



Training

Training documentation

zenon Basic Training (zenon 7.60)



©2017 Ing. Punzenberger COPA-DATA GmbH

All rights reserved.

Distribution and/or reproduction of this document or parts thereof in any form are permitted solely with the written permission of the company COPA-DATA. Technical data is only used for product description and are not guaranteed qualities in the legal sense. Subject to change, technical or otherwise.

Contents

1. Welcome to the COPA-DATA Training	8
2. Introduction	9
3. Example setup	10
4. zenon product family	11
4.1 Special zenon editions	12
4.2 Main components of zenon	12
5. Licensing	13
6. Startup Tool	14
7. Topic: Visualization	15
7.1 A new project	15
7.1.1 Creating a new workspace	15
7.1.2 Creating a new project	17
7.1.3 Configuration of the project	18
7.2 Variables	20
7.2.1 Drivers	20
7.2.2 Data Types	23
7.2.3 Variables	27
7.3 Frames	31
7.3.1 Creation of a frame	31
7.4 Screens	33
7.4.1 Creation of a screen	33
7.5 Screen elements	34
7.5.1 Vector elements	34
7.5.2 Functions	37
7.5.3 Dynamic elements	39
7.5.4 Screen Functions	43
7.6 The tank	44
7.6.1 Adding the tank symbol	44

7.6.2	Adding the fill level display.....	44
7.6.3	Adding the inflows and outflows.....	45
7.6.4	Adding the pipelines.....	46
7.7	Language switch.....	47
7.7.1	Language table	47
7.7.2	Creation of a language file.....	47
7.7.3	Creation of key words	48
7.7.4	Using the key words	48
7.7.5	Activate language switching	49
7.8	Color switching.....	49
7.8.1	Creating a color palette	50
7.8.2	Creation of a color	50
7.8.3	Using the color palette	50
7.8.4	Activate color switching	51
7.9	Styles.....	52
7.9.1	Creation of a style.....	52
7.9.2	Using a style.....	53
7.10	Runtime.....	55
7.10.1	Start the Runtime	55
7.10.2	Tips and tricks for navigation in Runtime	55
7.10.3	Exit Runtime	56
7.11	Adapt and expand project.....	56
7.11.1	Reload project	56
7.11.2	Substitution of an element.....	58
7.11.3	Substitution of a screen.....	60
7.11.4	Script management	60
7.12	Questions about visualization.....	62
8.	Topic: Event Handling	64
8.1	General.....	64
8.2	Special screen types	64
8.3	Chronological Event List (CEL)	65
8.3.1	CEL screen	65
8.3.2	Screen switch - CEL.....	66
8.3.3	CEL in Runtime.....	70
8.3.4	Configuration of the CEL.....	72

8.4	Alarming	72
8.4.1	Defining alarms.....	72
8.5	Alarm Message List (AML).....	76
8.5.1	AML screen	76
8.5.2	Screen switching - AML	77
8.5.3	AML in Runtime	78
8.5.4	Configuration of the AML	79
8.6	Alarm Cause (Context List).....	80
8.6.1	Screen Context List	80
8.6.2	Screen switch - Context List.....	81
8.6.3	Context List in Runtime	82
8.6.4	Alarm causes in the AML	83
8.7	Questions on Event Handling.....	83
9.	Topic: Operation.....	84
9.1	User Administration	84
9.1.1	Types of login	84
9.1.2	Define users.....	85
9.1.3	Protecting functionality	86
9.1.4	Configuration of the user administration.....	88
9.1.5	Functions of the User Administration	88
9.1.6	Graphic display of blocked elements	89
9.1.7	Runtime changeable data.....	89
9.1.8	User administration in the Runtime	90
9.1.9	User Administration screen	92
9.1.10	Read back data that can be changed in Runtime	92
9.2	Recipes	93
9.2.1	Creating Recipes	93
9.2.2	Use of Recipes	94
9.2.3	Recipes in the Runtime.....	95
9.3	Questions about Operation	97
10.	Topic: Network.....	99
10.1	General.....	99
10.1.1	Requirements	100
10.1.2	Network topologies	101
10.2	Client-server network	101

10.2.1	Configuring the server	102
10.2.2	Transferring the Runtime files with Remote Transport	102
10.2.3	Configuring the clients.....	103
10.3	Client-server network with redundancy	107
10.3.1	Configuring a Standby Server	107
10.3.2	System variables	108
10.4	Multi-server network	108
10.4.1	Definition of the hierarchical structure in the Editor	109
10.4.2	Network topology.....	110
10.4.3	Special requirements for the integration project.....	111
10.5	Circular redundancy	116
10.6	Web functionalities	117
10.6.1	Web Server	117
10.6.2	HTML Web Engine	117
10.7	Questions about the Network.....	118
11.	Topic: Diagnosis.....	119
11.1	Information collection tool - SIC	119
11.2	Project analysis - CRL light	119
11.2.1	Use of variables and functions in the project.....	119
12.	zenon online test	121
12.1	Execution of the zenon online tests.....	122
12.1.1	Registration for the zenon online test.....	122
12.1.2	E-Mail with link to the zenon online test	123
12.1.3	Starting the zenon online test	124
12.2	Q&As about the zenon online test.....	125
13.	Further zenon training sessions.....	126
13.1	Online training	126
13.2	zenon Logic training	127
13.3	zenon Analyzer training	127
13.4	zenon VBA training.....	127
13.5	zenon VSTA training	127
13.6	zenon VBA/VSTA training.....	128
13.7	zenon design & usability training.....	128

13.8	zenon Historian training.....	128
13.9	zenon Network & security training	128
13.10	zenon Energy Edition training	129
13.11	zenon Pharma Edition training.....	129
13.12	zenon Batch Control training	129
14.	zenon individual training	129
15.	Glossary	130
15.1	Glossary for visualization	130
15.2	Event handling glossary	130
15.3	Glossary for operation	131
15.4	Network glossary	132
16.	Exercises	133
16.1	Exercise 1: Switch with its own text.....	133
16.2	Exercise 2: Counter	133
16.3	Exercise 3: Pump display	133
16.4	Exercise 5: Automatic language switching with user logged in	134
16.5	Exercise 6: Language switching with a button	135
16.6	Exercise 7: Close a screen yourself	135
16.7	Exercise 8: Monitoring of safety shut-off mats	135

1. Welcome to the COPA-DATA Training

GENERAL HELP

If you cannot find any information you require in this help chapter or can think of anything that you would like added, please send an email to documentation@copadata.com (<mailto:documentation@copadata.com>).

PROJECT SUPPORT

You can receive support for any real project you may have from our Support Team, who you can contact via email at support@copadata.com (<mailto:support@copadata.com>).

LICENSES AND MODULES

If you find that you need other modules or licenses, our staff will be happy to help you. Email sales@copadata.com (<mailto:sales@copadata.com>).

2. Introduction

Welcome and thank you for choosing zenon. We want to cover the following questions with you for the introduction to zenon:

- ▶ What is zenon and why is zenon the way it is?

In a nutshell: zenon is a tool that is used in order to be able to complete the work better and more efficiently.

- ▶ The follow-up question: How can I create a good project with this tool?

As with all other projects, whether I am building a house or buying a new car, it is always important to have a plan. An automation solution also needs an organized procedure and good planning.

- ▶ In which industry is zenon used?

zenon is used in many different industries, there are 4 core industries: Automotive, Energy & Infrastructure, Food & Beverage and Pharma.

- ▶ What can I use the demo project for?

The demo project can be used as a template and to provide ideas. It offers an initial basis, give many pieces of information and provides numerous ideas in relation to how you can approach a project depending on the industry. Furthermore, you can export data such as the color palette from the demo project and reuse it in your project.

- ▶ What scope does a zenon project have?

The scope of a zenon project is, among other things, influenced by the number of data points and screens, network compatibility and required modules. If the scope of a project is defined, it makes sense to issue structured names for drivers, variables, screens, functions etc. It also makes sense to consider the reuse of objects that have been saved centrally and locally.

- ▶ How can I design the project with a clear overview?

With the determination of the design (colors, fonts, operating concept and possibly screen splitting), we look at the question of who is to operate the project. Each target group has different requirements that need to be taken into account in order to achieve the best possible usability.

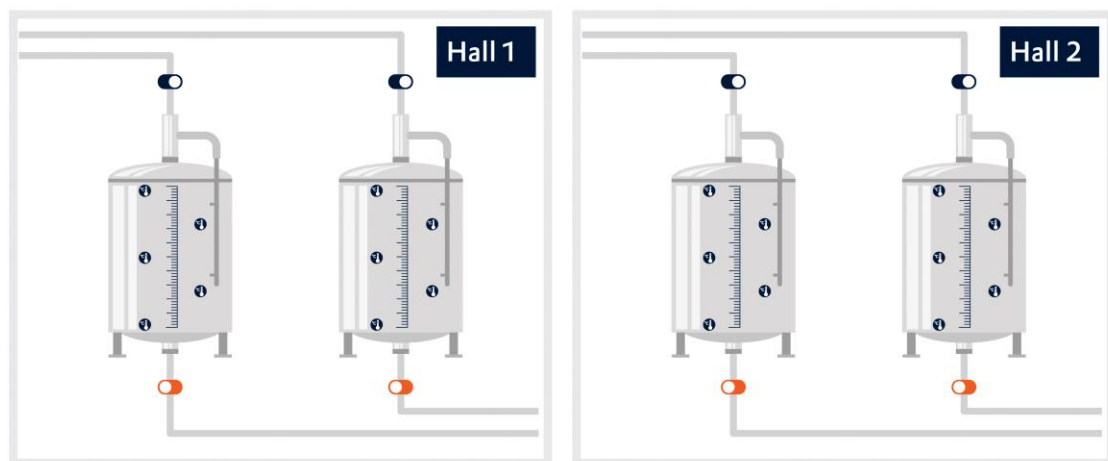
With this tutorial, you get to know the fundamental operation of zenon. You learn how to create a project and how to draw screens which display the values from your PLC in dynamic elements.

3. Example setup

During the training course , we will create an example project together with you; step by step we will add different zenon functions.

The example project simulates a tank facility that consists of two identical production halls, each with two tanks. The tanks are visualized with inflows and outflows and with the corresponding valves. With the help of a driver, production is simulated and this allows an example setup that changes.

This picture shows you an exemplary illustration:



We will simulate different temperatures for the different temperature sensors in the tanks (5 per tank). Visualize the tank fill level and the corresponding inflows and outflows with the help of a bar chart. The temperature sensors in the tanks are implemented with variables and given limit values and alarms. We then show these alarms – as well as system messages – in clear lists. We will create users with different rights and define recipes to set complete parameter sets. We will then analyze the network functionalities of zenon as well as the possibilities for remote maintenance.

4. zenon product family

<As automation software for HMI, SCADA and reporting, CD_PRODUCTNAME> provides a wide range of integrated functions for engineering and Runtime, in platform-independent form.

With zenon you can:

- ▶ Create, distribute, edit and execute automation projects
- ▶ Use integrated Soft PLC zenon Logic
- ▶ Collect and evaluate equipment-spanning data from a plethora of sources

The zenon product family consists of:

ZENON OPERATOR

The zenon Operator is a cost-effective product that is adapted to machine operation. The principle function conforms with that of the zenon Supervisor. However, compared to zenon Supervisor, zenon Operator is limited in terms of functionality.

zenon Operator consists of Editor and Runtime and can be used on all PC operating systems.

ZENON SUPERVISOR

The zenon Supervisor is a comprehensive tool for creating and executing automation projects. It consists of Editor and Runtime, permits the quick creation of automation networks - also via a web connection - and can be used on all current Windows operating systems.

ZENON LOGIC

zenon Logic is the programming environment integrated into zenon in accordance with IEC 61131. It is available as Editor and Runtime (Soft PLC) for zenon Supervisor and zenon Operator.

ZENON ANALYZER

zenon Analyzer offers, with cross-system analysis, a management-level and controlling-level view of all available online and offline data. It summarizes tasks which were separated until now and it eliminates friction losses which emerge due to parallel set-ups, maintenance and the coordination of different IT systems.

zenon Analyzer:

- ▶ condenses and accounts data from different sources in different formats
- ▶ compares them to one another
- ▶ display the results in graphical form as report

4.1 Special zenon editions

zenon is also available in two special editions: These editions are optimized for the respective application and contain special elements such as symbols, wizards, etc.

ENERGY EDITION:

The zenon Energy Edition is a package with special functionality for the energy sector and the procedural technology. The user benefits from easy-to-implement functions that allow for an individual adjustment of the application to the physical environment.

PHARMA EDITION:

The zenon Pharma Edition is an expansion of the standard scope with special features that are needed in the pharmaceutical industry most of all, such as Batch Control.

4.2 Main components of zenon

zenon consists of two main components, the zenon Editor and the zenon Runtime.

ZENON EDITOR:

Projects are created, configured and maintained with the zenon Editor. You can, for example, use symbols to design a display element for temperature values. The Editor also serves to create Runtime files.

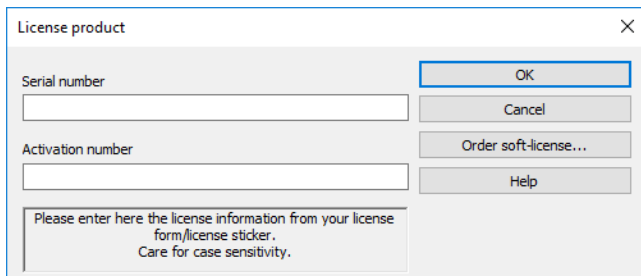
ZENON RUNTIME:

Runtime is the program in which projects are executed. The current temperature can be observed and controlled in Runtime for example.

5. Licensing

zenon Editor is your control center for simple and clear project configuration. All modules use the same data basis and can be easily activated at any time with licensing.

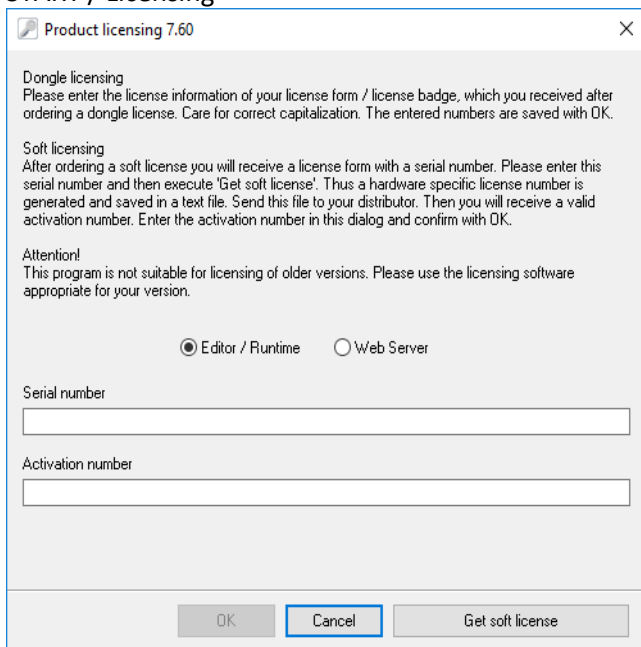
- Use the zenon Editor **File** menu and the **General configuration / License product ...** command.



The 'License product' dialog box contains the following elements:

- Serial number:** A text input field.
- Activation number:** A text input field.
- Buttons:** OK, Cancel, Order soft-license..., and Help.
- Instructions:** A text box stating: "Please enter here the license information from your license form/license sticker. Care for case sensitivity."

- You can also use the licensing program of the current version for the product licensing: Windows **START / Licensing**



The 'Product licensing 7.60' dialog box contains the following elements:

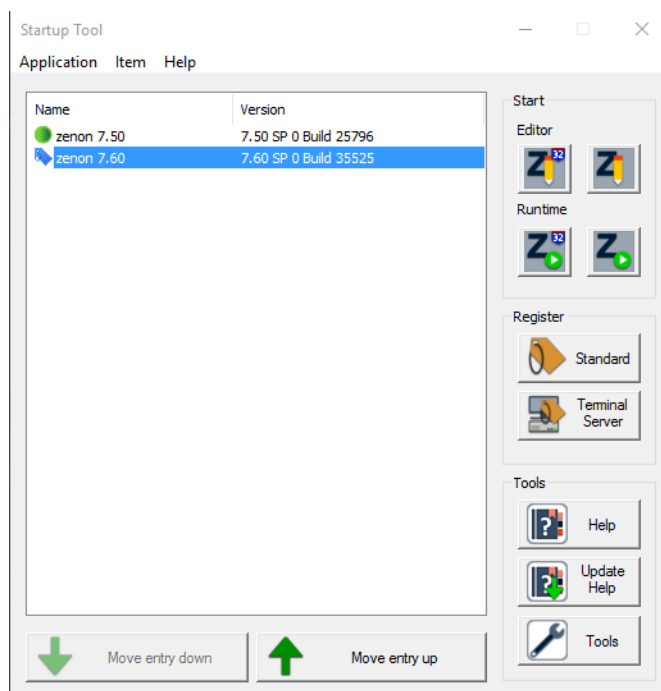
- License Types:**
 - Dongle licensing:** "Please enter the license information of your license form / license badge, which you received after ordering a dongle license. Care for correct capitalization. The entered numbers are saved with OK."
 - Soft licensing:** "After ordering a soft license you will receive a license form with a serial number. Please enter this serial number and then execute 'Get soft license'. Thus a hardware specific license number is generated and saved in a text file. Send this file to your distributor. Then you will receive a valid activation number. Enter the activation number in this dialog and confirm with OK."
- Attention!** "This program is not suitable for licensing of older versions. Please use the licensing software appropriate for your version."
- Mode Selection:**
 - ☒ Editor / Runtime
 - ☐ Web Server
- Serial number:** A text input field.
- Activation number:** A text input field.
- Buttons:** OK, Cancel, and Get soft license.

Product serial number and activation number are entered in the `zenon6.ini` file.

6. Startup Tool

The Startup Tool offers the possibility to administer the installed zenon versions. These can be listed there.

Settings for the network configuration can be adjusted via the Startup Tool. Various other programs can be started up here.



7. Topic: Visualization

Learning objectives:

- ▶ You get to know the Editor as a graphic user interface to create and maintain projects. In doing so, you will find out how to design your working environment in such a way that it is possible to work efficiently.
- ▶ You will be able to create workspaces in which you administer your projects. You will also get to know different types of projects and the how to centrally administer properties such as color palettes, fonts, etc.
- ▶ You will be able to establish communication to your hardware with the help of a driver. You will also find out that variables are the central interface to driver objects and data types, and how you can create your own different data types by yourself.
- ▶ You will create frames as the basis for your screens and get to know their benefits.
- ▶ You will be able to design screens with different elements and display the variable values in these. You will group symbols and reuse them effectively.
- ▶ You will be able to implement the operation of your project with the help of predefined functions.

7.1 A new project

In this section, you will learn what workspaces and projects are. We will create a workspace and then an example project.

Workspace:

A workspace serves as the directory of all projects in the Editor. Several projects can be saved and managed within one workspace.

Project:

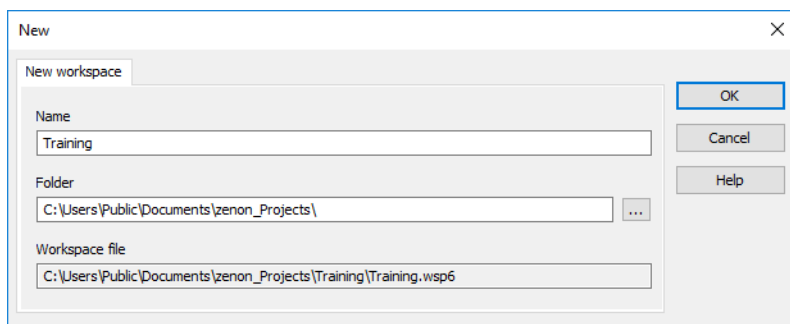
A project in the Editor contains all data and settings for the creation of the Runtime files. Projects must be assigned to a workspace.

7.1.1 Creating a new workspace

- ▶ In the File menu, use the **Workspace / New...** command.
- ▶ Give the name **Training** to the workspace.

If the workspace has not been created yet, after confirming the dialog box with **OK**, the `C:\Users\Public\Documents\zenOn_Projects\Training` directory is created and a file called `Training.WSP6` is created in it.

The directory name and the name of the workspace file are issued automatically by zenon, but they can be changed later while being created.



Hint: Video - the workspace in zenon

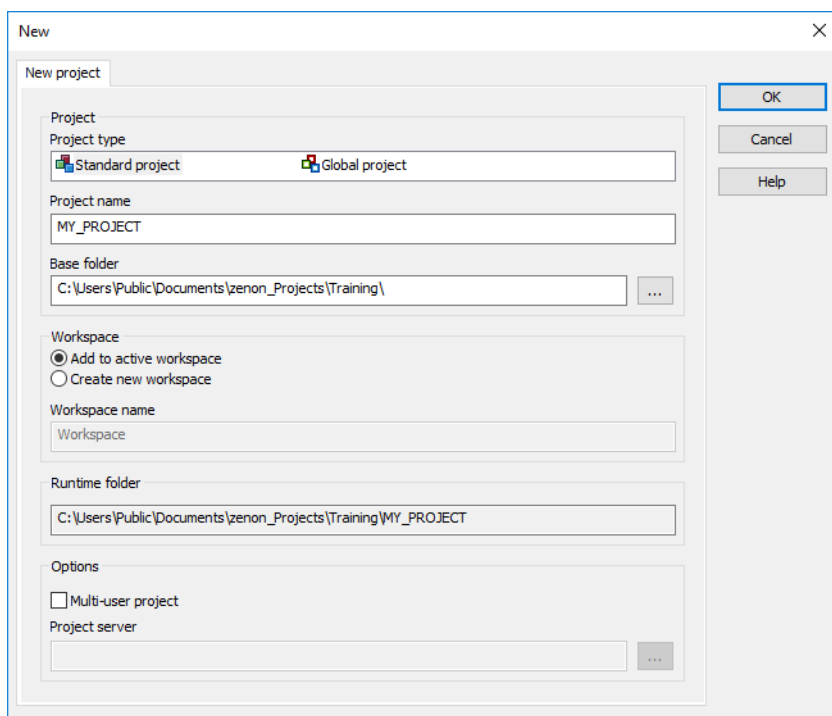
The video shows how a workspace in zenon is created.

Optional for the QR code: www.copadata.com/tutorials



7.1.2 Creating a new project

- In the **File** menu, select the **New project** command.
- Enter **MY_PROJECT** as a project name and accept the proposed path.



We use a **standard project** for our example.

A **global project** has only limited functionality. The objects (templates, fonts, colors etc.), that are created in a **global project** are also available in all other projects of the workspace. For instance, the frames of the **global project** in the other projects can be selected under the name 'g_name' in the other projects.

The **multi-user project** option makes it possible to create a project that can be edited by several people working on the project at the same time. <In doing so, CD_PRODUCTNAME> ensures that an object is not being edited by two people working on the project at the same time. We will not use this option in our project.

- Confirm the settings with **OK**.

This step takes a little bit longer than the creation of a workspaces, because there is now not just one individual file; all editor files and the SQL database of the project are created.

If, in your editor, VBA or VSTA is activated (default setting), then the **project wizard** is started automatically. Wizards are VBA or VSTA macros, with which you can automate the work in the Editor. In this example, we will not use a wizard, we will close the dialog with **Close**. We confirm the query in the next dialog with **No**.

Tip: Video - the project in zenon

The video shows how a project is created in zenon.

Optional for the QR code: www.copadata.com/tutorials



7.1.3 Configuration of the project

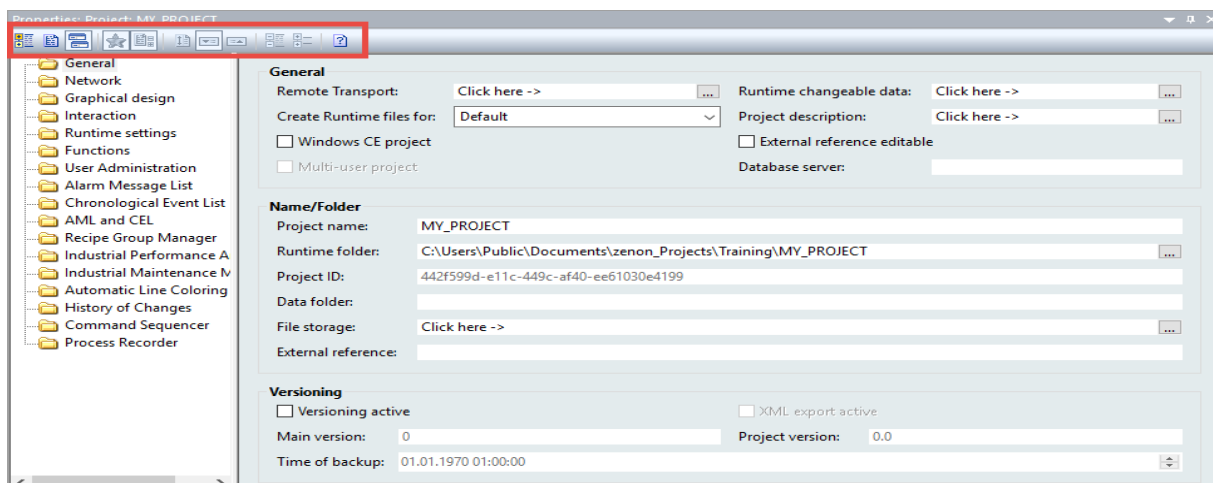
- In the **Project Manager**, select your **Training** workspace.

In the **detail view** you now see a list of the projects belonging to this workspace. In this case, it is only one project at the moment.

- In the **detail view**, select your **MY_PROJECT** project.

In the **properties window**, the properties of the **MY_PROJECT** project are now displayed and can be adapted.

We will use the **properties window** often. In principle, it shows the properties of the object selected in the **detail view** of the **project manager**. Icons on the top border of the **properties window** allow you to show the properties in different views:



In doing so, the individual icons offer the following functionality:

Parameter	Description
Grouped	The properties are combined into logical groups.
All properties	All properties are shown in a row.
Dialog view	The properties are displayed as dialog boxes.
Show/hide favorites	In the favorites, you can put together the most frequently used properties by using the context menu of the property window. Here, you can show or hide the favorites.
Show/hide all properties	If the favorites are shown, you can hide all other properties with this icon for a better overview.
Sorted logically	With this icon, the displayed properties are sorted according to their logical connectedness.
Sorted Ascending	With this icon, the displayed properties are sorted in alphabetically ascending order.
Sort descending	With this icon, the displayed properties are sorted in alphabetically descending order.
Expand all	By clicking on the '+' on the left border of the property window, you can open a closed node. This icon automatically expands all closed nodes.
Collapse all	By clicking on the '-' on the left border of the property window, you can close an expanded node. This icon automatically closes all expanded nodes.
Display properties help	In a separate window, a short description of the currently-selected property is displayed if you have the help displayed.

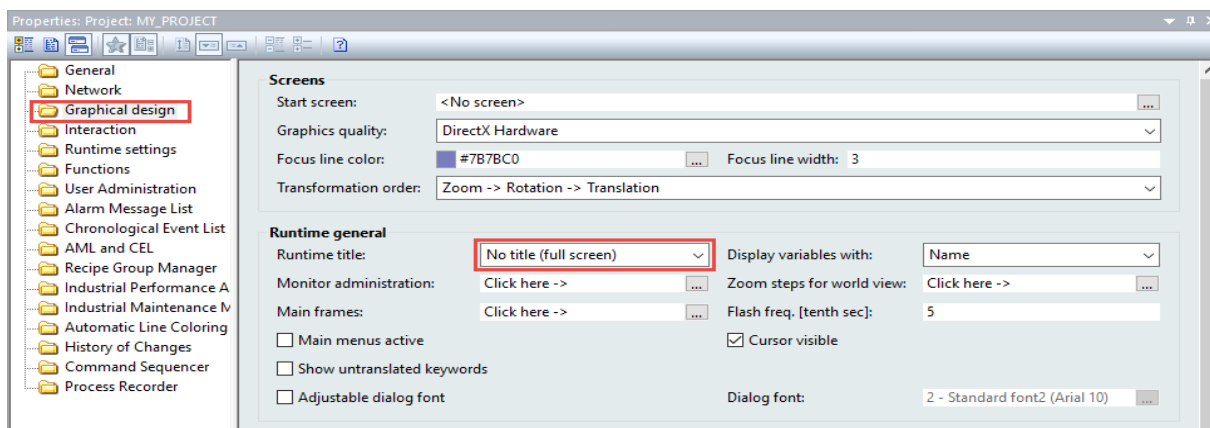


Information

You can add or remove frequently-used properties to or from the favorites with drag&drop or by using the context menu of a property. By doing so, you save the opening of the corresponding properties group for each change.

- ▶ Switch to the dialog view.
- ▶ Open the **General** group.
- ▶ Open the **Graphical design** group.

- Change the **Runtime title** property to No title (full screen).



This means the program window of the Runtime will be displayed without a title bar.

7.2 Variables

In this step, you will learn how zenon connects to a process and how it receives values from the PLC.

To do so, we will create a driver and define our own data types. We use the driver and the data types to define the variables.

7.2.1 Drivers

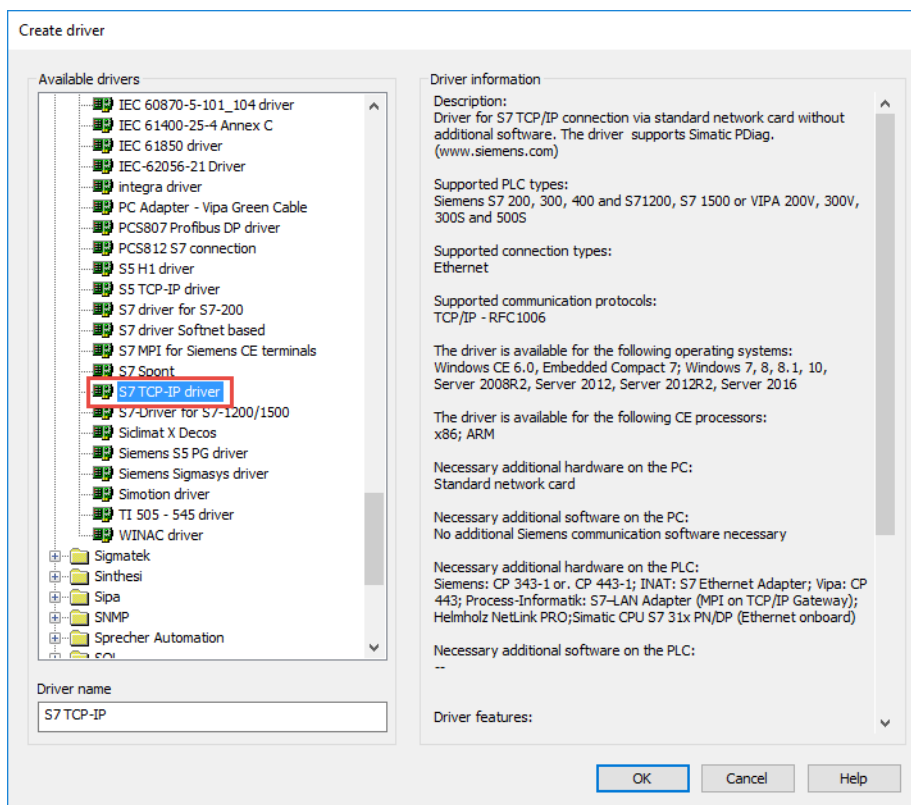
To communicate with a data source (PLC field bus, etc.) it is necessary to connect to a driver. Depending on the project requirements, the desired drivers must be defined (depending on the PLC, etc.) and their attendant process variables must be created. If necessary, several drivers, or even one driver, can be created more than once in a project.

- Open the **Variables** node in the **project manager**.
- Now select the **Driver** subnode.

As you can see, in our project there are already drivers created for **internal variables**, **mathematics variables** and **system variables**. These drivers are provided as standard, they do not need to be licensed and their variables are also not counted for the license.

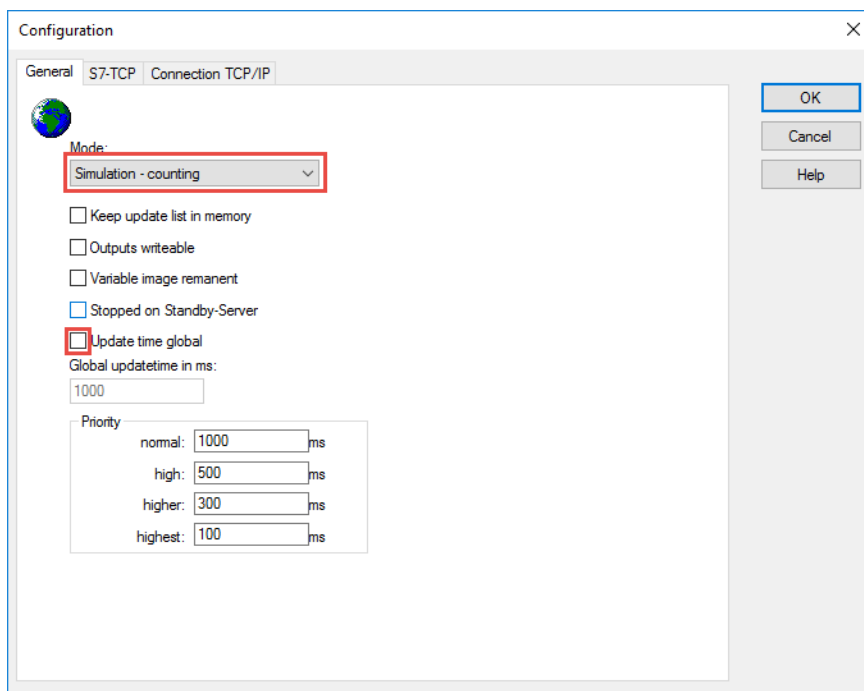
- Click on the **New driver...** icon in the top left of the detail view.

The dialog box to select a driver opens.



- In the selection for available drivers, scroll until the **Siemens** folder appears.
- Select the **S7 TCP-IP driver**. In **Driver name**, the entry is applied automatically as displayed in the **Definition of a driver...** dialog. Leave the dialog box by confirming with **OK**.

Now the dialog for the configuration of the driver opens.



- Set the mode to **Simulation (counting)**.

In **Hardware** mode, zenon would immediately attempt to establish a connection to the PLC when Runtime is started. However, because there is no PLC available at the moment, zenon would display all values as "invalid".

- Switch off **global update time**.

With this setting, we have the possibility to assign one of four different update times to each individual variable when defining variables. These four update times can be set in the lower part of this dialog, under **Priority**.

The other tabs of this dialog are driver-specific, i.e. different according to the selected driver. Because we do not have a PLC available at the moment, we can disregard the other settings in this example.



Information

*In the **driver documentation**, you can find more detailed information about the possible driver-specific settings and all driver-specific dialogs. The import of the process variables for the respective driver is also described in detail.*



Tip: Video - create a driver in zenon

The video shows how a driver is created in zenon.

Optional for the QR code: www.copadata.com/tutorials



7.2.2 Data Types

A data type is a variable template without connection to the process. In addition to the already pre-defined IEC data types, it is possible to define your own data types in zenon. Two possibilities are available here: Simple data types and structure data types. The advantage of self-defined data types is that there is the possibility to be able to make changes to them centrally. If a property of a data type is changed, this is automatically applied to all linked variables.

Variables are based on both data types and driver object types. The driver object types depend on the selected driver.

Data types are in principle independent from the driver. However, not all driver object types support all data types. With the corresponding selection, you are always only offered the data types that are also supported by the selected driver object type.

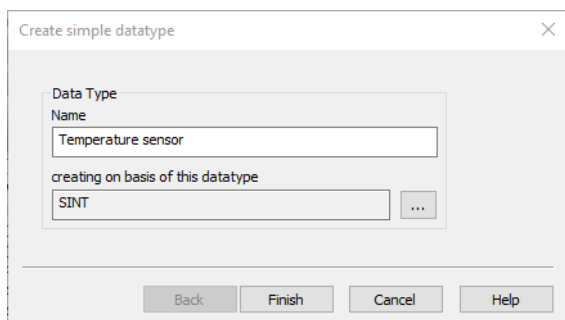
In this step, you will learn how to create simple data types and structure data types.

Creation of a simple data type

- ▶ Open the **Variables** node in the **project manager**.
- ▶ Now select the **Data Types** node.

In the **detail view** you can see that there is already a list of pre-defined data types.

- Click the **New simple datatype...** icon at the top left of the detail view.



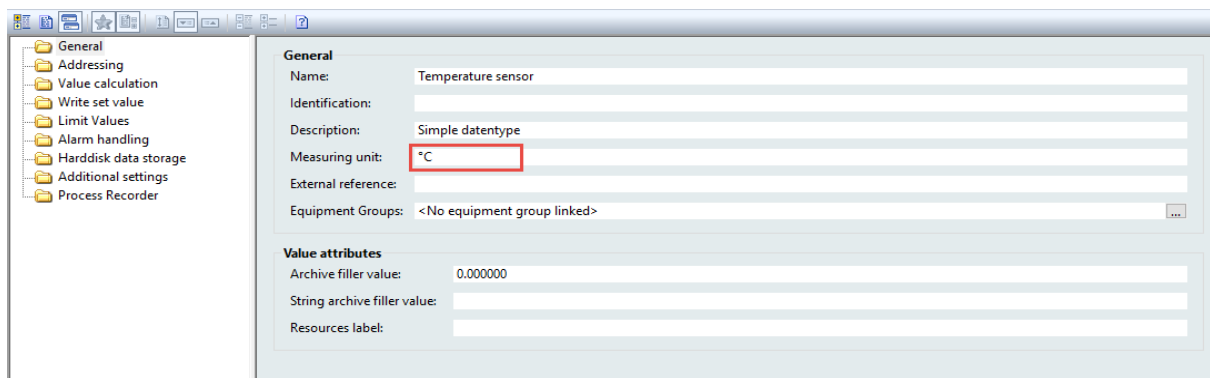
- Enter as name of the new simple data type **Temperature sensor**.
- Select **SINT** as the basic data type.

By clicking on **Finish**, the data type is created and will then be available in the list of data types.

- Select the **Temperature sensor** data type in the list.

In the **properties window**, the properties of the **Temperature sensor** data type are now shown and we can make changes.

- Select the **General** group.
- Change the **Measuring unit** property to °C.



- Select the **Value calculation** group.

The **Value range PLC** shows you one of the possible value ranges of the raw value from the controller here. You can define the scaling of the variables under **Value adjustment linear**.

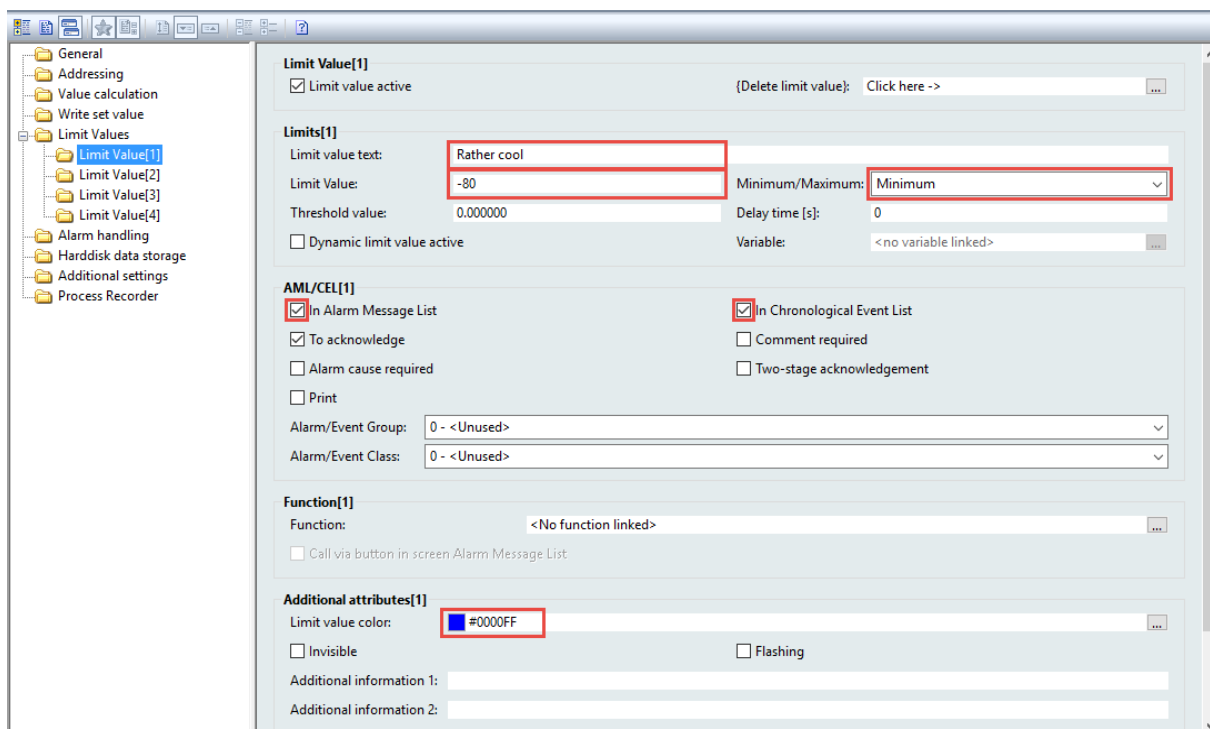
We now want to define limit values for the **Temperature sensor** data type.

- Select the **Limit values** group in the properties window.
- Click on the {new limit value} [...] button.

A new section with the name **Limit value[1]** is created.

- Select the **Limit value[1]** sub-group.

- Change the properties of the limit value as shown in the illustration.




Information

Create three further limit values for the 'temperature sensor' data type:

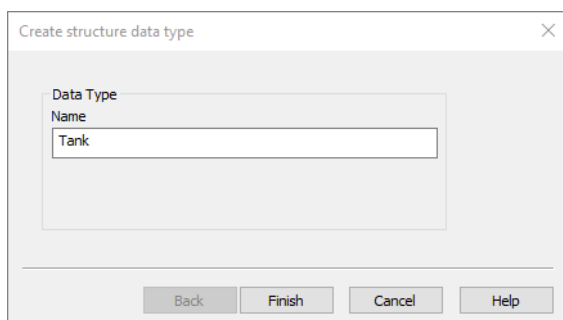
Limit value 2: -50; MIN; 'cool'; light blue; no alarm

Limit value 3: 50; MAX; 'warm'; light red; no alarm

Limit value 4: 80; MAX; 'very warm'; dark red; alarm

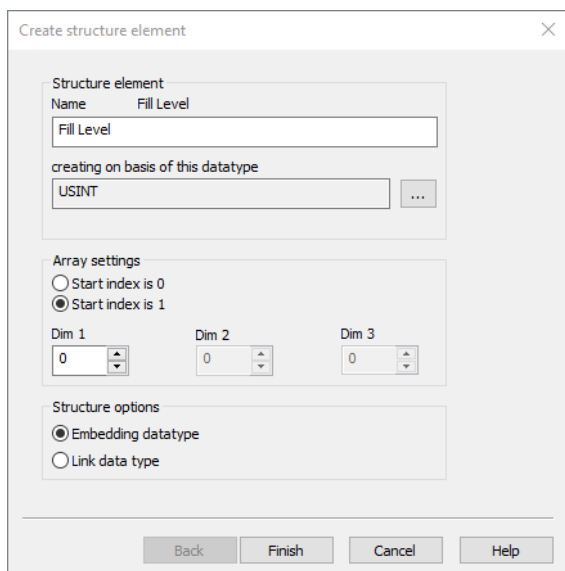
Creating a structure data type

- In the **project manager**, select the **Variables** node.
- Now select the **Data Types** node.
- Click on the second icon, **New structure data type...**, at the top left of the detail view.



- Enter **Tank status** as the name for the new structure data type.

After you have completed the structure data type, a dialog box will open, in which you can define the first structure element of this data type.



The dialog box titled "Create structure element" contains the following fields and options:

- Structure element Name:** A text box containing "Fill Level".
- Fill Level:** A text box containing "USINT".
- creating on basis of this datatype:** A text box containing "USINT" with a dropdown arrow.
- Array settings:**
 - ☐ Start index is 0
 - ☒ Start index is 1
 - Dim 1:** A spinner box set to 0.
 - Dim 2:** A spinner box set to 0.
 - Dim 3:** A spinner box set to 0.
- Structure options:**
 - ☒ Embedding datatype
 - ☐ Link data type
- Buttons:** Back, Finish, Cancel, Help.

- Enter **Fill level** as a name for the new structure element.
- Select **USINT** as the basic data type.
- The data type should be embedded.

If a basic data type is embedded in a structure data type, the properties of that data type can be changed in the structure element independently from the basic data type.

After **Finishing** the structure element, the structure data type is created in the list.

However, we want to more structure elements in this data type:

- With the right mouse button, open the context menu of the **Tank status** structure data type.
- Select the **New structure element...** menu entry.



Information

Create further structure elements for the **Tank status** structure data type:
 Structure element 2: Inflow; BOOL; embedded
 Structure element 3: Outflow; BOOL; embedded

7.2.3 Variables

A process variable is the interface between the data source (PLC, field bus, etc.) and zenon. For correct recording, control and regulation of processes, both the exchange of data and the input of setpoint values and commands are necessary. Process data and settings are defined for single process variables and the parameters are entered and changed in the variable list.



Information

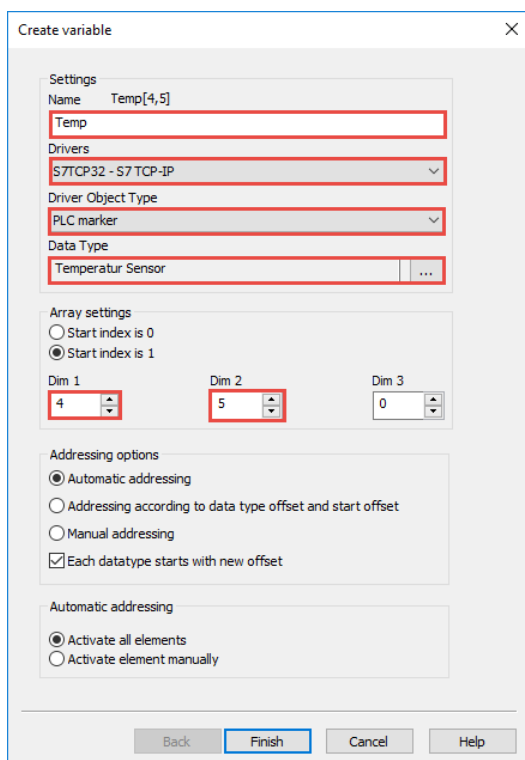
Many zenon drivers offer you the possibility to import variables from the PLC or from the controller directly, so that you do not need to create these variables manually. You will find the necessary information for the variable import in the according driver documentation.

Because we have neither a PLC nor an S7 project in this example project, we will create the variables manually.

Creation of a variable with a simple data type

- ▶ Select the Variables node in the **Project Manager**.
- ▶ Click on the **New variable...** icon in the top left of the detail view.

The dialog box for creating a variable opens.



The 'Create variable' dialog box is shown with the following settings:

- Settings**
 - Name: Temp[4,5]
 - Temp
 - Drivers: S7TCP32 - S7 TCP-IP
 - Driver Object Type: PLC marker
 - Data Type: Temperatur Sensor
- Array settings**
 - ☐ Start index is 0
 - ☒ Start index is 1
 - Dim 1: 4
 - Dim 2: 5
 - Dim 3: 0
- Addressing options**
 - ☒ Automatic addressing
 - ☐ Addressing according to data type offset and start offset
 - ☐ Manual addressing
 - ☒ Each datatype starts with new offset
- Automatic addressing**
 - ☒ Activate all elements
 - ☐ Activate element manually

Buttons at the bottom: Back, Finish, Cancel, Help.

- ▶ Enter **Temp** as a name.

As a **driver object type**, select PLC marker.



Attention

The **communication details** driver object type does not establish a connection to values in the controller, but provides information on the quality of the connection to the controller.

- ▶ Select **Temperature sensor** as a data type.

A simple variable is created with these settings. We can however create several variables of the same type by changing the corresponding settings in the array settings area.

Because we have 4 tanks with 5 temperature sensors in our example setup, we set these settings.

- ▶ Set **Dim 1** to the value 4.
- ▶ Set **Dim 2** to the value 5.
- ▶ Confirm the settings with **Finish**.

The variables are now added to the variable list in the detail window of the variable node. All properties of our **Temperature sensor** data type are applied to the variables. The properties of the selected variables can be checked and changed in the **properties window**.

Because we have changed the array dimension, twenty variables have been created at the same time. These variables are automatically addressed due to our settings, in this case we only need to issue one **Start Offset**. zenon calculates the other addresses automatically.

- ▶ Select the **Variables** node.
- ▶ Select the **Temp** entry in the detail view.
- ▶ In the **Addressing** property group, set the **Start offset** property to 11.

We thus achieve the following addressing:

Parameter	Description
Temp[1,1]	Offset 11
Temp[1,2]	Offset 12
Temp[1,3]	Offset 13
...	Offset n

The **last used offset** property shows, accordingly: 30/7 (Offset 30 / Bit 7).



Information

The **Start Offset** setting has no meaning in simulation mode, but you must be aware of it for a real PLC connection.

The variable list in the detail view of the **project manager** should now have the following entries.

Status	Name	Identification	Measur...	Net address	Data block	Offset	Bit num...	Drivers	Data Type	Start offset
	Filter text	Filter text	Filter...	Filter text	Filter text	Filter text	Filter...	Filter text	Filter text	Filter text
[-]	Temp			0	0		0	S7TCP32 - S7 TCP-IP	Temperature sensor	11
[-]	Temp[1,1]		*C	0	0	11	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[1,2]		*C	0	0	12	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[1,3]		*C	0	0	13	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[1,4]		*C	0	0	14	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[1,5]		*C	0	0	15	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[2,1]		*C	0	0	16	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[2,2]		*C	0	0	17	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[2,3]		*C	0	0	18	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[2,4]		*C	0	0	19	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[2,5]		*C	0	0	20	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[3,1]		*C	0	0	21	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[3,2]		*C	0	0	22	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[3,3]		*C	0	0	23	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[3,4]		*C	0	0	24	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[3,5]		*C	0	0	25	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[4,1]		*C	0	0	26	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[4,2]		*C	0	0	27	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[4,3]		*C	0	0	28	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[4,4]		*C	0	0	29	0	S7TCP32 - S7 TCP-IP	Temperature sensor	
[-]	Temp[4,5]		*C	0	0	30	0	S7TCP32 - S7 TCP-IP	Temperature sensor	

21 total / 21 filtered / 0 selected | 20 tags used / unlimited tags available



Information

Always give the process variables logical names!

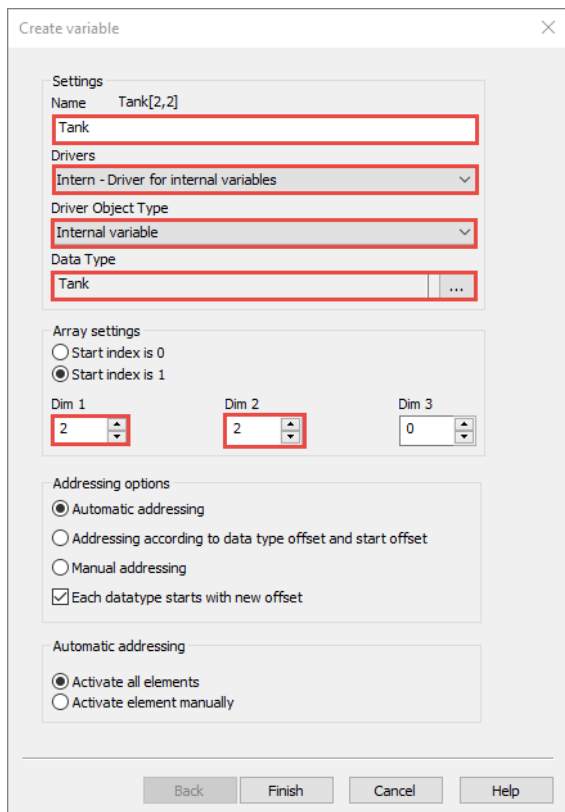
Creation of a variable with a structure data type

We now want to create other variables, but this time for a structure data type. To do this, proceed as follows.

- ▶ As a **name**, enter a **Tank**.
- ▶ Select **Tank status** as a **data type**.
- ▶ As a **driver**, select **Driver** Internal - driver for internal variables.
- ▶ As a **driver object type**, select **Internal** variable.
- ▶ Under **Array settings**, set **Dim 1** and **Dim 2** each to 2.

In contrast to hardware drivers, we do not need to think about the addressing with the internal driver. zenon automatically ensures clean addressing here.

Now the variables for four tanks are created, whereby the four tanks each consist of three variables.



The 'Create variable' dialog box is shown with the following settings:

- Name:** Tank[2,2]
- Tank:** Tank
- Drivers:** Intern - Driver for internal variables
- Driver Object Type:** Internal variable
- Data Type:** Tank
- Array settings:**
 - ☐ Start index is 0
 - ☒ Start index is 1
 - Dim 1:** 2
 - Dim 2:** 2
 - Dim 3:** 0
- Addressing options:**
 - ☒ Automatic addressing
 - ☐ Addressing according to data type offset and start offset
 - ☐ Manual addressing
 - ☒ Each datatype starts with new offset
- Automatic addressing:**
 - ☒ Activate all elements
 - ☐ Activate element manually

Buttons: Back, Finish, Cancel, Help

Your variable list in the detail area of the project manager should now look as follows:

Status	Name	Identification	Measur...	Net address	Data block	Offset	Bit num...	Drivers	Data Type	Start offset
+	Temp			0	0	0	0	S7TCP32 - S7 TCP-IP	Temperature sensor	11
-	Tank			0	0	0	0	Intern - Treiber für in...	Tank	0
-	Tank[1,1].Fill level			0	0	0	0	Intern - Treiber für in...	USINT/<Embedded> 1	
-	Tank[1,1].Supply			0	0	0	8	Intern - Treiber für in...	BOOL/<Embedded> 1	
-	Tank[1,1].Drain			0	0	0	9	Intern - Treiber für in...	BOOL/<Embedded> 2	
-	Tank[1,2].Fill level			0	0	0	0	Intern - Treiber für in...	USINT/<Embedded> 1	
-	Tank[1,2].Supply			0	0	0	8	Intern - Treiber für in...	BOOL/<Embedded> 1	
-	Tank[1,2].Drain			0	0	0	9	Intern - Treiber für in...	BOOL/<Embedded> 2	
-	Tank[2,1].Fill level			0	0	0	0	Intern - Treiber für in...	USINT/<Embedded> 1	
-	Tank[2,1].Supply			0	0	0	8	Intern - Treiber für in...	BOOL/<Embedded> 1	
-	Tank[2,1].Drain			0	0	0	9	Intern - Treiber für in...	BOOL/<Embedded> 2	
-	Tank[2,2].Fill level			0	0	0	0	Intern - Treiber für in...	USINT/<Embedded> 1	
-	Tank[2,2].Supply			0	0	0	8	Intern - Treiber für in...	BOOL/<Embedded> 1	
-	Tank[2,2].Drain			0	0	0	9	Intern - Treiber für in...	BOOL/<Embedded> 2	

34 total / 34 filtered / 1 selected | 20 tags used / unlimited tags available

7.3 Frames

In zenon there is a - not-directly imposed but on closer inspection extremely useful - concept to arrange screens on the monitor and to arrange them consistently - the concept of frames. A frame defines a screen area in which the assigned screens are called up.

If, for example, a logo and time are to be shown at the top edge in a uniform manner in the complete project, a **Header** frame is defined for this area. This subsequently defines the size and position of the header. At the lower edge of the screen, there is always a menu bar for example – a **Button bar** frame is thus created for this. Between this, there are the process screens; a **Process screen** frame is created for this. All screens consequently created then are assigned to one of these frames and opened therein in Runtime. This has the benefit that in each screen, the same frame is shown in the same place and in the same size.

If the screen distribution is to be changed because, for example, the header is to become a footer – no problem. Repositioning is sufficient for the three frames. The screens remain unaffected by this; they are automatically shown in the new position by zenon.

A further benefit of the frame concept is to be able to refer to screen areas functionally. I can, for example, close the screen that is currently called up as a menu without needing to know which screen it is in detail.

In addition, the user need not worry about closing the screens, because it is always only one screen of a frame that can be open. If another screen of the same frame is opened, the previous one is closed automatically.

So before we can create the first screen, we still need to define our frames.



Attention

*The frame that is created first is automatically used for each new screen. If possible, create the **process screen** as the first frame.*

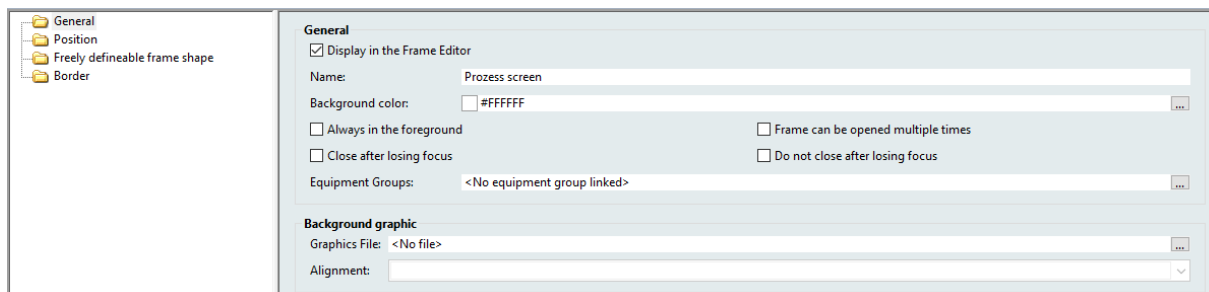
7.3.1 Creation of a frame

- ▶ In the **Project Manager**, open the **Screens** node.
- ▶ Select the **Frames** subnode.
- ▶ Click on the **New frame...** screen on the top left in the detail view.

A frame with the name **Frame 0** is created and shown in the **Frame editor**. You can change the properties of the frame in the **properties window**.

- ▶ The size of the frame is always set to the screen resolution currently set by default. We will change this so that there is a distance at the top and bottom. We need the upper space for alarm messaging and the lower space for a subsequent button bar.

- Give the frame the name **Process screen**.

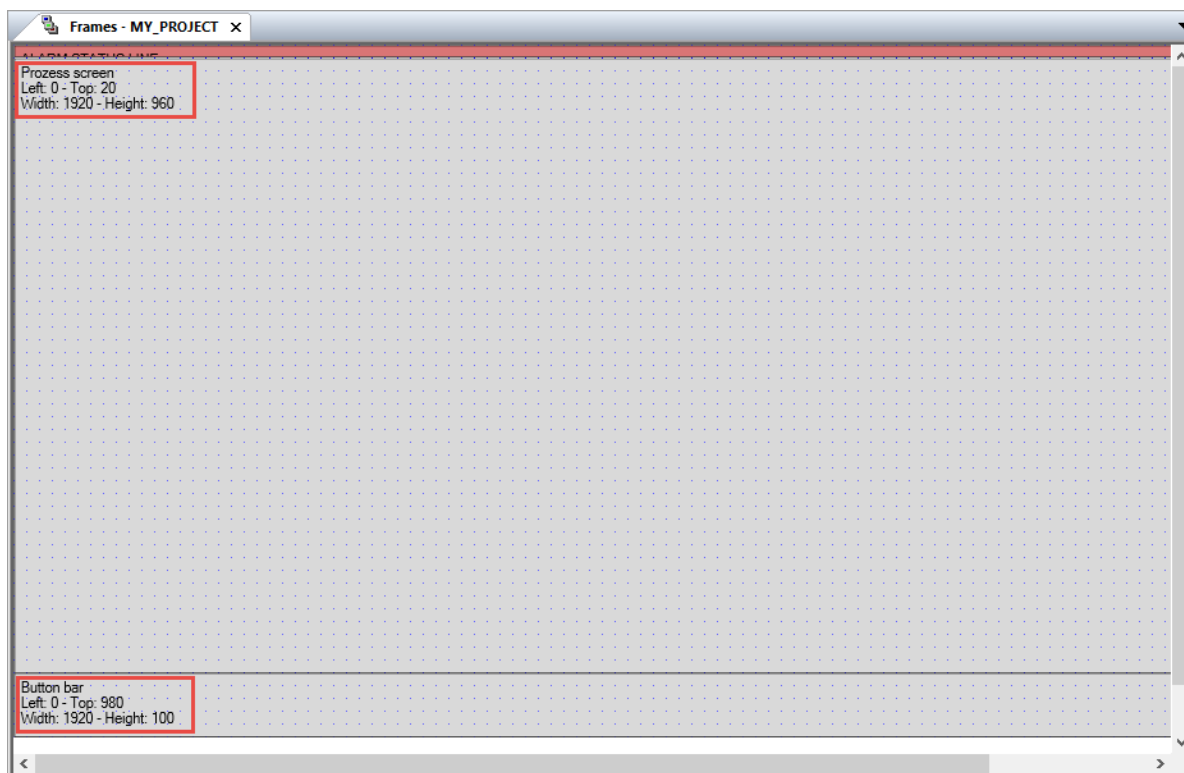


The screenshot shows the 'General' tab of the frame editor for a frame named 'Prozess screen'. The settings are as follows:

- General**
 - ☒ Display in the Frame Editor
 - Name: Prozess screen
 - Background color: #FFFFFF
 - ☐ Always in the foreground
 - ☐ Close after losing focus
 - Equipment Groups: <No equipment group linked>
 - ☐ Frame can be opened multiple times
 - ☐ Do not close after losing focus
- Background graphic**
 - Graphics File: <No file>
 - Alignment: (dropdown menu)

You can also change the size and position of the frame in the frame editor directly. The frame editor also offers the possibility to call up a new frame using the context menu.

- Create a second frame with the name **Button bar** for the button bar. The frame should be high enough to be able to draw the buttons.
- Match the size of the **Process screen** frame and the **Button bar** frame so that they do not overlap.
- This is how our frames should now look in the frame editor:





Hint: Video - frames in zenon

The video shows how frames are created in zenon.

Optional for the QR code: www.copadata.com/tutorials



7.4 Screens

A screen is a window with special predefined properties. Each screen must be assigned a frame.



Information

You can also create a screen without having created a frame beforehand. In this case however, zenon automatically creates a frame with the default settings in the background in the full screen size.

7.4.1 Creation of a screen

- ▶ In the **Project Manager**, select the **Screens** node.
- ▶ Click on the **New screen...** icon in the top left of the detail view.

A screen with the name **Screen 0** is automatically created. You can change the properties of the screen in the **properties window**.

- ▶ In the **General** group, name the screen **Start screen** and leave the **screen type** as **standard**.
- ▶ In the **Frame** property, ensure that the screen is assigned to the **Process screen** frame.

The screen is automatically opened in the screen editor during creation.



Information

Because this is the first screen that we have created, it is automatically used as a start screen for the Runtime. You can change the Runtime start screen in the project properties under **Graphical appearance - Start screen**.

- ▶ Create a screen with the name **Hall**. Use the **Process screen** frame for this screen.
- ▶ Create a screen with the name **Button bar**. Assign the **Button bar** frame to this screen.



Tip: Video - screens in zenon

The video shows how a screen is created in zenon.

Optional for the QR code: www.copadata.com/tutorials



7.5 Screen elements

Roughly speaking, there are two different types of elements that you can use in a screen:

Parameter	Description
Vector elements	The appearance of these elements in the Runtime always stays the same.
Dynamic elements	These elements change their appearance in the Runtime (usually depending on the value of a variable).

7.5.1 Vector elements

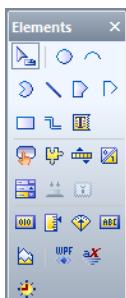
Vector elements are used for the graphic display of background information; their appearance in Runtime does not change.

Static text

We will now place some text in our start screen.

- Open the **Start screen**.

Open the **Elements** menu or use the **Elements** tool bar.



- ▶ Select the **Static text** element.
- ▶ In the **Start screen** screen, press the left mouse button on the desired location and hold it down while pulling up a rectangle.

You can change the properties of the screen in the **properties window**. You can change the properties of an element at any time; you only need to select the corresponding element in the screen – click on it with the left mouse. The properties are then available in the **properties window** again.

This procedure is the same for all elements.

- ▶ Now, in the **Text** group in the **Text** property, write `My first zenon project`.



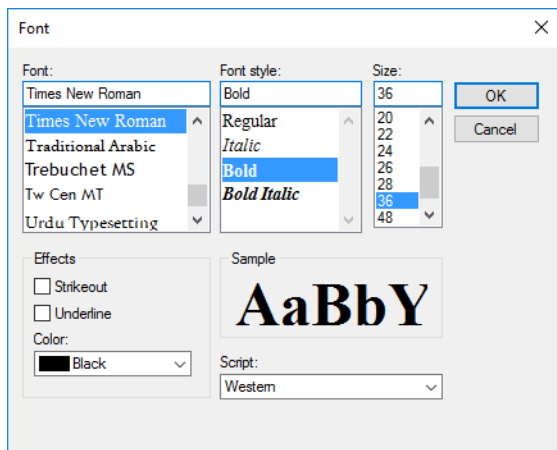
FONT

We now want to create a new font for the text element.

- ▶ In the **project manager**, open the **Screens** node.
- ▶ Select the **Fonts** node.
- ▶ Now, in the detail view, open the context menu and select the **New font...** entry.

The Windows standard dialog to define fonts opens.

- Select the font, the font type and the font size as desired.



In the detail view of the **project manager**, there is now **Font 6**, the **properties window** shows the properties of the newly-created font type.

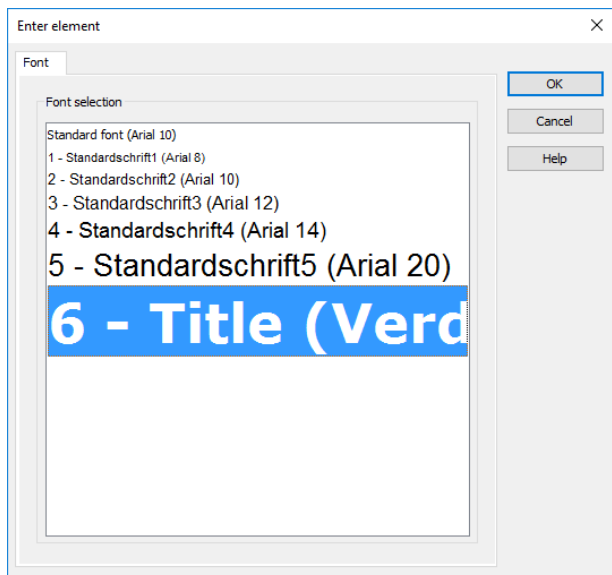
- Change the name of the font to **Title**.



The font types created here can be used in all elements that contain text.

We now want to use this font in our text element.

- In the **Start screen**, select the text element and select the **Text** group in the **properties window**.
- Use the **Font** property to open the dialog to select a font.



- ▶ Select the **Title** font.
- ▶ To save it, click on the **Save screen** icon or select the **save screen** entry in the screen's context menu.

7.5.2 Functions

The project is implemented using functions. For example, you can call up your newly-designed screens using buttons, with functions which have been linked to screen switching. However, functions are not just used for screen elements, we will create many different functions in our project.

Function administration

- ▶ In the **Project Manager**, select the **Functions** node.

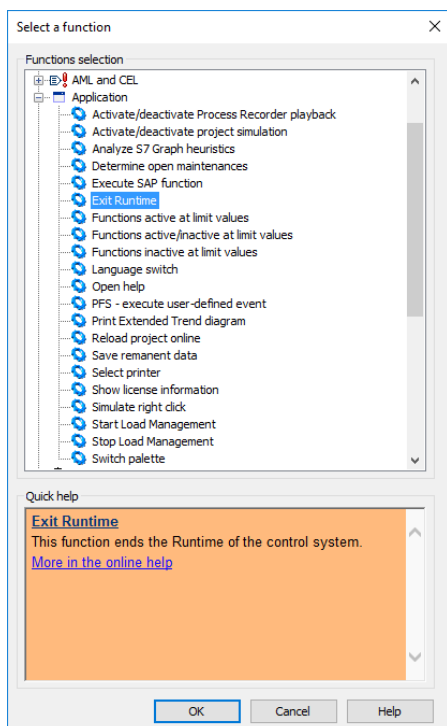
The empty function list now appears in the detail view of the **Project Manager**. The functions you have created are listed later in the functions you have created.

- ▶ Click on the **New function...** icon, at the top left of the detail view.

A dialog box with the pre-defined functions opens; these are arranged in groups.

Firstly, we want to create a function to close Runtime.

- ▶ Open the **Application** group.
- ▶ Select the **Exit Runtime** function and confirm the selection with **OK**.



The function is now shown in the function list in the detail area of the **Project Manager**. The function can be changed as required in the **properties window**.

- Change the name of the function.

This function does not need any parameters; the definition is therefore ended here.



Information

You can add or remove frequently-used functions to or from the favorites with drag&drop or by using the context menu. You thus save having to open the corresponding group of the function each time.

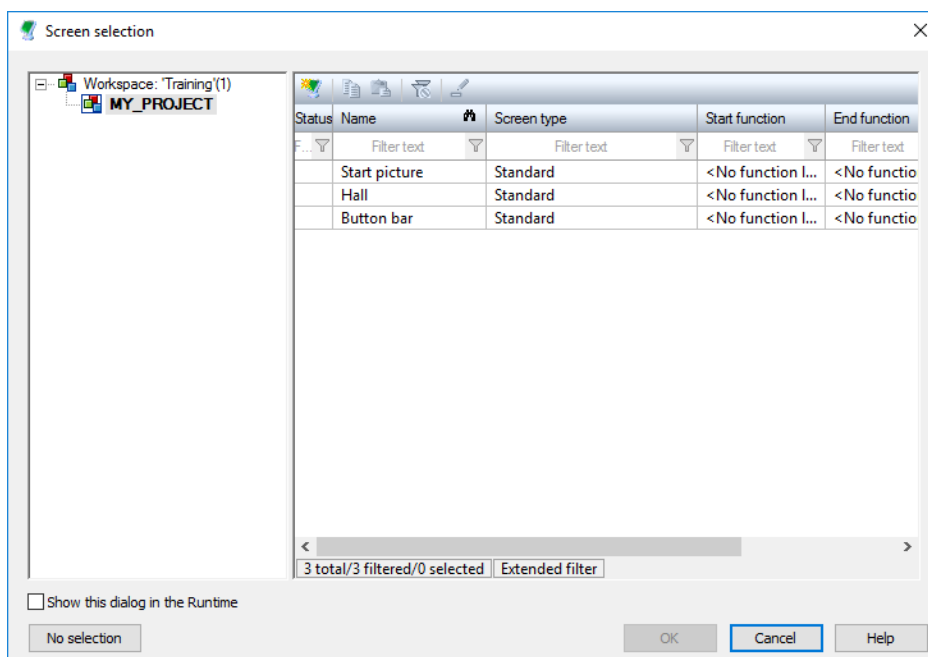
The next thing to do is to create the functions for calling up the screens, the so-called screen switch function.

- Click on the **New function...** icon, at the top left in the detail view.
- Select the **screen switch** function from the **Favorites** and confirm the selection with **OK**.

Because this function needs parameters, the dialog to enter the required parameters now opens.

The screen that is to be opened is needed as a parameter.

- Select the **Start screen** screen and confirm the selection with **OK**.



Further parameters are not required and the definition of this function is completed.

- Create the corresponding screen switch function for the **Button bar** and **Hall** screens.

Our function list should now look as follows:

Status	Name	Type	Parameter
Filter text	Filter text	Filter text	Filter text
	Exit Runtime	Exit Runtime	
	Switch to button bar	Screen switch	Button bar (Standard)
	Switch to Hall 1	Screen switch	Hall (Standard)
	Switch to Start picture	Screen switch	Start picture (Standard)



Tip: Video - screen switching in zenon

The video shows how the screen switching function and a button in zenon are created.

Optional for the QR code: www.copadata.com/tutorials



7.5.3 Dynamic elements

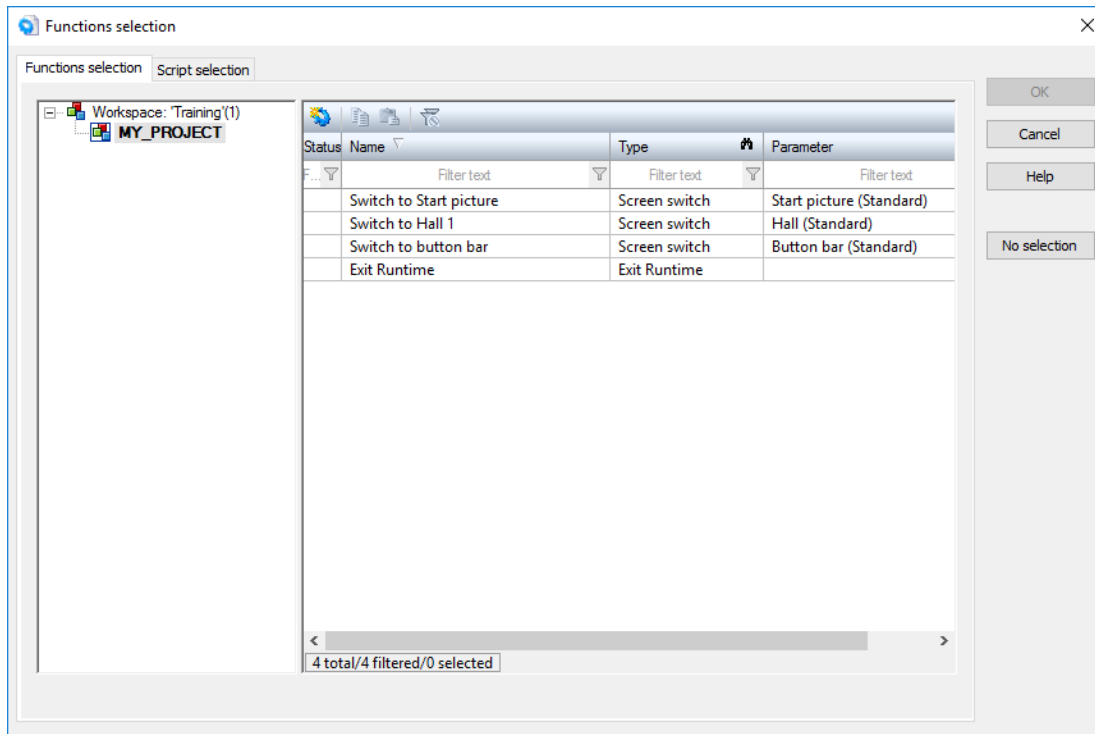
As opposed to vector elements, dynamic elements change their appearance in the Runtime. They are used to display variable values or to execute functions.

Button

First we will create buttons for our **Button bar**, with which it is possible to switch between the **Start screen** and the **Hall**.

- ▶ Open the **Button bar** screen.
- ▶ Select the **Button** entry in the **Elements** menu.
- ▶ Now, with the left mouse button held down, drag the element to the **Button bar** screen in the desired size.

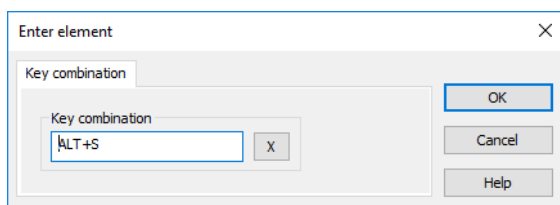
The dialog box for selecting functions opens automatically after you release the mouse button.



- ▶ Select the **screen switch start screen** function.
- ▶ In the **properties window**, open the **Text** group.
- ▶ For **Text line 1**, enter the text `Start screen` and, for **Text line 2**, the text `Alt+S`.
- ▶ Open the **Runtime** group.
- ▶ Under **Keyboard operation**, select the **Key combination [...]** property.

The dialog for the definition of the key combination opens.

- ▶ Enter `Alt+S` by clicking in the input area and then pressing that key combination.
- ▶ Confirm the entry with **OK**.



In Runtime, you can now execute this function either by clicking the button with the mouse or by pressing this key combination.

- ▶ Create a corresponding button for the **Hall** screen with the label **Hall 1** and `Alt+1`, as well as the corresponding keyboard shortcut.

- Create another button for the **Exit Runtime** function, with the label **Exit** and the keyboard short cut ALT+E.

For the button labels, it is recommended that a certain font is defined and that this is used for all buttons that are used. This is because if you want to change the font or the size at a later time, this is only changed at a central point and all buttons are adapted automatically.

- Create a new **Buttons** font.
- Select the buttons in the **Button bar** screen.

To do this, you can either click on the first button and then select the others with the Caps key pressed, or press the Ctrl+A key combination in the screen; all elements are selected in the screen as a result.

You can now change the properties of all selected elements in the **properties window** at the same time.

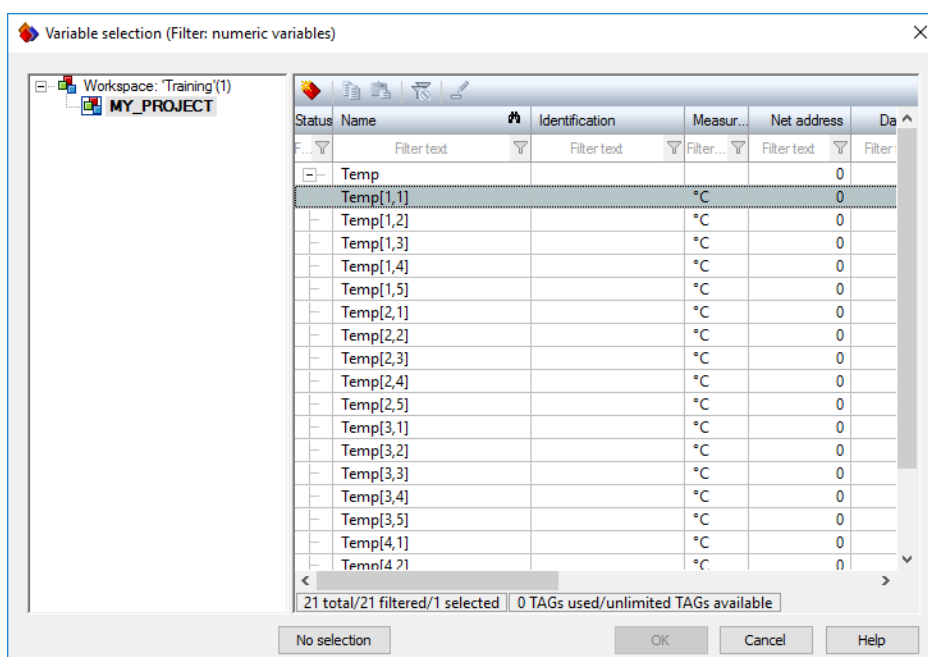
- Open the **Text** group.

In the **Font** property, select the **Buttons** font that has just been created.

Numeric value

- Open the **Start screen** screen.
- In the **Elements** menu, select the **Numeric value** entry.
- With the left mouse button held down, drag the the element to the **Start screen** in the desired size.

The dialog box to select variables opens automatically.

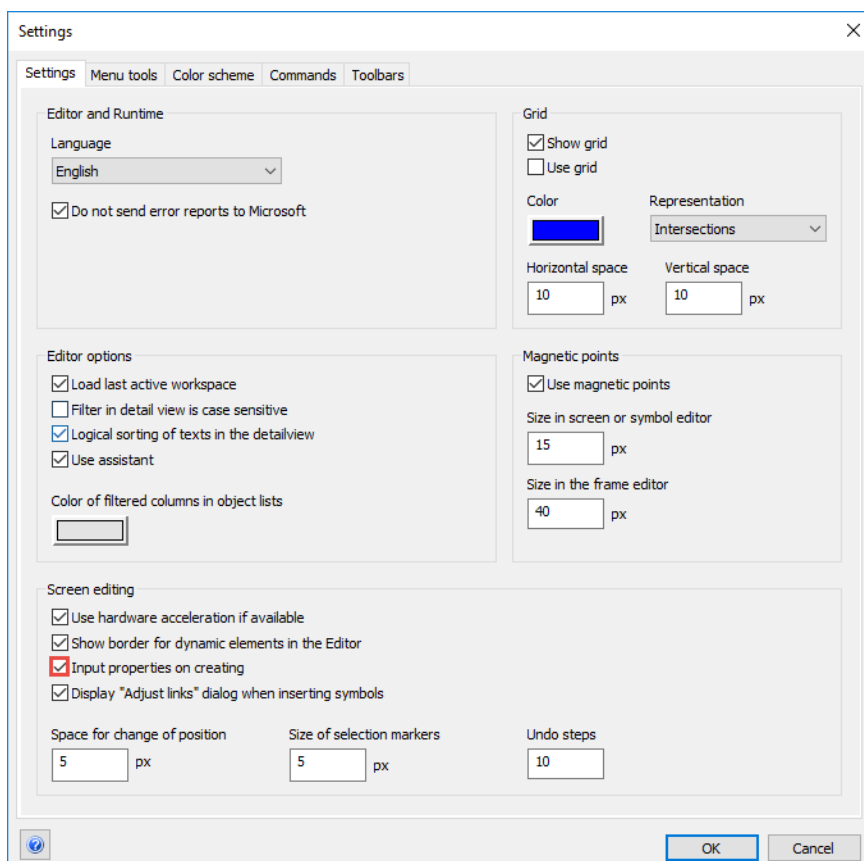


- Select the variable **Temp[1,1]** and the press the **OK** button.

If the **variable selection** does not open automatically, proceed as follows:

- ▶ Open the global zenon **Extras** menu.
- ▶ Select the **Settings** menu.

The following dialog opens:



- ▶ Activate the **Input properties on creating** option in the **Screen editing** group and then press the **OK** button.

Changing an element:

- ▶ You can also open the variable selection by double clicking on the element.
- ▶ To change the size of the element, click on one of the corner points (a double arrow appears instead of the mouse pointer). Hold down the mouse button and drag the element to the desired size.
- ▶ Double-clicking on one of the marking points opens a context menu in which you can enter the pixel coordinates precisely.
- ▶ To move the dynamic element, click on the middle of the element with the mouse (an arrow cross appears), hold down the left mouse button and drag the element to its new desired position.

- ▶ If the mouse pointer is on the knob, you can move the point with the cursor buttons or change the size.
- ▶ You can activate a grid in the screen, on which you can position the elements.

Creation of further elements:

- ▶ Now create further numeric value elements for the variables **Temp[1,2]**, **Temp[1,3]**, **Temp[1,4]** and **Temp[1,5]**.
- ▶ For each **numeric value**, create a **dynamic text field** and write the name of the variable in the text field.
- ▶ Click on the **Save screen** icon or select the **Save screen** entry in the context menu.



Information

Create a numeric value element and the attendant text field and copy the elements with copy/paste.

7.5.4 Screen Functions

You can link functions to each screen, which will automatically be executed on opening or closing the screen. We will now use this functionality to call up the **Button bar** automatically with the **Start screen**.

- ▶ In the **Project Manager**, select the **Screens** node.
- ▶ In the detail view of the **Project Manager**, highlight the **Start screen** screen.
- ▶ Open the section **Execution** in the properties window.
- ▶ In the **Start function** [...] property, select the **Screen switch** function **Button bar** from the list.

This means that in Runtime, whenever the **Start screen** screen is opened, the **Button bar** screen will also be opened automatically.



Attention

*The function must not refer to its own screen. That means that you must NOT link the **Screen switch button bar** function to the **Button bar** screen.*

7.6 The tank

In this section, we create a screen that visualizes a tank with all inflow and outflow valves. The tank is supplied by a pump.

- ▶ Open the **Hall** screen to do this.

7.6.1 Adding the tank symbol

We will not draw the tank itself, but will select a symbol for this. Symbols can be stored in a standard project in two places. Firstly, there is, under the screens node, a symbol library subnode, which can include the project-specific symbols. Secondly, you can find the general symbol library at the bottom in the project tree, whose symbols are available in all projects, but not in the project backups.

The general symbol library already contains symbols. We have predefined these symbols for you, in order to make the design of your project easier. You can use all symbols from this and change them as you wish.



Information

If symbols are also used in other projects or in multi-user projects, they can also be stored in a global project.

Because we have not created our own symbols in this project, the local symbol library is currently still empty.

- ▶ Select the **General symbol library** group.
- ▶ Open the **Container** group.
- ▶ Drag **Container 2** with the left mouse button held down to the middle of the left half of the screen and press the Ctrl key, before you release the mouse.
- ▶ Change the size of the symbol.

If you have not pressed the Ctrl key when inserting a symbol, a link to the symbol in the library is created in the screen. The symbol is copied to the screen with the Ctrl key pressed down. We will look at that in more detail later however.

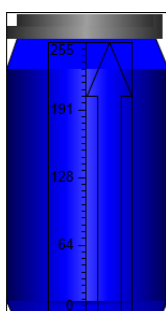
7.6.2 Adding the fill level display

Now we want to have the fill level displayed in our tank.

- ▶ In the **Elements** menu, select the **Bar chart** element.
- ▶ With the left mouse button held down, drag the element in the tank to the desired size.

- ▶ In the **variable selection** dialog box, select the **Tank[1,1].Fill.Level** variable and press the **OK** button.
- ▶ In the **properties window**, open the **Display** group.
- ▶ Set the **bar color** to green under **Bar fill**.
- ▶ In the **properties window**, open the **Filling** group.
- ▶ Activate the **Transparent** property here.

Your tank should like something like this:



7.6.3 Adding the inflows and outflows

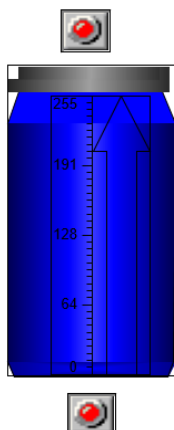
In this step, we add the valves for the inflow and outflow in our screen. To do this, we will use the **Switch** element, which visualizes the status of a binary variable.

- ▶ In the **Elements** menu, select the **Switch** element.
- ▶ With the left mouse button held down, drag the element above the tank to the desired size.
- ▶ In the **Variable selection** dialog, select the **Tank[1,1].Inflow** variable and then click on the **OK** button.
- ▶ In the **properties window**, open the **Filling** group.
- ▶ Under **pre-defined graphics**, select the bitmap pair for a valve.

You can also select your own graphics files for the statues instead of the pre-defined bitmaps. Instead of a switch, you can also use the **Combined element** element for the display of different statuses of a variable. The combined element makes it possible for you to display different graphics files or symbols for more than two statuses.

Also add a corresponding **Switch** element for the outflow valve, which displays the status of the **Tank[1,1].Outflow** variable.

This is something like what our screen looks like:



7.6.4 Adding the pipelines

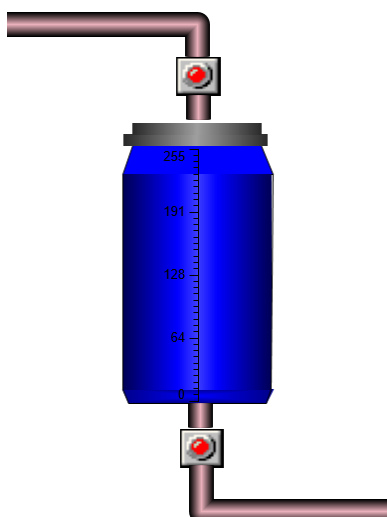
We will now draw the corresponding inflow and outflow pipelines. To do this, we can use the elements **Line**, **Polyline** or **Pipe**. In this example, we use a **Pipe**.

- ▶ Select the **Pipe** element.
- ▶ Connect the inflow valve to the tank.

You add a new joint to the Pipe with each click. To close the Pipe, double-click or press the **Esc** key.

Add the other Pipes as shown in the illustration below.

This is something like what our tank should look like:



- ▶ Now add your tank to the symbol library of the project.

7.7 Language switch

This functionality is essential for the realization of international projects. Language switching makes it possible to switch between different languages used for text information shown in Runtime. For project configuration, a language table has been created in the detail view of the Language file node.

In order for texts to always be language switchable, they must always start with the @ character or be enclosed with two @ (for example: @Hall@ 1) . These texts are marked as key words in zenon and entered into a language table. After an odd number of "@" characters, the text is translated, whereby the counting starts at the beginning of the entry.

Examples:

Keyword	Meaning
@Text	The whole text will be translated.
Text@	This text will not be translated.
@Text@	The text between the two "@" will be translated.
@user@@sample@	Everything is translated
@user@sample	Sample is not translated

7.7.1 Language table

To configure language switching, a language table must be created in the Project Manager. This language table is based on text files and these can be created in the context menu of the Project Manager or in the detail view of the language file.



7.7.2 Creation of a language file

To create a new language file, open the **language file** node in the **Project Manager**.

- ▶ In the detail view menu, click on **New language file**.
- ▶ Enter the name **German** into the dialog.
- ▶ Confirm with OK and the new file is added to the language table on the far right side.
- ▶ Click on the **New language file** icon in the detail view menu.
- ▶ Enter the name **English** into the dialog.
- ▶ Confirm with OK and the new file is added to the language table on the far right side.

We now have the necessary table for language switching.

7.7.3 Creation of key words

To create a new key word, scroll to the last line of the language table. Here you can find an empty line in which new key words can be added.

- ▶ Enter the name of the keyword in the empty row, in the column **keyword**.
- ▶ Confirm your entry with the Enter key.
- ▶ The key word you have entered is now copied to all available language files.
- ▶ To change the entries in the language files, click in the cell of the table and change the text.

Create the following key words:

Keyword	German	English
Hall	Halle	Hall
Reload project online	Nachladen	Reload
Exit	Beenden	Exit
Start screen	Startbild	Start screen



Information

In the language table, all key words are entered without the @ character. Existing control characters are removed automatically.

7.7.4 Using the key words

To be able to use the key words in Runtime, you must change the text in the respective elements. We want to make our buttons in the **Button bar** language-switchable.

- ▶ Open the **Button bar** screen.
- ▶ Click on the button for screen switching on the **Start screen**.
- ▶ Change the **Text line 1** property in the **Text** properties group from `Start screen` to `@Start screen`.

Repeat this step for all key words, we have created in the previous step.

7.7.5 Activate language switching

To be able to change the language in Runtime, a function must be created for this.

- ▶ Create a new function.
- ▶ In the **Application** group, select the **Language switch** function.
- ▶ Click on the ... button.
- ▶ Select the language file English.txt and click on **Open**.
- ▶ Close the dialog with **OK** and change the name of the function.
- ▶ Create a second function for the language file German.txt.
- ▶ Create new buttons for these functions in the **Button bar** screen.
- ▶ Give these buttons a unique name.

To see the effects of the language switching, you must switch to Runtime. When clicking on the button for English, the labeling of the other buttons switches to the English text.



Hint

You can assign the buttons to the respective flag as background graphics.

7.8 Color switching

zenon makes it possible for you to not just adapt the language in Runtime, but also the colors of the elements. This is controlled using the color palettes.

Color palettes make it possible to summarize individual colors into color palettes. You can define a color set such as this, which can be easily edited, both in the Editor and in the Runtime. All colors that are defined with the help of palette colors can be easily changed.

The uniform design of corporate designs can be completed so very easily. If necessary, the design can be change completely (switching palettes) or only individual colors (color switch in palette) can be changed centrally.

In our example, we will set up color switching for a day view and a night view.

7.8.1 Creating a color palette

To create a new color palette, open the **Screens** node in the **Project Manager**.

- ▶ Now select the **Color palettes** subnode.
- ▶ In the detail view menu, click on the **New color palette** icon.
- ▶ Change the name of the color palette in the **Color palette name** properties to `Night`.
- ▶ Create a second color palette with the name `Day`.

Note: You only see the color palette when you create a new color.

7.8.2 Creation of a color

- ▶ Click on the **New color...** icon in the detail view menu.
- ▶ Change the name of the color by double clicking on the color.
- ▶ Change the individual colors in both color palettes by double clicking on the respective color. You can also enter the color code directly.

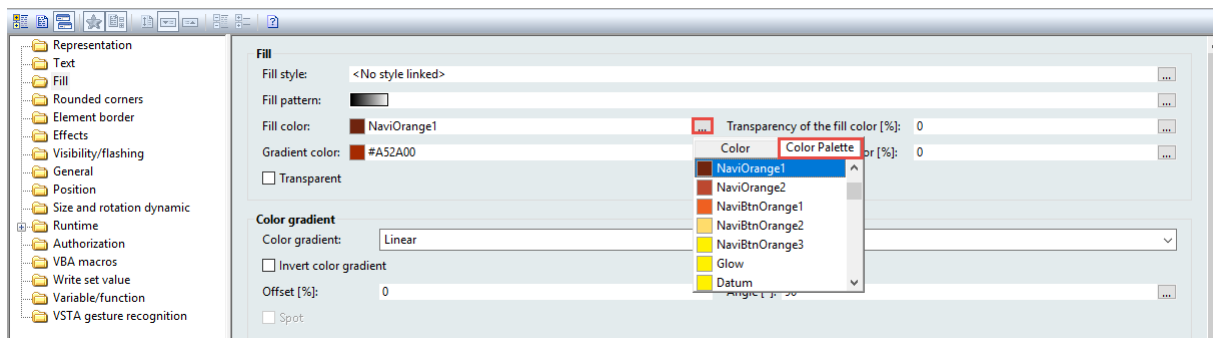
Create some colors.

7.8.3 Using the color palette

In order to be able to use the color palette in Runtime, you must link the entries of the color palette to the respective elements. We want to make our buttons in the button bar color switchable.

- ▶ Open the **Button bar** screen.
- ▶ Click on a desired button.
- ▶ Open the **Filling** properties group and click, in the **Fill color** property, on the **...** button.
- ▶ Next to the **Colors** entry, you now see an entry with **color palette**. Click on the **Color palette** entry.

- You now see the colors that you defined previously; select the desired color.



Repeat this step for all buttons in the **Button bar** screen:

7.8.4 Activate color switching

To be able to change the colors of buttons in Runtime, a function must be created for this.

- Create a new function.
- Select, in the **Application** group, the **Switch color palette** function.
- In the **Color palette** drop-down menu, select **Night**.
- Close the dialog with **OK** and change the name of the function.
- Create a second function for the **Day** color palette.
- Create new buttons for these functions in the **Button bar** screen.
- Give these buttons a unique name.

You must switch to Runtime to see the effects of the color switching. When clicking on the button for **Night**, the colors of the buttons switch to the **Night** color palette.

7.9 Styles

Styles comprise graphical properties of screen elements in zenon that belong together in logical units. As an example, the "Line" style comprises the properties Color, Thickness, Sample and the line ends.

Several styles are compiled into a style group. This makes it possible to administer the complete graphical properties of a zenon element (with the exception of the size) independently and centrally in the (global) project and to apply it to as many elements as desired.

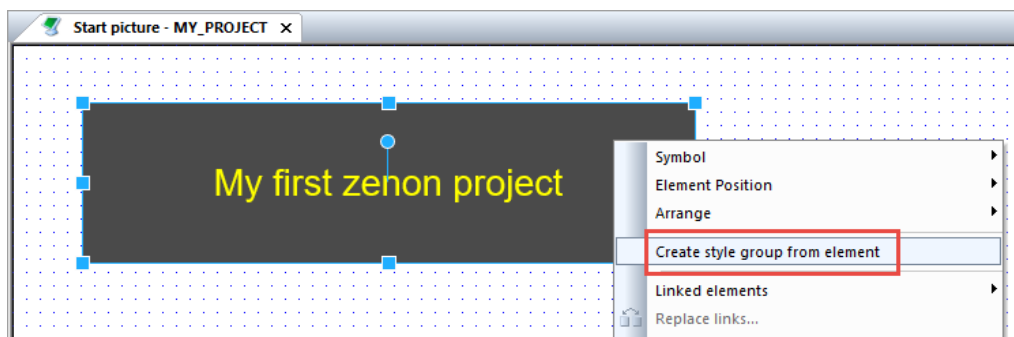
Style groups can be extracted from existing zenon screen elements. As a result, the usual configuration is made possible to create the style, at the element directly.

The styles serve as an aid to project configuration and support you in the definition, visualization and safeguarding of the consistency of the graphical user interface and allow quick implementation of the projects through reusability. They also make it easier for you to incorporate your company-specific designs into your zenon project.

7.9.1 Creation of a style

You cannot create a style directly, you must do it via the zenon elements.

- ▶ Create a new element (button, text element, etc.), or configure an existing element.
Select a special design for this element. Such as, for example, colors, shading, rounded corners or a font type for the text, etc.
- ▶ As soon as you have completed the element, click on the element with the right mouse button.
- ▶ Select **Create style group from element** in the context menu.



- ▶ Enter a name for the style group in the subsequent dialog.

A style group was created. You can now edit the properties of this style centrally in it.

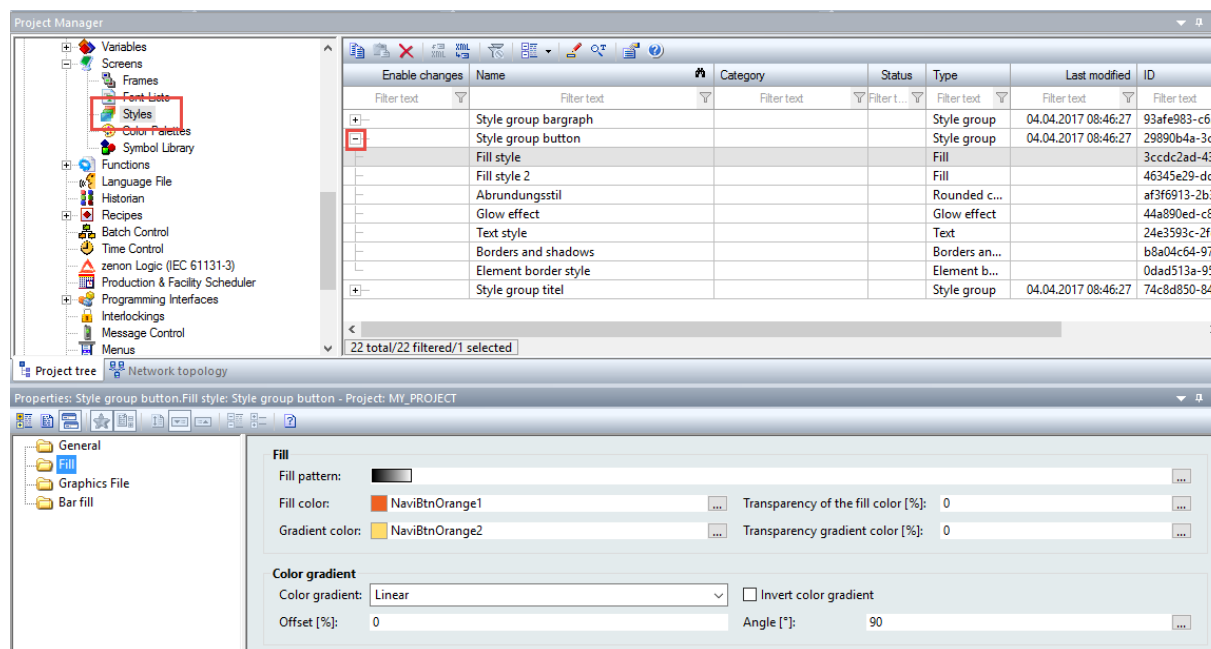
- ▶ In the **Project Manager**, open the **Screens** node.
- ▶ Select the **Styles** subnode.

You can see previously-created style groups in the detail view.

Click on the + on the left edge to see the individual styles of the style group.

- You can change this in the properties by clicking on the styles.

Your styles should look something like this:



- Create a style group for a button and a style group for a static text field.

7.9.2 Using a style

You can apply a style in two different ways.

1. Styles can be assigned to an element by using **Drag&Drop**.
2. Styles can be assigned to an element by using the **element properties**.



Information

In addition to the assignment of style groups to an element, it is also possible to assign individual styles from the style groups to an element.

When assigning single styles, it is also possible to do this by using Drag&Drop or the element properties.

1. The first thing we will do is to assign our buttons in the **Button bar** screen a style group by using **Drag&Drop**.
 - a) In the **Project Manager**, open the **Screens** node.
Open the **Button bar** screen by double-clicking on the name.

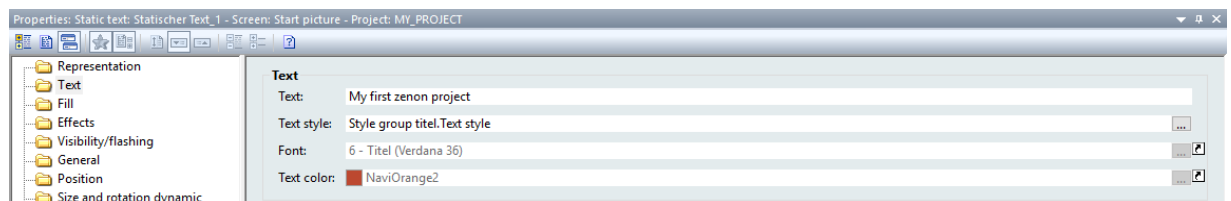
- b) In the **Project Manager**, open the **Screens** node.
- c) Select the **Styles** subnode.
- d) Now move the button style group by dragging & dropping the first button in the **Button bar** screen.
- e) The display of the button is adapted to the style group.

In the element properties (such as fill), you now see the linked style in each subgroup.

2. The second thing to do is to assign our static `My first zenon project` text field in the **Start screen** a style group, using the **element properties**.

- a) In the **Project Manager**, open the **Screens** node.
Open the **Start screen** screen by double-clicking on the name.
- b) Click on the `My first zenon project` static text field.
- c) Open the **Text** properties group and change the **Text style** property.
- d) In the subsequent dialog, select the text style from the text style group.
- e) Confirm your selection with **OK**.

The properties of your static text element should look like this:



Information

As soon as an element has been assigned a style, you can no longer change the properties. This must happen using the style, or the style must be deleted.

7.10 Runtime

In this step, you learn how to start and close the Runtime. In addition, you can see how you operate your project in the Runtime.

7.10.1 Start the Runtime

The Runtime can be started in three ways:

Description
By clicking the Start Runtime (F5) button
With the F5 key in the Editor
In the Start menu in the program group All programs\COFA-DATA\zenon Runtime 8.00

After starting the Runtime two images are loaded. First, the **Start screen** (because this was entered as the start screen during project configuration) and then the **Button bar** image (because it was called up with the start function of the start screen).

7.10.2 Tips and tricks for navigation in Runtime

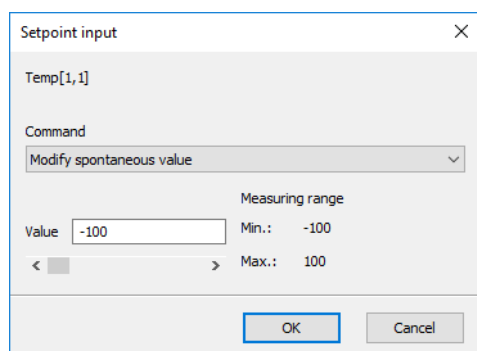
- ▶ Right-click somewhere in an empty area of the screen.

If you have pressed the right mouse button long enough, the name of the screen appears on the mouse pointer.

- ▶ Now right-click on the numerical value element for the variable **Temp[1,1]**.

The name of the variable that is linked to it is now shown on the left above the element.

- ▶ Now left-click on the numerical value element for the variable **Temp[1,1]**.



The standard dialog box for setting values opens and allows you to change the value of this variable.

- Set the value of the variable **Temp[1,1]** to 300 and close the dialog box with OK.
- Set the value of the variable **Temp[1,2]** to 35 and close the dialog box with OK.
- For the variable **Temp[1,3]**, change the command in the setpoint input to switch to substitute value and close the dialog box with OK.

You now see a red square at the right upper edge of the element. This means that you do not currently have a connection to the controller for this variable. This red square is also shown in the event of an interruption to the controller.

- ▶ Click on the **Hall 1** button.

Switch to the respective screen by clicking on the screen switch function button.

- ▶ Click on the button for color switching.

The color of the buttons changes to the settings that you made in the Editor.

- ▶ Click on the button for language switching.

The language of the buttons adapts to the settings that you made in the editor.

7.10.3 Exit Runtime

- ▶ Press the **End** button to close Runtime.



Information

If you have not defined a button for stopping the Runtime and if the Windows title bar is not available, the Runtime can also be closed with the key combination Alt+F4.

7.11 Adapt and expand project

In this step, you learn how to easily adapt and expand existing projects.

7.11.1 Reload project

In order for us to not have to close and restart the Runtime each time the project is changed, we will now take a look at the **Reload** function. This functionality allows us to make changes in the Editor and to reload these in the Runtime at any time.

- ▶ In the **Project Manager**, open the **Functions** node.
- ▶ In the context menu of the **Functions**, open the **New function..** entry.
- ▶ In the **Application** group, select the **Reload** function.

- ▶ Create a new button in the **Button bar** screen, with the text @reload, which executes the **Reload** function.
- ▶ Start the Runtime.
- ▶ Switch to the Editor using the Windows task bar or with the Alt+TAB keyboard combination.

We will then see how the reload function can be used after the individual changes in the project.

Change project

We will undertake changes to our project and apply these in Runtime, without closing and restarting Runtime.

Change set value input

- ▶ Open the **Start screen** screen.
- ▶ Select the numeric value for the variable **Temp[1.1]**.
- ▶ Open the **Write set value** properties group.
- ▶ Change the **Write set value via** to **Element**.

The standard dialog box is then no longer opened when the value of the **Temp[1.1]** variable is changed in the Runtime. Instead, we can enter the set value in the element directly and then confirm the entry with the Enter key.

Reload changes in the Runtime

Before we can reload the changes in the Runtime with the reload function, we must first save the screen and create new Runtime files.

- ▶ Click on the **Save screen** icon or select the **Save screen** entry in the context menu.

There are two possibilities for the creation of Runtime files. All Runtime files can be newly created, or only those that have changed.

New Runtime files can be created using the context menu of the project in the project manager or by using the corresponding icons in the tool bar.

- ▶ Open the context menu of the project by clicking on the name of the project with the right mouse button in the **Project Manager**.
- ▶ Select the **Runtime files / Create changed** entry.

Or:

- ▶ Click on the **Create changed Runtime files** icon.

We have thus created the new Runtime files. We now only need to apply these in the Runtime.

- ▶ Use the Windows task bar or the keyboard shortcut Alt+TAB to switch to the Runtime.
- ▶ Click on the **Reload** button.

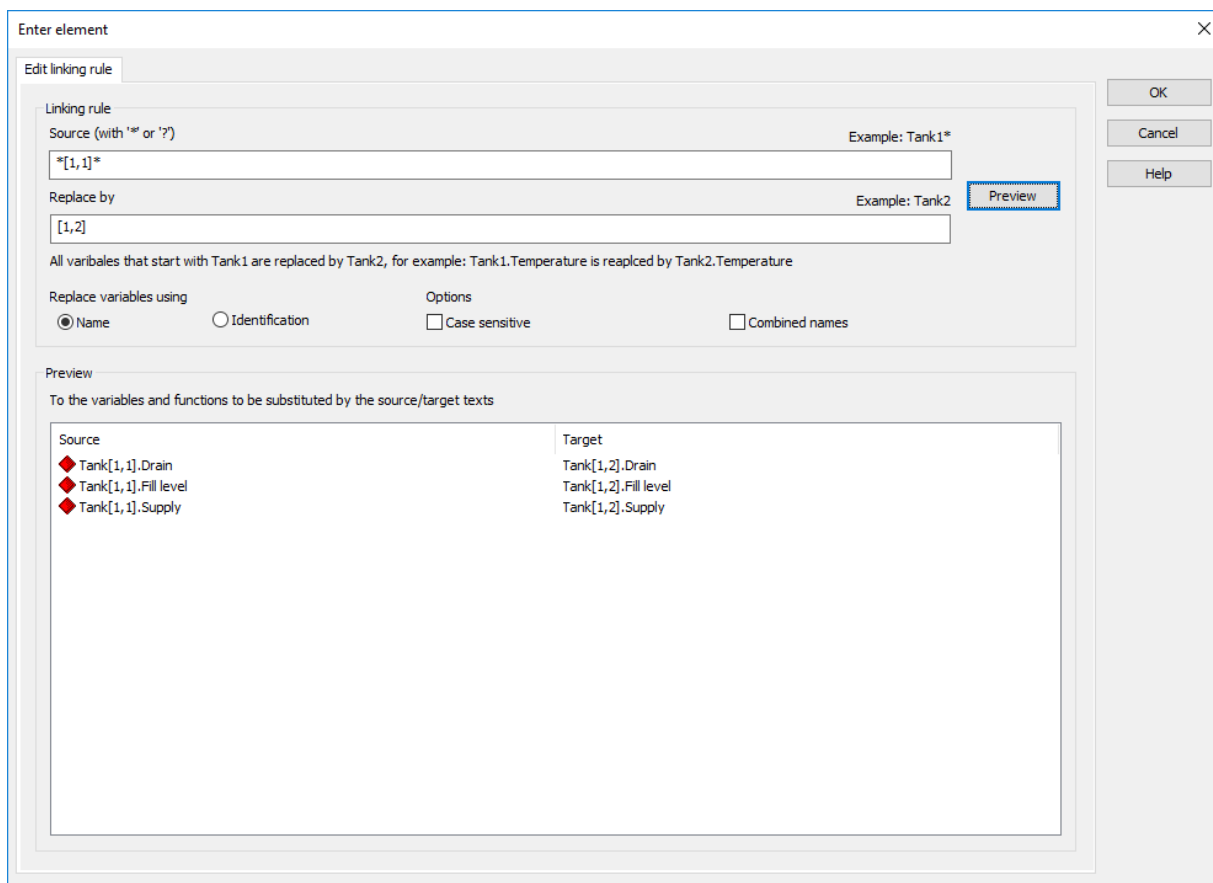
7.11.2 Substitution of an element

The replacement of links is like the classic find&replace for variables and function in screens. It is automatically offered in many places in zenon, but can be executed manually to a highlighted group of screen elements. In doing so, a list of all variable and function references is created, which can then be replaced by other ones or wild cards in groups. Substitution then replaces the old reference and is saved permanently with the screen. This method is particularly helpful if parts of a screen are copied to another and are to be linked to other variables.

In the next step, we want to use this functionality to display a second tank in our **Hall** screen.

- ▶ Open the symbol library of your project.
- ▶ Select the previously-added tank and add it next to the first tank in the **Hall** screen.

The tank element from the symbol library is added in the screen. A new dialog for substitution opens automatically.



Source	Target
Tank[1,1].Drain	Tank[1,2].Drain
Tank[1,1].Fill level	Tank[1,2].Fill level
Tank[1,1].Supply	Tank[1,2].Supply

All variables and functions that are linked to our tank are offered in a list for replacement.

- ▶ In **Source**, enter `*[1.1]*`.
- ▶ In **Replace with**, enter `[1.2]`.
- ▶ Click on the **Preview** button.

Now, in the right column of the list, you can check whether the replacement will be made as you wish. Your substitution is applied and the variables or functions are replaced in the element by clicking on the **OK** button.

You can call this dialog up at any time.

- ▶ To do this, open the context menu of the selected elements with the right mouse button.
- ▶ Select the **Replace links** entry from the context menu.

The dialog to replace links is thus opened again.

7.11.3 Substitution of a screen

Screens that have already been created can be used more than once with the help of indirect addressing. This functionality makes it possible to replace variables and functions with others when opening a screen. You thus only need to create one screen for the same equipment parts and can then open it by using different screen switchi functions with other respective groups of variables and/or functions.

We will use indirect screen addressing for our **Hall** screen. We will now switch between **Hall 1** and **Hall 2** with two buttons, but refer to the same screen in the process.

- ▶ Create a new **screen switch** function.
- ▶ As a parameter, select the **Hall** screen and confirm the selection with **OK**.

zenon detects, after the screen has been selected, that there are already dynamic elements with variables and/or functions in the screen. Another dialog box is therefore automatically opened. We are already familiar with this dialog from substitution. We will disregard the additional page to replace indexes here.

You can see the variables and functions contained in the screen in the lower area of the Linkings dialog box. You can replace the variables individually by double clicking on a line. Because we have adhered to a consistent structure for issuing names with our variables, we can replace complete groups.

- ▶ Enter, as a **source**, * [1*.
- ▶ Enter, as a **target**, [2.
- ▶ Click on the **Apply** button and confirm the subsequent query with **Yes**.

In the lower part of the dialog box, you can now check whether the replacements correspond to what you expect. In our example, for all variables in which the name [1 appears, [1 is replaced with [2.

- ▶ Close the dialog box with **OK**.
- ▶ Create a new button in the **Button bar** screen with the function that has just been created and the labeling **Hall 2**.

7.11.4 Script management

If several functions are summarized and processed in a certain sequence, they must be saved in a script. There are some reserved names in the system, such as scripts that are automatically processed when starting and ending Runtime.

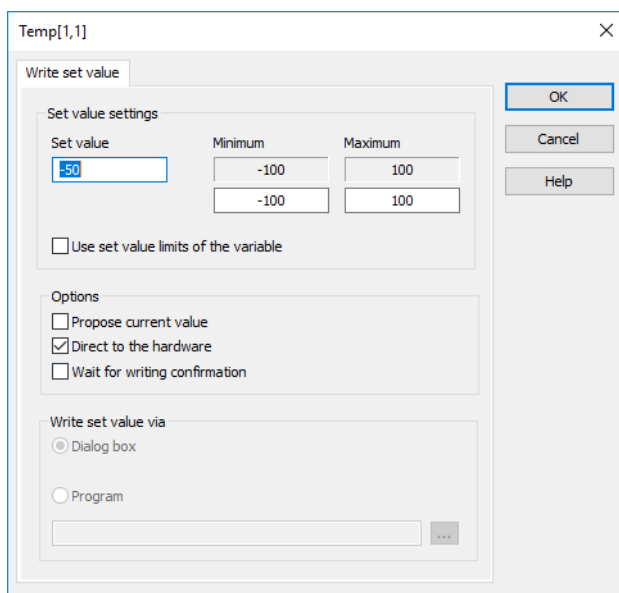
Parameter	Description
AUTOSTART	This script is automatically executed when the Runtime starts before the start screen is called up.
AUTOEND	This script is executed automatically when the Runtime is closed.

Creation of the write set value function

- Create a new function.
- In the **Variable** group, select the **Write set value** function.

Select the variable **Temp[1,1]** and confirm the selection with **OK**.

Because this function needs additional parameters, a further dialog opens.



- Set the **set value** to -50.
- Activate the **Direct to hardware** option.
This option causes the set value entered above to be written (=sent to the controller) without the Runtime expecting a confirmation or allowing a change.
- Close the dialog with **OK** and give the function a unique name.

Create a function for the variable **Temp[1,2]** that writes the set value as -70.

Creation of an AUTOSTART script

- ▶ In the **Project Manager**, open the **Functions** node.
- ▶ Select the **Scripts** subnode.
- ▶ Open the context menu in the detail view and select the **New script** entry.
- ▶ Give the new script the name 'AUTOSTART', by typing it in or selecting from the list of reserved script names.

Click on the **Add functions** button.

The dialog box for multiple selection of functions opens.

- ▶ Select both **Write set value** functions by either double clicking each or simply selecting them and clicking on the **Add** button.
- ▶ If both functions are present in the lower dialog box list, confirm the selection with **OK**.

The script's functions are now shown to you in the detail view of the **Project Manager**.

Status	Name
Filter text	Filter text
<input checked="" type="checkbox"/>	AUTOSTART
<input type="checkbox"/>	write value Temp 1,1: Write set value - Temp...
<input type="checkbox"/>	write value Temp 1,2: Write set value - Temp...

7.12 Questions about visualization

1. What is a multiuser project?
 - a) A project on which in the Runtime several users can log on at the same time, with all of them having the same rights.
 - b) A project on which in the Runtime several users can log on at the same time, with every user having different rights.
 - c) A project which can be edited by several project engineers on several computers at the same time.
 - d) A workspace containing several projects with every individual project being able to be edited by a different project engineer. However, a project cannot be edited by several project engineers at the same time.
2. You have a variable representing a value from a controller (PLC). The driver object type of the variable can be a:
 - a) PLC marker
 - b) Internal Variable
 - c) Communication details
 - d) Input

8. Topic: Event Handling

Learning objectives:

- ▶ You learn the functionality of special screen types and how to use them.
- ▶ You know that you can display system messages in the Chronological Event List in filtered form, print them and export them.
- ▶ You know that limit values/alarms can be defined using limit values of variables or via Reaction Matrices.
- ▶ You are able to administer your alarms in Runtime (acknowledge, print or export, for example).
- ▶ You know the difference between an alarm comment and an alarm cause.

8.1 General

zenon offers, with its basic functionality, two types of lists; we will get to know them in this section.

The **Chronological Event List (CEL)** shows system messages. Some of these messages – such as the login of users and editing users, or the registration of clients in the network – are shown here in principle; others can be configured, such as the editing or writing of recipes or the write set value actions. Furthermore, limit value breaches can also be logged in the CEL too, if desired.

The **Alarm Message List (AML)** shows alarms and their current status. These can be: occurred, cleared, acknowledged.

For AML and CEL, there are pre-defined separate control elements available; we need the "special screen types" for this.

8.2 Special screen types

We will use the screen for the Chronological Event List to show how a "special screen type" is created.

The special screen types make it easier for you to configure a project using special control elements and with the provision of screen templates. These templates are included with each installation. Existing templates can be changed and saved as a new template.

Furthermore, the special screen types offer a range of special tasks. After selecting a special screen type, special user elements for the respective screen type are available in the **Control elements** menu. The **Add template** menu item opens a selection dialog to add pre-defined layouts with certain control

elements at defined locations. There is now also the possibility to change these special screens to your own requirements and to save them as a standard template.

8.3 Chronological Event List (CEL)

8.3.1 CEL screen

- ▶ Create a new screen.
- ▶ Change the name.
- ▶ In the **General** properties group in the **Screen type** property, select the `Chronological Event List` screen type.

Selection of a template

- ▶ Open the screen by double-clicking on the name.
- ▶ Open the **Control elements** menu and select the **Add template** option.
These templates are pre-defined as standard for the operation of the Chronological Event List screen type and are shown in the dialog with a preview screen.
- ▶ Select the desired template and click on **Apply**.

After the desired template has been selected, the corresponding elements are added to the screen.

Creation of a template

To create a template, a screen of the desired type must be created (or opened).

- ▶ Make some changes to the existing CEL screen (position of the elements, colors, etc.)
- ▶ Select **Create template for screen type** in the screen's context menu or in the tool bar.
The dialog to create your own templates is opened
- ▶ Select an existing folder or create a new one.
- ▶ Enter a name for the new template. You can add a description as an option.
- ▶ Confirm with **OK**.

The template is added to the selected folder.

8.3.2 Screen switch - CEL

The setting of the content of the Chronological Event List is not carried out in the screen itself, but in the filter settings of the screen switch function for the CEL screen. This offers you the possibility to access the same screen with different buttons/functions, with different filter settings.

- ▶ Create a new screen switching function for the CEL screen.
- ▶ You can stipulate the filter settings for the Chronological Event List in the filter settings.

The filter dialog consists of several tabs; the General and Time tabs will be explained below. You can find information about the other tabs in the zenon Help.

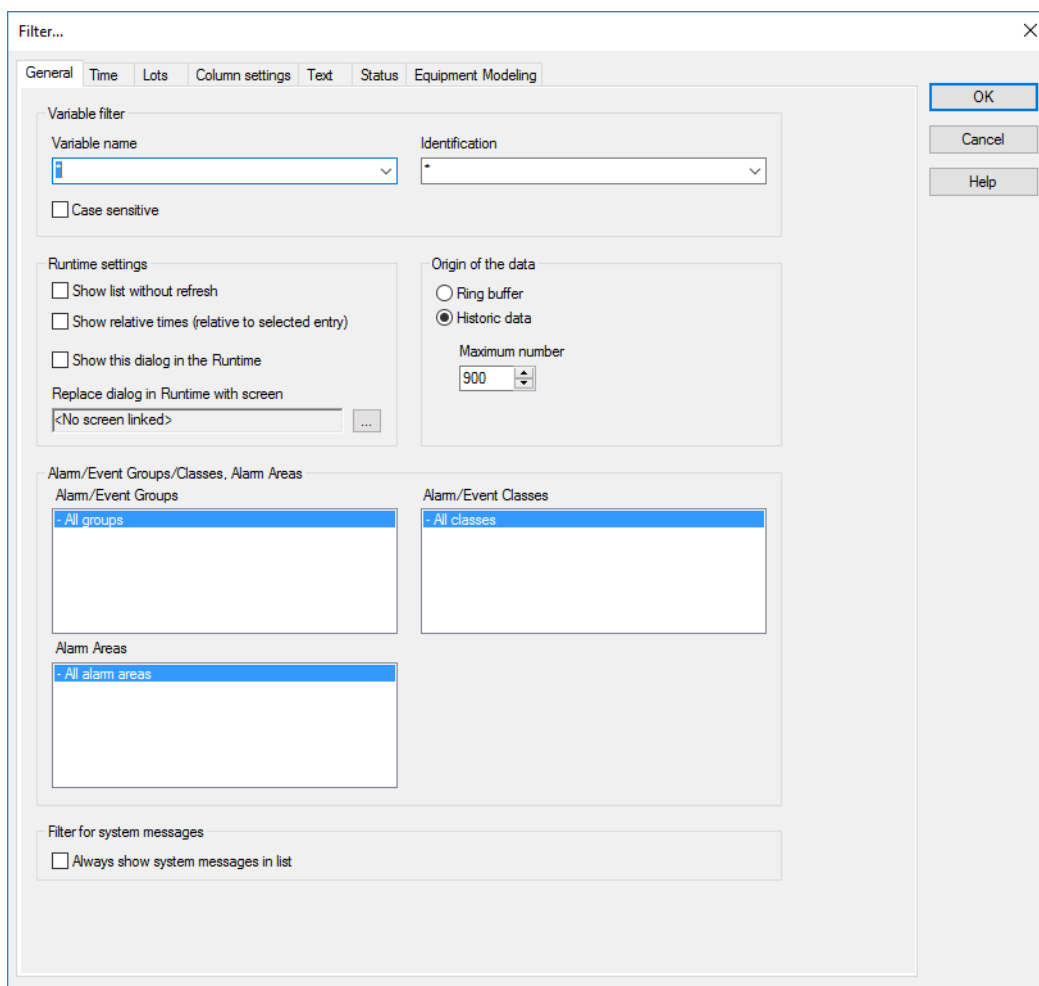


Hint

The **Chronological Event List Filter** offers a simplification of the filter settings. In this screen, you can offer the user certain filter setting possibilities and deliberately not offer others.

Filter dialog - General filter

In the first tab, entitled **General**, you define which events are displayed and what kind of access you have to the settings in Runtime.



The screenshot shows the 'Filter...' dialog box with the 'General' tab selected. The dialog has a title bar with a close button (X). Below the title bar are tabs: General, Time, Lots, Column settings, Text, Status, and Equipment Modeling. The 'General' tab contains the following sections:

- Variable filter:**
 - Variable name: A dropdown menu with a small icon on the left.
 - Identification: A dropdown menu with a small icon on the right.
 - ☐ Case sensitive
- Runtime settings:**
 - ☐ Show list without refresh
 - ☐ Show relative times (relative to selected entry)
 - ☐ Show this dialog in the Runtime
 - Replace dialog in Runtime with screen: A text field containing '<No screen linked>' and a button with three dots.
- Origin of the data:**
 - ☐ Ring buffer
 - ☒ Historic data
 - Maximum number: A spinner box set to 900.
- Alarm/Event Groups/Classes, Alarm Areas:**
 - Alarm/Event Groups: A list box containing '- All groups'.
 - Alarm/Event Classes: A list box containing '- All classes'.
 - Alarm Areas: A list box containing '- All alarm areas'.
- Filter for system messages:**
 - ☐ Always show system messages in list

On the right side of the dialog, there are three buttons: OK, Cancel, and Help.

The events available can be distinguished by:

- ▶ Variables

These options allow you to limit to events of certain variables. You can filter for variable names, as well as variable identification.

- ▶ Runtime settings

With this option, you control the behavior of the screen switching function in Runtime. With the **Show this dialog in the Runtime** option, you can also provide the filter settings in Runtime.

- ▶ Origin of the data

You can use the data origin to control whether you want to have current or current and historic events shown. Current data is saved in the working memory by means of a ring buffer; this is

limited to 100 entries by default. Historical data is saved to the hard drive directly and not limited in size.

The data can be saved in two different memories, in the working memory (via the ring buffer setting) or to the hard drive directly (using the historical data setting).

- ▶ Alarm/event groups, classes and alarm areas

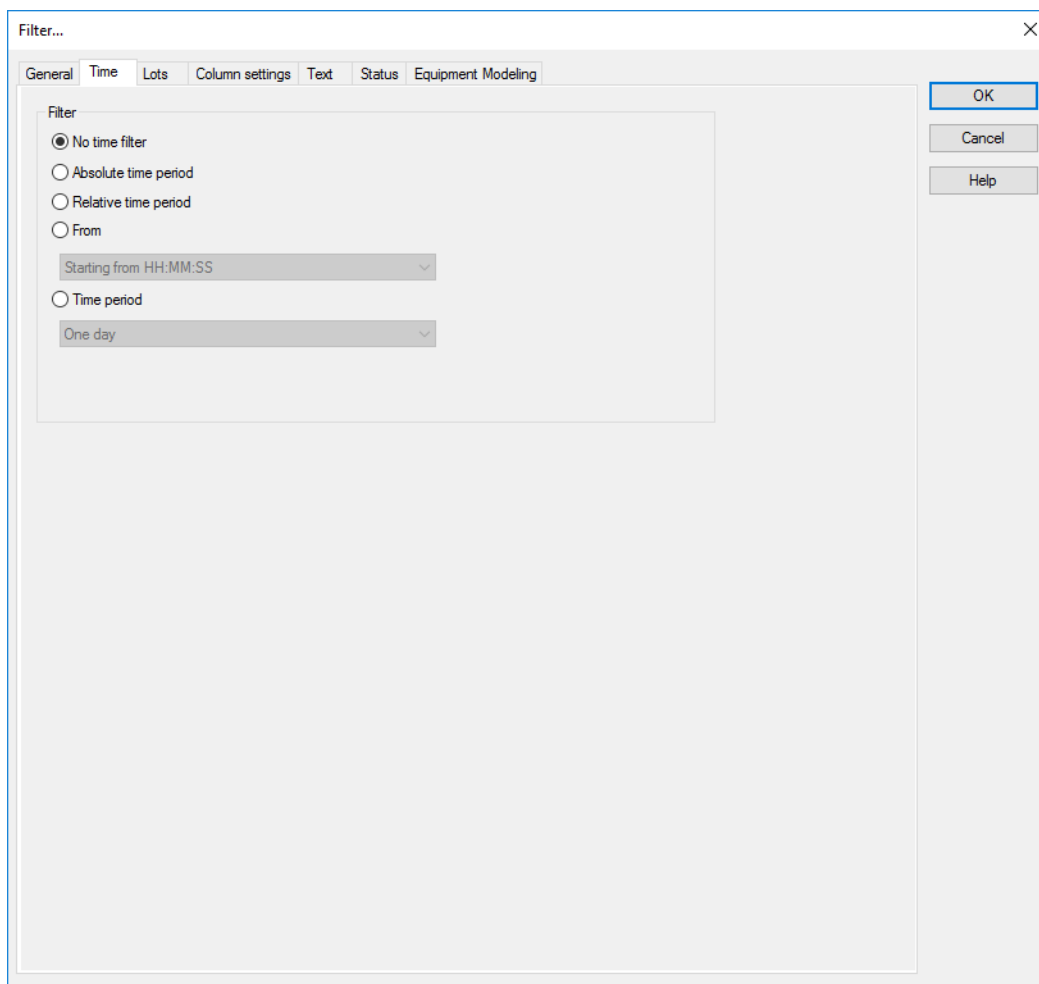
Allows filtering according to alarm groups, alarm classes and alarm areas. This option is only available if you have also defined these alarm settings.

There is only a small change needed for our example:

- ▶ Activate the **Historic data** option.
- ▶ Set the **Maximum number** to 900.

Filter dialog - Time tab

The second tab, entitled **Time**, allows the entry of a time filter.



Time filters make it possible to limit the data to be displayed or exported in terms of time. The settings made here determine the time period of the data shown when calling up a CEL screen.

The time filters available can be distinguished according to:

- ▶ No time filter

The data is shown with no time filtering or limitation in the CEL table.

- ▶ Absolute time period

You define a fixed time period with the absolute time period filter. When the function is executed, the defined absolute time period is exactly used.

- ▶ Relative time period

You define a relative time period with the relative time period filter. This time period is updated constantly and runs with the current time.

Filter profiles in the CEL

You can save filter settings that you have created in Runtime in the filter profiles.

- ▶ Open the filter dialog with the **Filter** button and select another filter.
- ▶ Close the filter dialog with **OK**.
The filter is now applied to the CEL list.
- ▶ You can save this filter in Runtime.
 - Write the name of the filter to the **Filter profiles** text box.
 - Save the filter with the Save button.
- ▶ Switch to a different Runtime screen and then back to the CEL screen again.
- ▶ The **Filter profiles** combo box now provides a selection of the saved filter profiles. In our case, our previously-defined filters.
- ▶ If you select this filter, it is applied to the CEL list again.

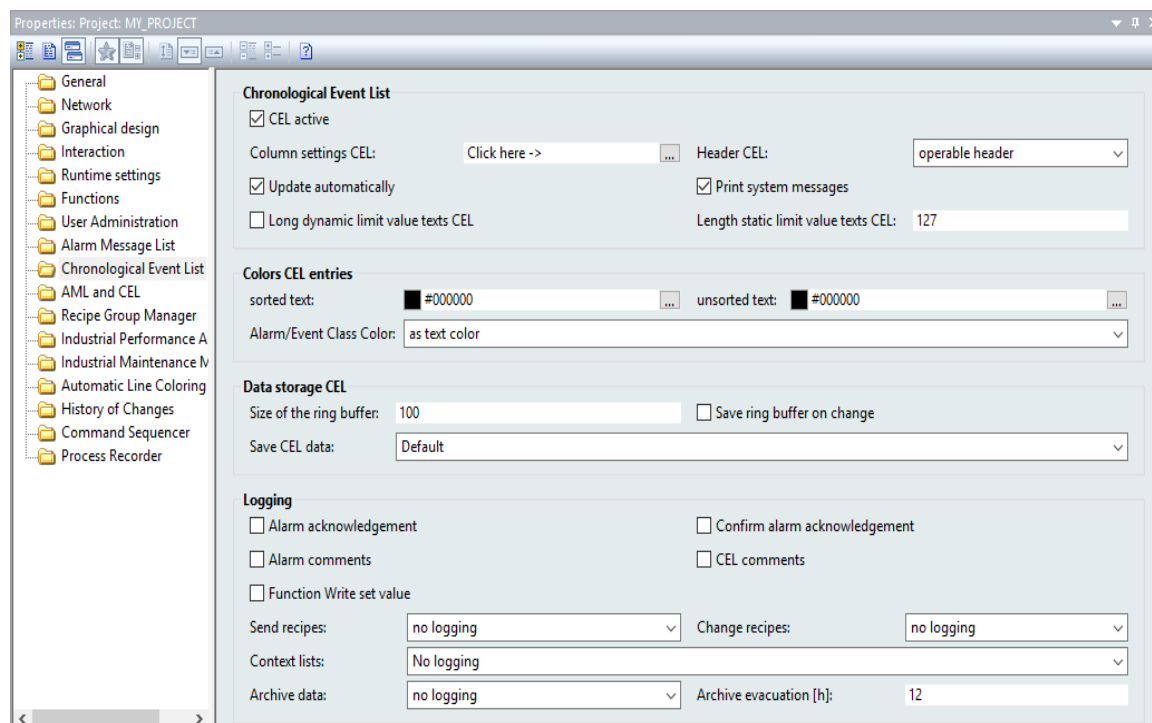


Information

The profiles of all screens that support this functionality are saved in the Runtime directory in the **project.zrt** file. This file is stored in the network on the local computer; these profiles are therefore not available to other clients.

8.3.4 Configuration of the CEL

A detailed or global configuration of the Chronological Event List is possible in the project properties under **Chronological Event List**. You can find more detailed information on the settings options in the help.



8.4 Alarming

8.4.1 Defining alarms

There are two possibilities for defining alarms in zenon:

- ▶ Limit Values
- ▶ Reaction Matrices

Limit Values

Limit values can be defined either centrally in the data type or extra for each variable. You can stipulate, change or delete the limit values in the properties of the variable and the data types.

Limit values are **not** automatically also an alarm; this is controlled using the **In Alarm Message List** property. We have already defined two of our four limit values as alarms.

Reaction matrices (REMAs)

In contrast to limit values, reaction matrices are initially defined independently of variables. They can be linked to one or more variables after being created.

- ▶ Open the **Variables** node in the **Project Manager**.
- ▶ Open the **Reaction Matrix** subnode.
- ▶ Click on the **New Reaction Matrix..** node at the top left of the detail view.

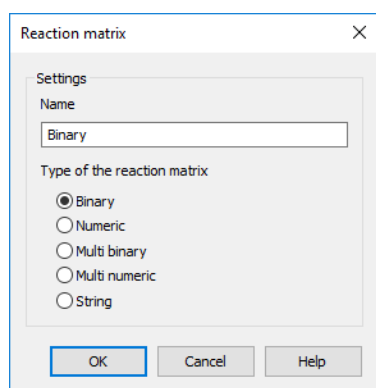
zenon makes a distinction between five types of reaction matrices:

1. Binary Reaction Matrices
2. Numeric Reaction Matrices
3. Multi-binary Reaction Matrices
4. Multi-numeric Reaction Matrices
5. String Reaction Matrices

The Binary and Numeric Reaction Matrices are explained in more detail below.

Binary Reaction Matrix

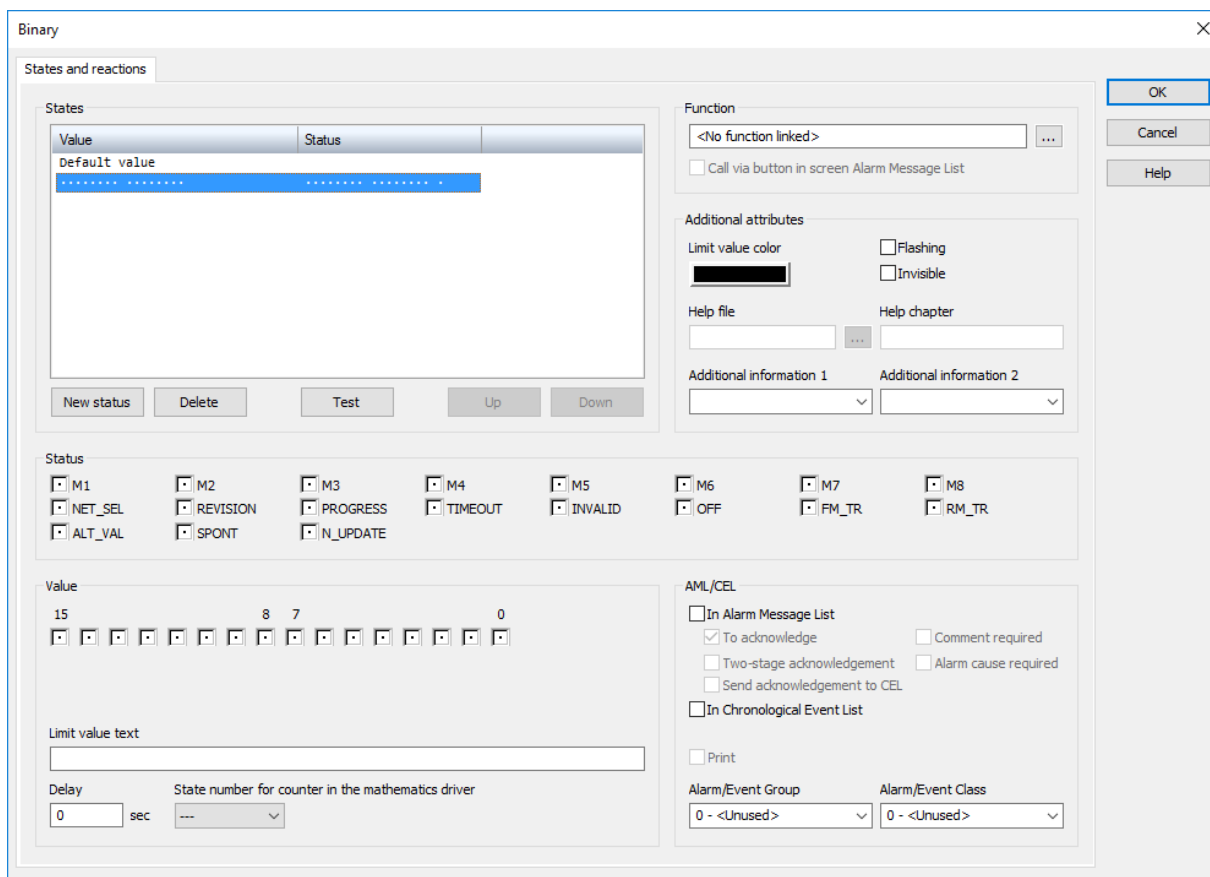
To create a binary reaction matrix, select the **Binary** entry in the dialog.



- ▶ Enter `Binary` as a **Name**.
- ▶ Confirm the settings with **OK**.

A reaction matrix is now created and the dialog box to define the binary reaction matrix is opened.

- Click on the **New status...** button.



A new status for the reaction matrix is created. For each status, you can define a bit pattern, against which the linked variables are checked.

- Set bit 6 and bit 7 to 1 for the first status.
- Switch the option **In Alarm Message List** to active.
- Enter The last bits as **limit value text**.
- Create a second status in which bit 4 and bit 5 are set to 1.
- Enter The other bits as **limit value text** and activate the **In Chronological Event List** option.

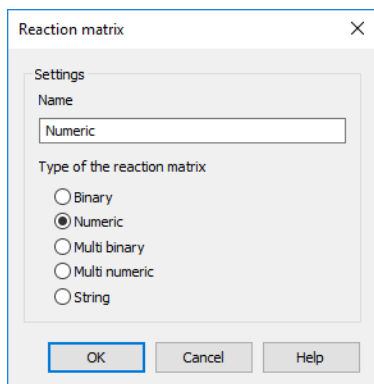
There are now variable values in which both defined states could be true. The sequence of the states is important in this case. The individual states are processed in sequence by zenon, from top to bottom. As soon as the first applicable status is achieved, the actions defined therein are executed. zenon ignores all further states. You can change the order at any time with the **Up** and **Down** buttons.

For each status, colors and texts, as you are familiar with them from the creation of limit values, can be defined.

- Click on **OK** to confirm the settings.

Numeric Reaction Matrix

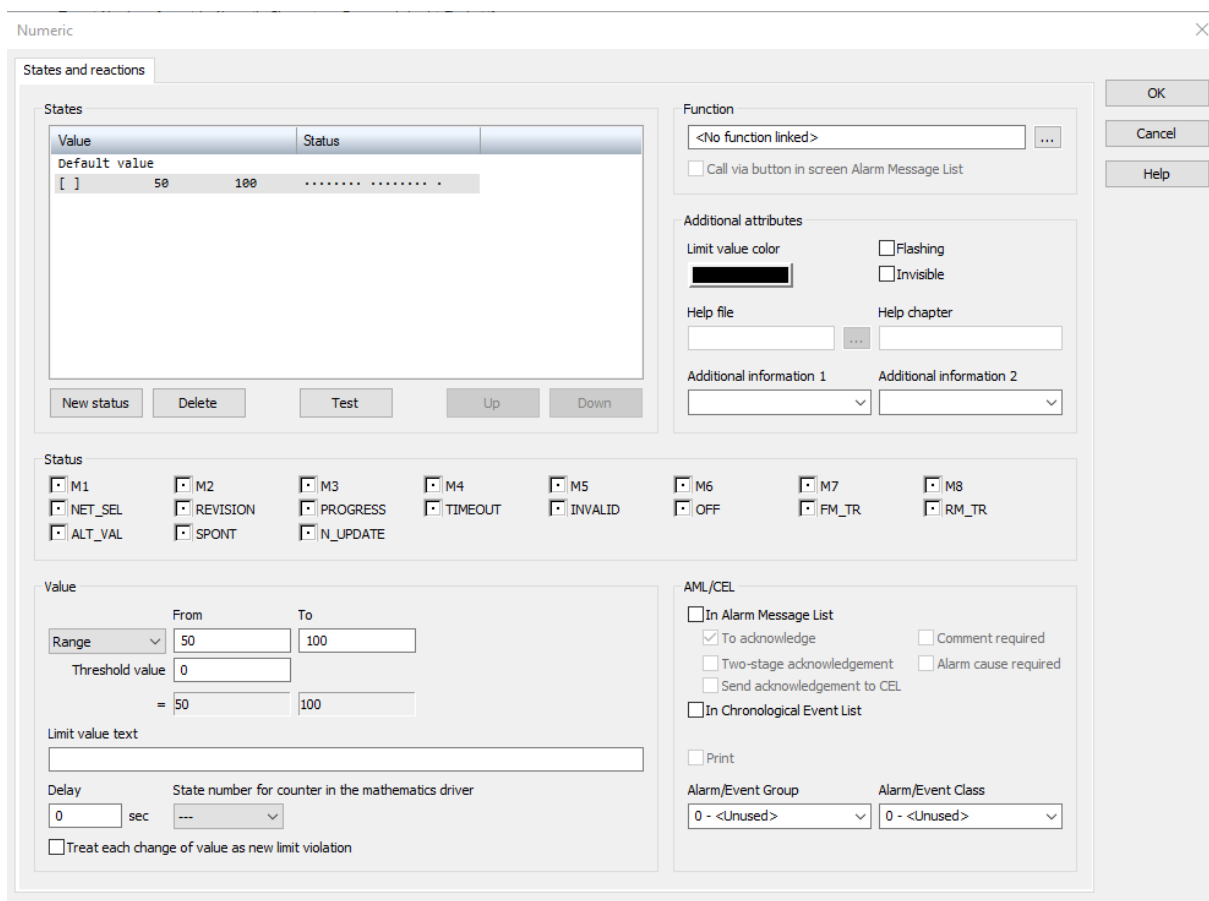
To create a numeric reaction matrix, select the **Numeric** entry in the dialog.



The 'Reaction matrix' dialog box is shown. It has a 'Settings' section with a 'Name' field containing 'Numeric'. Below it, 'Type of the reaction matrix' has five radio buttons: 'Binary', 'Numeric' (selected), 'Multi binary', 'Multi numeric', and 'String'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

- Enter **Numeric** as a **Name**.
- Confirm the settings with **OK**.

The reaction matrix is now created and the dialog box to define the numeric reaction matrix is opened.



The 'Numeric' dialog box is shown. It has a 'States and reactions' tab. The 'States' section contains a table with columns 'Value' and 'Status'. The 'Value' column has a 'Default value' of 50 and a range from 50 to 100. The 'Status' column has a default value of 100. Below the table are buttons for 'New status', 'Delete', 'Test', 'Up', and 'Down'. The 'Function' section has a dropdown menu set to '<No function linked>' and a checkbox for 'Call via button in screen Alarm Message List'. The 'Additional attributes' section has checkboxes for 'Limit value color', 'Flashing', and 'Invisible'. The 'Help file' and 'Help chapter' fields are empty. The 'Additional information 1' and 'Additional information 2' dropdowns are set to '0 - <Unused>'. The 'Status' section shows a grid of checkboxes for various states: M1, M2, M3, M4, M5, M6, M7, M8, NET_SEL, REVISION, PROGRESS, TIMEOUT, INVALID, OFF, FM_TR, RM_TR, ALT_VAL, SPONT, and N_UPDATE. The 'Value' section has a 'Range' dropdown set to 'Range', with 'From' and 'To' fields set to 50 and 100 respectively. The 'Threshold value' is set to 0. The 'Limit value text' field is empty. The 'Delay' is set to 0 seconds. The 'State number for counter in the mathematics driver' is set to 0. The 'AML/CEL' section has checkboxes for 'In Alarm Message List', 'To acknowledge', 'Comment required', 'Two-stage acknowledgement', 'Alarm cause required', 'Send acknowledgement to CEL', 'In Chronological Event List', and 'Print'. The 'Alarm/Event Group' and 'Alarm/Event Class' dropdowns are set to '0 - <Unused>'.

- Click on the **New status...** button.

A new status for the reaction matrix is created.

- ▶ Under **Value**, select `Area`.
- ▶ Enter a value range of 50 to 100.
- ▶ Click on **OK** to confirm the settings.

Link reaction matrices to variables

- ▶ Open the **Variables** node in the **project manager**.
- ▶ In the variable list, select the variables **Temp[1.3]** and **Temp[1.4]**.
- ▶ Select the **Limit values** group in the properties.
- ▶ Under **Reaction matrix**, select the `Numeric` reaction matrix.



Information

It is not possible to link a limit value and a reaction matrix to variable or a data type at the same time. As soon as you link the reaction matrix, the properties for the limit values are grayed out.

8.5 Alarm Message List (AML)

The alarm list is for the administration alarms. It shows alarm messages line by line in the Runtime.

8.5.1 AML screen

To create a screen for the Alarm Message List, we need a new screen with a special screen type.

- ▶ In the **Project Manager**, open the **Screens** node.
- ▶ Create a new screen and give it a unique name.
- ▶ Under **Screen type**, select the `Alarm Message List` entry.
- ▶ Add a screen template using the **Control elements** menu.

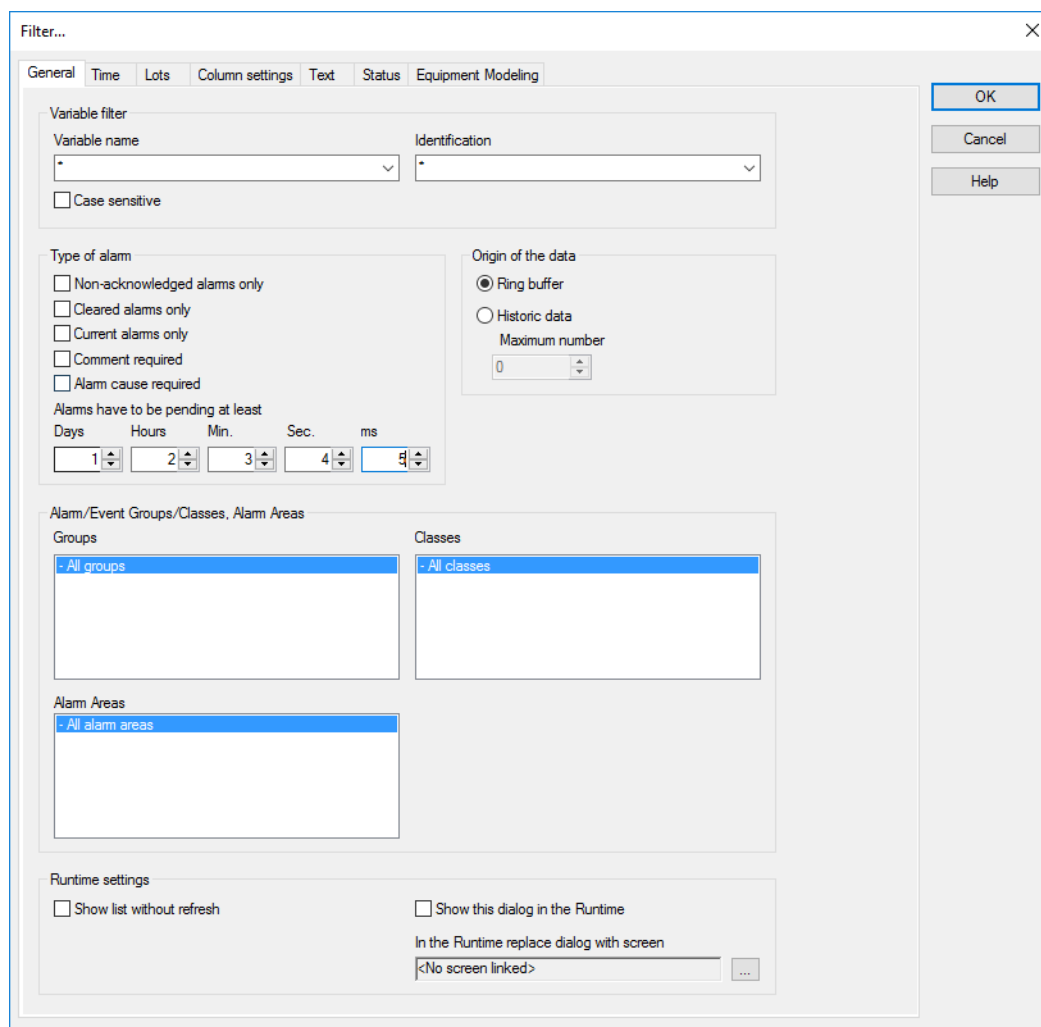
8.5.2 Screen switching - AML

The same as in the CEL, the setting of parameters for the AML is not in the screen itself, but by means of the filter settings for the screen switch function for the AML screen.

- ▶ Create a new screen switch function for the AML screen.
- ▶ You can stipulate the filter settings for the Alarm Message List in the subsequent dialog.

The filter settings are only slightly different to those of the CEL. The differences in the General tab are explained in more detail below.

Filter dialog - General filter



Filter...

General | Time | Lots | Column settings | Text | Status | Equipment Modeling

Variable filter

Variable name: * Identification: *

☐ Case sensitive

Type of alarm

☐ Non-acknowledged alarms only
☐ Cleared alarms only
☐ Current alarms only
☐ Comment required
☐ Alarm cause required

Origin of the data

☒ Ring buffer
☐ Historic data
Maximum number: 0

Alarms have to be pending at least

Days: 1 Hours: 2 Min.: 3 Sec.: 4 ms: 5

Alarm/Event Groups/Classes, Alarm Areas

Groups

- All groups

Classes

- All classes

Alarm Areas

- All alarm areas

Runtime settings

☐ Show list without refresh
☐ Show this dialog in the Runtime
In the Runtime replace dialog with screen: <No screen linked>

OK Cancel Help

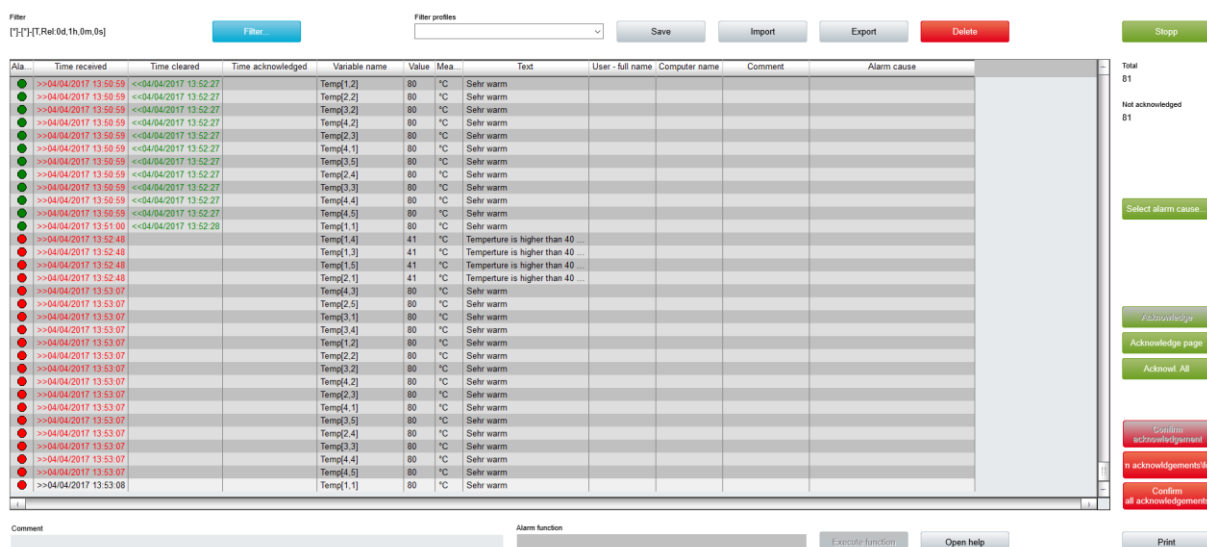
In addition to the already-known filter settings for the CEL, there are further filter options for the AML.

Parameter	Description
Only non-acknowledged alarms	Only alarms that have not yet been acknowledged by the user are displayed.
Only cleared alarms	Only alarms that have already passed, i.e. whose values no longer in the critical range, are displayed.
Only current alarms	Only alarms that are still active, i.e. whose values are still in the critical range, are displayed.
Comment required	Only alarms for which it is necessary to leave a comment are displayed.
Alarm cause required	Only alarms for which it is necessary to leave an alarm comment when acknowledging are displayed.
Alarms have to be pending at least	Use the spin control to define the minimum time that an alarm should be active in order for it to be displayed.

8.5.3 AML in Runtime

- ▶ In the button bar screen, add a button for the screen switching function to the AML screen and give it a unique name.
- ▶ Start the Runtime or click on the Reload button.
- ▶ Open the Alarm Message List by clicking on the new button.

Your Alarm Message List in Runtime should look similar to this.



- ▶ Select an alarm in the list and press the **Acknowledge** button.

If you note the number of unacknowledged alarms at the top left of the screen, you will establish that more than just one alarm has been acknowledged. In fact, all of the same alarms (same variable, same limit value) are acknowledged with this. If, in contrast, you delete an alarm, only the selected occurrence of the alarm from the active alarm is deleted and moved to the historical alarms.

Comments in the AML

You can add a comment for each entry in the Alarm Message List.

- ▶ Select a desired alarm from the list
- ▶ You can now enter free text in the **Comments** field below the list.

If you select the entry again, this text is shown in the **Comment** field. You can also read the text in the Comments column.

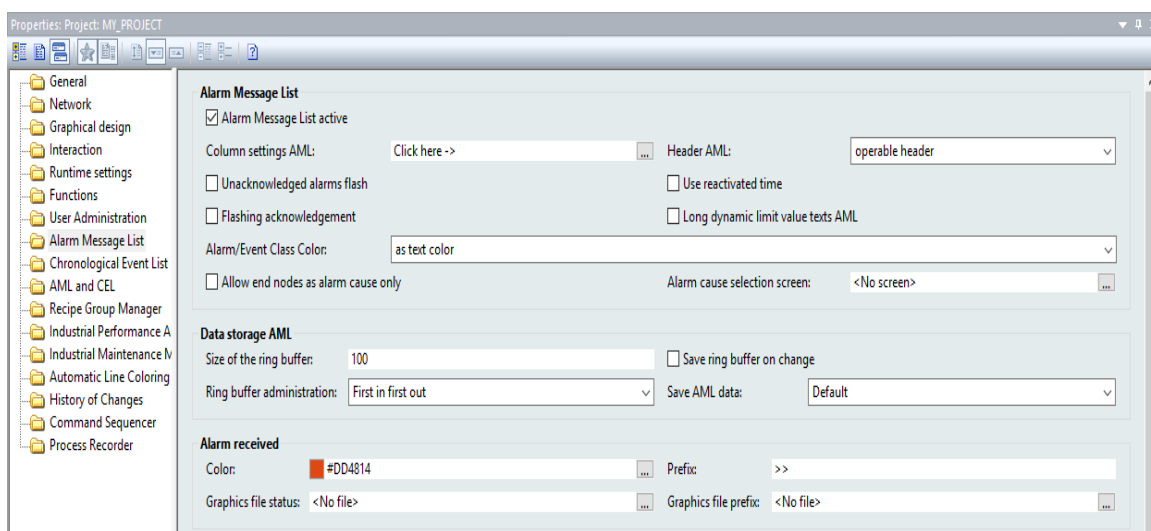


Information

In the Editor, you can stipulate for each alarm whether a comment is required or not. If this setting has been activated, you cannot acknowledge the alarm without a comment.

8.5.4 Configuration of the AML

A detailed or global configuration of the Alarm Message List is possible in the project properties under **Alarm Message List**. You can find more detailed information on the settings options in the help.



8.6 Alarm Cause (Context List)

Alarm Causes are very similar in the comments in the Alarm Message List, however the difference is that the Alarm Causes are predefined and saved in a Context List. The user therefore only has certain selection possibilities for the cause of an alarm and no free text field in which a free comment can be entered. As a result, you automatically reduce the probability of a typing error or an inaccurate comment. This is primarily important for subsequent analyses of the sources of errors or evaluation in reports, for example.

Context Lists generally allow the central administration of hierarchically-structured texts in Runtime. Context Lists are currently only used for the central administration of alarm causes in zenon. These are entered into the context list and can be assigned to an alarm in the Alarm Message List.

The following is applicable for context lists:

- ▶ Levels

Several nodes can be arranged in parallel or hierarchically in a Context List. Each node can contain several entries.

Context Lists are limited to a maximum hierarchy level of 5 levels and the language cannot be switched.

- ▶ Persistence

Context Lists are persistent. They therefore **cannot be deleted**, only hidden from the user. Gaps in reports are thus avoided.



Information

Alarm causes can only be created in Runtime.

8.6.1 Screen Context List

To create a screen for the Context List, we need a new screen with a special screen type.

- ▶ In the **Project Manager**, open the **Screens** node.
- ▶ Create a new screen and give it a unique name.
- ▶ Under **Screen type**, select the `Context list` entry.
- ▶ Add a screen template using the **Control elements** menu.

8.6.2 Screen switch - Context List

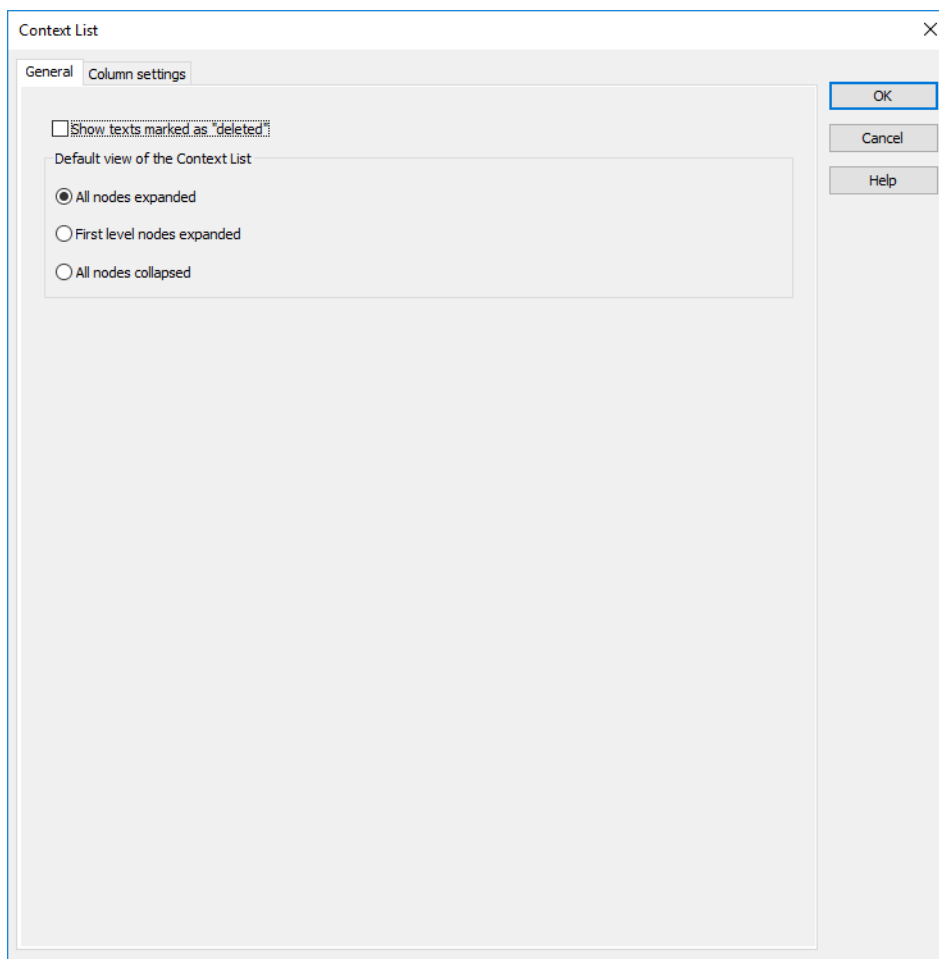
The setting of the content of the context list is not carried out in the screen itself, but in the filter settings of the screen switch function for the screen switch function. This offers you the possibility to access the same screen with different buttons/functions, with different filter settings.

- ▶ Create a new screen switch function for the context list screen.
- ▶ You can stipulate the filter settings for the context list in the subsequent dialog.

The filter dialog consists of several tabs; the General tab will be explained below. You can find information about the other tabs in the zenon Help.

Filter dialog - General filter

In the first tab, entitled **General**, you define which events are displayed and what kind of access you have to the settings in Runtime.

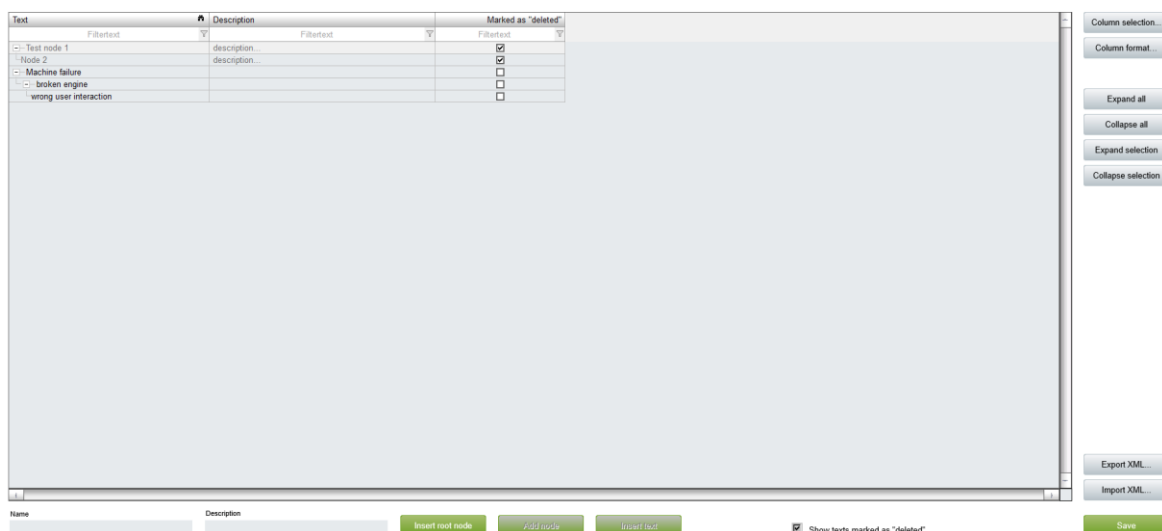


You can set whether as deleted marked entries are shown or not and how the context list is shown by default.

8.6.3 Context List in Runtime

- ▶ In the **Button bar** screen, add a button for the screen switch function to the context list screen and give this a unique name.
- ▶ Start the Runtime or click on the Reload button.
- ▶ Open the context list by clicking on the new button.

Your context list in the Runtime should look similar to this.



- ▶ Click in the **Name** text field to add a new entry.
- ▶ Enter the name of the first entry.
- ▶ If necessary, you can add a description to the entry.
- ▶ You add a new root node with the **root node** button.
- ▶ With the **Add node** button, add a new subnode at the highlighted root node.

Attention

Because you cannot deleted entries in the context list, you should also create these with forethought.

8.6.4 Alarm causes in the AML

You can add an alarm cause for each entry in the Alarm Message List.

- ▶ Open the Alarm Message List.
- ▶ Select a desired alarm from the list
- ▶ Click on the **Select alarm cause...** button.
- ▶ You can now select an alarm cause in the dialog.
- ▶ The selected alarm cause is assigned to the alarm by clicking on the **OK** button.

This text is shown in the **Alarm cause** column whenever you select the entry again.



Information

In the Editor, you can stipulate for each alarm whether an alarm cause is required or not. If this setting has been activated, you cannot acknowledge the alarm without an alarm cause.

8.7 Questions on Event Handling

1. You would like to temporarily change the order of the columns in your Chronological Event List. How to proceed?
 - a) You change the column order in the project properties under "Chronological Event List / CEL column settings".
 - b) You change the filter settings of the screen switch function for the Chronological Event List.
 - c) In the Runtime, you change the column order with the "Filter..." button.
 - d) In the Runtime you change the column order and save.
2. How can you allocate colors to individual alarms?
 - a) In the limit values
 - b) Via alarm classes
 - c) Via alarm areas
 - d) In the screen elements

9. Topic: Operation

Learning objectives:

- ▶ You know that operation is subject to user administration in that there are different authorization levels to which you can assign individual elements.
- ▶ You know what different possibilities there are to write set values, whereby the operability of individual variables or variable groups can be defined via the data type, variable definition or element properties (with inheritance or individually).
- ▶ You understand the correct handling of Runtime-changeable data.
- ▶ You are in a position, with the help of recipes, to define parameter sets and execute them in Runtime, as well as to administer your recipes.

9.1 User Administration

The concept of zenon user administration assumes that different users have different operating rights (authorization levels and function authorizations). Administrators also have different rights, but have additional administrative rights, such as the administration of users. Users can be administered via zenon and the Windows Active Directory.

Users can be grouped in user groups.

9.1.1 Types of login

zenon provides two types of login:

- ▶ Temporary login
- ▶ Permanent login

Temporary login

Temporary login means a temporary login for one individual interaction in the Runtime. If a user in the Runtime encounters a password-protected element, they are asked for their user name and password. They can then execute the action in accordance with their authorization or are notified that they are not authorized. If the action has been completed, the user is logged out again immediately.

Temporary login can be deactivated in the user administration, in the project properties. If it is deactivated, a user who is not logged in cannot execute any protected actions. In this case, they are notified that they are not authorized.

Permanent login

Permanent login describes the one-time login of a user who remains logged in until they log out or the Runtime is closed.

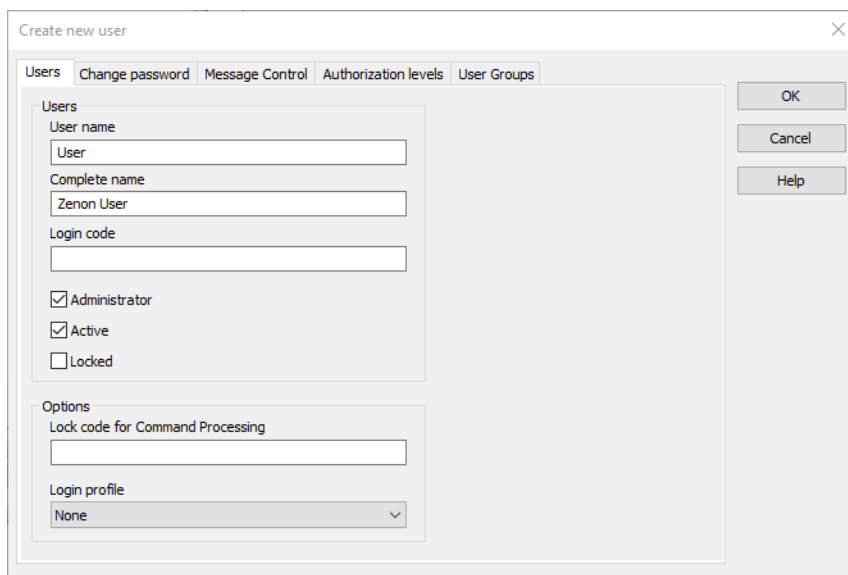
The users log in using a **login** function. If the logged-in user encounters an element for which they have authorization, they do not notice that there is password protection for it. If, in contrast, they encounter an element for which they are not entitled, they immediately get the message that they are not authorized.

zenon knows in this case at all times what the logged-in user has authorization for. Password-protected buttons/menus can therefore be set to invisible for solely permanently-logged in users. This happens in the project properties in the **User administration** properties group using the **Locked buttons** or **Locked menu items** property.

9.1.2 Define users

- ▶ In the **Project Manager**, open the **User administration** node.
- ▶ Select the **User** node.
- ▶ You can now create a new user at the top left in the detail view or by means of the context menu.

The following dialog to define a user now opens:



The image shows a 'Create new user' dialog box with a close button (X) in the top right corner. The dialog has four tabs: 'Users' (selected), 'Change password', 'Message Control', and 'Authorization levels'. The 'Users' tab contains the following fields and options:

- User name:** A text input field containing 'User'.
- Complete name:** A text input field containing 'Zenon User'.
- Login code:** An empty text input field.
- Administrator:** A checked checkbox.
- Active:** A checked checkbox.
- Locked:** An unchecked checkbox.
- Options:** A section containing:
 - Lock code for Command Processing:** An empty text input field.
 - Login profile:** A dropdown menu currently set to 'None'.

On the right side of the dialog, there are three buttons: 'OK', 'Cancel', and 'Help'.

- ▶ Enter the user name in the **User name** field.

- ▶ Give it a **complete name**.

Note: The user name is used in Runtime for logging in, whilst the complete name is shown in various lists.

- ▶ Activate the **Administrator** option.

Note: Only administrators can edit, delete or create other users in Runtime.

- ▶ Define a password for the user in the **second tab**.

Note: The minimum requirement for the password can be changed in the project properties.

- ▶ Define the authorization levels for the user in the **third tab**.

- Assign the user the first ten authorization levels.

- ▶ Confirm the settings with **OK**.



Information

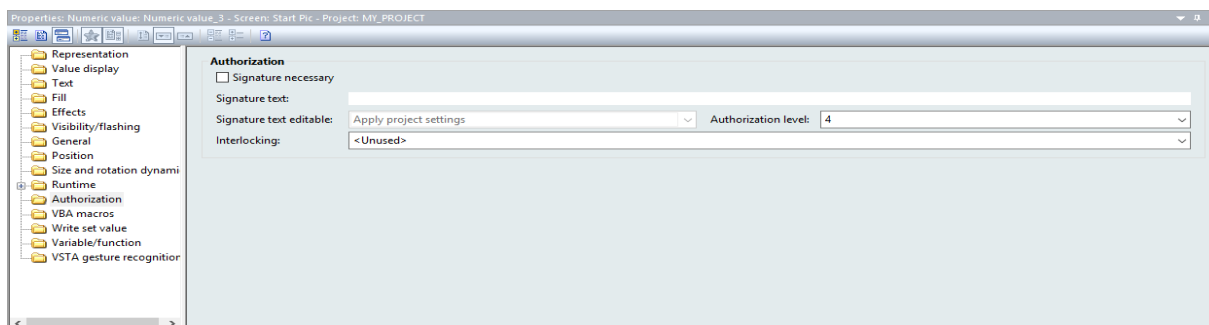
- ▶ You can manage the users' access rights to the elements by means of authorization levels.
- ▶ Each user always has authorization level 0.
- ▶ The authorization levels are in no way hierarchical, i.e. a user who only has authorization level 5 cannot execute an action protected with authorization level 3.

9.1.3 Protecting functionality

In zenon, **authorization levels** are put on elements to protect linked functionality. You cannot protect just a screen, but only the button with which the screen is called up.

- ▶ Open the **Start screen** screen.
- ▶ Select the numeric value element that is linked to the variable **Temp[1.1]**.
- ▶ Open the **authorization** properties group.

- Set the **Authorization level** property to the value 4.



Only users with the authorization level **4** can now change the value of the variable **Temp[1.1]**.

Set the **authorization level** of the elements with the variables **Temp[1.2]** and **Temp[1.3]** to the value 7.

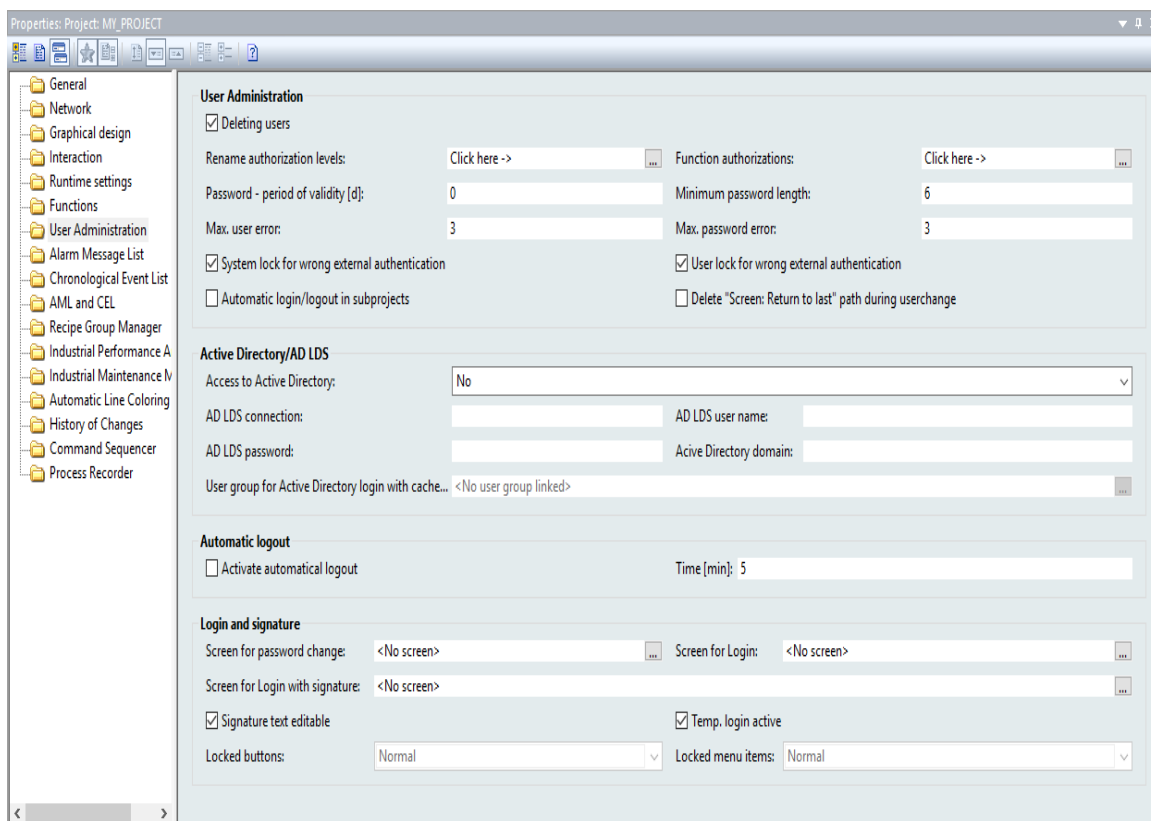
Signature

Particularly important values or functions can also be protected by means of signing. In this case, the user must enter their password again, even if they are logged in and have the appropriate rights. In addition, an entry in the **Chronological Event List** is created here for traceability.

- For the element of the **Temp[1.3]** variable, activate the **Signature required** property.
- As a **signature text**, enter *Yes, I have done it.*

9.1.4 Configuration of the user administration

You set the configuration of the user administration in the project properties under **User administration**.



The screenshot shows the 'Properties: Project: MY_PROJECT' dialog box with the 'User Administration' tab selected. The left sidebar lists various project settings categories, with 'User Administration' highlighted. The main area contains the following configuration options:

- User Administration**
 - ☒ Deleting users
 - Rename authorization levels: Click here -> [button]
 - Function authorizations: Click here -> [button]
 - Password - period of validity [d]: 0
 - Minimum password length: 6
 - Max. user error: 3
 - Max. password error: 3
 - ☒ System lock for wrong external authentication
 - ☒ User lock for wrong external authentication
 - ☐ Automatic login/logout in subprojects
 - ☐ Delete "Screen: Return to last" path during userchange
- Active Directory/AD LDS**
 - Access to Active Directory: No [dropdown]
 - AD LDS connection: [text field]
 - AD LDS user name: [text field]
 - AD LDS password: [text field]
 - Active Directory domain: [text field]
 - User group for Active Directory login with cache... <No user group linked> [button]
- Automatic logout**
 - ☐ Activate automatical logout
 - Time [min]: 5 [text field]
- Login and signature**
 - Screen for password change: <No screen> [button]
 - Screen for Login: <No screen> [button]
 - Screen for Login with signature: <No screen> [button]
 - ☒ Signature text editable
 - ☒ Temp. login active
 - Locked buttons: Normal [dropdown]
 - Locked menu items: Normal [dropdown]

- Activate the **Signature text editable** property.

The above-defined **signature text** is then not simply written to the **Chronological Event List**. Instead, a dialog is opened in Runtime, in which we can edit the text beforehand.

9.1.5 Functions of the User Administration

- In the **Project Manager**, open the **Functions** node.
- Create a new function, select the **User administration** node and then the **Login with dialog** function.
- In the **Button bar** screen, add a new button for this function and give it a unique name.
- Create the **Logout** function. Create a new button for this function.
- Create the **Change user** function. Create a new button for this function.

For a better overview in Runtime, we want to have the name of the currently-logged-in user shown to us by a system variable.

- ▶ Create the system variable **User full name**.
- ▶ In the **Start screen** screen, add a new dynamic text field and link it to this variable.

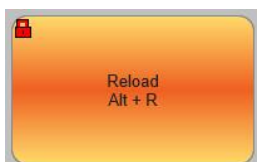
9.1.6 Graphic display of blocked elements

zenon makes it possible to highlight blocked elements graphically.

- ▶ Select the **Graphical design** group in the project properties.
- ▶ In the **Locked/Interlocked elements**, activate the **Graphical identification active** property.

In principle, you can select between graphics files and a lock symbol. Furthermore, you can select colors for the lock symbol.

The display of a locked button, via the lock symbol, looks like this:



9.1.7 Runtime changeable data

Users can be created, edited and deleted in both the Editor and in the Runtime. If you create new users in the Runtime, they do not yet exist in the Editor. When creating new runtime files from the Editor, these new users would disappear. To prevent this unwanted deletion of users, zenon contains a safety mechanism.

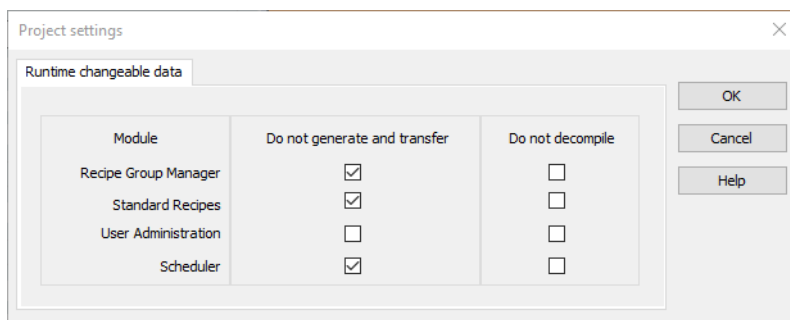
Before we can look at the user administration in the Runtime, we must still carry out some preparation.

- ▶ Create new Runtime files.
- ▶ If it is not yet open, have the **output window** displayed via the **Option** menu.

Here we can see that the password.cmp file will not be overwritten. The editor user data will thus not be available in Runtime. We need this data in the Runtime however.

- ▶ Open the **General** group in the project properties.

- Open the following dialog in the **Runtime changeable data** property.



- Remove the tick for the **User Administration**.
- Create new Runtime files again.

In the output window, you can see that **password.cmp** is now actually created. If you want to ensure that future changes are not overwritten in the Runtime, you can set the tick again.

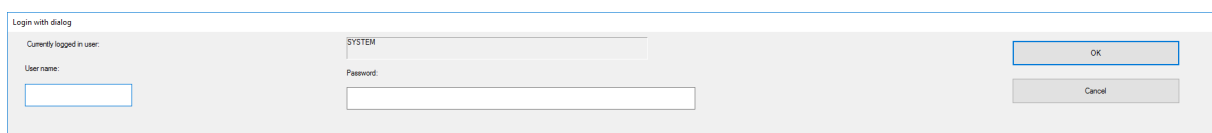
9.1.8 User administration in the Runtime

- Start Runtime and/or reload the project in the Runtime.

The lock symbol shows us on the **Start screen** that elements for the variables **Temp[1.1]**, **Temp[1.2]** and **Temp[1.3]** cannot currently be operated. The system variable **Complete user name** shows us that the user SYSTEM user is currently logged in.

- Click on the element for the **Temp[1.1]** variable.

With the following dialog, you are now requested to enter your user name and password.



Note: Here you can see the functionality of the temporary login.

- Enter user name and password.

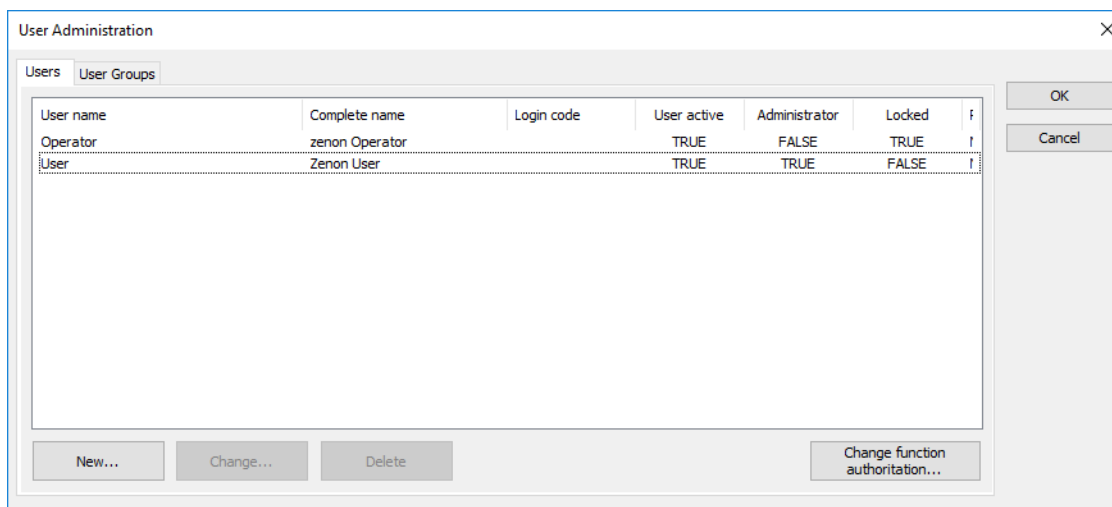
Only now can you write the set value of the variable. However, our dynamic **user** text field still displays **SYSTEM**. By clicking on the element directly, you are only logged in temporarily and are logged out again immediately. Take a look at the Chronological Event List; you can find a corresponding entry there. All actions in relation to the user administration are automatically logged there.

- Click on the **Login** button.

The same dialog opens. However, you are permanently logged in this time, which is also shown in the **User** dynamic text field.

- Click on the **Change user** button.

The following dialog opens:



We now want to create a new user.

- ▶ Click on the **New...** button.

You are familiar with the dialog box that now opens from the Editor.

Create a new user.

Note: Because, even as an administrator you only gave authorization levels 1 to 10, you also cannot approve other authorization levels for other users.

- ▶ Log in with the new user.

You are requested to change your password immediately, because not even the administrator should know the password of other users.

Security mechanisms:

- ▶ Log in again with this user, but intentionally mistype the password.
An error message is shown for a login attempt with an incorrect password.

- ▶ Login in a second time with an incorrect password.

You are informed that the user has been blocked, because only three incorrect logins are permitted. You can define the number of login attempts using the project properties.

- ▶ This is how you unlock users:

- Log in again as the Administrator user, this time with the correct password.
- Click on the **Change user** button.
- Select the blocked user and click on the **Change...** button.

You now see that the user is blocked and can unlock them here again. A look at the CEL shows you that our actions have been logged.

- Log in with an incorrect user name.

An error message is shown for a login attempt with an incorrect user name.

- Log in twice with an incorrect user name.

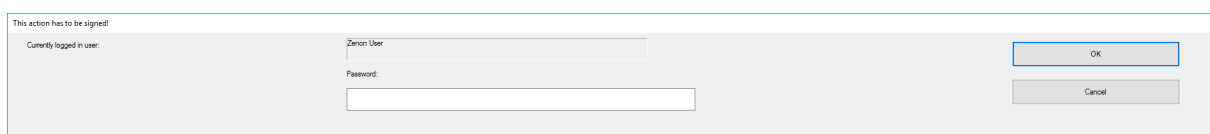
You are informed that the system has been blocked, because only three consecutive incorrect user entries are permitted.

The system is only unlocked again when an administrator logs in.

- Ensure that you are logged in with your Admin user.

- Click on the element with the **Temp[1.3]** variable.

Although you have the rights, to set this value, you are nevertheless asked for the password. However, this time you do not need to log in; you are asked for your signature.



Confirm with OK; you can now change the value.

9.1.9 User Administration screen

There are also special screen types available for the display and administration of users. Users and user groups can be created, edited and deleted in the Runtime.

- Switch back to the Editor.
- Create a new screen with the special **User Administration** screen type and add a screen template.
- Create the corresponding screen switch function with a button in the **Button bar** screen.
- Create the Runtime files.
- Start the Runtime.

Note: If you do not want to overwrite the users who have previously been overwritten in Runtime, you must activate the **Runtime changeable data** setting again.

- Switch to Runtime and create another user - using the new screen.

9.1.10 Read back data that can be changed in Runtime

Our two new users currently do not exist in the Runtime. However, we need these under certain circumstances in the the Editor too, be it because we want to make a backup of it or because we want to convert the project to a more recent zenon version later.

- ▶ Change back to the Editor.
- ▶ Open the project's context menu and the **Runtime files** submenu there.
- ▶ Go to the **Import submenu**.
- ▶ The data is read back from the Runtime by clicking on Import.

Because this is only supposed to happen deliberately, a request for confirmation is made before reading it in.

- ▶ Confirm the query with **Yes**.
- ▶ In the **Project Manager**, open the **User administration** node.

Select the **User** node.

All users will now be shown. The Admin user, who we created in the editor, but also our two users from the Runtime.

9.2 Recipes

Recipes collect set values in a list of required values that can be executed in the Runtime by means of a function. Recipes can be created in both the editor and in the Runtime.

A collection of several recipes is created and administrated with the help of the Recipegroup Manager

Note: You need an extra license for the Recipegroup Manager; the standard recipes are included as standard.

9.2.1 Creating Recipes

- ▶ In the **Project Manager**, open the **Recipes** node.
Here, you can find, firstly, the **Standard Recipes**, which are the subject of this section, and secondly, the **Recipegroup Manager**, an optional module that must be licensed separately.
- ▶ Select the **Standard Recipes** subnode.
- ▶ Open the context menu of the recipe to create a new recipe.
- ▶ Select the **New recipe...** entry.

A new recipe, named **Recipe 0**, is created.

- ▶ Change the name of the recipe in the properties to **Cake**.

Note: As an option, you can also enter an authorization level for the recipe. Only users with the corresponding authorization level can then edit this recipe in the Runtime. However, all users can still execute this recipe.

- ▶ Click on the entry for our **Cake** recipe.

- ▶ Add variables using the **Add variable** button.

Select the variables **Temp[1.1]** to **Temp[1.5]**.

- ▶ You can now enter values for the recipe in the **Set value** column. The values that you set for the respective variable here are written to the variable when the recipe is executed.



Information

The standard recipes do not permit any selection of string variables. If you need this functionality, you can use the optional Recipegroup Manager module.

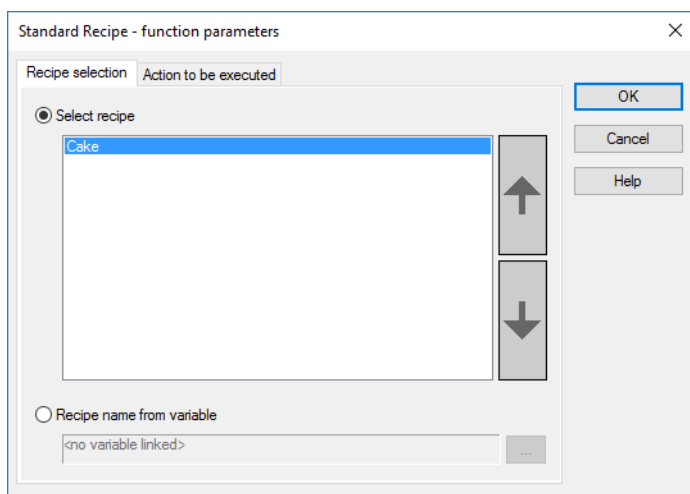
9.2.2 Use of Recipes

zenon offers two possibilities for using recipes in Runtime:

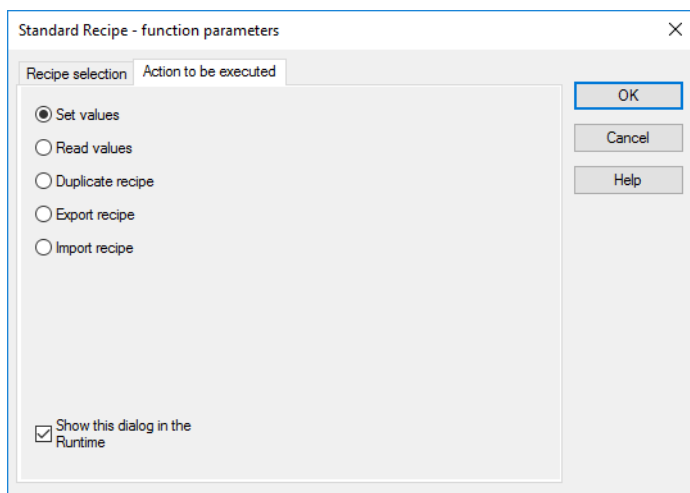
- ▶ A function
- ▶ A special screen type

Standard recipe function

- ▶ Create a new function.
- ▶ In the function node **Recipes** select the **Standard recipe** function.
- ▶ Select our **Cake** recipe in the subsequent dialog.



- In the second tab, select the **Set values** setting.



These functions offer the possibility to, for example, write the values of the recipe to the corresponding variables or to read the current values of the variables and to have this written to the recipe. However, we can have it duplicated or export and import the recipe.

This dialog can also be opened in the Runtime before execution of the selected action; the selection would thus be made in the Runtime.

Recipe screen

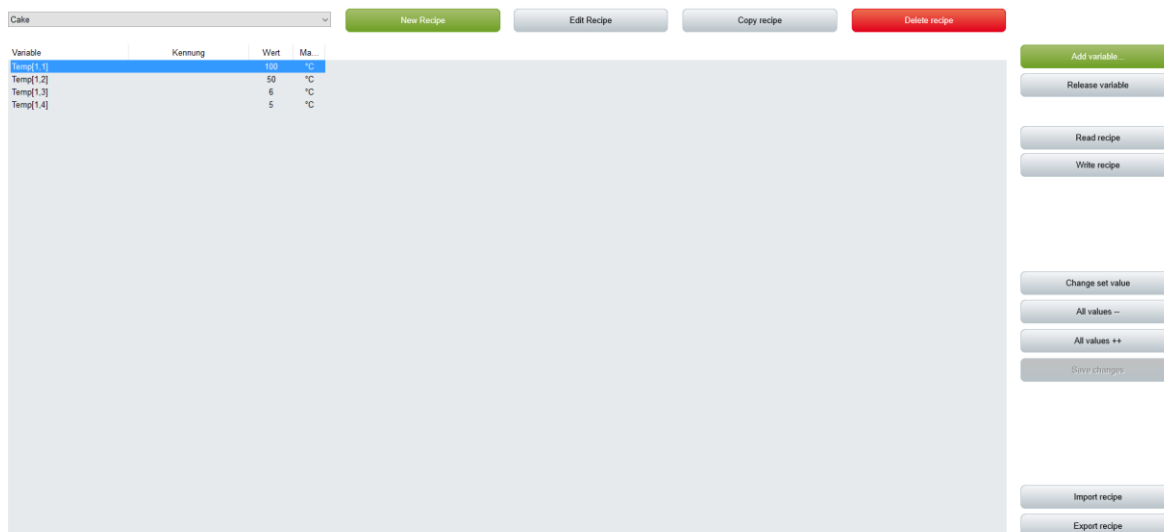
- Create a new screen with the special **Standard Recipe** screen type and add a screen template.
 - Create the corresponding screen switch function with a button in the **Button bar** screen.
- Accept the default settings in the filter dialog.

Attention: Before you create the new Runtime files, you must – as with the user administration – change the **Runtime changeable data** setting.

9.2.3 Recipes in the Runtime

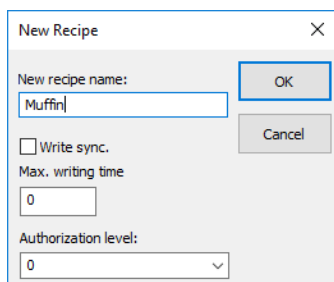
- Start the Runtime or reload the changes.
- Open the recipe screen.

Your recipe screen in the Runtime should look something like this.



You can write the values of the selected recipe to the variables using the **Execute recipe** button. With the **Read recipe** button, the current values are read into the recipe and the recipe can then be saved with the **Save changes** button.

- Click on the **New recipe** button.



- Create a new recipe with the name **Muffin** and confirm with the **OK** button.
- Add any desired variables using the **Add variable** button.
- Change the set values in the table directly.
- Click on the **Save Changes** button to save the changes.

We have thus created a new recipe in the Runtime, which is not yet in the Editor. Just as with the users, you can also read back recipes into the Editor.

- Go back to the Editor.
- Open the project's context menu and the **Runtime files** submenu there.
- Go to the **Import** menu item.
- The data is read back from the Runtime by clicking on Import.

Because this is only supposed to happen deliberately, a request for confirmation is made before reading it in.

- ▶ Confirm the query with **Yes**.
- ▶ In the **Project Manager**, open the **Recipes** node.

Select the **Standard Recipe** node and check whether your recipe has been imported.

A second way to export a recipe is via data import/export in the Runtime.

- ▶ Switch back to the Runtime.
- ▶ Click on the **Export recipe** button.

The recipe currently shown is exported to a text file. You can edit this file externally. You can find the file in the `Workspace/project/export` folder.

You can now edit and reimport this file. Recipes can be edited in an automated manner this way.

- ▶ Open the file with the Windows Notepad.
- ▶ Change the name of the recipe to **Bread** in the exported file.
- ▶ Change the values of the recipe, but not the variables.
- ▶ Save the file with a new name.
- ▶ Switch back to Runtime.
- ▶ Import the new recipe.
- ▶ Save the changes.

Your recipe list should now include 3 recipes.



Attention

What is important when editing a recipe with the Editor is that the last line ends with an Enter.

9.3 Questions about Operation

1. In a network project, you use internal driver variables for interlocking elements. How do you have to configure the variable so that the interlocking takes place on all computers?
 - a) As local variable
 - b) As remanent

- c) As network variable
 - d) Internal variables cannot be used for interlocking.
2. What types of login are offered by zenon?
- a) Local login
 - b) Temporary login
 - c) Global login
 - d) Permanent login

10. Topic: Network

Learning objectives:

- ▶ You know that a zenon network is platform-independent and version-independent, and you are familiar with the requirements for a network project.
- ▶ You are familiar with the different setups in zenon networks, such as client-server networks or distributed networks.
- ▶ You can set up a redundant system and understand circular redundancy.
- ▶ You know how the server and standby server sync in Runtime.
- ▶ You get to know remote transport as a possibility for the transfer of Runtime files.

10.1 General

The network functionality of zenon makes it possible to deploy projects in a distributed manner on different computers. You can thus create very efficient, complex network setups with it. In doing so, setups can also be configured in such a way that project content, for example, is only visible on a certain computer. The zenon Editor supports users in creating and configuring such configurations.

The integrated topology administration creates interrelationships for the individual projects in the process, with the attendant computers in graphical form. A testing routine checks the configured structure to see that it is complete and that there are no configuration errors.

With the network nodes function, zenon also checks to see if the selected network topology can work.

zenon networks can be set up quickly and securely; they allow you, among other things, to do the following:

- ▶ Full access to the Runtime of different computers
This way, actions such as the acknowledgment of alarms at a workspace on all other computers in the network thus become visible.
- ▶ Centralized logging and archiving
- ▶ Creation of redundant systems
- ▶ Redundancy switching with integrated rating methods
- ▶ Setup of distributed systems
- ▶ Use of strong encryption
- ▶ Concurrent work on a project, on several computers

10.1.1 Requirements

A requirement for network operation in zenon is a functional Windows network. In detail the following requirements have to be fulfilled:

1. TCP/IP as the network protocol
2. A functional naming resolution
The naming resolution can be solved as DNS, WINS or local HOST files.
3. Administrator rights
There must be at least local administrator rights during installation on individual computers, because zenon must register two services for operation in the network.
4. Ports
 - a) When using a firewall, the ports 1100, 1101 and 1102 for the web server must be unlocked



Information

*You can easily check at least the first two of these requirements by opening the DOS command prompt (cmd) and entering the command:
ping COMPUTER NAME and receiving a positive response.*

zenon needs two services for operation in a network, which are automatically installed with zenon:

Parameter	Description
zenSysSrv.exe	This service is automatically started with the operating system and is required for Editor network communication. This service uses TCP port 1101, which must not be blocked by any other application.
zenNetSrv.exe	This service is automatically started with zenon Runtime and is responsible for Runtime network communication. This service uses TCP port 1100, which must not be blocked by any other application.

10.1.2 Network topologies

zenon supports two network topologies:

Parameter	Description
Client-server network	Here, one and the same project runs on the server and on all clients.
Multi-server network	Here, a client can access several servers at the same time and thus display the data from different projects at the same time.



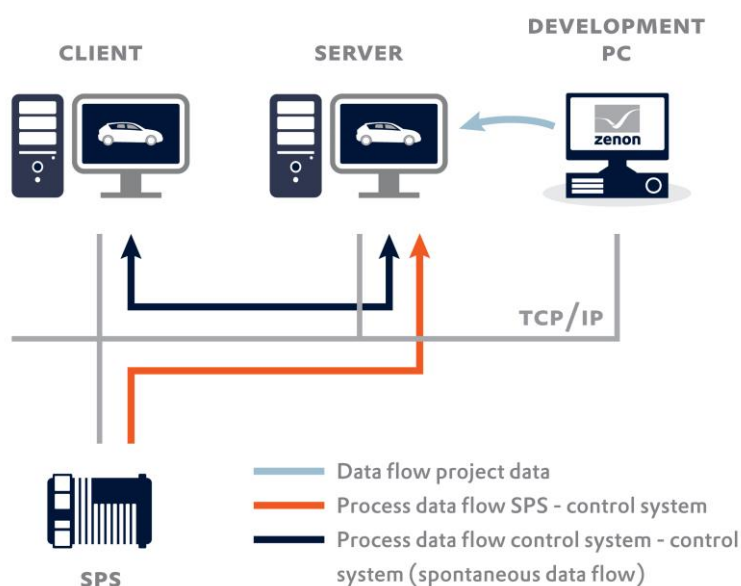
Information

These two topologies can also be mixed at any time!

10.2 Client-server network

zenon makes it easy to set up a server-client network: In the project properties in the network group, one computer is defined as the server. This means that only one computer has a direct connection to the hardware, to the PLC. Even more, it also has the full and exclusive administration of the process data (e.g. online data, archive data, alarms, recipes, etc.) and the project data (e.g. screens, functions, variables, etc.).

Each other computer, which starts the same project, is automatically recognized and defined as a client. Each time Runtime is started, this client establishes a connection to the server, syncs the project data and displays the current process data.



10.2.1 Configuring the server

- ▶ Open the project properties of your current zenon project.
- ▶ The network settings can be found in the **Network** group.
- ▶ Activate the property **network active**.

Now all the other properties of this section are available and we can go on configuring the server.

Note: You will find a more detailed explanation of the single properties in the property help.

- ▶ In the **Server 1** property, enter the name of the computer that has the connection to the hardware (PLC).

Attention: The IP address is not sufficient here! The name of the computer must be entered.

You can select the computer name either from the list offered from the [...] button or type it in manually.

If the development station, on which you created the project, is the Runtime server at the same time, the configuration of the server now is complete. At the moment we leave the other properties unchanged.



Information

*If you do not know the computer name, open the DOS command prompt (cmd) and enter the **hostname** command.*

10.2.2 Transferring the Runtime files with Remote Transport

- ▶ On the development station create the Runtime files of the project.



Information

Password files, recipes etc. can be write-protected and are not created, which can lead to error messages later.

*In order to be able to generate these files too, you must change **Runtime changeable data**. Deactivate all options.*

We now want to transfer the Runtime files to our server. To do this, we need the **Remote Transport** tool bar.



- ▶ Define the server's computer name and the destination location for the files.

- Click on the first icon, **Remote Transport: Connection Settings** of the **Remote Transport** tool bar.
- In the **Project settings** dialog, in the **Computer name or IP address** field, enter the computer name of your server.
- Confirm your entry with the **OK** button.
- ▶ Establish a connection to the server.
 - Click on the second icon, **Remote Transport: Establish connection**.
 - In the **Establish connection** dialog, you can define a password; no password is defined as standard.
 - Do not make any changes and close the dialog with **OK**.

In the output window of the Editor, you now get information about the computer to which you have established a connection.

After the connection has been established successfully, the other icons in the **Remote Transport** tool bar are available.



With the previous steps, you have established a connection to the sever, but not yet transferred any data.

- ▶ Click on the button **Remote: Transfer changed Runtime files**.
All Runtime files are thus transferred to the server.
- ▶ Set the start project on the server with the following button: **Remote: Set start project**.
- ▶ You can now start Runtime on the server. Click on the **Remote** button: **Start Runtime**.
- ▶ With the buttons **Remote: exit Runtime** and **Remote: reload project**, you can remotely control Runtime on the server.

10.2.3 Configuring the clients

A client can be set up in two ways:

1. The development computer transfers the Runtime files to the client, via the **Remote Transport** tool bar.
2. The client gets the data from the server manually.

Client gets the files from the development computer

Carry out all steps from the **Transferring the Runtime files with Remote Transport**, with the client's computer name.

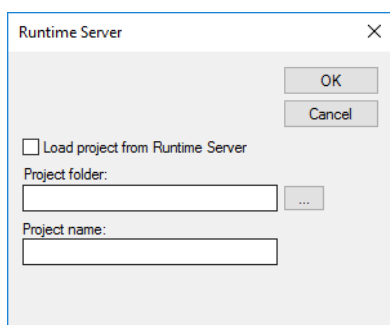
Client gets the files from the server

- ▶ Close the Editor and the Runtime on the client.
- ▶ Open the Windows Explorer.
- ▶ Switch to the %CD_SYSTEM% directory.
- ▶ Open the file **zenOn6.ini** with a text editor.
- ▶ Remove the line **VBF30=....**

The project that the Runtime loads on starting is saved in this line. Once we have removed this line, the Runtime no longer knows which project it should load.

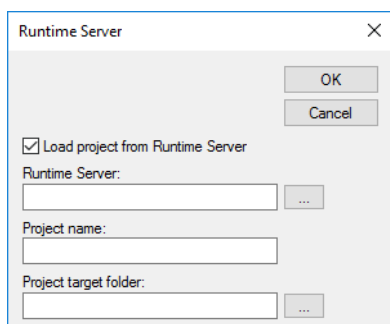
- ▶ Start the Runtime using the Startup Tool (without opening the Editor).

The Runtime now asks, in a dialog box, which project is to be loaded.



- ▶ Activate the **Load project from Runtime server** setting.

The appearance of the dialog box has now changed and allows us to enter the necessary parameters for setting up a client.



- ▶ Enter the server of the project under **Runtime Server**.

You can either type in the computer name or select from the list offered using the ... button.

- ▶ Under **Project name**, enter the name of the project that is running on the server.

- ▶ Stipulate the local directory under **Project destination folder**.

You either can select an existing directory using the ... button or type it in by hand. If you type in the directory, it can also be a directory that does not yet exist. It is created automatically.

- ▶ Confirm the settings with **OK**.

The Runtime now establishes a connection to the server and copies the Runtime files from there to the project target directory. The Runtime is started once the copy process has been completed successfully.

The entry **VBF30=...** in the file zenOn6.ini is written to the project target directory, so that Runtime starts the network project on the client automatically after each start.

Repeat this process for each further client.

The behavior in the Runtime

The project can be operated from the server and from all clients in exactly the same way. Generally speaking, the user cannot tell if they are working on the server or on a client.

However, only the server has a connection to the hardware and thus administers the process data. This means that the clients get the current values of the variables, the CEL system messages, AML alarms, recipes and archived data from the server only. This is done spontaneously and event triggered.

Monitoring the connection:

When using the default setting of 30 seconds for the **Network communication timeout** property in the Startup Tool, the network service (zenNetSrv.exe) of each client sends a watchdog to the network service (zenNetSrv.exe) of the Primary Server every 10 seconds during online operation. If the Primary Server responds to at least one of the three watchdogs within the 30 seconds, the client assumes that the network connection is working. If there is no response, the clients display blue squares at the top right corners of the elements. This should clearly show that the clients do not get any current values from the server.

This standard setting can be changed in the Startup Tool in the network configuration:

- ▶ Start the Startup Tool
- ▶ In the **Application** menu, select the "**Options**" option
- ▶ Open the **Network configuration** entry.

Here, you can change the Timeout property, which is set to 30 seconds by default

Project changes in online operation

You can make changes in the Editor at any time whilst a project is running on the server and/or on the client. The updating of Runtime files is possible without restarting Runtime. Once you have carried out the desired changes in the Editor, proceed as follows:

- ▶ Create the desired Runtime files on the development computer.
If the development computer is the project server at the same time, you only need to switch to Runtime and reload the changes. Otherwise:
- ▶ Establish a remote connection to the server.
- ▶ Send all Runtime files to the client with Remote Transport.
- ▶ Carry out the reloading on the server with Remote Transport.
- ▶ Stop the online connection.

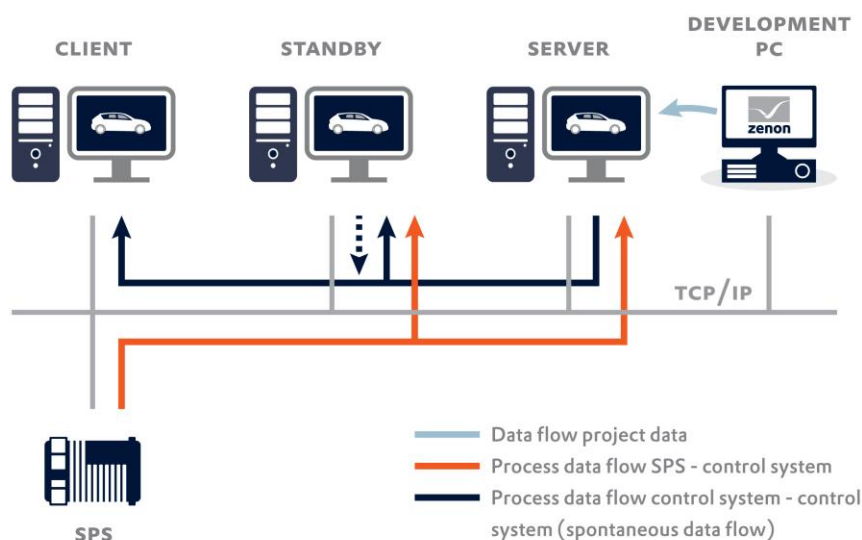
You do not need to worry about the clients. The changes in the project are automatically transferred to the connected clients by the server and updated there. If a client is not online during the changes, they automatically get (only) the changed project data from the server the next time Runtime starts.

10.3 Client-server network with redundancy

Redundant SCADA networks are used if 100% process control and data security is demanded, even if a server fails.

You achieve this fail safety by defining a second server, a so-called standby server, along with the project server. This standby server automatically recognizes a server failure and automatically assumes the complete functionality of the server.

In order to avoid data loss in the time period between the server failure and the detection of the failure, the Standby Server always buffers all data. After a failure of the server, this buffer is merged with the last data from the server and the new incoming data, so that no data is lost. zenon thus guarantees seamless redundancy.



As soon as the server is running again and has obtained the Runtime data from the standby, the standby automatically becomes the standby only again.

10.3.1 Configuring a Standby Server

- ▶ Open, in the project properties in the Editor, the **Network** group.
- ▶ In the **Server 2** property, enter the computer name of the Standby Server.

zenon offers different redundancy modes. You determine the behavior of both defined servers:

Parameter	Description
Non-dominant	After a failure of Server 1, Server 2 remains the primary server until it fails itself.
Dominant	Server 2 takes over after a failure of Server 1. If Server 1 comes back onto the network again, it automatically takes on the role of the primary server again.
Evaluated	Definable metrics are used to determine which of the two defined servers is the primary server.

10.3.2 System variables

zenon offers you a number of variables, with the help of which you can get an overview of your network.

- ▶ Create a new variable and select the system driver.
- ▶ Select the **Network** theme.

You now see an overview of the pre-defined variables for networks.

- ▶ Select the variable **Current Primary Server** and click on **Add ->**.
- ▶ Select the variable **Current Standby Server** and click on **Add ->**.
- ▶ Select the variable **Names of the connected clients** and click on **Add ->**.

These variables are string variables; they can be shown with the **dynamic text** element on the start screen of the **INTEGRATION_PROJECT**.

- ▶ Add the **dynamic text** elements for the three variables.
- ▶ Start the Runtime.

On the start screen of the **INTEGRATION_PROJECT**, you see the computer names of the server and standby in the new text fields, as well as the names of all connected clients.

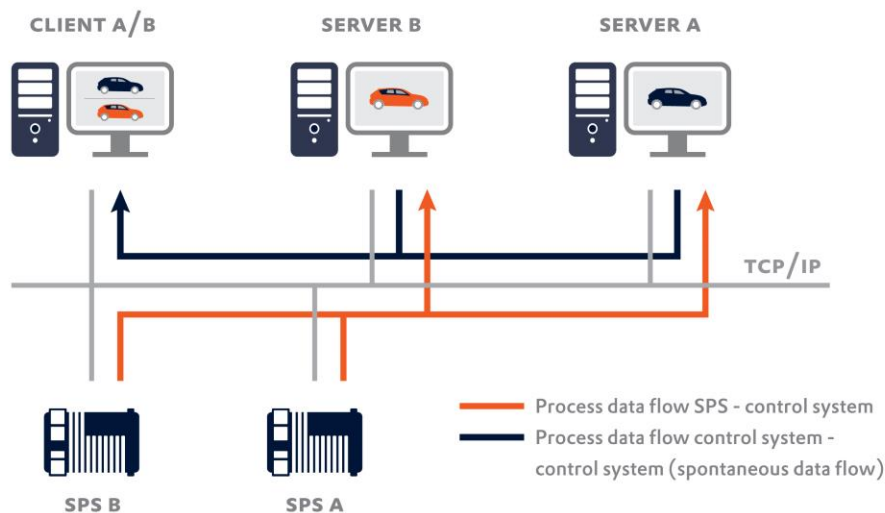
- ▶ Remove the network cable from the computer that has the role of server.
- ▶ Go to the second computer.

After a maximum of 30 seconds, the standby takes on the role of the server.

10.4 Multi-server network

zenon offers the possibility of running more than one project on a computer at the same time. Due to the exact definition of the server, standby and the clients in the project, it is possible to have different servers, clients and standby projects running at the same time together on one computer.

In conjunction with the ability of the project manager to create hierarchical tree structures, known as integration projects, a number of new possibilities arise here.



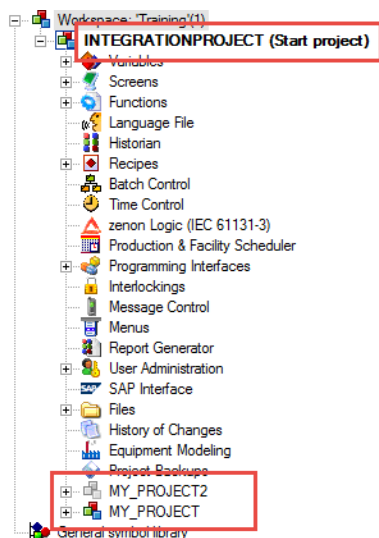
10.4.1 Definition of the hierarchical structure in the Editor

The structure of the network is created in the Editor by simply dragging & dropping the projects. For our example, we need at least three projects, two normal projects (**MY_PROJECT1** and **MY_PROJECT2**) and an integration project.

- ▶ Rename your project to **MY_PROJECT1**.
- ▶ Create a project backup.
- ▶ Restore the project backup as a new project; give the new project the name **MY_PROJECT2**.
- ▶ Create a completely new project and call it **INTEGRATION_PROJECT**.
- ▶ Drag **MY_PROJECT1** and **MY_PROJECT2** to below the **INTEGRATION_PROJECT** by using drag&drop.

The two projects are now displayed in the project manager as a branch of the **INTEGRATION_PROJECT** project. You have thus created the hierarchical structure of the multi-server network.

Your structure should look like this:



Set up the server and standby so that each back up each other's projects.

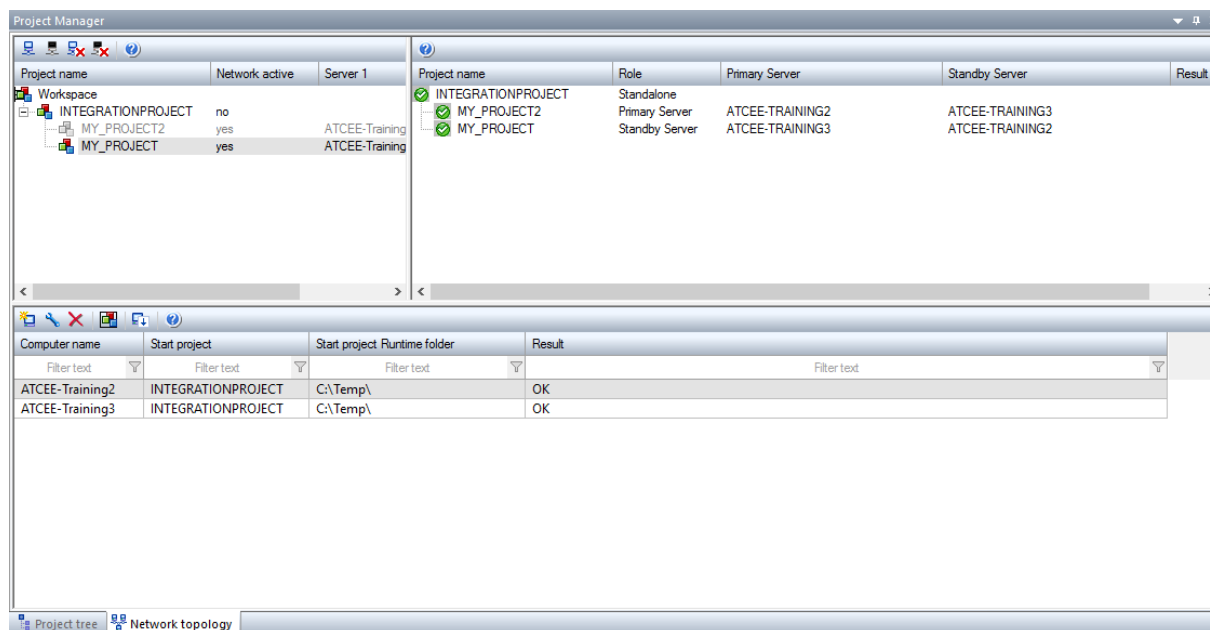
- ▶ In the **MY_PROJECT1** project, define **PC A** as the server and **PC B** as the standby.
- ▶ In the **MY_PROJECT2** project, define **PC B** as the server and **PC A** as the standby.
- ▶ Use Remote Transport to send all Runtime files to both servers, i.e. to **PC A** and **PC B**.

10.4.2 Network topology

zenon provides, in addition to the project manager, a special view for the network topology. The following tasks are possible there:

- ▶ Creation and display of multi-hierarchical projects.
- ▶ Overview of the structure of multi-hierarchical projects and the roles of the individual computers for the respective projects (Server 1, Server 2 or client).
- ▶ Checking to see if the configured topology is possible.
- ▶ Simultaneous transfer of Runtime files to several computers

Your network topology should look something like this:



10.4.3 Special requirements for the integration project

When designing the integration project, we must note some special requirements in order to be able to define access to the lower-level projects.

Screens always in the foreground

In order to be able to switch from a lower-level project into a different one or back into the integration project, you must design a small frame in the integration project, which remains in the foreground. Based on this template, we then create a screen with the navigation buttons for the subprojects.

- ▶ Switch to the **INTEGRATION_PROJECT**.
- ▶ Create a new frame.
- ▶ Activate the **Always in the foreground** property.

This property ensures that the screen always remains in the foreground in Runtime, even if we switch to the screen of a subproject. This is how we can switch to screens in the integration project or to screens of a different subproject.

- ▶ Create a screen that is based on the previous screen.

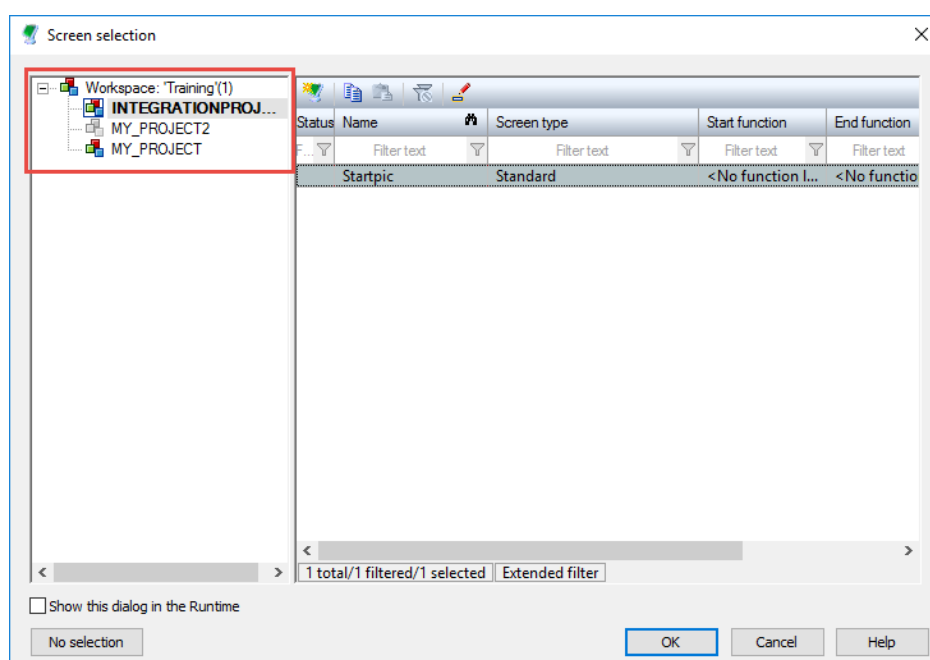
Screen switch to subprojects

Switching between screens of different projects is implemented with normal screen switch functions.

- ▶ Create two new screen switch functions in the **INTEGRATION_PROJECT**, with the respective buttons.
- ▶ Change the background color of the **Start screen** from **MY_PROJECT1** and **MY_PROJECT2** to different colors.

You thus easily see, when the **Start screen** is called up, the project to which it belongs.

If there is more than one project present in the current workspace, an enhanced dialog box for screen selection is shown. In addition to the screen names, you can also select the project too.



Access to elements of the subprojects

You can access data from the subprojects with more than just the screen switch function. You also have the possibility to access variables, recipes or alarms from the subprojects. Furthermore, there is the possibility of a joint Alarm Message List or a joint Chronological Event List.

We will look at a couple of examples for this in this chapter.

Variables

You can access variables from the subprojects at any time in the **INTEGRATION_PROJECT** using normal dynamic elements.

- ▶ Open the start screen of the **INTEGRATION_PROJECT**.
- ▶ Add a new **counter value** dynamic element.

In the variable selection dialog, you can now select between three projects, as with the screen switch function.

- ▶ Click on **MY_PROJECT1** and select a variable from there.

Because in Runtime both projects, the **INTEGRATION_PROJECT** and **MY_PROJECT1**, are loaded, it will work faultlessly in Runtime.



Attention

zenon does not check to see if the network structure in Runtime actually allows access to the selected project and its variables.

Recipes

zenon allows you to write values to variables from different projects of the workspace in a recipe.

- ▶ In the **INTEGRATION_PROJECT**, open the Recipes node.
- ▶ Create a new recipe under Standard recipes.
- ▶ Add variables to this recipe.

In the variable selection dialog, you can now select between variables from your three projects.

- ▶ Click on **MY_PROJECT1** and select the variables from this.

Because the **INTEGRATION_PROJECT** is a client for the server of **MY_PROJECT1**, it must work faultlessly in Runtime.

Alarms and CEL

zenon allows you to display system messages and alarms from different projects from one workspace together in a list. These entries then can be filtered, displayed, printed or exported just the data from normal Alarm Message Lists or Chronological Event Lists.

- ▶ Create an **Alarm Message List** screen.
- ▶ Add a screen template to the screen.
- ▶ Create a **screen switch function** on this screen.

The filter dialog for the Alarm Message List will open.

In the filter dialog, you can find additional **Project** tabs; you can filter for different projects in these. You can now also select several projects at the same time.

Authorization in the network

A network project can be operated in the same way from all stations (server, standby and client). So it can happen, that two users on two different stations set different values for the same variable. This is not a problem for zenon; both actions are executed and the value entered later overwrites the earlier one.

Nevertheless, in zenon you have the possibility to allow operation of the project from only one station at a time. In this case, a user has to get authorization before they can operate the project. In this case operating means an active intervention in the process (setting values, executing recipes, acknowledging alarms, etc.). Opening screen, reading lists, etc. still is possible from all stations.

In the Editor

This operating authorization in the network must be activated in the project properties.

- ▶ Open the project properties of the **INTEGRATION_PROJECT** project.
- ▶ Under **Operating authorizations - operating authorizations in the network**, select the **Global operating authorizations** property.

With this setting, only one computer can still operate the project at any time.

The operating authorization is thus activated for this project. However, we must give the users the possibility to get the operating authorization on the respective computer or to allow it again.

- ▶ In the **Project Manager**, open the **Functions** node.
- ▶ Create a new function.
- ▶ Select the **Operating authorization in the network** function from the network functions.

Select the **Fetch** parameter in the subsequent dialog.

This function allows the users to get operating authorization on their own computer.

- ▶ Create a further **Operating authorization in the network** function.
This time, select the **Release** parameter.
- ▶ Create new buttons in the **INTEGRATION_PROJECT** start screen for both functions.

In the Runtime

We now want to try out the behavior in the Runtime.

- ▶ Restart the Runtime.
- ▶ Stipulate a set value for any desired variable.
The set value is not sent to the hardware. Instead, a dialog opens that informs you that you do not have operating authorization for this project.
- ▶ Fetch the operating authorization with the corresponding button.
- ▶ Stipulate a set value for any desired variable.
This time, the value is sent to the hardware, because you now have operating authorization for this project.
- ▶ Go to another computer on which this project runs.
- ▶ Stipulate a set value for any desired variable.
The value is not written again. Instead, a dialog opens that informs you that another client has operating authorization.
- ▶ Go to the first computer.
A dialog box has opened here in the mean time. You are asked whether you want to cede the operating authorization to the other computer. You have three possibilities for reacting:

Parameter	Description
Yes	Operating authorization is passed over to the other computer.
No	Operating authorization remains on this computer.
Do nothing	<p>You see a countdown in the dialog box. As soon as this time has passed, the operating authorization is released automatically and the other computer gets it.</p> <p>This ensures that a project remains operable, even if there is no one at the computer with operating authorization.</p> <p>This time is set to one minute by default. You can find the Timeout for query [s] property in the project properties under Network. You can change this time if required.</p>

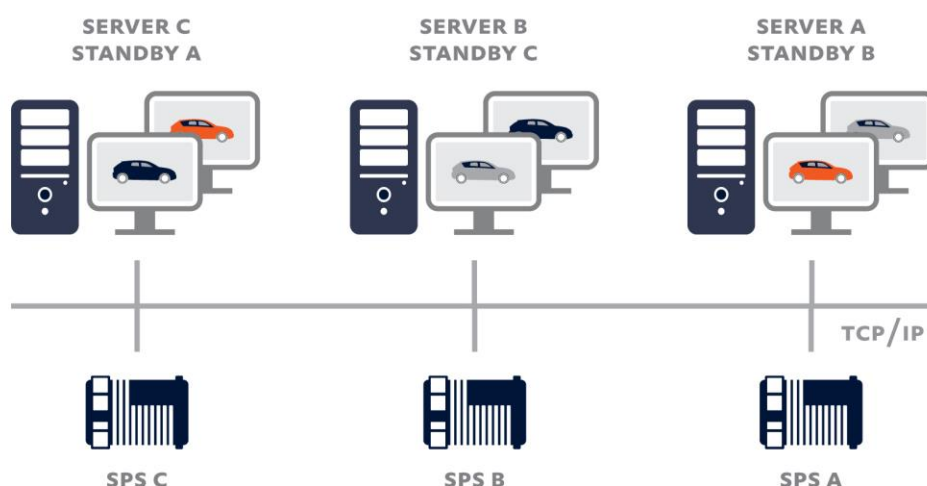
10.5 Circular redundancy

Circular redundancy is a type of network design for connecting several redundant projects to one another very efficiently. Circular redundancy is a cost-effective possibility to increase equipment availability with multi-project administration and redundancy.

There are normally always two PCs required for a redundant project, one PC runs as a server and one as a standby. With two projects, you accordingly need four PCs, etc.

With circular redundancy, you use the possibility of multi-project administration: Several projects can run simultaneously on one PC. Each PC is the server for one project and at the same time the standby server for the neighboring project; and additionally, it can be the client for other projects. This results in a circle.

As a result, there is an enormous potential for savings with hardware and software.



Example:

There are three machines in your production hall for car-care products. The first produces products, the second fills the bottles and the third packs the bottles for transport.

A visualization runs as a server project on each machine. In a redundancy situation, you need a total of six PCs for this. This is not the case with circular redundancy: A second project also runs on each PC as a standby: The filling project on the production machine, the packing project on the filling machine and, in turn, the production project on the packing machine. Each project is thus executed redundantly. In the event of a failure of one PC, the two others assume the complete functionality!

You have thus increased equipment availability and ensure that there is no data loss in the event of a server failure or in the event of maintenance works.

10.6 Web functionalities

10.6.1 Web Server

With the zenon web server, visualization content can be called up and displayed with a web browser. In doing so, no installation of zenon Runtime is necessary on end devices. All components for the display of the user interface on the end device are provided automatically. Additional web clients can thus be incorporated into the system dynamically.

zenon Web Server, Web Server Pro and Web Server Pro Light constitute the complete scope of functions of a zenon visualization. Whereby the client connected to the zenon Web Server only acts as a viewer. The required web browser plug-in can be installed automatically the first time the web client is started.

zenon Web Server, zenon Web Server Pro and zenon Web Server Pro Light require a license.

10.6.2 HTML Web Engine

The HTML Web Engine is for the provision of zenon screens as an HTML5 web page. The user interface is called up and displayed on the visualization end device using a web browser. No special software installation - or any browser plug-ins - are required on the end device. Process data for the visualization is taken from zenon Runtime.

FUNCTIONALITIES OF THE HTML WEB ENGINE

Overview of the functions of the HTML web engine:

- ▶ Session-based provision of HTML5 visualization content on HTML web clients.
- ▶ Display of basic visualization content that was created in the zenon Editor.
- ▶ Data view:
 - Variable values
Displays are updated dynamically in the HTML Web Engine.
 - Chronological Event List (CEL)
CEL entries are updated dynamically in the HTML Web Engine.
 - Extended Trend (ETM)
The display in the ETM is not updated dynamically in the HTML Web Engine. This means that the ETM only draws with existing data when called up and is then no longer updated.
 - Alarm Message List (AML) messages.

AML entries are updated dynamically in the HTML Web Engine.

- ▶ Forwarding of process information, such as variable values, alarm messages or event messages from zenon Runtime to one or more HTML web clients.
- ▶ Support of active operations, such as write set value.
- ▶ Mobile, location-independent operation and observation.
- ▶ No installation and/or configuration on the end device i.e. the client is necessary. Platform-independent display in HTML5 standard.
- ▶ Operation of the HTML web server on a different computer, such as is possible in a DMZ for example.
- ▶ Secure network communication via HTTPS, based on SSL certificates.
- ▶ Protection of sensitive visualization areas or processes by means of user authentication and support of user levels.

The HTML web engine supports authentication of a web engine client with increased security in comparison to zenon user authentication and Active Directory. Login is effected by entering the user name and password.

10.7 Questions about the Network

1. Prerequisites for a functioning zenon network are:
 - a) A functioning name resolution, a TCP/IP network.
 - b) A TCP network, network topology with managed switch.
 - c) All computers involved must be logged onto a domain.
 - d) None of the mentioned.
2. The server fails in a redundant zenon project. What implications does this have for the user from the client side?
 - a) A messagebox is displayed on the client that the server has failed.
 - b) The system administrator receives a message and will initiate the transfer of data without loss from the server to the standby server. The client computer does not notice this.
 - c) The client automatically connects to the standby, which takes over the role of the server. All required Runtime files were buffered by the drivers at the standby and are taken over without interruption when the server failure is recognized.
 - d) The client successfully connects with the standby. The dynamic elements display blue points as hints.

11. Topic: Diagnosis

11.1 Information collection tool - SIC

The tool **SIC – System Information Collector** is part of zenon and is for the collection of information that is necessary to be able to process support queries quicker and more effectively. All information such as that about the system, log files, registry, are collected into a package and can be used by support employees to enable them to work more quickly. In addition to the description of the project and the error, as well as the project backup, this information is the primary basis for working on queries.

You can find the **SIC – System Information Collector** tool in the Startup Tool under **Tools**.

This tool is automatically installed with the zenon setup and offers the following options:

Parameter	Description
Full	Collects information about the system and zenon. Dump files are not collected.
System	Only collects information about the system.
Scada	Collects information about zenon, without dump files.
Scada with dump files	Collects information about zenon and includes dump files.

11.2 Project analysis - CRL light

With the cross-reference list, all references between variable, functions, recipes etc. are shown in a table overview. A project analysis by means of "Cross Reference List Light" only takes variables and functions into account.

All relevant modules are searched through using an option in the context menu and the points of usage are determined. The results are shown in the results list.

11.2.1 Use of variables and functions in the project

- ▶ Highlight the desired variable or function in the detail view.
- ▶ In the context menu select the **Variable use** or **Function use** menu item.

After selecting the option, the results are shown in their own window. You can call up an element by double-clicking or via the context menu.

You can also look for all variables/functions that are not used.

- ▶ Click on the **Variables** project tree node.
- ▶ Open the context menu and select **Show unused variables...**

After selecting the option, the results are shown in their own window. You can call up an element by double-clicking or via the context menu.

12. zenon online test

Around the world there are many well trained customers and partners that have excellent knowledge about zenon. COPA-DATA offers various types of training courses (www.copadata.com/training) that are adapted to the respective zenon products and modules.

We really want to ensure proof of the knowledge achieved on an individual level but also for entire companies. zenon Certification is on a personal level, but companies with certified employees can also become a member of the COPA-DATA Partner Community (www.copadata.com/partner) where additional benefits are offered.

There are currently three levels of zenon Certificates: Basic, Advanced and Professional. Each of these levels are reached by completing different training courses and the attendant online tests (www.copadata.com/online-tests).

Here is a brief overview of the levels of zenon certification:



The zenon online tests are created by COPA-DATA and required as part of the zenon Certification.

For each training stated in the training overview there is a corresponding online test as well.

zenon Additional Training Courses

zenon Individual Training (x days)	zenon VBA Training (2 days)	zenon VSTA Training (2 days)	 	
zenon Analyzer Advanced Training (1 day)	zenon Logic Advanced Training (1 day)	zenon Historian Training (1 day)	zenon Network Training (1 day)	zenon Design & Usability Training (1 day)
zenon Analyzer Training (2 days)	zenon Logic Training (2 days)	zenon Energy Training (2 days)	zenon Pharma Training (2 days)	zenon Batch Training (2 days)

zenon Standard Training Courses

zenon Basic Training (3 days)	zenon Operator Training (4 days)	zenon Supervisor Training (5 days)	
-------------------------------	----------------------------------	------------------------------------	--

12.1 Execution of the zenon online tests


The execution of the zenon online tests consists of 3 stages.

1. You register via the web site.
2. You are sent a personalized link your zenon online test.
3. You execute the zenon online test via this link.

12.1.1 Registration for the zenon online test

After attending a zenon training course, the corresponding zenon online tests are available to you on the COPA-DATA website at www.copadata.com/online-tests.

Simply select the zenon online test you'd like to complete. The names of the zenon online tests are the same as the corresponding zenon training courses. All zenon online tests are available in English and German.



Questions? Contact us at:
training@copadata.com

COPA-DATA Training - zenon online tests

With a few clicks you're ready to go – start your online test today and prove your zenon skills!

Request your zenon online test link now!

How to get started with your zenon online test!

You can select your zenon online test in the radio buttons adjacent to the test name. The online tests are named as the corresponding zenon training courses, so that you can select the test adapted to the training you have done. If you are unsure which test you should do, please see the overview of all training courses [here](#).

zenon online test in short

- ✔ Select the test you'd like to complete and submit the form
- ✔ Receive your zenon online test link in your inbox
- ✔ Answer the questions: the tests consist of multiple choice questions from all chapters of your training.
- ✔ Receive your test result immediately after completing

” After completing the zenon Supervisor training I had absolutely no problem reaching the minimum correct answers of 60% in the test. It was great to be able to see the results immediately and what I really like is the back-up this test gave me – I now know I'm a zenon guru! “

Select the corresponding test below!

Select the online test you'd like to complete below. The names of the tests are the same as the corresponding training course. All online tests are available in English and German.

zenon Supervisor

Mr

Peterson

Peterson & Son

90009

John

john.peterson@peters

United States

Los Angeles

Submit

12.1.2 E-Mail with link to the zenon online test

You will receive an e-mail with the link to the zenon online test. This contains two links for the zenon online test; one for the English version and one for the German version. Select your preferred language and click on the link.

Dear Mr. Peterson,

We're pleased you've registered for the zenon online test. When you select the online test link below you will be forwarded to a new platform where information on how to proceed with the test is given. If you have any questions regarding the online test, please contact training@copadata.com.

Chosen training course:

zenon Supervisor Online Test English: <https://www.classmarker.com/online-test/start/?quiz=nmb58c65470d78a3> | zenon Supervisor Online Test German: <https://www.classmarker.com/online-test/start/?quiz=f6n58c653fe8aa86>


Best regards

COPA-DATA Headquarters

123

12.1.3 Starting the zenon online test

You will be redirected to the zenon online test when you click on the link. Fill in the necessary information, follow the instructions and complete the zenon online test.



COPADATA
do it your way

7.60 zenon Supervisor Test EN

First name
John

Last name
Peterson


Email address
john.peterson@peterson.com

Company
Peterson & Son

Training type, date and place:
Supervisor, Copadata HQ

Start ▶

Start the online test.



COPADATA
do it your way

7.60 zenon Supervisor Test EN

Time left: 0:59:52

John Peterson

Question 1 of 10

In the Runtime, in a client-server project you have saved different profiles on a client for your Chronological Event List. Now you would like to also have these profiles available on the other clients. How to proceed?

- ☐ You copy the file „cel.bin“ from the client to the other clients.
- ☐ You export the profiles and import them to a different client.
- ☐ You transport the profiles with Remote Transport to the other clients.
- ☐ Profiles from the CEL are automatically synchronized by the system.

Next ▶

12.2 Q&As about the zenon online test

- ▶ Who can carry out an zenon online test?

Anybody who has completed zenon training carried out by a COPA-DATA sales representative or at COPA-DATA HQ can complete the corresponding zenon online test.

- ▶ When should the zenon online test be taken?

The zenon online test can be taken right after the training course at the earliest. It is possible to take it for 1 year.

- ▶ How do I access the zenon online test(s)?

The online tests are available on the COPA-DATA website at www.copadata.com/online-tests.

- ▶ How many questions does a zenon online test include?

A zenon online test always consists of 10 questions in total.

- ▶ What is needed to pass a zenon online test?

To pass a zenon online test, you need to give the correct answers to at least 60 % of the 10 questions. The questions are randomized every time the test is taken again. All tests are updated on a regular basis to reflect the latest zenon release information. The tests are personal and cannot be handed over to someone else.

- ▶ What happens if I don't pass the zenon online test the first time?

You can re-take the zenon online test up to three times. After this, you need to request the possibility to re-take the online test via your local COPA-DATA Sales Representative.

- ▶ Are there zenon online tests for individual training sessions as well?

There are not online tests available for all individual training sessions carried out. However, if your individual training session covers the same content as a regular zenon training course, you are entitled to take the online test for the respective zenon training course.

- ▶ How much time do I have to finalize the test?

There is a time limit of one hour for all ten (10) questions. After this time, the test will end automatically. Depending on the given answers, you will either pass or fail the test and it will count as one completed try.

- ▶ In which languages are the zenon online tests available?

The zenon online tests are always available in English and German. However, there may be some zenon online tests available in other languages. An overview of the languages is available at www.copadata.com/online-tests.

13. Further zenon training sessions

COPA-DATA offers, in addition to this training, much other training, the precise details of which you can find in the following chapters.

You can find detailed information on the COPA-DATA home page. For training queries, please contact your regional distributor or **sales@copadata.com**.

13.1 Online training

MORE FLEXIBLE ZENON TRAINING COURSES WITH E-LEARNING

Online training is not tied to a particular time or place. As a result, there is a new possibility for COPA-DATA customers, partners and employees to learn parts of zenon step by step. The usual training offered has proven itself over many years and makes it possible for zenon specialists to act competently around the globe. The more flexible e-learning training offered supports mobility in learning and thus better addresses the participants. If zenon knowledge is required, the information and content can be called up immediately from the online training.

EASY LEARNING AT YOUR OWN PACE

The new online zenon training courses consist of videos, accompanying texts and revision tests, as well as educational embedded practice examples. The course participants are presented with the learning material in a clear and manageable form – they can now build up their knowledge about zenon through individual e-learning modules, without having to take into account the learning pace of other participants or the trainer.

ONLINE TESTS AND ZENON CERTIFICATION

As with training with participants present, there is also the possibility after the e-learning course to obtain official zenon certification with a zenon online test. The content of these tests is based on the respective training course for both options – the learning objectives are always the same. Anyone who meets all conditions and passes the test at the end gets a certificate.

You can find details about the training we offer at www.copadata.com/training.

13.2 zenon Logic training

In the zenon logic training, we tell you about the basics of zenon logic. You learn step by step how you operate the zenon Logic Workbench – the development environment of zenon Logic – and how communication with a PLC or zenon Runtime works. After a short time, you will independently create a zenon logic project and test this in zenon Logic Runtime. You will get well-founded knowledge about zenon Logic, about possible cases where it can be used and the basics of the programming language in accordance with IEC 61131-3.

zenon Logic Training is divided into standard training and advanced training.

13.3 zenon Analyzer training

In the zenon Analyzer training, we explain the basics of zenon Analyzer to you. You get to know the different components step by step and get an overview of the different types of reports. You can already create a report after a short time.

In training, no in-depth zenon supervisor knowledge is required. For each report, we explain to you the functions behind zenon Editor and zenon Runtime. You get well-founded knowledge about zenon Analyzer and about possible instances where the different reports can be used.

zenon Analyzer training is divided into standard training and advanced training.

13.4 zenon VBA training

In the zenon VBA training, we tell you the basics of the COM interface in zenon. You learn step by step how you can operate the VBA editor and how communication between the COM interface and zenon works. After a short time, you can write a VBA macro yourself and execute it in the zenon Editor. During the course of the second training day, you learn how to execute VBA macros for zenon Runtime and to execute them in it. We will introduce you to the zenon object model and show you some standard functions. Using practical examples, you get well-founded knowledge about the COM interface in zenon.

13.5 zenon VSTA training

In the zenon VSTA training, we tell you the basis of the COM interface in zenon. You learn step by step how you can operate the VSTA editor and how communication between the COM interface and zenon

works. After a short time, you can write a VSTA macro yourself and execute it in the zenon Editor. During the course of the second training day, you learn how to execute VST macros for zenon Runtime and to execute them in it. We will introduce you to the zenon object model and show you some standard functions. Using practical examples, you get well-founded knowledge about the COM interface in zenon.

13.6 zenon VBA/VSTA training

In the zenon VBA/VSTA training, we tell you the basis of the COM interface in zenon. You learn step by step how you can operate the VBA editor and VSTA editor and how communication between the COM interface and zenon works. You create not just different VBA macros for the zenon Editor or zenon Runtime, you also learn to program VSTA macros and their applications in the same training.

We will introduce you to the zenon object model and show you some standard functions. Using practical examples, you get well-founded knowledge about the COM interface in zenon and the differences between VBA and VSTA.

13.7 zenon design & usability training

In the zenon design & usability training, you master the basics of design, usability and ergonomics – both in theory and in practice – when configuring zenon projects. You learn, step by step, how you can design your project so that it can be operated in a user-friendly and efficient manner. You get a comprehensive introduction to the basics of design and usability.

13.8 zenon Historian training

In the zenon Historian training, we show you the archiving and evacuation of archives with zenon. You learn step by step how you can save the data from your project or your equipment in an archive. We introduce you to the "Report Generator", "Report Viewer" and "Extended Trend" modules and show you how you can prepare your data with them. You get well-founded knowledge about archiving in zenon and about instances where it could be used.

13.9 zenon Network & security training

In the zenon network & security training, you get to know the module step by step, including an insight into the security mechanisms that zenon provides you. We show you how you can set up a network with smart network topology with zenon. You get to know the multi--Server-network, as well as a terminal

server and zenon Web Server. We will present this comprehensive module using clear examples. There is also sufficient possibility to discuss your own requirements and approaches for a solution.

13.10 zenon Energy Edition training

In the zenon Energy Edition training, you get to know zenon Energy Edition step by step: from the configuration of a topologically-colored network (ALC), to the application of command input and possible protocol specifics. We show you how you, with your topologically-structured network, can carry out a search for topological errors. You get well-founded knowledge about zenon Energy Edition and about possible instances in which it can be used in the energy industry.

13.11 zenon Pharma Edition training

In the zenon Pharma Edition training, we introduce you to the special functionalities of zenon Pharma Edition. You learn step by step how you can use these additional possibilities and thus complete your project for the Pharma industry. We show you how zenon Pharma Edition helps you, with archiving and special reporting, to implement the regulations of the Pharma industry. You get well-founded knowledge about zenon Pharma Edition and instances where it could be used.

13.12 zenon Batch Control training

In the zenon Batch Control training, you get to know the module step by step: from the configuration of units to the handling of communication errors, from the creation of simple recipes to the use of partial recipes and control strategies. We will present this comprehensive module using clear examples. There is also sufficient possibility to discuss your own requirements and approaches for a solution.

14. zenon individual training

In the zenon individual training, we show you how you can solve your individual problems with the help of zenon. Using general examples, we will work out a way to a solution for your situation in zenon together.

You discuss the precise topic for the training with us. Please note that we can only provide you with expertise in relation to our own products.

We would be happy to create an individual offer for you.

15. Glossary

15.1 Glossary for visualization

Editor: Graphical user interface for creating, editing and maintaining zenon projects.

Project tree: Structured graphic display of all projects and modules in a workspace.

Backup: Additional saving of data (for example: projects or workspace) for subsequent restoring.

Frame: Defines the size and location of an area on the monitor in which screens are displayed in Runtime. Each screen is based on a frame.

Screen type: Defines the type of screen and the elements which are available in it as standard. For each screen type, there are elements (buttons, lists, etc.) available, which are adapted for the functions.

Symbol: Summary of different dynamic elements and vector elements for a new element. A change to the size has a proportional effect on all elements included.

Data type: Defines fixed area of values (fixed top and bottom limit) with an exactly defined number of values. That is why real numbers can only be depicted as float-point numbers with a certain accuracy.

Variable: Are the interface between the data source and zenon; also called process variable or data points.

Function: A pre-defined action that can be instigated in Runtime, such as a screen switch or the writing of a set value.

Script: Several user-defined functions that can be connected in sequence.

15.2 Event handling glossary

Chronological Event List (CEL): Central administration of events in zenon. The CEL is listed twice: Once as a ring buffer in the memory and once as historic data in files on the hard drive. In contrast to the AML, the CEL only administers the "received" time stamp.

Alarm Message List (AML): The Alarm Message List is for the administration of limit value violations and alarms.

Limit values: Limit values have the task to trigger a reaction, as soon as a limit value or a status is violated or reached.

Ring buffer: A ring buffer constantly saves a definable amount of data. If the memory space is full, the data is overwritten, either on a FIFO or LIFO basis.

Reaction matrix (REMA): A compilation of states which can be linked with any number of variables. Every time a variable changes its value, the reaction matrix is evaluated and the first applicable status triggers the configured reaction. The following reactions are possible: Color changes, alarms, execution of functions etc., in contrast to limit values, can also monitor the status information.

Alarm status line: Information line that is always displayed in the foreground in front of all programs in Runtime; it displays either the oldest or the most recent alarm that has not been acknowledged.

Recipes: Recipes collect set values and commands in a list which can be executed in the Runtime with the help of a function.

15.3 Glossary for operation

Keyboard: A virtual input device that is displayed on the screen. It contains a number of control elements that allow you to enter characters.

Interlockings: Allow you to control the access to certain zenon objects in Runtime using variables. Operation can be blocked or released depending on variables.

User group: Summary of one or several authorization levels under a common name. Users can be assigned to user groups.

Authorization level: Determines whether a user can carry out an action. An authorization level can be assigned to each control element. For selected functions such as Acknowledge alarm, load project in the Editor, etc. authorization levels can also be issued. There are 128 levels (from 0-127), whereby 0 is the system level, which every user has.

User administration: Program or part of a program in which you can define which users exist in a system, which rules apply to them when logging on to the system and which rights they have in the system.

Variable Diagnosis: Screen type for the display of variable values, time stamps and status. Variable values can also be written.

Mathematics driver: Driver that is integrated into Runtime and that provides mathematical functions.

Temporary login: The user is only logged in for the execution of a certain action and is logged out again immediately.

Permanent login: Is a one-off login that remains in force until the user logs out or Runtime is restarted.

Files that can be changed in Runtime: Files in zenon that can be changed in both the Editor and Runtime.

15.4 Network glossary

Redundancy: 100% process control and data security of the SCADA network, even if a server fails.

Server: A server is a computer or a computer program that provides functions such as service programs, data or other resources so that other computers or programs (clients) can access it, usually via a network.

Standby: Program that has all current data and server connections at its disposal and that can take over spontaneously if the server fails.

Naming resolution: The naming resolution is the process that makes it possible to translate and provide names of computers.

Service: A program, routine or process that carries out a certain system function to support other programs, especially at a low (hardware) level.

Port: Is part of an IP address and defines a unique communication channel. For communication with TCP/IP, the IP address and a port number are always required, in order to address a recipient unambiguously.

System variable: Variable that is based on the system driver; these variables contain information about the project and/or hardware.

Network dongle: Hardware on which a certain number of usable software licenses are saved at the same time. Access is gained via the network.

16. Exercises

16.1 Exercise 1: Switch with its own text

Create a new BOOL variable on the internal driver with the name "Switch_Exercise".

Configure a switch with two different texts.

When a switch is activated, "On" with a green background should be visible.

If the switch is not activated, "Off" with a gray background color should be visible.

16.2 Exercise 2: Counter

Create a new INT variable on the internal driver with the name "Counter".

Configure two write set value functions, "+1" which counts up and "-1" which counts down.

Draw a counter value element and two buttons on which you can link the functions.

The counter element is for display only. Only set values from -10 to +10 can be written.

16.3 Exercise 3: Pump display

Exercise 3: Pump display

Structure data type and variable:

Create a structure data type with the name "Pump" with the following elements:

- Temperature (°C); SINT data type
- Speed (rpm); UINT data type
- Throughflow rate (m³/h); UINT data type
- Power consumption (A); INT data type

Create three internal variables, Pump_01, Pump_02 and Pump_03, which are based on this structure data type.

Structure data type:

Pumpe	Struktur Datentyp	0
Temperatur	Struktur Element	SINT/ <eingebettet> 1
Drehzahl	Struktur Element	UINT/ <eingebettet> 3
Durchflussmenge	Struktur Element	UINT/ <eingebettet> 4
Stromaufnahme	Struktur Element	INT/ <eingebettet> 1

Structure variable:

Pumpe_01		
Pumpe_01.Temperatur		°C
Pumpe_01.Drehzahl		U/min
Pumpe_01.Durchflussmenge		m³/h
Pumpe_01.Stromaufnahme		A
Pumpe_02		
Pumpe_03		

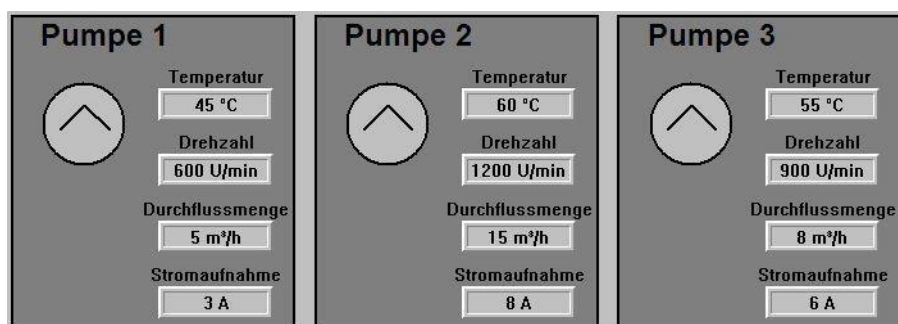
Symbol:

Create a pump symbol (circle and line vector elements) and visualize the variables of the Pump_01 with the dynamic elements.

Create a symbol from all the elements and give it the name "Pump".

Add the symbol to the symbol library project.

Use the "Pump" symbol in a screen with the three variables "Pump_01", "Pump_02" and "Pump_03".



Expand symbol:

Expand the symbol by clicking an on/off switch

To do this, add a new BOOL "on/off" structure element to the structure data type. The pump symbol should get a color surround between the gray and green.

16.4 Exercise 5: Automatic language switching with user logged in

The language should be automatically switched by the logging in of a certain user.

A switch to the default language should made automatically when the user logs out.

16.5 Exercise 6: Language switching with a button

The language is to be changed by clicking on a button. The original language is activated again when the same button is clicked on again.

Addition: The country flag symbols of the respective language should be used for switching instead of a button. Therefore, when actively using German the English flag should be shown, when actively using English the German flag should be shown.

16.6 Exercise 7: Close a screen yourself

A pop-up screen is to be opened by clicking on a button. This screen should close again independently after 10 seconds or the button to close should only be displayed after this time.

16.7 Exercise 8: Monitoring of safety shut-off mats

In an equipment layout of a conveyor system, the monitoring of safety shut-off mats is to be displayed. In doing so, polygons are to be created for the areas to be monitored. When activating a safety shut-off mat, (BOOL variable "Safety shut-off mat 001" to "Safety shut-off mat 004"), these polygons are to be colored red.

This display of safety shut-off mats is monitored in terms of time. An alarm is triggered if one of the monitoring variables for these safety shut-off mats is longer than 10 seconds.