being updated, the response will contain the data immediately after the operation execution has been initiated, but it is good to note that the data may have not been updated yet.

6.4 WebSocket protocol

WebSocket allows bidirectional connection between the client and the server and has been designed to enable bidirectional communication with web applications. The WebSocket protocol is defined in RFC 6455[11].

Control Software also provides WebSocket support. The HTTP API can be used for requesting the data, i.e., asking whether new data is available. To overcome this limitation, the API also supports the WebSocket protocol.

While HTTP is request-based, WebSocket establishes a continuous bidirectional stream. Once a connection is established, it is possible to send and receive packets asynchronously.

[11] https://tools.ietf.org/html/rfc6455

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

As with HTTP API, it is possible to read and set values, but also subscribe to listening changes in values. If the data is being listened to, the server will asynchronously send updates to the client when the data changes.

The listened values are unique to each connection, so it is possible to establish multiple connections for different purposes. Once the connection terminates, all connection specific listeners will be cleared.

This section defines the communication scheme shared by all commands. Command-specific details can be found in Appendix I.

6.4.1 Initiating the connection

As defined above in Section 6.1.2, WebSocket services are accessed via paths that start with **ws** following the endpoint name.

If an access key is needed, it can be specified as a query parameter **key**.

For example, *wss://localhost:49099/ws/values/?key=00000000-1111-2222-3333-444444444444* would refer to Secure WebSocket endpoint *values* with an API key for authentication.

If access is denied, an HTTP response with the code 503 is returned.

6.4.2 Communication scheme

The WebSocket connection is a bidirectional pipe where the client or server may send data at any time. The communication is divided into packets. Each data packet[12] is a complete JSON object.

Whenever the server sends data, the client should receive data until it has received a full JSON object. The same applies for packets sent by the client. Whenever the client sends a packet, it should send the data as a full JSON packet, which is considered to be complete when the whole package has been sent.

While the server and client can both send packets asynchronously at any time, all the packets follow certain rules to keep the traffic under control.

The messages sent by the client are always control commands and they can be sent at any time. The packets sent by the server are either responses to commands initiated by the client or asynchronous messages that may or may not be related to earlier commands.

[12] A packet in this context refers to a complete piece of data, i.e. JSON object, sent through the connection. WebSocket tra-c is also divided into packets, which may or may not follow the same structure, but this protocol level splitting of messages should not be confused to this.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

6.4.2.1 Data flow

The normal data flow in the endpoint follows the request-response pattern. Whenever the client sends a command, the server will respond. The response is in two parts. For all commands, the server first sends a response with the status **RECEIVED** to notify that it has received the command and started processing it. If the command is processed successfully, the server will send a response with status **SUCCEEDED** containing the data.

Packets sent by the server that are related to a specific command that was sent are bound to it by an ID. When the server sends an initial response to a command, it contains an ID, which is automatically generated and remains constant over the whole communication. Depending on the command, the subsequent asynchronous notifications that are being sent as a consequence may or may not contain the ID. The user may also supply an ID that will be used instead of a generated one. The use of an ID enables multiple operations to be in progress simultaneously.

In addition, the server may send asynchronous events with the status **NOTIFICATION**. They may be sent by the server at any time, e.g., due to unexpected general conditions that need to be notified about, or bound to a specific command, e.g., starting listening changes in a certain value. All events bound to a command use the same ID as the command that they are related to. If no command is involved, the ID does not exist.

> NOTE: Some endpoints may deviate from the standard dataflow. The deviations are described in the corresponding sections of the endpoint reference.

At any part of the command execution process, or in response to a general error condition, an error message can be sent as a response to a command, or asynchronously. The error packets are identified by the status **ERROR**. The command-related errors will have the command-specific ID except if parsing the ID from the initial failed packet. If an error message is received, there will be no further messages related to the command. For example, if **ERROR** is received instead of **RECEIVED**, the command is terminated, and the server will not send any further **RECEIVED** or **SUCCEEDED** responses in the same command chain with the same ID.

However, it is guaranteed that the server will respond with **RECEIVED** and eventually **SUCCEEDED** or **ERROR** in place of either of them.

6.4.2.2 Packet structure

All packets are JSON objects with the same basic structure.

There are four types of packets: commands, success responses, error responses, and asynchronous events.

Commands

Each command may contain up to three JSON elements:

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

- **id**: This is a unique string identifier for binding all elements of a transaction together. This is optional. If not specified, it will be generated. It must be composed from hexadecimal numbers and may contain hyphen (-) and underscore (_) characters, e.g., 0123-4567_89ab-cdef.
- **command**: This is a command name that defines the action to take, e.g., *set*, *read* or *listen*.
- **data**: This is a command-specific payload. Also, this is a JSON object, which has a command-specific structure.

Success responses

The server responds to successful commands with one or more response packets, which have the following JSON elements:

- **id**: This is a unique string identifier for binding elements of a transaction together.
- **status**: A status indicates the state and condition of the command execution. The successful responses have either a **RECEIVED** or **SUCCEEDED** status.
- **data**: This is a JSON object containing the actual packet payload, which is specific to a command.

Error responses

If some part of the command fails, an error will be returned instead. Instead of supplying the data, an error code will be returned.

- **id**: This is a unique string identifier for binding elements of a transaction together. If the error is a response to a general condition, the ID does not exist.
- **status**: The status indicates the state and condition of the command execution. This is always **ERROR** for error messages.
- **code**: This is a numeric code indicating the error type.
- **description**: This is a human-readable explanation of the error code.
- **details**: The details are in the form of a JSON object containing additional command-specific data, e.g., the offending payload.

Asynchronous events

Asynchronous events can be sent by the server at any time.

- **id**: This is a unique string identifier for binding elements of a transaction together. If the notification is a response to a general condition rather than a specific command, the ID does not exist.
- **event**: This is a command-specific type of event describing the purpose, e.g., value update or disconnect.
- **status**: The status indicates the type of the message, which, for asynchronous events, is **NOTIFICATION**.
- **data**: This is a JSON object containing the actual packet payload, which is specific to the event-type notification and the command it is related to.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

6.5 Authentication and security

The server provides lightweight security mechanisms to provide access control for the server.

The server has different ports for unencrypted and secured connections. By default, port 49099 is used for unencrypted connections and 49098 is used for secure connections.

The unencrypted port is only accessible from the same computer to make it easier to use the API from other programs. However, it is inherently insecure to use it with external services. The encrypted port is open for external communication.

Both connections may or may not use the access keys for authentication.

The program has been designed to be used in an internal network and protected by a firewall, but it supports and enforces certain security practices to reduce the attach surface. That may seem like overkill but following good security practices is strongly encouraged because there is always a risk of misconfiguration or security breach even in a properly set up environment.

6.5.1 Access keys

The server uses access keys to grant access to the server. The API configuration includes a list of generated access keys and a list of endpoints. The keys can be configured to grant or deny access to each endpoint and operation type separately, e.g., by only reading values.

The key list also has a special key called **<unauthenticated>** that can be used for controlling access without a key. Only unencrypted connections allow unauthenticated access.

A secure connection does not permit unauthenticated connections and requires a key as a parameter for both HTTP and WebSocket connections.

A WebSocket connection is continuous, and the operations are executed after connection has been established. As with HTTP the key is supplied as a query parameter for WebSocket connections. If the key has been defined and permission for the corresponding endpoint exists, access is granted. The key permissions are also checked when a command is being executed. If the key, or permission from the key, is removed after establishing the connection, the connection will remain open, but all subsequent commands will be denied.

6.5.2 Design considerations

When designing services that may be accessible from the internet, it is important not to store the keys in an external client-side application as it would compromise the whole security scheme.

It is possible to use the API directly, e.g., from a single-page web browser application. However, if the browser communicates directly with the API, the key is stored in the application itself and transmitted to the end user's machine, which

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

compromises the security scheme. In such cases, it is recommended to use a proxy server, which contains the actual key and handles the communication with both the Control Software instances and end users. The proxy server should provide its own session management and access control for end users.

7 Important information

7.1 Intended use

WARNING

Only use the software for the intended purpose it has been designed for.

7.2 Safety

WARNING

The Control Software interfaces directly with the operation of the dilution refrigerator. You are responsible for the correct performance of any operation. Always read and allow the instructions, safety information, and warnings stated in the system user manual. In correct action or operation may cause critical malfunction or system breakdown.

7.3 Custom applications

IMPORTANT

The information contained within this publication covers a wide range of applications and may not specifically apply to your equipment layout or custom setup.

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

Contact us directly (support@bluefors.com) if you have any question about the specifications or any other content contained within this publication.

7.4 Customer service

For support, contact support@bluefors.com.

7.5 Disclaimer

OPERATION OF THE SOFTWARE AND ITS FEATURES IN CONJUNCTION WITH ANY THIRD-PARTY DEVICE IS PROVIDED “AS-IS” WITHOUT WARRANTY OF ANY KIND (EITHER EXPRESSED OR IMPLIED) AND DONE AT YOUR OWN DISCRETION AND RISK WITH THE AGREEMENT THAT YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGE, LOSS OR ERROR (INCLUDING LOSS OF DATA) THAT MAY RESULT FROM SUCH USE. IN NO EVENT SHALL BLUEFORS BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY DIRECT OR INDIRECT DAMAGE OR LOSS RESULTING FROM SUCH ACTIVITIES WHETHER IN CONTRACT, TORT (INCLUDING, WITHOUT LIMITATION, NEGLIGENCE), OR OTHER THEORY AND REGARDLESS OF WHETHER BLUEFORS WAS ADVISED OF OR WAS AWARE OF THE POSSIBILITY OF SUCH DAMAGES OR LOSSES.

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

Appendix I: API Reference

This appendix contains descriptions of the endpoints provided by the Control API and a list of error codes the API may return.

HTTP endpoints

system

Description http://address:49099/**system**/endpoint/path/?param1=value1&parm2=value2

System endpoint provides general information about the system such as system name and version. System endpoint supports only the GET operation.

Endpoint path http://address:49099/system/**endpoint/path**/?param1=value1&parm2=value2

Not used

Parameters http://address:49099/system/endpoint/path/?**param1=value1&parm2=value2**

- **key**: API key for authentication
- **prettyprint**: If set to 1, output will be indented and split to multiple lines to improve readability.

Supported operations

- **GET**: Retrieves the data

Response structure

- *data* (Object): JSON object containing the data
  - *system_name* (String): Name of the system
  - *system_version* (String): Core software version
  - *api_version* (String): Version string similar to system version

Example 1 Requests system information in pretty printed format.

- **Request type**: GET
- **Request URL**: http://localhost:49099/system/?prettyprint=1
- **Request content**: <None>
- **Response content returned by the server**:

```
{
  "data": {
    "system_name": "Test system",
    "system_version": "1.4",
    "api_version": "1.4"
  }
}
```

values

Description http://address:49099/**values**/endpoint/path/?param1=value1&parm2=value2

Values endpoint provides access to the value tree. This endpoint can be used for retrieving any node or branch from the value tree including the node contents.

The endpoint supports GET operation for reading data and POST operation for updating the data and calling functions in value tree. Data being sent in POST

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

values

is structured similarly with GET operation with flat style, but content only specifies data being updated.

Endpoint path http://address:49099/values/**endpoint/path**/?param1=value1&parm2=value2

The endpoint path defines the target value tree node or branch to access. If request affects multiple nodes, the paths in the request are relative to the path defined in endpoint path.

With POST command, the path specified in the URL will be ignored as full path must be always specified in the content.
It is also worth noting that the nodes listed in POST operation are not guaranteed to be executed in the same order. To guarantee ordering, split the operation to multiple requests.

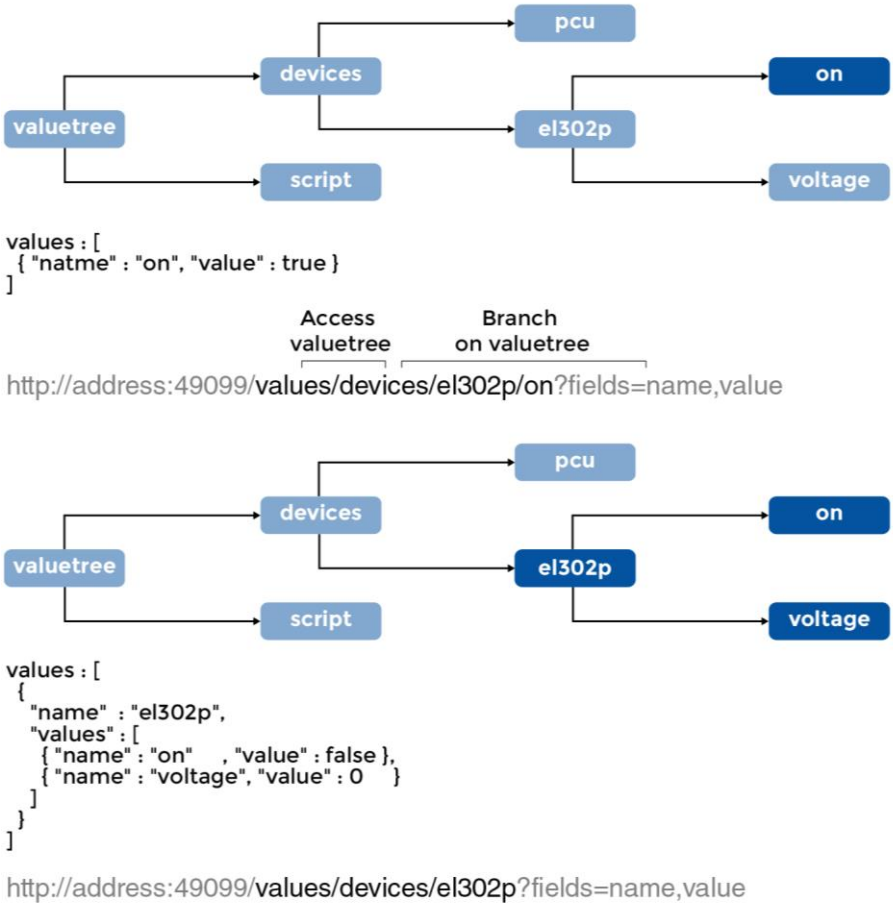


Figure 50: Endpoint path relation to value tree path.

Parameters http://address:49099/values/endpoint/path/**?param1=value1&parm2=value2**

- **fields**: Semicolon separated list of field names to include in the response. This can be used for filtering out unnecessary information. By default, everything will be included.
- **key**: API Key used for authentication
- **prettyprint**: If set to 1, output will be indented and split to multiple lines to improve readability.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

values

- **recursion**: Sets recursion depth. Defines how many levels of child nodes to include in output. -1 is unlimited (default), (only GET).
- **style**: Defines the response structure to be flat (default) or tree. Flat returns all nodes in a single flat list while in tree representation each node object has its own list of children. (only GET)
- **must_exist**: If 1, existence of all nodes and their contents will be checked before executing any commands. Default is 0 (only POST)
- **wait_response**: If 1, wait for commands to be completed before returning response. If not set, this returns response immediately after starting the operations, so returned value nodes may or may not be updated. Default is 1 (only POST)

Supported operations

- **GET**: Retrieves a value tree node or branch
- **POST**: Updates one or more value nodes in value tree or executes methods

Response structure when style=tree

- *data* (Object): Response root node
  - *name* (String): Name of the current node
  - *type* (String): Type of the content
  - *children (Object): JSON object containing the children as named objects*
  - *content* (Object): Node content. This is specific to node type

Response structure when style=flat

- *data* (Object): Response root node
  - *<path.to.node1>* (Object): Tree node 1
    - *name* (String): Name of the node
    - *type* (String): Type of the content
    - *content* (Object): Node content. This is specific to the node type.
  - *<path.to.node2> (*Object): Tree node 2
  - ...
  - *<path.to.nodeN>* (Object): Tree node N

Content structure for value

- *latest_valid_value* (Object): Latest acquired value that had status of being valid and is not too old
  - *value* (String): Value as a string
  - *outdated* (Boolean): Is value more recent than its maximum age
  - *date* (Integer): Timestamp of the sample as Unix timestamp
  - *exception* (String): May contain related Java exception information if the sample is invalid
- *latest_value* (Object): Latest acquired sample. Structure is similar to latest valid value
- *maximum_age* (Integer): Maximum age in milliseconds. Used for determining whether the latest valid sample is still valid
- *lockable* (Boolean): If true, the value can be locked for exclusive access
- *read_only* (Boolean): If set to true the value cannot be written
- *owner* (String): Component that is responsible for updating the value inside the Control software
- *value* (String): Present only in POST operation. Used for setting value for value nodes.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

values

Content structure for call

- *parameters* (Array): List of parameters that the function call expects. With POST operation this contains the values of parameters.
- *description* (String): Short description about purpose of the call
- *call* (Integer): Present only in POST operation. Used for calling the function, if the content type is a call.

Example 1 Requests flowmeter value in pretty printed format.

- **Request type**: GET
- **Request URL**: http://localhost:49099/values/mapper/bf/flow?prettyprint=1
- **Request content**: <None>
- **Response content returned by the server**:


```
{
  "data": {
    "name": "mapper.bf.flow",
    "type": "Value.Number.Float",
    "content": {
      "read_only": true,
      "maximum_age": 5000,
      "lockable": false,
      "locked": true,
      "owner": "driver.vc.flow",
      "latest_valid_value": {
        "value": "1.23",
        "outdated": false,
        "date": 1631106116076,
        "status": "SYNCHRONIZED",
        "exception": ""
      },
      "latest_value": {
        "value": "1.23",
        "outdated": false,
        "date": 1631106116076,
        "status": "SYNCHRONIZED",
        "exception": ""
      }
    }
  }
}
```


Example 2 Opens valves V1 and closes V2. The output is also in pretty printed format and only fields *name*, *value* and *status* will be outputted to simplify the output.

- **Request type**: POST
- **Request URL**: http://localhost:49099/values/?prettyprint=1&fields=name;value;status
- **Request content**:


```
{
        "data" : {
             "mapper.bf.valves.v1" : {
                    "content" : {
                           "value" : "1"
                    }
             },
                "mapper.bf.valves.v2" : {
                    "content" : {
                           "value" : "0"
                    }
             }
        }
}
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

values

- **Response content returned by the server:**


```
{
        "data": {
             "mapper.bf.valves.v1": {
                    "name": "mapper.bf.valves.v1",
                    "content": {
                           "latest_valid_value": {
                                  "value": "1",
                                  "status": "SYNCHRONIZED"
                           },
                           "latest_value": {
                                  "value": "1",
                                  "status": "SYNCHRONIZED"
                           }
                    }
             },
             "mapper.bf.valves.v2": {
                    "name": "mapper.bf.valves.v2",
                    "content": {
                           "latest_valid_value": {
                                  "value": "0",
                                  "status": "SYNCHRONIZED"
                           },
                           "latest_value": {
                                  "value": "0",
                                  "status": "SYNCHRONIZED"
                           }
                    }
             }
        }
}
```


Example 3 Runs the script and tries to access non-existent value without enforcing the existence, so all operations will be executed but error response with all data will be returned.

- **Request type**: POST
- **Request URL**: http://localhost:49099/values/?prettyprint=1
- **Request content**:

```
"data" : {
        "does.not.exist" : {
             "content" : {
                    "value" : 1
             }
        },
        "script.legacy.run" : {
             "content" : {
                    "call" : 1
             }
        }
}
```

- **Response content returned by the server:**


```
"error": {
    "code": 10013,
    "name": "Data error",
    "description": "May be caused by incorrect data or request caused an exceptio...
    "query": "POST /values/mapper/bf/valves/",
    "query_data": "{\"data\": {\"script.legacy.run\": {\"content\": {\"call\” : 1...
    "details": {
      "script.legacy.run": {
        "name": "script.legacy.run",
        "type": "Method",
        "content": {
          "parameters": [],
          "description": "Runs the script",
          "return": null
        }
      },
      "does.not.exist": {
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

values

```
        "error": {
          "id": null,
          "status": "ERROR",
          "code": 3001,
          "description": "Value not found"
        }
      }
    }
  }
}
```

resources

Description http://address:49099/**resources**/endpoint/path/?param1=value1&parm2=value2

This endpoint can be used for retrieving static resources used by the Control Software, e.g., user interface layout and other assets.

Endpoint path http://address:49099/resources/**endpoint/path**/?param1=value1&parm2=value2

The endpoint path is the path to the requested resource.

Parameters http://address:49099/resources/endpoint/path/**?param1=value1&parm2=value2**

- **key**: API key used for authentication

Supported operations

- **GET**: Retrieves a resource from given path

Response structure

- The requested file will be returned as is.

Example 1 Requests *layout.xml*, which is the main layout file for the user interface.

- **Request type**: GET
- **Request URL**: http://localhost:49099/resources/layout.xml
- **Request content**: <None>
- **Response content returned by the server**:

```
<ui>
  <uimodule order="1" include="frontpanel.xml" />
  <uimodule order="2" include="plots.xml" />
  <uimodule order="4" include="script_editor.xml"/>
  <uimodule order="5" include="separator.xml"/>
  <uimodule order="6" include="status.xml"/>
  <uimodule order="7" include="bfbridge.xml"/>
  <uimodule order="9" include="lakeshore.xml"/>
  <uimodule order="10" include="e1302p.xml"/>
  <uimodule order="11" include="fse.xml"/>
  <uimodule order="12" include="separator.xml"/>
  <uimodule order="15" include="config.xml" />
</ui>
```

WebSocket endpoints

ws/system – System information

System endpoint can be used for querying general information about the software.

System endpoint supports only the **read** command.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

read

Description
Returns general information about the system.

The server will first send a RECEIVED notification as an acknowledgement of receiving the message, following a SUCCEEDED message that contains the payload.

Command data
- *<none>*

Response data
- *id*(): Id of message that started the operation
- *Status*(): Status of operation
- *data*(): Inner JSON object that contains endpoint specific fields
- *system_name*(): Name given to system. Given by user
- *system_version*(): Version of the main system. Given in format X.Y.Z
    - X is major version: Changes in functionality
    - Y is minor version: Functionality is same but there are minor changes that may break something
    - Z is patch number: Bugs were fixed but it should be fully compatible (except potential changes due to fixed buggy functionalities)
- *api_version*(): Similar to **system_version** but reflects the part of the system providing API.

Notifications
Not used

Example 1 Connecting to a secure port in local host with specified API key and using custom id for command
- **URL**: wss://localhost:49098/ws/system/?key=0352ebaa-6de5-4f1d-9091-17678d11dfd6
- **Communication**
    - **sent**

```
{
        "command" : "read",
        "id" : "2b64707c-17b0-11ec-827e-14dae904baea"
}
```

    - **received**

```
{
        "id" : "2b64707c-17b0-11ec-827e-14dae904baea",
        "status" : "RECEIVED",
        "data" : {
             "command" : "read",
             "id" : "2b64707c-17b0-11ec-827e-14dae904baea"
        }
}
```

    - **received**

```
{
        "id" : "2b64707c-17b0-11ec-827e-14dae904baea",
        "status" : "SUCCEEDED",
        "data" : {
             "system_name" : "",
             "system_version" : "1.4",
             "api_version" : "1.4"
        }
}
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

ws/values – Values

Values endpoint provides access to the data in value tree.

The value tree may contain values and function calls as a content. This endpoint provides same functionalities as its HTTP API counterpart; reading and writing values and calling functions. As WebSocket provides a bidirectional channel between the core and the client, this endpoint also supports listening changes in value data.

The endpoint provides following commands:

- **read**: Read data from value tree node(s).
- **set**: Updates values and call methods in the value tree
- **listen**: Listens one or more value tree nodes for changes
- **unlisten**: Stops listening one or more value tree nodes for changes
- **status**: Returns list of values that are currently being listened for changes

read
Description Returns a single node from the value tree. The resulting data structure contains the value tree node and content.
Command data • *target* (String): Target node from the value tree structure which value will be read. The target may be a leaf node or branch node, but only the corresponding node without children will be returned. • *style* (String): Optional parameter that defines the output style. If the value is **tree** (default), the nodes are outputted as a tree structure. If the value is **flat** the nodes are returned as a JSON object containing all the values named by the node. • *recursion* (Integer): Specifies recursion depth when fetching the child nodes. 0 (default) refers to only the node specified by target, 1 refers to the target node and its immediate children, and so on. -1 is infinite.
Response data • Identical to HTTP values GET operation
Notifications None
Example 1 Retrieve the current flow value • **URL**: wss://localhost:49098/ws/wsvalues/?key=0352ebaa-6de5-4f1d-9091-17678d11dfd6 • **Communication** ○ **Sent** `{ "command" : "read", "id" : "6352827e-1ac3-11ec-bdf6-14dae904baea", "data" : { "target" : "mapper.bf.flow" } }` ○ **Received**

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

read


```
{
        "id" : "6352827e-1ac3-11ec-bdf6-14dae904baea",
        "status" : "RECEIVED",
        "data" : {
             "command" : "read",
             "id" : "6352827e-1ac3-11ec-bdf6-14dae904baea",
             "data" : {
                    "target" : "mapper.bf.flow"
             }
        }
}
```


- **Received**


```
{
        "id":"6352827e-1ac3-11ec-bdf6-14dae904baea",
        "status":"SUCCEEDED",
        "data": {
             "name" : "mapper.bf.flow",
             "type" : "Value.Number.Float",
             "content" : {
                    "read_only" : true,
                    "maximum_age" : 5000,
                    "lockable" : false,
                    "locked" : true,
                    "owner" : "driver.vc.flow",
                    "latest_valid_value" : {
                           "value" : "1.23",
                           "outdated" : false,
                           "date" : 1632218701084,
                           "status" : "SYNCHRONIZED",
                           "exception" : ""
                    },
                    "latest_value" : {
                           "value" : "1.23",
                           "outdated" : false,
                           "date" : 1632218701084,
                           "status" : "SYNCHRONIZED",
                           "exception" : ""
                    }
             }
        }
}
```


Example 2 Retrieve information about the script run command

- **URL**: wss://localhost:49098/ws/wsvalues/?key=0352ebaa-6de5-4f1d-9091-17678d11dfd6
- **Communication**
  - **Sent**


```
{
        "command" : "read",
        "id" : "acf0e246-1ac7-11ec-bdf6-14dae904baea",
        "data" : {
             "target": "script.legacy.run"
        }
}
```


  - **Received**


```
{
        "id" : "acf0e246-1ac7-11ec-bdf6-14dae904baea",
        "status" : "RECEIVED",
        "data" : {
             "command" : "read",
             "id" : "acf0e246-1ac7-11ec-bdf6-14dae904baea",
             "data" : {
                    "target" : "script.legacy.run"
             }
        }
}
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

read

- **Received**

```
{
        "id" : "acf0e246-1ac7-11ec-bdf6-14dae904baea",
        "status" : "SUCCEEDED",
        "data" : {
             "name" : "script.legacy.run",
             "type" : "Method",
             "content" : {
                    "parameters" : [
                    ],
                    "description" : "Runs the script"
             }
        }
}
```

set

Description
Set command can be used for updating a value in value tree. The structure of the data is identical to HTTP value endpoint POST operation.

Command data

- *must_exist* (Integer): If set to 1, checks existence of all nodes before executing any operations. Default is 0
- *data* (Object): List of tree nodes to set in flat format
  - *<path.to.node1>* (Object): Node 1 data
    - *content* (Object): The node content data
      - *value* (String): Updated value (only for values)
      - *call* (Integer): Must be set 1 to initiate the call (only for method calls)
      - *parameters* (Array): Array containing the parameter for the call (only for method calls)
  - *<path.to.node2>* (Object): Node 2 data
  - ...
  - *<path.to.nodeN>* (Object): Node N data

Response data

- Identical to read command with flat style and containing only the specified nodes.

Notifications
None

Example 1 Close valve V1 and open valve V2

- **URL**: wss://localhost:49098/ws/wsvalues/?key=0352ebaa-6de5-4f1d-9091-17678d11dfd6
- **Communication**
  - **Sent**

```
{
        "command": "set",
        "id": "4014ffe4-3729-11ec-974d-14dae904baea",
        "data": {
             "must_exist": 1,
             "data": {
                    "mapper.bf.valves.v1": {
                           "content": {
                                  "value": 0
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

set

```
                        }
                    },
                    "mapper.bf.valves.v2": {
                        "content": {
                            "value": 1
                        }
                    }
                }
            }
}
```

- **Received**

```
{
        "id" : "9ae4bbda-3729-11ec-974d-14dae904baea",
        "status" : "RECEIVED",
        "data" : {
            "command" : "set",
            "id" : "9ae4bbda-3729-11ec-974d-14dae904baea",
            "data" : {
                    "must_exist" : 1,
                    "data" : {
                        "mapper.bf.valves.v1" : {
                            "content" : {
                                "value" : 0
                            }
                        },
                        "mapper.bf.valves.v2" : {
                            "content" : {
                                "value" : 1
                            }
                        }
                    }
            }
        }
}
```

- **Received**

```
{
        "id" : "9ae4bbda-3729-11ec-974d-14dae904baea",
        "status" : "SUCCEEDED",
        "data" : {
            "mapper.bf.valves.v1" : {
                    "name" : "mapper.bf.valves.v1",
                    "type" : "Value.Number.Integer.Enumeration.onOffE...
                    "content" : {
                        "read_only" : false,
                        "maximum_age" : 5000,
                        "lockable" : true,
                        "locked" : true,
                        "owner" : "driver.vc.ch.c31",
                        "latest_valid_value" : {
                            "value" : "0",
                            "outdated" : false,
                            "date" : 1635341237559,
                            "status" : "SYNCHRONIZED",
                            "exception" : ""
                        },
                        "latest_value" : {
                            "value" : "0",
                            "outdated" : false,
                            "date" : 1635341237559,
                            "status" : "SYNCHRONIZED",
                            "exception" : ""
                        }
                    }
            },
            "mapper.bf.valves.v2" : {
                    "name" : "mapper.bf.valves.v2",
                    "type" : "Value.Number.Integer.Enumeration.onOffE...
                    "content" : {
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

set


```
                        "read_only" : false,
                        "maximum_age" : 5000,
                        "lockable" : true,
                        "locked" : true,
                        "owner" : "driver.vc.ch.c21",
                        "latest_valid_value" : {
                                "value" : "1",
                                "outdated" : false,
                                "date" : 1635341237760,
                                "status" : "SYNCHRONIZED",
                                "exception" : ""
                        },
                        "latest_value" : {
                                "value" : "1",
                                "outdated" : false,
                                "date" : 1635341237760,
                                "status" : "SYNCHRONIZED",
                                "exception" : ""
                        }
                }
        }
}
```


listen

Description
Start listening changes in one or more values.

Command data

- *target* (String): Target node in the value tree
- *all* (Boolean): If true, send notification from all updates, e.g., same value with updated timestamp, and if false, send updates only when the actual value or its status changes.
- *recursive* (Boolean): If true, recursively listens all child nodes. Default: false

Response data

- Response is similar to the status command with only values that were added by this command

Notifications

changed

description: Event caused by listening value. Generated each time the value or its status is updated.

data:

- Similar to status data content with only the value that has been changed

Example 1 First start listening all changes in flow, then all valves by only listening actual value changes

- **URL**: wss://localhost:49098/ws/wsvalues/?key=0352ebaa-6de5-4f1d-9091-17678d11dfd6
- **Communication**
  - **Placeholder**

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

listen

- **Sent:**

```
{
        "command" : "listen",
        "id" : "be4b7d90-3bc8-11ec-84e8-14dae904baea",
        "data": {
                "target" : "mapper.bf.flow",
        }
}
```

- **Received:**

```
{
        "id" : "be4b7d90-3bc8-11ec-84e8-14dae904baea",
        "status" : "RECEIVED",
        "data" : {
                "command" : "listen",
                "id" : "be4b7d90-3bc8-11ec-84e8-14dae904baea",
                "data" : {
                        "target" : "mapper.bf.flow",
                }
        }
}
```

- **Received:**

```
{
        "id" : "be4b7d90-3bc8-11ec-84e8-14dae904baea",
        "status" : "SUCCEEDED",
        "data" : {
                "mapper.bf.flow" : {
                        "status" : "OK",
                        "only_changes" : false,
                        "id" : "be4b7d90-3bc8-11ec-84e8-14dae904baea",
                        "content" : {
                                "read_only" : true,
                                "maximum_age" : 5000,
                                "lockable" : false,
                                "locked" : false,
                                "owner" : "driver.vc.flow",
                                "latest_valid_value" : {
                                        "value" : "1.23",
                                        "outdated" : false,
                                        "date" : 1635849386887,
                                        "status" : "SYNCHRONIZED",
                                        "exception" : ""
                                },
                                "latest_value" : {
                                        "value" : "1.23",
                                        "outdated" : false,
                                        "date" : 1635849386887,
                                        "status" : "SYNCHRONIZED",
                                        "exception" : ""
                                }
                        }
                }
        }
}
```

- **Received:**

```
{
        "id" : "be4b7d90-3bc8-11ec-84e8-14dae904baea",
        "status" : "NOTIFICATION",
        "event":"changed",
        "data" : {
                "mapper.bf.flow" : {
                        "name" : "mapper.bf.flow",
                        "type" : "Value.Number.Float",
                        "content" : {
```


Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

listen


```
                                "read_only" : true,
                                "maximum_age" : 5000,
                                "lockable" : false,
                                "locked" : false,
                                "owner" : "driver.vc.flow",
                                "latest_valid_value" : {
                                        "value" : "1.23",
                                        "outdated" : false,
                                        "date" : 1635849391886,
                                        "status" : "SYNCHRONIZED",
                                        "exception" : ""
                                },
                                "latest_value" : {
                                        "value" : "1.23",
                                        "outdated" : false,
                                        "date" : 1635849391886,
                                        "status" : "SYNCHRONIZED",
                                        "exception" : ""
                                }
                        }
                }
        }
}
```


unlisten

Description
Stops listening changes in one or more values.
If only target is specified, this applies to all nodes that matches the target. If only id is specified, this matches to all targets that were associated to given id.

Command data

- *target* (String): Target node from the value tree structure. If not specified, this matches to all nodes matching the command id
- *recursive* (Boolean): If true, stops listening the current value and all child nodes in the tree
- *id* (String): Command id used for setting the listener. If not specified, this applies to all listeners matching the target

Response data

- Response is similar to status command with only values that were removed by this command

Notifications
None

status

Description
Returns list of all nodes that are being listened to

Command data

- *command* (String): Command specifies what operation is intended to start by message

Response data

- *path.to.node* (Object): First node being listened to

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

status
○ *status* (String): Reference status: OK: Value exists, UNLINKED: Target does not exist, WRONG_TYPE: The target node content is not a value. ○ *all* (Boolean): See parameter all for listen command ○ *id* (String): The unique identifier used for adding this listener. ○ *content* (Object): Identical to read command tree node contents • *path.to.second.listened.node* (Object): Second node being listened to • ...
Notifications None

Error codes

Table 5: Error codes

General errors
1001 – Parameter error One or more parameters in the request were incorrect or missing.
1002 – Invalid JSON JSON supplied in API request is not valid JSON.
1003 – Missing JSON The request does not contain the mandatory JSON content.
1005 – JSON not required The request contains JSON data, but the endpoint does not use it.
1008 – Path parameter not supported The specified path is not supported by this endpoint.
10012 – Method not supported The endpoint does not support the used method.
10013 – Data error Data supplied in the request is incorrect.
100111 – Unknown error The program reported an error with unknown error code.
Value errors
3001 – Value not found Value tree node not found.
3003 – Read only Value is read only.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

3005 – Value edit error Failed to update the value.
3006 – Wrong type The target value tree node content is unexpected type.
3007 – Target value has no content The target value tree node does not have any content.
3008 – Method call exception Calling method caused an exception.
6001 – Command not found WebSocket command not found.
Server errors
2001 – No key supplied Access key is needed for accessing this resource
2002 – Content does not exist Requested content does not exist
2003 – Access denied Supplied key has no rights to the content
2004 – Method type error HTTP method is not supported
2005 – Fatal WebSocket error WS connection encountered an unexpected exception. Connection will be closed.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Appendix II: Script language definition

Bluefors scripting language definition

The Bluefors ValveControl script language is a simple state-based language. It resembles very closely to many imperative programming languages and much of the functionality is common. However, the exact execution model is slightly different which may be confusing at first.

Execution model

The execution of the script is state based, which means that instead of plain sequential code execution, the script is divided into distinct states and only one state contents is executed at a time in a loop until the state is changed. The states are identified by unique integer numbers which can be arbitrarily chosen with one exception: all the code defined in state 0 is executed always like it would be part of every state. The initial state is 1.

Each state contains a list of statements which are executed in a loop when the state is active. The execution of the statements does not change the state of the program directly. Instead, the state contents are evaluated first and then applied to program at once. This also means that if multiple operations to same variables or other state values are applied, the last one will be the final outputted value. For example, the following code does only set v1 on and wait only 5 seconds before executing the next cycle.

```
START
ON v1
WAIT 7
OFF v1
WAIT 6
ON v1
WAIT 5
END
```

During each execution cycle, the following will be done:

- **Skip the execution if wait counter is active**: If wait counter is active skip the loop execution until the counter has reached zero.
- **Read current state**: Reads index of current state which is used for deciding which parts of script will be executed
- **Read system status**: The interpreter takes snapshot of all system variable values to be used in state execution
- **Evaluate the statements**: The interpreter evaluates the statements belonging to current state and state 0. The interpreter does not make any changes to system state at this point
- **Stop execution, if needed**: Stops the script execution in case of variable access error or because of encountered **ERROR** or **STOP** statement during evaluation.
- **Update program variables**: The interpreter updates the control card and numeric variables to the system based on final values of

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

corresponding variables calculated when evaluating the statements. At this point the values are actually written to the system.
- **Set wait counter value**: If the **WAIT** statement was encountered when evaluating the statements activate the wait counter
- **Set the next state of the state machine**: It **SET_STATE** statement was encountered when evaluating the script change the current state to the new state

Variables

The valve control software has two types of variables defined which can be both used as part of the expressions, but their values are set by using different commands. List of these can be found in Appendix III.

Control card boolean variables

The devices which are controlled via Bluefors relay control card, e.g., valves, pumps and heaters, can be turned on and off using commands **ON**, **OFF** and **OFF_ALL**. They can be used as part of an expression as well. In that case the values are considered to be 1 if corresponding channel is on and 0 if the channel is off.

Numeric status values

In addition, there is more numeric variables which can be used also as part of an expression and some of them can be set by using **SET** command.

Expressions

The expressions used in **SET** and **IF** are used for calculating values from mathematical expressions. The expressions follow similar rules as in many other commonly used programming languages.

It is possible to use both control card boolean values and numeric variables as part of the expressions.

The expressions follow the normal precedence rules which can be altered by using parentheses. Following operations are supported and listed in precedence order:

- **Comparison**: Comparison operators evaluates to 1 if values are **==** equal, **!=** not equal, **<=** less or equal, **>=** greater or equal, **<** less or **>** greater and 0 otherwise
- **Addition and subtraction**: **+** and **-** works just like their mathematical counterparts.
- **Multiplication, division and modulo**: ***, /** and **%** works just like their mathematical counterparts.

Commands

The script is constructed from several lines where each line may be either empty or contain one commands. A command is in this case defined as any non-empty

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

line which all have a purpose. Each command line starts with keyword and additional command specific parameters may follow, e.g., "*STATE 100 Start state*". It is allowed to have an arbitrary number of spaces between different elements in the line.

The commands can be divided into three groups: structural commands, statements, and comments.

Comments

C

```
C <comment text>
```

Comments are namely commands, but they serve no other purpose than providing information to the user. Comments start with keyword **C** and may be followed anything until next line. Comments can be placed anywhere in the script.

Structural commands

The structural commands defines the overall structure of the program. These are **START**, **END** and **STATE**.

START

```
START
```

The **START** command must be the first command in the code. Only comments are allowed before it. Statement commands placed right after **START** will be considered to belong to state 1.

END

```
END
```

The **End** command must be the last command in the code. Only comments are allowed after it.

STATE

```
STATE <numeric index> <state description message>
```

The **STATE** command defines startpoint of a new state definition with given id and optional description. Each statement command after STATE command and until **END** or next **STATE** command will be executed when the corresponding state is active. It is possible, although not recommended, to define multiple states with same id and contents of those states will be all executed when the state is active.

Statement commands

The statements define the actual functional characteristics of the program. First, all the statements inside one state are evaluated and once the changes to the

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

system are defined they will be all written to system at once at the end of the execution loop.

ON and OFF

```
ON <control card channel name>
OFF <control card channel name>
```

The **ON** and **OFF** statements turn on and off valves, pumps, heaters, etc. These commands only control the devices which are directly connected to the relay control card in the control cabinet.

OFF_ALL

```
OFF_ALL
```

The **OFF_ALL** statement switches off all the relay control card channels.

SET

```
SET <numeric variable name> = <expression>
```

The **SET** command sets the value of some numeric variable in the system. Naturally, this works only for variables which are writable.

STOP

```
STOP
```

The **STOP** command stops the script execution after current execution cycle.

ERROR

```
ERROR <error message>
```

The **ERROR** statement stops the script execution with specified error message after current execution cycle.

SET_STATE

```
SET_STATE <numeric index>
```

The **SET_STATE** statement sets the next state where the execution will be transferred to after this execution cycle.

WAIT

```
WAIT <integer number of seconds to wait>
```

The **WAIT** command sets time to wait in seconds until next execution cycle is started.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

IF ... ELSE ... ENDIF

```
IF <expression>
<statements executed if expression result was nonzero>
ELSE
<optional else statements which are executed if expression result was
zero>
ENDIF
```

The **IF-ELSE-ENDIF** is slightly different set of statements. It defines conditional execution rules.

If the expression defined after **IF** command is evaluated as nonzero, the statements until **ELSE** or **ENDIF** will be evaluated, otherwise skipped.

It is possible to define optional **ELSE** between **IF** and **ENDIF**. If **ELSE** is defined, the statements after **IF** will be executed if the **IF** statement expression result was zero and skipped if the result was nonzero.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Appendix III: Variable name

Variables

The system contains two kinds of variables: channel variables for switching on and off valves, pumps, etc., and numeric variables for measurements, status values, numeric controllable values, etc.

Channel variables

The channel variables represent the digital IO channels in the Bluefors control card and nothing else. In standard Bluefors system these are connected to valves, pumps, and heaters.

These channels have only two possible states: on and off. Their state can be read and written, although the read value always indicates the status of the electrical control signal rather than the actual status of the component e.g., valve. For example, if a valve is switched on, it appears open when read if the power is applied to it, but it is possible that the pneumatic valve itself has failed to open.

These can be set by using commands **ON**, **OFF** and **OF_ALL** in scripts or **on**, **off** and **switch** in telnet server.

These cannot be set by using set command in the scripts, but these can be used as part of an expression in the scripts, e.g., IF v1==1.

Name	Description
v1	Valve V1
v2	Valve V2
v3	Valve V3
v4	Valve V4
v5	Valve V5
v6	Valve V6
v7	Valve V7
v8	Valve V8
v9	Valve V9
v10	Valve V10
v11	Valve V11
v12	Valve V12
v13	Valve V13
v14	Valve V14
v15	Valve V15

BF1000-1234517327-71

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

BLUEFORS

Name	Description
v16	Valve V16
v17	Valve V17
v18	Valve V18
v19	Valve V19
v20	Valve V20
v21	Valve V21
v22	Valve V22[13]
v23	Valve V23[13]
scroll1	Scroll pump 1
scroll2	Scroll pump 2
turbo1	Turbo pump 1
turbo2	Turbo pump 2[13]
compressor	Compressor
hs-still	Still Heat Switch
hs-mc	Mixing Chamber Heat Switch
ext	Ext heater
pulsetube	Pulse Tube
heater	4K Heater

Numeric variables

In addition to the channel variables, there are also many numeric status variables which are all readable and some of them can be written as well.

Currently, these are accessible only from the scripts and only pressures *p1–p6* are accessible via telnet protocol by using dedicated commands.

Some of these exists in every system, e.g., flow, but many others are device dependent and exists only if corresponding device is present in the system.

These can be used as part of an expression in scripting language and writable ones can be written by using **SET** command in scripting language.

Name	Device	Description
flow	Standard	Flowmeter reading

[13] Only exists if second turbo has been installed.

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Name	Device	Description
time	Standard	Time since starting of the script
var1	Standard	User variable 1
var2	Standard	User variable 2
var3	Standard	User variable 3
var4	Standard	User variable 4
var5	Standard	User variable 5
var6	Standard	User variable 6
ctrl_pres_ok	Standard	Control pressure ok
ctrl_pres_err	Standard	Control pressure too low
ctrl_pres	Standard	Control pressure sensor present
p1	MaxiGauge	Pressure channel P1
p2	MaxiGauge	Pressure channel P2
p3	MaxiGauge	Pressure channel P3
p4	MaxiGauge	Pressure channel P4
p5	MaxiGauge	Pressure channel P5
p6	MaxiGauge	Pressure channel P6
p1_on	MaxiGauge	Pressure channel P1 on
p2_on	MaxiGauge	Pressure channel P2 on
p3_on	MaxiGauge	Pressure channel P3 on
p4_on	MaxiGauge	Pressure channel P4 on
p5_on	MaxiGauge	Pressure channel P5 on
p6_on	MaxiGauge	Pressure channel P6 on
tvpower1	Turbo-V	Output power
tvbearingtemp1	Turbo-V	Bearing temperature
tvcontrollertemp1	Turbo-V	Controller temperature
tvbodytemp1	Turbo-V	Body temperature
tvrot1	Turbo-V	Rotational frequency
tverr1	Turbo-V	Error status
tverrcode1	Turbo-V	Error code
tvstatuscode1	Turbo-V	Status
tvlife1	Turbo-V	Pump life
tvcommerr1 r	Turbo-V	Communication error
cptempwi	CP2800 Compressor	Input water temperature

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Name	Device	Description
cptempwo	CP2800 Compressor	Output water temperature
cptemph	CP2800 Compressor	Helium temperature
cptempo	CP2800 Compressor	Oil temperature
cpttime	CP2800 Compressor	Elapsed time
cperrorcode	CP2800 Compressor	Error code
cpavgl	CP2800 Compressor	Average low side pressure
cpavgh	CP2800 Compressor	Average high side pressure
cp2800commerr	CP2800	Communication error
nxdsf	nXDS	Rotational frequency
nxdspt	nXDS	Pump temperature
nxdsct	nXDS	Controller temperature
nxdst	nXDS	Run hours
nxdsbs	nXDS	Bearing service timer
nxdstrs	nXDS	Tip reseal service timer
nxdscommerr	nXDS	Communication error
cpastate	CPA series compressor	State
cparun	CPA series compressor	Running
cpawarn	CPA series compressor	Warning code
cpaerr	CPA series compressor	Error code
cpatempwi	CPA series compressor	Input water temperature
cpatempwo	CPA series compressor	Output water temperature
cpatempo	CPA series compressor	Oil temperature
cpatemph	CPA series compressor	Helium temperature
cpalp	CPA series compressor	Low side pressure
cpalpa	CPA series compressor	Average low side pressure
cpahp	CPA series compressor	High side pressure
cpahpa	CPA series compressor	Average high side pressure
cpadp	CPA series compressor	Delta pressure
cpacurrent	CPA series compressor	Current
cpahours	CPA series compressor	Running hours

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Name	Device	Description
cpascale	CPA series compressor	Scale
cpasn	CPA series compressor	Panel SN
cpamodel	CPA series compressor	Model
cpacommerr	CPA series compressor	Communication error
el302p_v	EL302P PSU	4K Heater output voltage
el302p_i	EL302P PSU	4K Heater maximum output current
el302p_on	EL302P PSU	PSU 4K Heater output on
el302p_vo	EL302P PSU	4K Heater actual voltage
el302p_io	EL302P PSU	el302p_io TTi EL302P PSU 4K Heater actual current
el302p_err	EL302P PSU	4K Heater error
pcu_pv	FSE	Pump Valve status
pcu_gv	FSE	FSE Gate Valve status
pcu_pos	FSE	Position
pcu_dst	FSE	Destination
pcu_warn	FSE	Warning code
pcu_probe_out	FSE	FSE Insert out
pcu_probe_in	FSE	FSE Insert in
pcu_torque_limit	FSE	Torque limited flag
pcu_destination_set	FSE	Destination set
pcu_motor_off	FSE	Motor o
pcu_rot_cw	FSE	Rotating CW
pru_rot_ccw	FSE	Rotating CCW
pcu_pos_compl	FSE	Position completed flag
pcu_going_home	FSE	Going home
pcu_no_ctrl_pressure	FSE	No control pressure
pcu_integrity_ok	FSE	Integrity test ok
pcu_maintenance_mode	FSE	In maintenance mode
pcu_probe_mounted	FSE	FSE mounted
pcu_vacuum t	FSE	Vacuum present
pcu_remote	FSE	In remote mode
ctr_pressure_ok	Standard	Control pressure ok

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**

Name	Device	Description
remote	Standard	Remote mode
option_heater	Standard	4K Heater option
option_turbo	Standard	Turbo option
sample	Temperature Control	MXC Heater (0=off)
still	Temperature Control	Still Heater (0=off)
t50k	Temperature Control	50K Flange temperature
t4k	Temperature Control	4K Flange temperature
tmagnet	Temperature Control	Magnet temperature
tstill	Temperature Control	Still Flange temperature
tmixing	Temperature Control	MXC Flange temperature
tfse	Temperature Control	FSE temperature

Bluefors Oy, Arinatie 10, 00370 Helsinki, Finland | VAT: FI 2183 2199
support@bluefors.com | +358 9 5617 4800 | **BLUEFORS.COM**