

Unified-E App Designer User Manual

Build HMI Visualizations in the HMI Editor

Software version 3.1.0.0, last updated: July 2025

Publisher: Unified-E AG, Winterthur, Switzerland



Content

1	Introduction	9
1.1	Software Overview	9
1.1.1	Software Components.....	9
1.1.2	Direct and Gateway Communication	10
1.2	Project File and App Package File	10
1.3	Important Terms.....	11
1.3.1	Operator App / HMI App.....	11
1.3.2	View	11
1.3.3	Views Element	11
1.3.4	Endpoint.....	12
1.3.5	Datapoint.....	13
1.3.6	Runtime	13
1.4	Quick Start	14
1.5	Licensing.....	14
1.6	Create a New Project	14
1.6.1	Templates for Displays and Default Layout Behaviors	15
1.6.2	User-defined Displays and Default Layout Behaviors	17
1.7	Major Areas of the HMI Editor	17
1.7.1	Project Navigation	17
1.7.2	Editor Area	18
1.7.3	Element Navigation	18
1.7.4	Properties Pane	18
1.7.5	Views Elements Pane	18
1.7.6	Workflow Bar.....	19
2	Manage Datapoints	19
2.1	The Datapoints Table	20
2.2	Create Endpoint Datapoints	21
2.2.1	The Endpoint Datapoint Editor at a Glance	22
2.2.2	Fix Communication Problems.....	23
2.2.3	Selecting Datapoints Online	24
2.2.4	Configure Formulas.....	25
2.2.5	Table Datapoints	25
2.3	Create Local Datapoints.....	26

2.3.1	Initialization of the Local Datapoint	27
2.3.2	Runtime of Local Datapoints	27
2.4	Creating Script Datapoints	27
2.4.1	Formula with "Read" Script Datapoint	28
2.4.2	Script Routine with "Write" Script Datapoint	30
2.4.3	Converter with Script Datapoint	31
2.4.4	Global Script Functions	31
2.5	Import and Export Datapoints	32
2.5.1	Importing from Excel via the Clipboard	32
2.5.2	Export to Excel from the Clipboard	33
2.5.3	Other Import/Export Options	34
2.5.4	Replace Datapoints	35
3	Designing Views in the Graphical editor	35
3.1	The View editor at a Glance	35
3.2	Layout Elements in a View	37
3.2.1	Set Layout Type on View	37
3.2.2	Absolute Layout:	38
3.2.3	Grid Layout	39
3.2.4	Flow layout	42
3.3	Display-specific Layout of Elements	43
3.3.1	Shared vs. Individual Layout	44
3.3.2	Create an Individual Layout	44
3.4	Use Images and Fonts	46
3.5	Edit Theme with the Appearance Template	47
3.6	Create and Use Types	48
3.6.1	Types at a Glance	48
3.6.2	Status Text Types	48
3.6.3	Image Types	50
3.6.4	Color Types	52
3.7	Tools for Editing	54
3.7.1	Positioning Grid in Absolute Layout	54
3.7.2	Group	54
3.7.3	Guides	55
3.7.4	Arrange Elements	56
3.7.5	Align Elements	56

3.7.6	Edit Display Text in the Graphical editor	57
3.7.7	Lock an Element in the editor	57
3.7.8	Hide Element in editor	58
3.7.9	Search for Objects	58
3.8	Reusability with Templates	60
3.8.1	Create a Template	60
3.8.2	Link View Elements to Template Parameters	62
3.8.3	Instantiate Template	63
3.9	Reusability with View Frame	64
3.9.1	View Frame Concept	64
3.9.2	Create a View Frame	64
3.9.3	Link View Frame with View	65
4	General Design Properties	66
4.1	Available Property Palettes	68
4.1.1	"General" Palette	68
4.1.2	"Configuration" Palette	68
4.1.3	"User Actions" Palette	70
4.1.4	"Size" Palette	72
4.1.5	"Position" Palette	72
4.1.6	"Appearance" Palette	72
4.1.7	"Alignment" Palette	73
4.1.8	"Margins" Palette	74
4.1.9	"Effects" Palette	75
4.1.10	"Visibility" Palette	76
4.1.11	"Permissions" Palette	77
4.2	Configure Color	79
4.2.1	Configure Element Color	79
4.2.2	Color Picker	79
4.3	Configure Font	81
4.4	Configure Border	82
4.5	Configure Number Format	83
4.5.1	Configuration in the Dialog	83
4.5.2	Textual Configuration of Number Format	83
4.6	Conditions for Datapoint Comparison	84
5	View Elements	85

5.1	Layout Panel Elements	86
5.1.1	Common Properties of the Layout Panels	87
5.1.2	Absolute Panel	88
5.1.3	Grid Panel	89
5.1.4	Flow Panel	90
5.2	Simple Elements	91
5.2.1	Text	91
5.2.2	Shape Elements	91
5.2.3	Image	93
5.2.4	Display Elements	95
5.2.5	Input Elements	100
5.2.6	Button Elements	105
5.2.7	View Navigation	110
5.2.8	Hyperlink	112
5.3	Extended Elements	113
5.3.1	Message Display Elements	113
5.3.2	Recipe Table	120
5.3.3	User Management Elements	132
5.3.4	System Button Elements	142
5.3.5	List Panel	144
5.3.6	Chart Panel	147
5.3.7	Chart Data Series	155
5.4	Special Elements	157
5.4.1	View	157
5.4.2	View Frame	160
5.4.3	Template	161
5.4.4	Template Element	161
5.4.5	Dialog	162
6	Alarms and Messages	163
6.1.1	Manage Alarm Message Events	164
6.1.2	Manage Message Classes	170
6.1.3	Manage Message Groups	172
6.1.4	Configure the Message Archive Storage	173
6.1.5	Configure Message CSV Files	174
7	User Management	176

7.1.1	Manage User Roles	177
7.1.2	Define Predefined Users	179
7.1.3	Login Configuration	180
8	Recipes	181
8.1	Important Terms.....	181
8.1.1	Recipe Type.....	182
8.1.2	Parameter Group and Parameter	182
8.1.3	Recipe Dataset	182
8.1.4	Recipe Dataset Version (optional).....	182
8.1.5	Activate Recipe Dataset.....	183
8.1.6	Steps.....	183
8.1.7	Dataset Folders.....	184
8.1.8	Process Highlighting	184
8.1.9	Parameter Adjustment Before Activation	184
8.2	Define Recipe Parameters	185
8.2.1	Properties of a Parameter Group	186
8.2.2	Properties of a Parameter	187
8.3	Predefined Recipe Datasets.....	190
8.3.1	Create and Configure a Recipe Dataset.....	191
8.3.2	Set Setpoints for the Parameters	192
8.4	Folder Permissions	192
8.4.1	Properties of a Folder Permission	193
8.4.2	Common Permissions	194
8.5	Dataset Folder	195
8.6	Configure Base Settings.....	195
8.6.1	Configure Comon Recipe Type Settings	196
8.6.2	Configure System Data Exchange in the Properties pane	197
8.7	Recipe Datasets in the File System.....	198
9	Trigger Actions.....	198
9.1	Manage Triggers.....	199
9.2	Trigger Conditions.....	200
9.3	Manage Trigger Actions	200
9.4	Runtime of the Trigger.....	201
10	Curve Recordings	202
10.1	Chart Recordings	202

10.1.1	Properties in the Table	203
10.1.2	Properties in the Properties Pane	204
10.2	CSV Recordings.....	205
10.2.1	Define CSV Recordings.....	205
10.2.2	Define Columns for CSV Recording	206
10.3	Recording Files in the File System	206
11	Datapoint Connections	207
12	Multilingual HMI Apps	208
12.1	Add New Language.....	208
12.2	Set Display Text in Active App Language.....	209
12.3	App Language at Runtime.....	210
12.4	Manage Text in the Language Editor	210
12.4.1	Language Editor Overview	210
12.4.2	Import and Export Texts	212
13	Test the HMI App.....	214
13.1	Test Endpoint Configuration	214
13.2	Test the HMI App in the App Simulator	215
14	Publish the HMI App on the Operator Device.....	216
14.1	Publish for Direct Communication	216
14.2	Publish for Gateway Communication.....	218
14.3	Start App in the Operator Device.....	219
15	Project Settings	220
15.1	App Properties	220
15.2	Displays	220
15.3	Images	222
15.3.1	Add and Configure Image.....	222
15.3.2	Add Images Using the View Editor	223
15.4	Fonts.....	224
15.4.1	Define the Default Font	224
15.4.2	Import Fonts	225
15.5	Appearance Template	225
15.6	Simulator Settings	227
15.7	System Error Settings	229
15.7.1	General Behavior in Case of System Errors	230
15.7.2	Email Notification in Case of System Errors	230

15.7.3	Push Notifications in Case of System Errors	230
16	General Features	231
16.1	Open Another App Designer Instance	231
16.2	General Editor Settings	231
16.2.1	Editor Settings	231
16.2.2	Others	232
17	Appendix.....	233
17.1	Tips and Tricks.....	233
17.2	Support and Further Information	234

1 Introduction

With the Unified-E App Designer, you can create HMI visualizations for the operation and monitoring of machines and plants. As part of the Unified-E software suite, the App Designer serves as a graphical development tool for the design of so-called HMI apps – user interfaces that can be executed on touch panels, industrial PCs, tablets or smartphones.

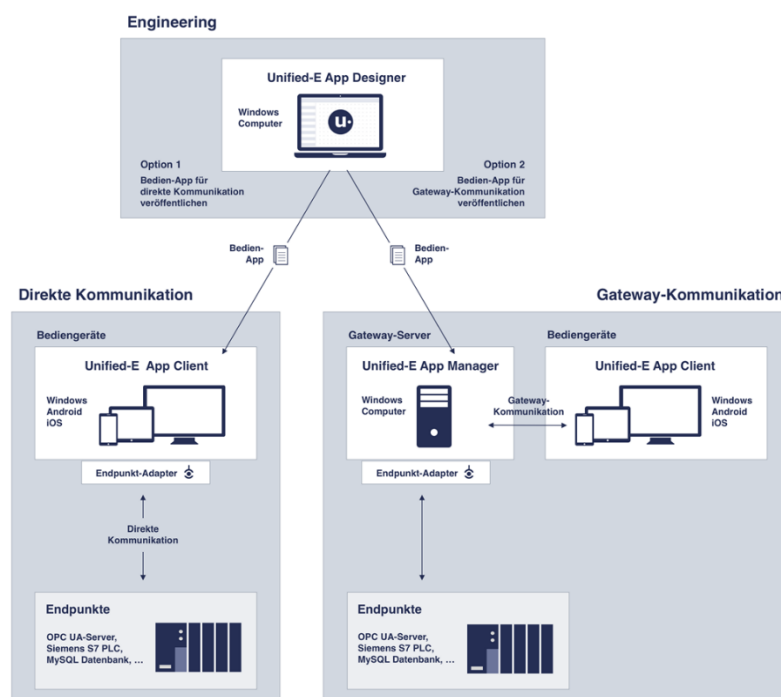
This guide is intended for people who create or edit visualizations using the Unified-E App Designer. It explains the structure of the user interface, the most important tools and the properties of the available objects – with the aim of making the creation of HMI visualizations as efficient and intuitive as possible.

The App Designer works closely with the Unified-E Client and the Unified-E App Manager, which will be presented in the next chapter.

1.1 Software Overview

1.1.1 Software Components

Unified-E consists of several programs or software components; the most important components are illustrated in the following figure. All programs can be downloaded from the Unified-E website – for development purposes, all programs are available free of charge.



Unified-E App Designer:

The Unified-E App Designer (short: App Designer) is an HMI editor that is used to configure the operator app or HMI app. The configuration is stored in a project file. For use at runtime, the App Designer uses the "Publish" function to create an app package file that contains the visualization in an optimized form.

Unified-E Client:

The Unified-E Client is installed as an HMI client on the respective operator device. It is used to execute the visualization created in the App Designer and thus enables the operation and monitoring of machines and plants.

The visualization is registered in the Unified-E Client via an app package file and then used at runtime.

For Android and iOS devices, the Unified-E Client (under the name "Unified-E App") is available in the Google Play Store and the Apple App Store, respectively. The Windows version of the client can be downloaded from the Unified-E website.

Unified-E App Manager (optional):

The optional Unified-E App Manager (short: App Manager) acts as an HMI server with central data storage. The server function is used when, for example, a smartphone is to communicate with the plant via the Internet or several operator devices access a plant at the same time. Since the HMI server acts as a Gateway from the client's point of view, this is also referred to as Gateway communication.

1.1.2 Direct and Gateway Communication

Direct communication:

Direct communication means that every operator device (e.g. a panel PC or smartphone) communicates directly with the PLC. This variant is particularly suitable for simple plants with a few operator devices, as no additional server is required and the operator device communicates with native communication protocols (without a web server).

Gateway communication:

Gateway communication, on the other hand, uses a central HMI server on a Windows computer with the Unified-E App Manager software that is connected to the PLC. All operator devices communicate exclusively with the Gateway, not directly with the PLC.

Gateway communication offers advantages for the following applications:

- Central data storage and preparation for several operator devices for the same plant
- Easily set up remote communication via the Internet

1.2 Project File and App Package File

Project file:

The configuration of an HMI app is managed as a HMI project in the Unified-E App Designer. The project is saved in a project file. All data, such as used images or fonts, are also

embedded there, external files are only needed for import into the project. The file extension is "uep".

The project configuration is technology-independent: A project can be run on all supported platforms (Windows, Android, iOS) – both for Direct communication and Gateway communication.

App Package File:

Both the Unified-E Client and the Unified-E App Manager use an app package file at runtime. This file can be created in the Unified-E App Designer using the "Publish" function via the toolbar and uses the configuration of the currently open project.

- **Direct communication:**
In case of Direct communication, this app package file can be added to the Unified-E Client as an HMI app. For mobile devices, the app package file can be downloaded directly from the Unified-E App Designer via QR code scanning.
- **Gateway Communication:**
This is where the app package file is registered as a new app in the Unified-E App Manager. There, in turn, operator devices can register with the help of the Unified-E Client.

1.3 Important Terms

1.3.1 Operator App / HMI App

The Unified-E App Designer can be used to configure operator apps – also known as HMI apps or HMI visualizations. These are executed with the Unified-E Client on the respective operator device and displayed for operation and monitoring.

1.3.2 View

A view is a related unit of controls that is displayed on the screen. It corresponds to the visible content of a page or screen area of the HMI app.

One of the central tasks when creating an HMI app is the design and configuration of views for the operation and monitoring of machines or plants.

1.3.3 Views Element

Unified-E provides a set of predefined view elements that can be inserted, placed, and configured into a view by dragging and dropping.

Typical elements are, for example, display elements for displaying status or actual values, buttons for machine operation or input fields for setting machine parameters.

The abbreviation "element" is often used in documentation.

1.3.4 Endpoint

What is an endpoint?

In most cases, an endpoint represents a PLC controller. Other data sources such as SQL servers or web servers can also be integrated into an HMI app as an endpoint.

An endpoint must meet the following criteria:

- **Available adapter:** A suitable endpoint adapter (communication driver) must be available for the respective endpoint – see the list of supported adapters below.
- **Monitor (read):**
The endpoint must be able to provide values through a unique datapoint address (e.g., temperature, pressure - typically a PLC variable). These values (also called datapoints) are then visualized, for example, in an HMI app view.
- **Operate (write):**
The endpoint must support writing values to defined datapoint addresses, for example to set a setpoint.

Endpoint Adapter:

To communicate with endpoints, appropriate endpoint adapters are required. Unified-E already includes several adapters for common systems. For more details on configuration and functions, please refer to the respective adapter manuals.

Supported adapters:

- Siemens S7-ISO-on-TCP
- Allen Bradley Ethernet/IP
- OPC UA
- Modbus TCP/Serial
- Beckhoff
- Jetter
- SQL
- JSON Web

For the Unified-E Client platforms Android and iOS, only the following adapters are supported for Direct communication:

- Siemens S7-ISO-on-TCP
- Allen Bradley Ethernet/IP
- OPC UA
- Modbus TCP/Serial

Note: For Gateway communication via the Unified-E App Manager, all adapters are also supported on the Android and iOS platforms.

Configure Endpoint:

An HMI app can communicate with one or more endpoints at the same time. Typically, an endpoint has an IP address and other communication parameters. These parameters can also be adjusted at runtime - e.g. in the app itself or via a higher-level configuration.

1.3.5 Datapoint

A datapoint (also as a tag in other HMI software) is a defined address within an endpoint (such as a PLC controller, SQL server, or web server) that can be used to read or write a single value. The datapoints thus represent the connecting layer between the HMI app and the variables of the endpoint.

Each datapoint has a unique address or name within the endpoint and is typically associated with a specific process variable – e.g. temperature, meter reading, operating status or setpoint.

Datapoints enable the following functions:

- **Read (Observe):**
The HMI app reads values from the endpoint via the datapoint address – e.g. for display in a visualization.
- **Write (operate):**
The HMI app writes new values to the datapoint address - e.g. to change a setpoint or start a motor.

The number of endpoints and datapoints in the project is not limited by the license. For more information on datapoints, see Chapter 2.

1.3.6 Runtime

The runtime refers to the state in which the HMI app is executed on an operator device – either real on a target device (e.g. panel PC, tablet, smartphone) or simulated in the integrated simulator of the App Designer. During runtime, the app actively interacts with the controller and displays real-time data.

In contrast, there is engineering, in which the view elements are configured and designed in the App Designer.

1.4 Quick Start

The operation of Unified-E has been designed in such a way that the possibilities can be intuitively discovered even without a manual.

Demo project:

After starting the Unified-E App Designer, you can open a demo project that already demonstrates many of the features of Unified-E. To open it, click on the blue link "Open demo project here" in the "New" or "Open" section. The HMI app can then be started via the "Simulate" button in the toolbar.

Getting Started Guide:

On the Unified-E website, you can find a "Getting Started" guide that describes how to create a simple HMI app.

Info icons in the Unified-E App Designer:

There are info icons ("i" symbols) in many places, and when hovering with the mouse, further information about the context is displayed there.

1.5 Licensing

The Unified-E App Designer can be used free of charge for the creation of HMI apps. Execution on operator devices is also possible free of charge for development purposes – with the following restrictions:

- **Direct communication:**
HMI visualizations can be tested directly on an HMI device (e.g., Windows Panel PC). This requires a runtime license per operator device. Without a license, the HMI app can be run during a test period of 48 hours; After that, it must be re-registered.
- **Gateway communication:**
With a developer Gateway license, the Unified-E App Manager software can be licensed on the Gateway PC and thus the operation on the operator device can be tested.

All information about licenses, pricing and activation can be found on the Unified-E website.

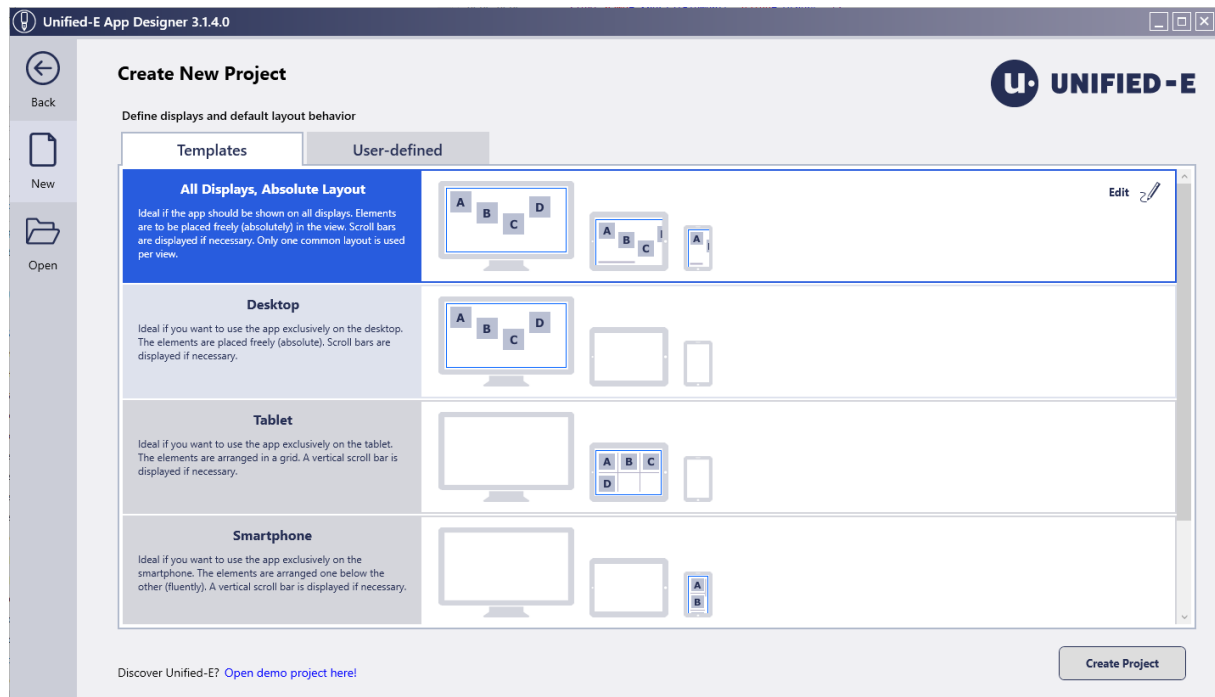
1.6 Create a New Project

With the Unified-E App Designer, HMI apps can be configured for different display sizes for the Windows, Android and iOS platforms.

If the size of a view to be displayed is larger than the display size, scrollbars will appear or the content to be displayed will be cut off, depending on the view configuration. If required, the view elements can be individually laid out for different display sizes, for example, a

Message Table can be displayed in table form on the main operating screen, and in a flowing layout with tiles on the smartphone. Details on this can be found in the chapter 3.3.

When creating a new project, depending on the use case, it is already defined which displays the HMI app should support and which standard layout behavior for the views may have to be selected for each display.



All settings here are available in the HMI project in the Displays editor (see Chapter 15.2) are customizable.

1.6.1 Templates for Displays and Default Layout Behaviors

For predefined use cases, templates for displays and the default layout behavior are available as follows.

1.6.1.1 Universal, absolute layout

Use case:

The HMI app is used on the large operating screen and also on the smartphone or tablet for remote monitoring.

Configuration:

Three displays (desktop, tablet, smartphone) are created. By default, there is only one shared layout in a view, which applies to all displays. By default, elements are laid out absolutely, i.e. with x, y position. If necessary, scroll bars are displayed.

1.6.1.2 Desktop

Use case:

The HMI app is only used for a local operator device with plant images etc.

Configuration:

A display (desktop) is created. By default, there is only one shared layout in a view, which applies to the single display (more displays can be added later). By default, elements are laid out absolutely, i.e. with x, y position. If necessary, scroll bars are displayed.

1.6.1.3 Tablet

Use case:

The HMI app is used for one or more operator devices with different sizes and resolutions.

Configuration:

A display (tablet) is created. By default, there is only one shared layout in a view, which applies to the single display (more displays can be added later). By default, elements are laid out in tiles, i.e. "responsive" in rows and columns. If necessary, there is a vertical scrollbar.

1.6.1.4 Mobile

Use case:

The HMI app is to be used on one or more smartphones with different sizes and resolutions.

Configuration:

A display (smartphone) is created. By default, there is only one shared layout in a view, which applies to the single display (more displays can be added later). Elements become flowing by default, i.e. the elements are laid out "responsively" in width. If necessary, there is a vertical scrollbar.

1.6.1.5 Tablet, Mobile

Use case:

The HMI app is intended to be used on smartphones as well as tablets with different sizes and resolutions. The view areas are to be displayed as single-column tiles on the smartphone and multi-column on the tablet.

Configuration:

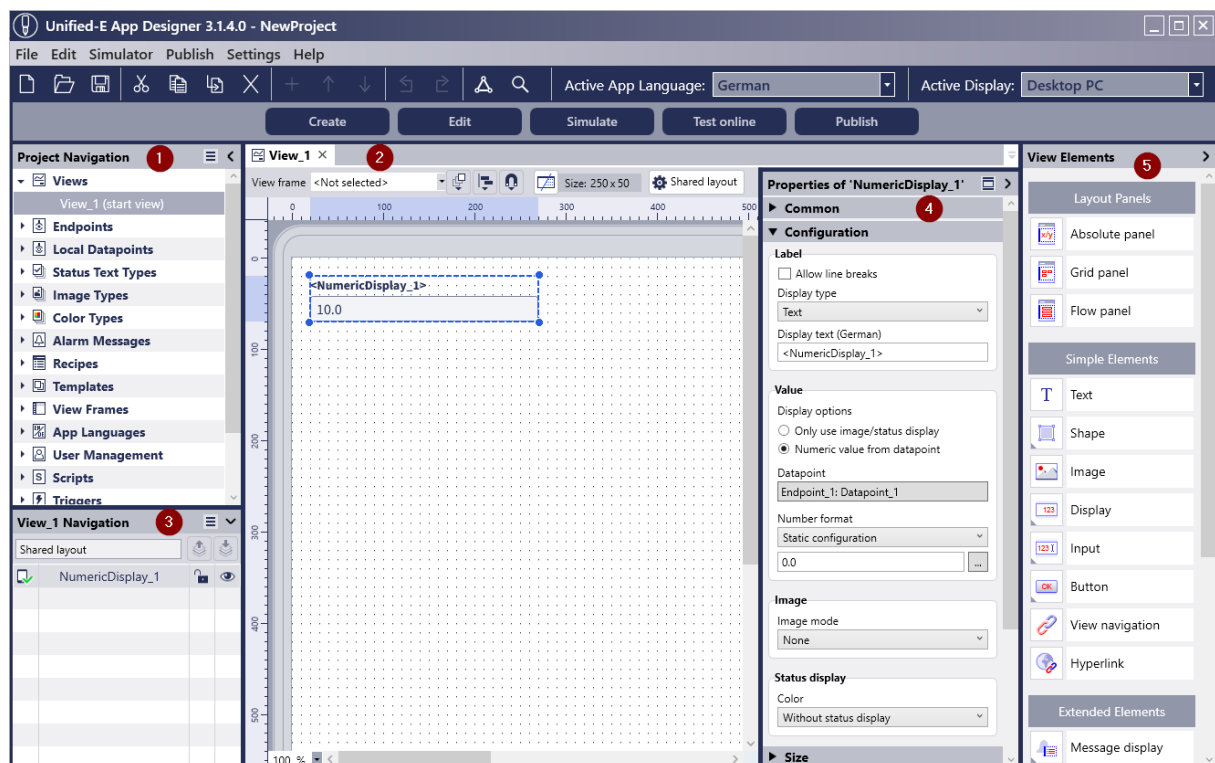
The displays (smartphone, tablet) are created. There is a shared layout by default, which is linked to the tablet. For smartphones, an individual layout is also defined in the view. View elements can thus be placed in the smartphone layout in the flowing layout and for the tablet in the grid layout.

1.6.2 User-defined Displays and Default Layout Behaviors

If you select the "Custom" tab under "Create new project", you can individually specify which displays are to be created in the HMI app and which standard layout is used for new views.

1.7 Major Areas of the HMI Editor

The editor is divided into different areas, which are described below.



1.7.1 Project Navigation

The Project Navigation (area 1 in figure) offers you a structured overview of all components and editors of your project. It is the central control to quickly access all objects and configuration items.

In the Project Navigation, editable objects and elements can be opened by double-clicking in the editor. Objects can be copied, duplicated, deleted, or renamed via the context menu.

1.7.2 Editor Area

The Editor area (area 2 in the figure) opens editors for complex objects such as views, endpoints, or configuration items.

Multiple editors can be open at the same time. Individual editor tabs can be removed by drag and drop and displayed in their own window. For example, an editor can be placed on a second monitor to work in parallel.

1.7.3 Element Navigation

The Element Navigation (area 3 in the figure) is active as soon as a view editor is open. It displays hierarchically all view elements that are contained in the currently open view.

The element navigation is particularly helpful for absolute layouts, as elements can overlap or be hidden in the editor interface.

Functions of the Element Navigation:

- Select an element: Navigates directly to the corresponding element in the editor.
- Move elements forward or backward:
The toolbar buttons can be used to move elements forward or backward one level. This feature is only available in the absolute layout.
- Hide element in active layout (first column): Useful if you want an element to be visible only in certain layouts (e.g. tablet, but not smartphone). This function has an effect on the runtime of the HMI app.
- Lock Element (Second Column): Prevents an element from being selected or accidentally moved in the editor.
- Hide element in editor: (third column)
Allows you to temporarily hide an element, e.g. to be able to edit an object behind it. This has no effect on the runtime of the HMI app and only serves to make it more efficient.

1.7.4 Properties Pane

In the Properties pane (area 4 in the figure), further settings can be made for the object selected in the editor.

The displayed properties are divided into property groups, which in turn are grouped together in so-called palettes. This enables structured and clear processing of even extensive objects.

1.7.5 Views Elements Pane

The Views Elements pane (pane 5 in the figure) is visible when a View editor, Template editor, or View Frame editor is active.

From there, you can drag and drop view elements such as Buttons, Texts, Images, or Layout Panels into the open view and place them there.

1.7.6 Workflow Bar

In the Workflow bar (area 6 in the figure), central actions are available to guide your project through the key editing steps.

The following functions are available:

- **Create:** Create new objects.
- **Edit:** Opens the editor of the selected object.
- **Simulate:** Shows the HMI app in the simulator window and uses simulated endpoints (see Chapter 13.2).
- **Test online:** Shows the HMI app in the simulator window and uses an endpoint connection with the configured address (see Chapter 13.2).
- **Publish:** Creates an app package file for the configured HMI app. The HMI app can be registered directly to a Gateway PC or used for Direct communication to an operator device (see Chapter 14).

The Workflow bar is located at the top of the Unified-E App Designer and is always visible for quick access to all phases of project development.

2 Manage Datapoints

Datapoints are objects that hold a value that can be read or written. For endpoint datapoints of a PLC, the current datapoint value is synchronized with the value of the associated PLC variable every second.

Communication with endpoints (e.g. a PLC) is always done via datapoints in the HMI app. For example, a PLC variable value can be visualized in a display element, or Images can be displayed or hidden depending on the PLC variable value.

Many objects in the HMI app can be linked to datapoint objects to influence the display, display PLC values, or send values to the PLC.

Unified-E supports the following types of datapoints:

- **Endpoint Datapoints:**
The datapoint value represents an endpoint value (such as a PLC variable). This requires constant communication with the endpoint.
- **Local Datapoints:** The datapoint value is managed locally by the operator device (Unified-E Client) or in the Gateway PC (Unified-E App Manager). Communication with external devices is not necessary here.
- **Script datapoints:**
A script datapoint is a type of "virtual" datapoint. Here, JavaScript code is used to define what should happen when reading or writing the datapoint value.

The different types of datapoints are described in more detail in the following subchapters.

2.1 The Datapoints Table

Datapoints are displayed in the datapoints table in the respective editor of the object (e.g. endpoint object, container object for local datapoints).

The datapoints table enables the editing of datapoints as well as sorting and filtering using input fields in the column headers.

The following is a list of the columns that all datapoints have in common.

Name column:

The label uniquely describes the datapoint within the container or endpoint.

Access Column:

The access setting defines whether the HMI app has read-only access or read and write access to the datapoint.

- "Read": With "Read", the HMI app may only access the value in a read-only manner and can therefore only be used to display or evaluate conditions.
- "Write, Read": In "Write, Read", the datapoint value may be read as well as written. This access therefore also allows the use of the datapoint for input fields or buttons.

Data Type Column:

The data type describes the type or kind of datapoint value. There are the following data types in Unified-E:

- "Numeric":
This type encompasses all numbers and is stored internally as a 64-bit floating point number. All numeric types such as INT/DINT must be mapped to this type.
- "Yes/No":
This type can only contain the values "Yes" (= 1) or "No" (= 0) and is comparable to a bit.
- "Text":
This type is also known as "string" in programming languages and comprises a string of characters.
- "Table":
This type contains a table with columns and rows. Datapoints with this type can currently only be used in the "List Panel" and "Chart" view elements. The data type "Table" is the only datapoint that does not belong to the simple data types (see Chapter 2.2.5).

Simulator Start Value Column:

In this column, the initialization value when simulating the HMI app with simulated endpoints must be entered, which can be started with the "Simulate" button in the Workflow bar.

Group Column:

Here you can enter a group name, which is only used for project configuration and can be helpful as a filter criterion for datapoint listings.

Cross-reference column:

This column is read-only and shows the number of links (cross references) to this datapoint. When you click on the navigation symbol, an area with the list of cross references appears at the bottom.

Comment column:

This column is used for project configuration to annotate datapoints.

Current Value Column:

This column is only displayed when simulating the HMI app and shows the current datapoint value.

- Simulate with "Simulate": The property "Current value" is editable, so the datapoint value can simply be set here.
- Simulate with "Test online": The "real" datapoint value of the endpoint or controller can only be read. Writing about this table is not allowed for security reasons.

Note: The HMI app simulation can be started in the Workflow bar in the upper editor area.

2.2 Create Endpoint Datapoints

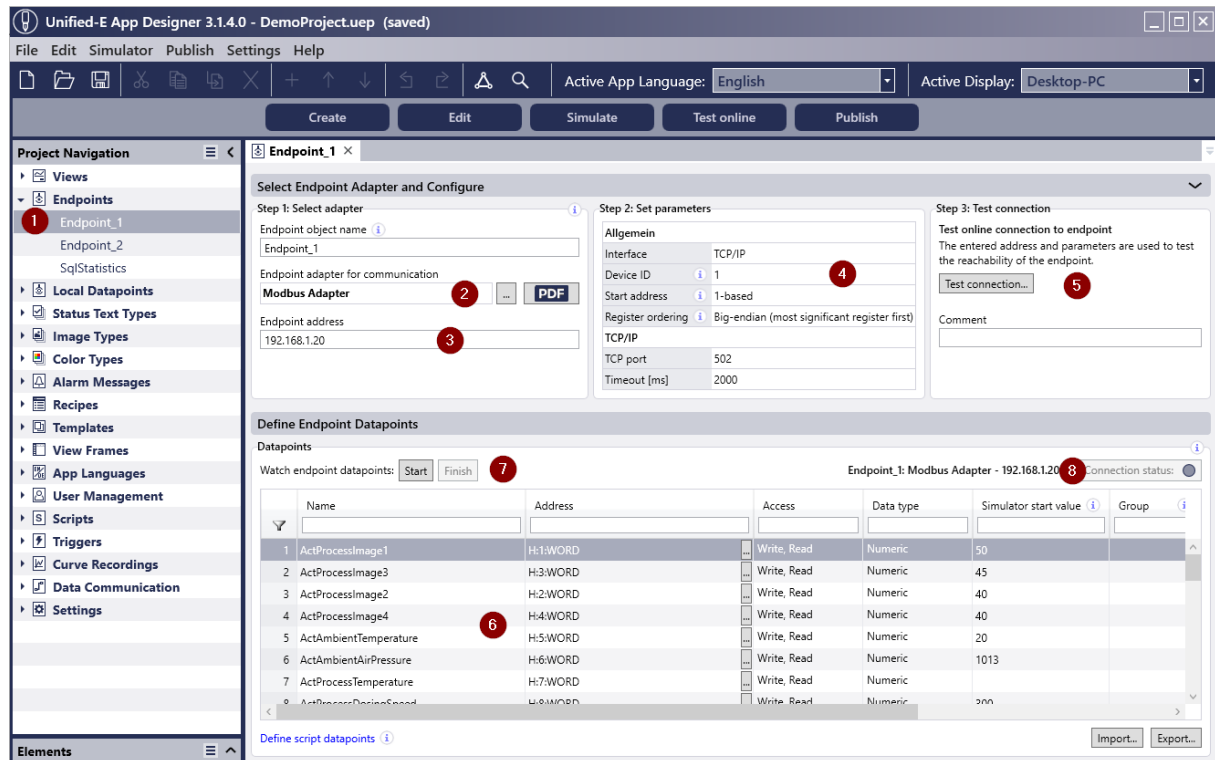
An endpoint datapoint is associated with a variable/value in the endpoint, such as a variable in PLC control. The datapoint value is read by the HMI app cyclically (usually per second), which requires a communication connection to the PLC or endpoint.

Address Column:

Endpoint datapoints, unlike local and script datapoints, also have an address that identifies the datapoint value in the endpoint. The structure of the datapoint address depends on the endpoint adapter and is documented in the respective PDF documentation of the adapter (see below).

2.2.1 The Endpoint Datapoint Editor at a Glance

The following is an example of an open endpoint datapoint editor.



Typical steps for managing the endpoint datapoints (numbered as shown in the figure):

1. Start the editor:
Double-click the endpoint object in the Project Navigation.
2. Select endpoint adapter:
Depending on the controller or data source, the appropriate endpoint adapter must be selected by clicking on "...".
3. Set address:
The address of the endpoint (PLC) must be entered. For most adapters, the IP address must be entered here.
4. Adjust parameters:
Depending on the selected adapter, different parameters will appear, which must be set accordingly depending on the endpoint. The PDF documentation of the endpoint can be very helpful, simply click on the "PDF" button to get a description of the parameters.
5. Test connection:
Clicking on "Test connection" checks whether a connection can be established with the endpoint. Make sure that the device is connected to the computer with the Unified-E App Designer open, is on the same network, and that there are no firewall settings blocking communication.

6. Manage datapoints:
In the datapoints table, the datapoints are edited, created, or deleted. To add a new datapoint, click the "Add new datapoint" button.
7. Check datapoints:
Before you use the created datapoints or link them to other objects, we recommend an initial test. By clicking on "Start", an endpoint connection is established and the datapoint values are cyclically retrieved from the endpoint (e. g. controller). The "Current value" column is displayed, and read errors are displayed in "red". An incorrect configuration (e.g. incorrect address) can thus be determined quickly.
8. In-depth diagnostics:
Errors during connection establishment or when reading the datapoint values can be displayed in more detail in the connection status dialog. This can be started during the simulation or while observing the endpoint datapoints by clicking on the "Connection Status" button.

2.2.2 Fix Communication Problems

2.2.2.1 Test the connection to the endpoint

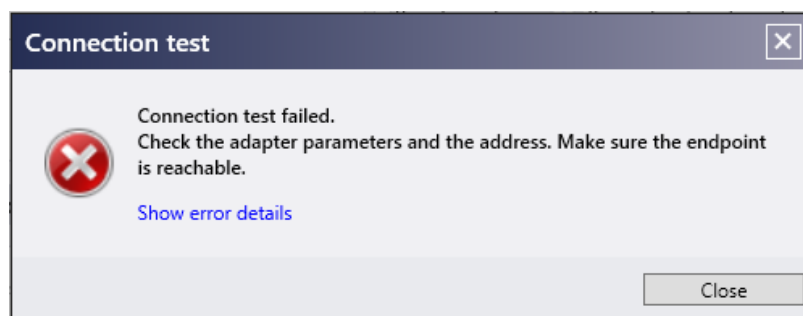
As already described above, clicking on "Test connection..." the connection to the endpoint can be tested.

Error "Could not connect within specified time":

This error means that communication with the endpoint could not be established within the selected timeout value. Typical causes are:

- Endpoint not reachable from the computer because of a different network
- Firewall rules block communication
- Timeout value too low in the parameters

Often, clicking on "Show error details" in the connection test dialog (see image below) helps with connection diagnostics to get more detailed information. In addition to network problems, a common cause is incorrectly set parameters.



2.2.2.2 Diagnosing Datapoint Errors

After the connection to the endpoint has been successfully established – as described in the previous chapter – the diagnostics of the datapoints can begin.

Start a real connection to the endpoint by either running "Test Online" from the Workflow bar or starting "Watch endpoint datapoints" in the Datapoints table.

Read errors are marked in red in the datapoints table – often with explanatory error messages.

A detailed diagnosis, including for spelling errors, is possible by clicking on "Connection status" above the datapoints table. A diagnostic dialog opens with extended error messages that provide a more detailed description of the problem.

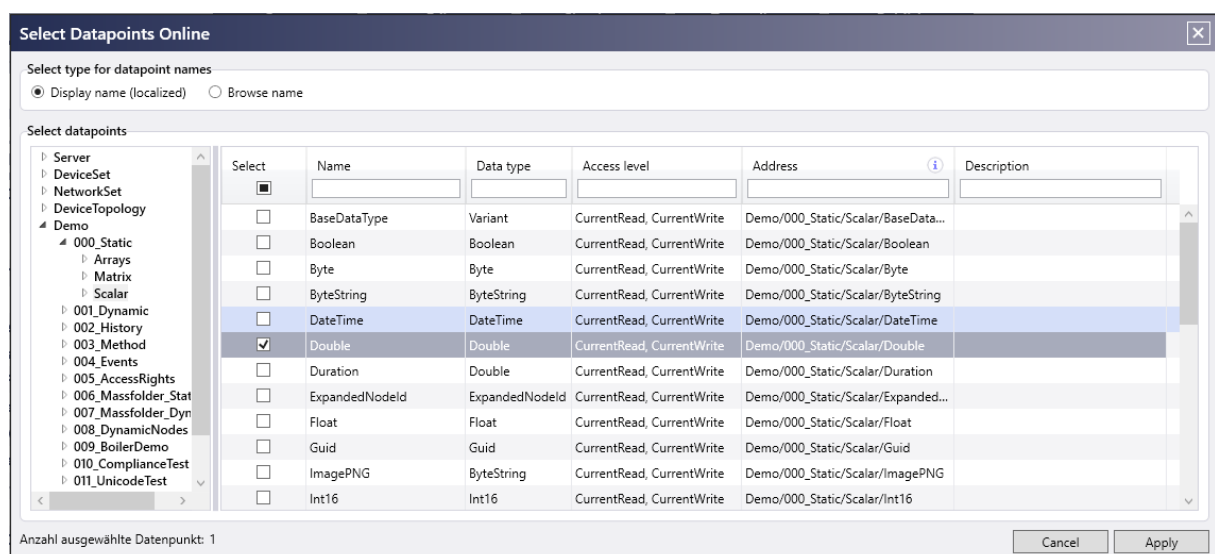
The most common causes of errors:

- Read Timeout: This can occur with large views with many datapoints and low network bandwidth. Often, increasing the timeout parameter at the endpoint or checking the network helps.
- Incorrect address: The datapoint address is incorrect. Check the address in the PDF documentation of the endpoint adapter.

2.2.3 Selecting Datapoints Online

When selecting the "OPC UA Adapter" endpoint adapter, the "Select datapoints online..." button appears above the datapoints table. This button allows datapoint addresses to be selected directly from the connected OPC UA server. After clicking, a dialog opens in which the desired variables can be selected and transferred to the HMI project as datapoints. The manual typing of the datapoint address is then not necessary.

This function connects to the endpoint, so the endpoint address and endpoint parameters must already be configured.



2.2.4 Configure Formulas

Formulas based on datapoints can be created in Unified-E with script datapoints.

Example:

The value "Datapoint1 + Datapoint2" should be displayed in a display element "Numeric Display":

Script datapoints are described in the chapter 2.4 described in detail. The Script Datapoint editor can be opened both via the Project Navigation and below the datapoints table via the "Define Script Datapoints" link.

2.2.5 Table Datapoints

Table datapoints are datapoints of the "Table" data type. Its value consists of a table with rows and columns.

A table datapoint is configured in the datapoints table as follows:

1. Set access to "Read"
2. Set data type to "Table"
3. Set address that returns a table

Using Table Datapoints:

Table datapoints are used in the following view elements:

- **Chart Panel:**
The table datapoint is mapped to a chart data series. The table datapoint is assigned to a chart data series. The table values are used to represent a data series, for example in a line or time chart (see Chapter 5.3.6).
- **List Panel:**
The list can be configured with multiple columns. The linked table datapoint provides the entries for the list (see Chapter 5.3.5).
- **Dropdown List:**
The Dropdown List can be configured so that the entries come from a table datapoint (see Chapter 5.3.6).

Table datapoints in endpoint adapters:

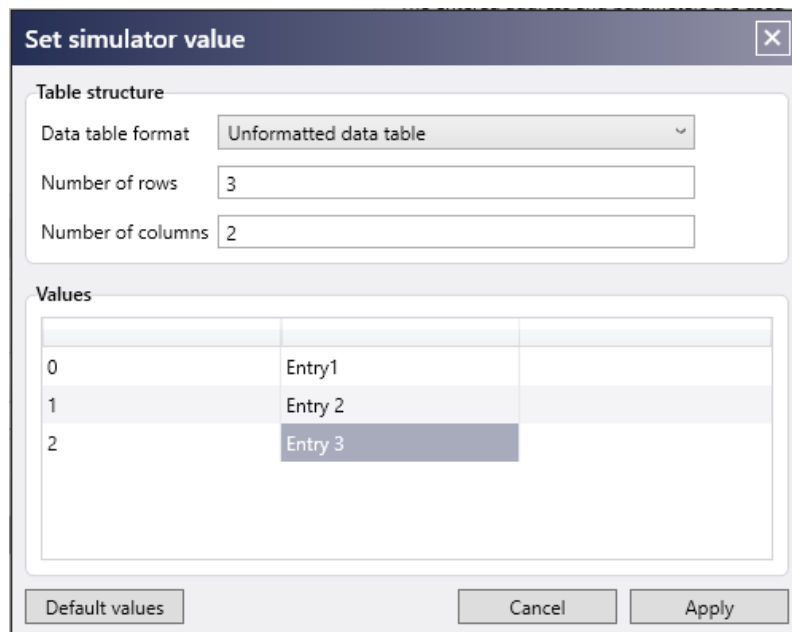
- **SQL adapter:**
The datapoint address can be used to define a "SELECT" command with several columns.

All endpoint adapters support arrays, which can be configured as a table.

Predefine simulator start value:

The starting value for the App Simulator can be set via the "..." in a dialog box. There you can enter the desired table value.

Similarly, during the simulation, the current table value can be displayed in the "Current value" column via the "..." are displayed.



The dialog box titled "Set simulator value" contains two main sections. The "Table structure" section has a "Data table format" dropdown set to "Unformatted data table", a "Number of rows" input field with the value 3, and a "Number of columns" input field with the value 2. The "Values" section displays a table with 3 rows and 2 columns. The first column contains indices 0, 1, and 2. The second column contains "Entry1", "Entry 2", and "Entry 3". The row with index 2 and "Entry 3" is highlighted. At the bottom, there are three buttons: "Default values", "Cancel", and "Apply".

0	Entry1
1	Entry 2
2	Entry 3

2.3 Create Local Datapoints

A local datapoint in Unified-E is a custom datapoint that is not connected to a controller (PLC) and is used exclusively within the HMI app. It is used to store states, variables or auxiliary values locally in the HMI, to display them or to use them for script operations – without communication with an endpoint.

Typical areas of application are:

- Control the visibility or state of UI elements: Tab switching can be easily implemented in this way.
- Caching of user input and processing it in the script.
- Auxiliary variables for scripts or trigger conditions.

Local datapoints do not require a connection to a PLC or other endpoint.

The editor for local datapoints can be opened in the Project Navigation by double-clicking on the corresponding container object under "Local datapoints".

2.3.1 Initialization of the Local Datapoint

For initializing and storing the local datapoint, there are two additional columns in the datapoints table:

"Initial Value" column:

Contains the start value, for example, when the HMI app is restarted for the first time.

"Initialization" column:

Describes how to initialize the datapoint when the app starts.

- "Start with initial value":
The datapoint is always initialized with the initial value.
- "Start with saved value":
The datapoint is started with the last stored value, and any value change is automatically saved. When starting for the first time, the initial value is used.

2.3.2 Runtime of Local Datapoints

The runtime property of a local datapoint determines whether the datapoint memory (instance) should be managed in the operator device or in the Gateway PC. In the case of Direct communication between the operator and the endpoint, the datapoint instance is always managed in the operator device.

Runtime Column:

"Operator device / App Manager":

The datapoint is managed centrally in the App Manager for Gateway communication. The datapoint can therefore be used in messages, recipe types, datapoint connections and curve recordings.

"Per operator device":

The datapoint is managed in the operator device regardless of the type of communication and is therefore not usable in the App Manager. Therefore, the datapoint cannot be used for messages, recipe types, datapoint connections, and curve recordings.

2.4 Creating Script Datapoints

Script datapoints can be used in the same way as endpoint datapoints or local datapoints. They can be selected for view elements or messages wherever datapoints with read or write access (depending on the property) are also allowed.

The editor for script datapoints is started in the Project Navigation under "Scripts" → "Script Datapoints".

Script Datapoint Value:

As with other datapoints, the value of a script datapoint can be read or written by the HMI app, depending on the access configured. However, unlike endpoint datapoints or local datapoints, the value is not stored directly.

When reading, a read script defines which value is returned. When writing, a writing script determines which datapoints are ultimately to be written.

Script with JavaScript syntax:

The script code for reading and writing is done with JavaScript ES6. The script code can access datapoints indirectly via mapped JavaScript variables to read or write data.

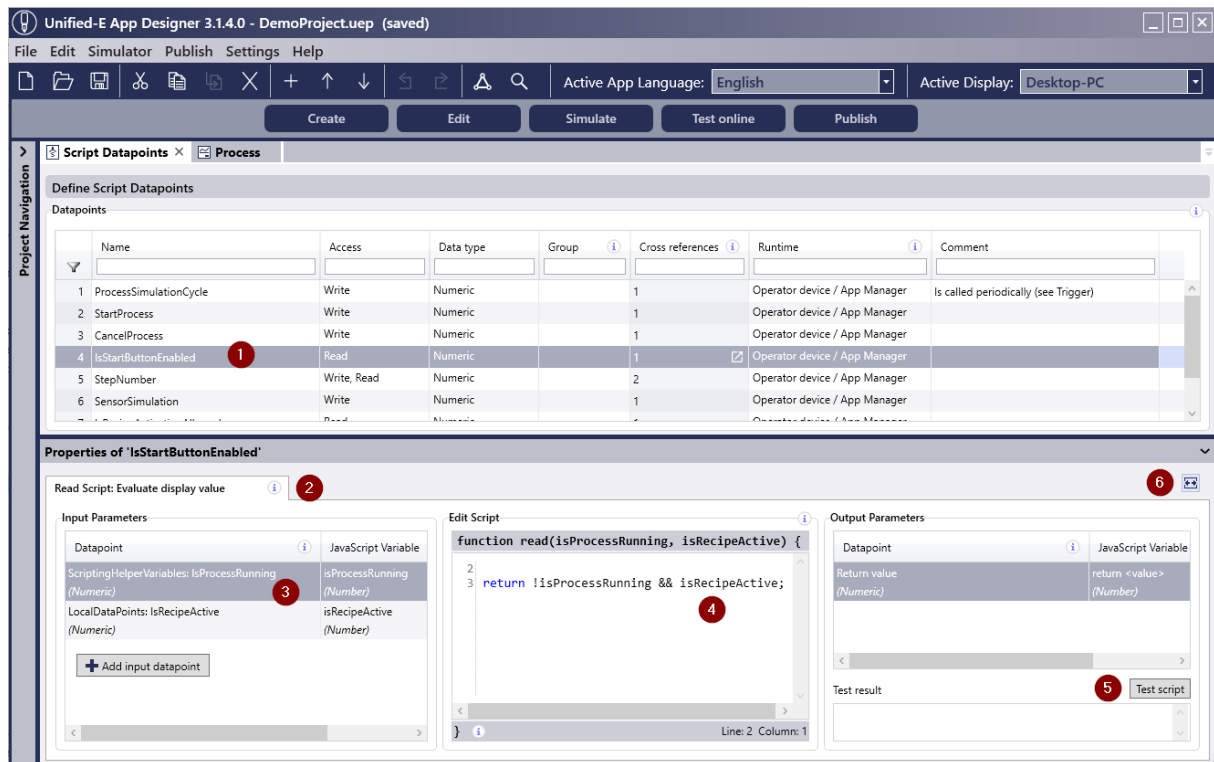
Script Datapoint Access Options:

The "Access" property has three different values for the script datapoint, as described below.

- "Read":
Only one read script is required, which calculates and returns the current value for each read access. The datapoint is not writable. Script datapoints with "read" access are ideal for implementing formulas based on one or more other datapoints.
- "Write, Read":
This access allows both reading and writing the datapoint value and can be used, for example, to display and enter in an input field. Both a read and a write script is required. This option is ideal for so-called converter datapoints, which convert entered values into another unit.
- "Write"
All that is required here is a write script. When writing (setting the datapoint value), a JavaScript function is called that sets other datapoints as part of its execution. Script datapoints with "write" access can only be used for buttons or trigger actions where a script is to be executed as soon as a certain condition or user action occurs.

2.4.1 Formula with "Read" Script Datapoint

The following image describes the creation of a "read" script datapoint, as is necessary for the creation of a formula, for example.



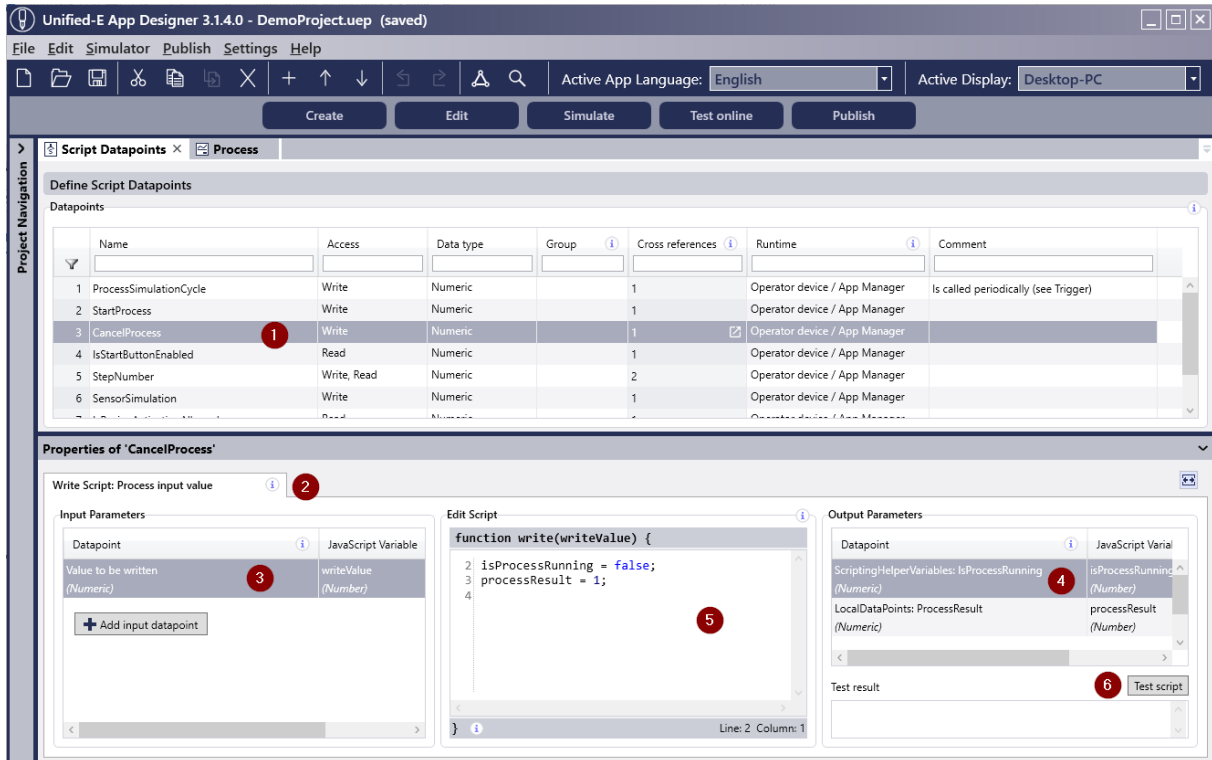
Description of the steps (numbering according to the figure):

1. Create datapoint:
The datapoint must be created and access must be set to "Read".
2. Select read script:
Define the script under the tab "Read Script: Evaluate display value" as follows.
3. Add and map input parameters:
Select all datapoints that the formula depends on and assign each one to a JavaScript variable. The variable name is freely selectable and can be used within the JavaScript code.
4. Implement JavaScript code for "Read":
With the help of JavaScript code and the JavaScript variables of the mapped input parameters, the formula value is determined and returned by means of "return".
5. Test script:
Already in the editor, you can test by clicking on "Test script" and check the return value in the results column under "Initial parameters". The mapped JavaScript variables of the input parameters can be predefined with test values under "Input parameters".
6. Enlarge code window:
Click the arrow icon to widen the code window and collapse the parameter lists. You can also use the upper splitter to increase the height of the code window.

2.4.2 Script Routine with "Write" Script Datapoint

A "write" script datapoint defines a script routine that is explicitly called by the associated object (e.g. button, trigger action).

All that is required is a writing script.



Description of the steps of the picture:

1. Create a datapoint: Create the datapoint and set the access to "Write".
2. Select write script:
The script can be configured under the tab "Write Script: Use Input Value" as follows.
3. Add and map input parameters:
All datapoints necessary for the JavaScript calculation must be selected here and mapped to a JavaScript variable.
4. Add and map output parameters:
All datapoints that are to be written with the mapped JavaScript variables after executing the script must be defined here.
5. Create JavaScript code:
Calculations are made with the help of JavaScript code and the JavaScript variables of the mapped input parameters. Then the mapped JavaScript variables of the initial parameters have to be set.
The "writeValue" variable, which contains the set value when the script is called (value when setting the script datapoint), can also be taken into account during execution.

6. Test script:

You can test in the editor by clicking on "Test script". The calculated values can be checked in the results column under "Initial parameters". The mapped JavaScript variables of the input parameters can be predefined with test values under "Input parameters".

2.4.3 Converter with Script Datapoint

Often, values have to be converted into different units during reading and writing, because the PLC or endpoint specifies a fixed unit that is not to be used in the HMI.

Example:

The PLC controller always provides the temperature value in Celsius in a datapoint "TemplnCelsius", and in a numerical input the input (and display) is to be made in Fahrenheit.

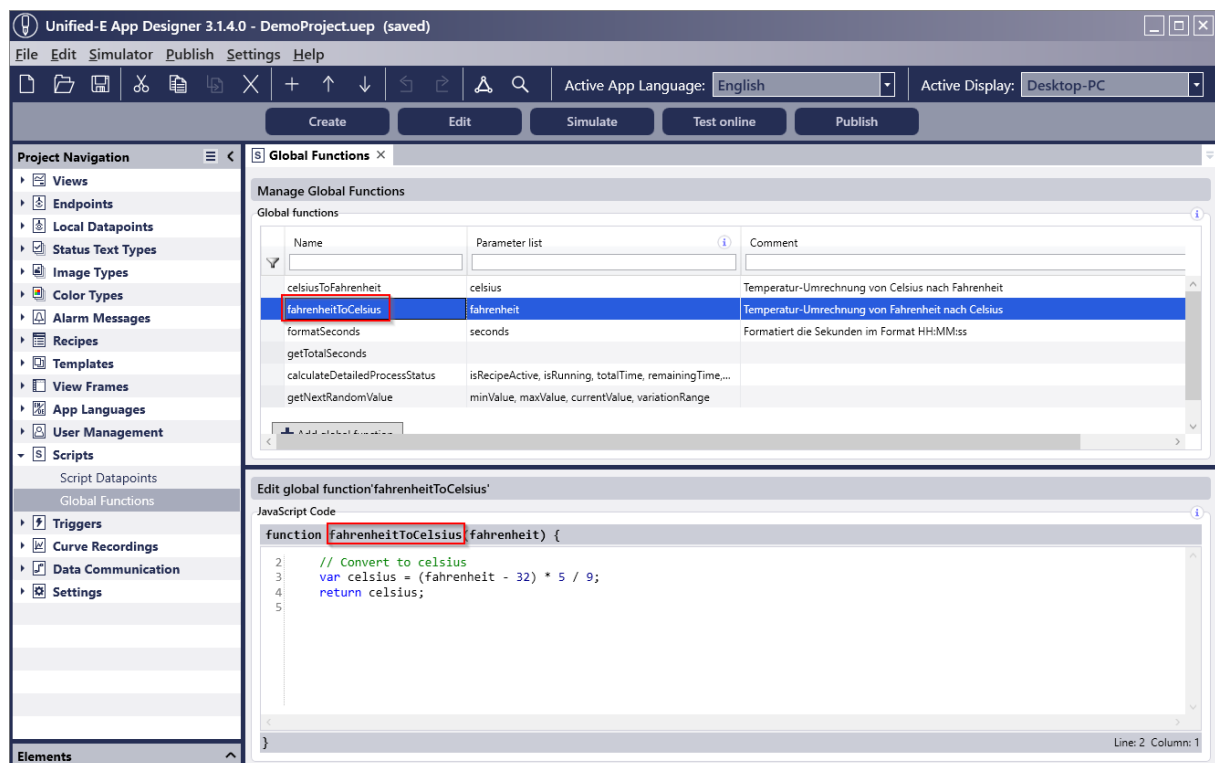
Solution:

1. Define the endpoint datapoint "TemplnCelsius".
2. Define the Script datapoint TemplnFahrenheit.
 - a. Set access to "Write, Read".
 - b. Create read script according to example 2.4.1. "TemplnCelsius" is to be used as the input parameter. The script code converts the value of "TemplnCelsius" to Fahrenheit and returns it with the return command.
 - c. Create a writing script according to the example 2.4.2: The script code converts the passed writeValue (value from the input field) to Celsius and sets the output parameter "TemplnCelsius" over the mapped script variable.
3. The value of the "Numeric Input" display element must be linked to the script datapoint "TemplnFahrenheit".

2.4.4 Global Script Functions

Global functions are used to provide shared script code centrally. They can be called in both read scripts and write scripts from script datapoints.

The editor for creating global script functions is opened in the Project Navigation under "Scripts" → "Global functions".



Steps to create a new global script function:

1. Add a function:
Clicking on "Add global function" creates a new function with standard properties.
2. Set function name:
In the "Name" column, set the object name, which also serves as the function name.
3. Define parameter list for input parameters:
In the Parameter List column, the function parameters are to be defined in a comma-separated list, which are to be used as variables in the code.
4. Program Code:
In the lower editor area, the code of the currently selected function appears in the list. The function header is already specified, only the functional body has to be programmed.

2.5 Import and Export Datapoints

The most efficient way to add datapoints is to use the import function below the datapoints table. Existing datapoints with changed addresses can also be quickly updated in this way.

2.5.1 Importing from Excel via the Clipboard

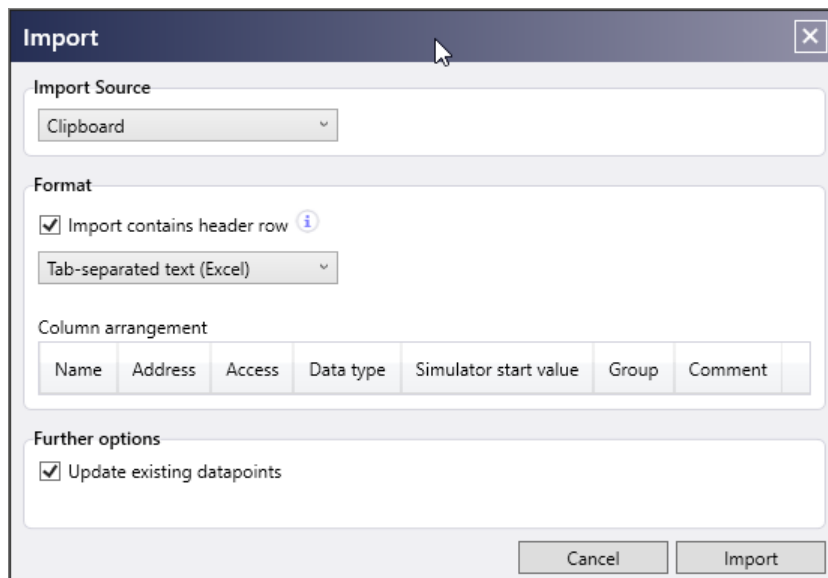
An easy way to import datapoints is through the clipboard with data from Microsoft Excel, as described below.

Use case:

Many datapoints are exported from a PLC engineering system (e.g. Siemens TIA Portal) and are to be transferred to Unified-E.

Steps:

1. Export the datapoints from the PLC engineering system to Excel.
2. Prepare the Excel file to match the format expected by Unified-E. You can determine this import format by performing an example export as described below.
3. In Excel, select all relevant rows including the header row and copy them to the clipboard.
4. Open the import dialog with the "Import..." button below the datapoints table
 - a. Set the import source to "Clipboard".
 - b. Under Format
 - i. Select the "Import contains header row".
 - ii. Select "Tab-delimited text (Excel)".
 - c. Under "Further options" you can specify whether new datapoint objects should be created for each row, or whether existing datapoints (identified by the name) should only be updated.
 - d. By clicking on "Import", the import process is carried out by processing the Excel data from the clipboard.



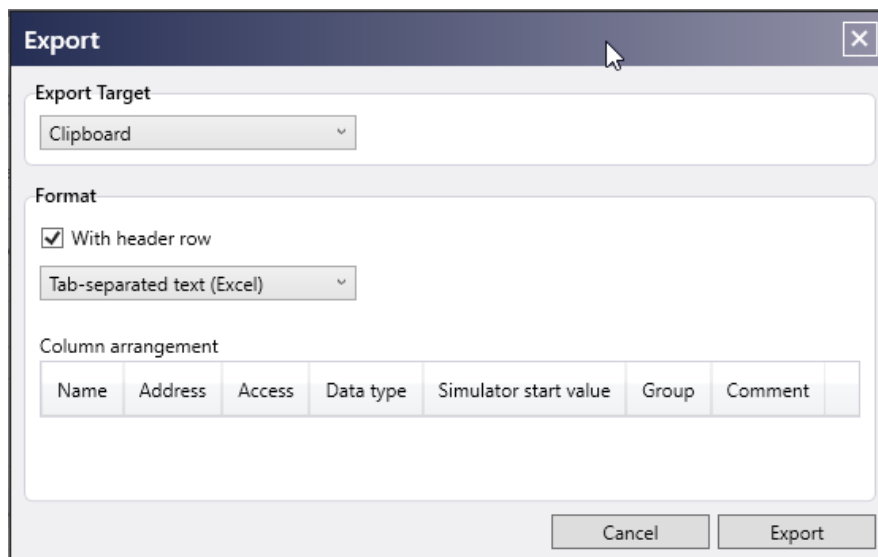
Column arrangement						
Name	Address	Access	Data type	Simulator start value	Group	Comment

2.5.2 Export to Excel from the Clipboard

The steps to export from the clipboard are as follows:

1. Open the Export dialog with the "Export..." button below the datapoints table

- a. Set export target to "clipboard"
 - b. Under Format
 - i. Select the "With header row" checkbox
 - ii. Select "Tab-delimited text (Excel)"
 - c. By clicking on "Export", the export process is carried out by storing the datapoint for Excel on the clipboard
2. Open Excel and select a blank worksheet
 - a. Select the first cell in the worksheet
 - b. Paste to paste the content from the clipboard



2.5.3 Other Import/Export Options

In addition to the import and export option via the clipboard, there are other options for exchanging datapoints.

Text file as an import/export option:

Instead of the clipboard, "Text file" can also be selected as the import source or export destination in the dialog. The full file name must then be selected here.

CSV format for text files:

While the tab character is recommended as a column separator for import/export with Excel, the CSV format is usually suitable for text files. If you select "CSV Format" in the dialog, the desired separator (e.g. comma or semicolon) can be explicitly defined.

2.5.4 Replace Datapoints

The "Replace datapoint" function can be used to replace all points of use of a datapoint.

Procedure:

1. Select a datapoint in the datapoints table
2. Open the context menu of the selected datapoint, select the menu item "Replace datapoint..." choose
3. In the "Replace Datapoint" dialog, select the new datapoint that should be used for the usage points of the old datapoint when replaced.

3 Designing Views in the Graphical editor

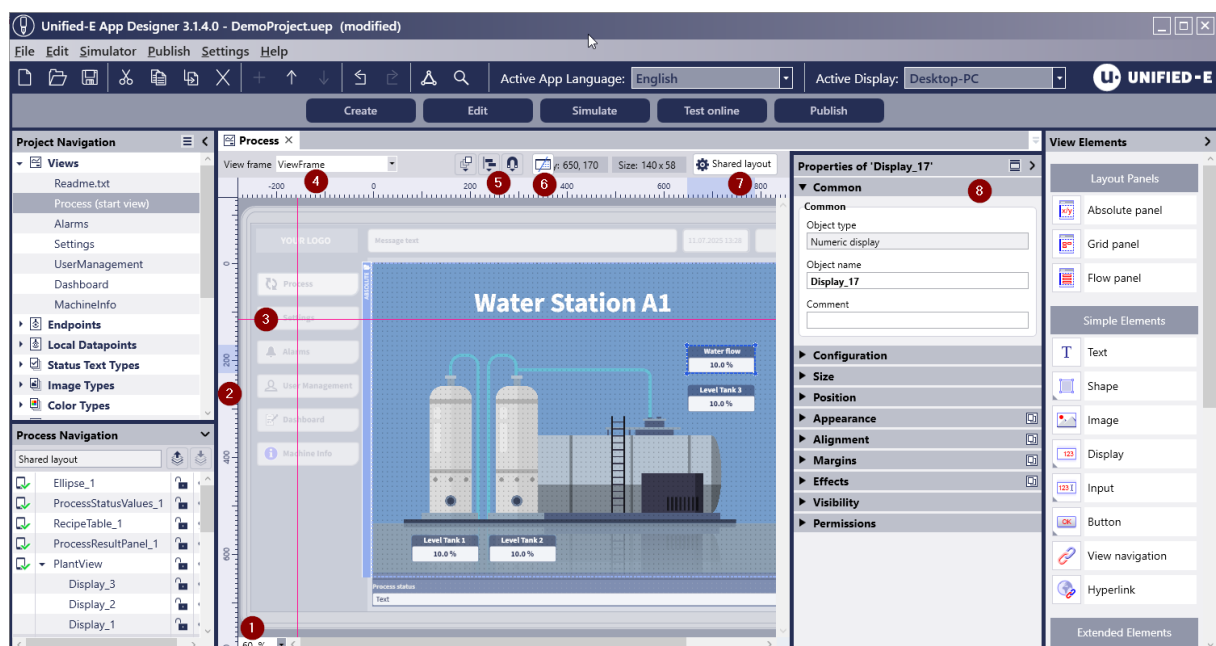
The operation and monitoring of a machine or plant is carried out via the views of the HMI app. In these views, machine states are displayed using view elements such as "Numeric Display", dynamic Images or Shapes.

Buttons and input fields also enable direct operation of the machine – via touchscreen, mouse or keyboard input.

The HMI interface is configured in the graphical editor by creating multiple views that users can navigate between.

3.1 The View editor at a Glance

The editor for the view configuration is opened in the Project Navigation by double-clicking on the view object under "Views".



Important elements of the editor (numbering as in the figure):

1. **Adjust zoom:**
The zoom input allows you to zoom in or out of the view area. Zooming is also possible with the mouse with CTRL + mouse wheel action within the editor.
2. **Ruler:**
With the help of the ruler, view elements can be positioned in a targeted manner or element sizes can be adjusted.
3. **Guides:**
Guides can be created by dragging with the mouse from the ruler. Elements automatically snap to a guide when they are moved close to it.
4. **Set View Frame (optional):**
Here you can optionally select a View Frame that contains common view areas (such as the header or the navigation area). The View Frame is semi-transparent in the editor (see image). Double-clicking on the View Frame area opens the View Frame editor. View Frames are described in the chapter 3.9 described in detail.
5. **Toolbar:**
The toolbar of the graphical editor includes functions for editing elements, which are described in the chapter 3.7 are described.
6. **"Show layout aids" button:**
With the "Show layout aids" toggle button, all tools such as positioning grids or guides can be quickly shown or hidden in order to display the view in the editor as if it were running time during engineering. Placeholders in container elements are also shown and hidden here.
7. **Configure the layout behavior of the view elements for multiple displays:**
This button can be used to configure individual layouts for different displays. If there are several displays, it must be determined whether a common or a display-specific layout should be used. Further information can be found in the chapter 3.3.
8. **Button "Always display one palette:**
If the toggle button is set then opening a palette ensures the other palettes get automatically closed.

Create a new view:

A new view can be created via the context menu in the Project Navigation on the "Views" folder by selecting the "Add View" command.

Set start view:

If multiple views are used, one view must be defined as the start view that appears first when the HMI app starts. The start view can be set in the Project Navigation by right-clicking the corresponding view entry and selecting Set as shared start view from the context menu (see also Chapter 15.2).

3.2 Layout Elements in a View

3.2.1 Set Layout Type on View

In the editor, view elements are layouted either via the properties or by drag-and-drop within a view. The size, position and rotation of the element can be determined.

Pixel specifications:

When layouts with absolute values, the inputs are made in pixels. These are "virtual" pixels – depending on the operator device.

For Windows: The pixel specifications refer to a screen resolution of 96 dpi.


For Android/iOS: The pixel specifications refer to a screen resolution of 160 dpi.

Define layout type:

The arrangement of the view elements within a view is determined by the layout type. This can be defined in the Properties pane under "General" → "Layout".

To edit the properties of the view, the view must be selected, e.g. by clicking on the editor background.

Layout

Active layout 
Desktop PC: Shared layout

Layout type

☒ ABSOLUTE
☐ GRID
☐ FLOW

Positioning grid X (px)
10

Positioning grid Y (px)
10

Scrollbar settings

☒ Display scrollbars and enlarge automatically
☐ No scrollbars and crop elements

Padding

Left	Right	Top	Bottom
0	0	0	0

The three layout types "Absolute", "Grid" and "Flow" are described in detail in the following subchapters.

3.2.2 Absolute Layout:

With the absolute layout, the elements can be freely placed on the view area, i.e. the properties X, Y, width and height can be freely defined via drag & drop or in the properties.

To layout an element via the properties, it must be selected in the editor so that its properties are visible in the Properties pane.

Configure the absolute layout of the view:

When selecting the "Absolute" layout type in the View Properties, all elements are absolutely laid out on the canvas. With the properties "Grid raster X" and "Grid raster Y", a positioning grid can be individually defined to which the elements are to be aligned during layout via drag and drop (snapping). A grid raster of 10 means that when dropping, an element can only be placed in increments of 10, such as position 10, 20, 30, etc.

For the layout of the element, the "Size" and "Position" palettes are available in the element's Properties pane.

"Size" palette: Set dimensions for the element:

Under "Size", the fixed width and height are defined in pixels. Alternatively, both the width and the height can be set dynamically over a datapoint in the "Dynamic Configuration" properties group and thus changed at runtime depending on machine conditions.

"Position" palette: Position element:

- **Position:** This is where the X, Y position of the element is set. This refers to the distance between the upper left corner of the element and the edge of the view.
- **Anchoring:** The anchoring is effective when the view size does not match the display size and can only be set if scroll bars are disabled when viewing. With the "left, top" standard, the anchoring has no effect. The element has at least one horizontal (left, right) and one vertical (top, bottom) anchorage.
 - Anchoring "Left": Default behavior - the configured distance to the left edge is retained regardless of the display size.
 - Anchoring "Right": the configured distance to the right edge is retained regardless of the display size.
 - Anchoring "Left, Right": The distance is maintained on both sides. The width of the element is adjusted if necessary.
 - The anchors "Top" and "Bottom" behave analogously to "left" and "right" - only in the vertical.
- **Rotation:** The element can be rotated around a rotation point by a certain angle in degrees. The rotation point can be set as follows:
 - Top left: The upper left corner is the rotation point
 - Center: The center of the element is the rotation point
 - Custom: The entered X, Y point relative to the left, upper element corner is the rotation point

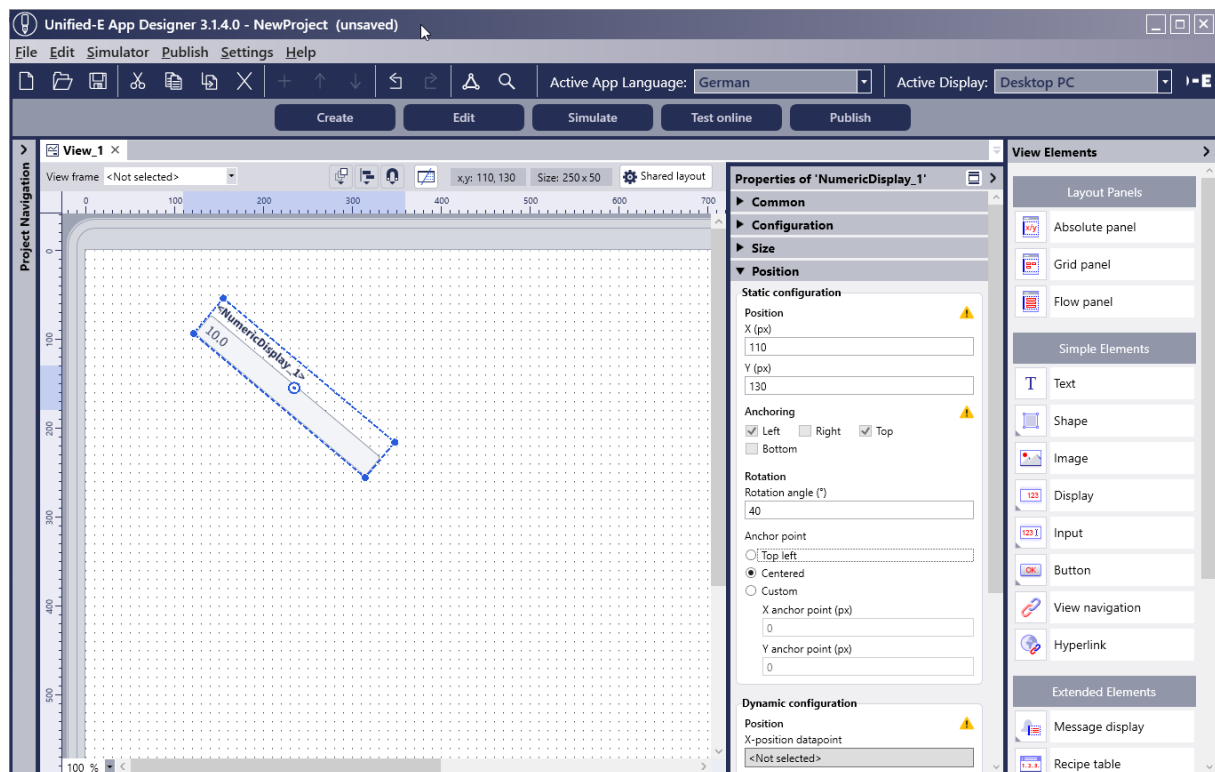
Positioning an element or changing its size can also be done graphically with the mouse.
Limitation: The size can only be adjusted graphically for non-rotated elements.

In general, the position properties in the "Dynamic Configuration" group can be changed dynamically at runtime with the help of datapoints. In this way, for example, an animation can be implemented.

Element level (z-index):

The level of the element can be changed in the context menu or in the editor toolbar using the functions in the "Arrange" menu (e.g. "Bring to Front").

Example:



In the example above, the element is rotated 40 degrees in the middle. The X,Y position in the properties always refers to the upper left corner of the element in the non-rotated state

3.2.3 Grid Layout

The grid layout defines a responsive layout in which the size of flexible areas adapts to the size of the device – e.g. for display on tablets.

The view elements are arranged in tiles within fixed rows and columns. An element can occupy either exactly one tile or several consecutive columns or rows.

Define the width of a column (or height of a row):

When setting the width of a column (or the height of a row), the following types are available:

- Automatic: The width is determined by the widest element it contains.
- Fixed pixel length: The width (or height) is manually entered in pixels.
- Fill: The column (or row) fills the remaining available range.
If multiple columns or rows are of type Fill, a ratio value is used to determine the division among them.

Scrollbars in the grid layout:

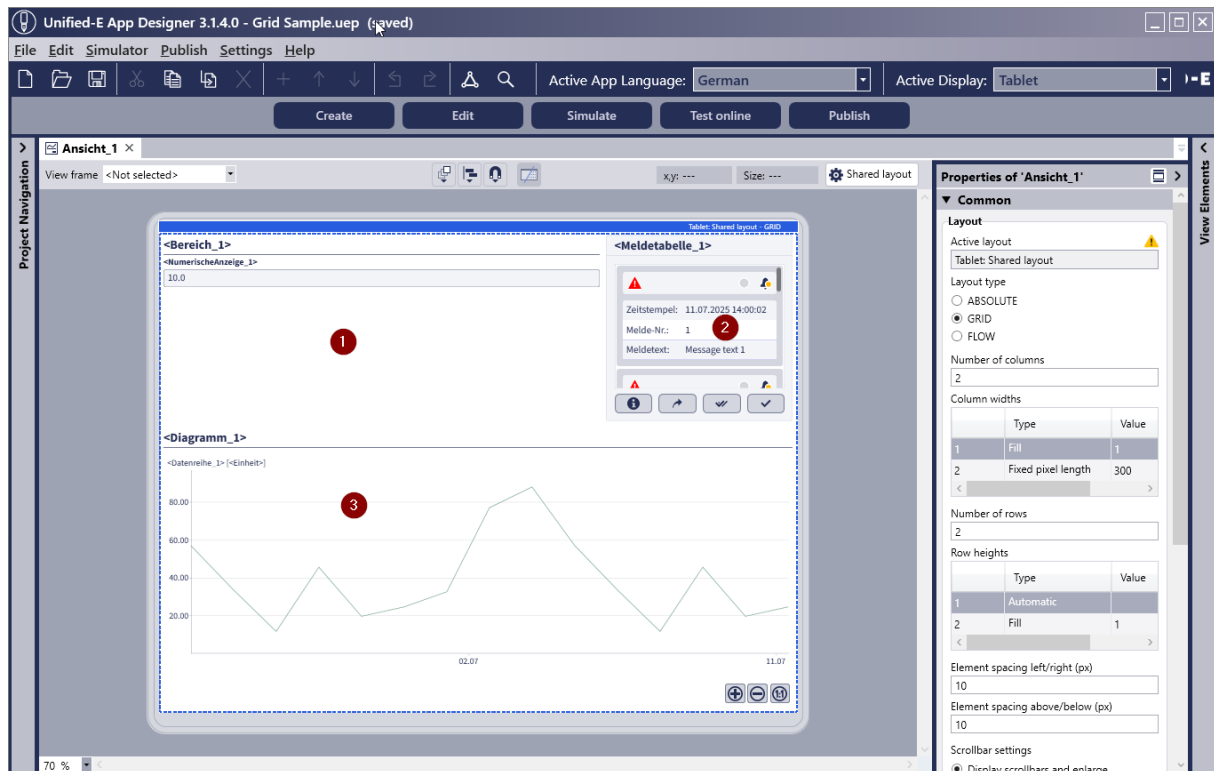
Basically, only the vertical scrollbar is supported.

Configure the grid layout for the view:

After selecting the layout type "Grid" in the view properties under "Layout" group, the following properties must be set to define the grid for the tiles:

- Number of columns:
Determines how many columns there are in the grid.
- Width per column:
In a table, the width can be defined individually for each column – as described above.
- Number of rows:
Specifies how many rows the grid contains.
- Height per row:
Here, too, the height can be defined individually for each row.
- Element Spacing Left/Right:
Determines the horizontal element spacing between two elements.
- Element Spacing Top/Bottom:
Defines the vertical element spacing between two elements.

Example:



The following example sets up a grid with two columns and two rows:

- Column 1 is set to "Fill"
- Column 2 has a fixed width of 300 pixels
- Line 1 is set to "Automatic", the largest height of the elements in line 1 is used
- Line 2 is set to "Fill"

Depending on the display size, the following changes:

- the width of Panel_1 (1 in the picture)
- the height and width of the diagram (2 in the picture)

Position element inside a tile:

The instantiation and placement of an element in a tile is done by dragging and dropping onto an empty tile labeled "Add element here".

The size, alignment and spacing of an element within its tile can be configured in the element properties under the "Size" palette.

The following properties are available:

- Span within grid:
The number of columns (number of tiles horizontal) and the number of rows (number of tiles vertical) are to be set. The default is 1.

- **Width:**
You can choose to use the full tile width for the element or set an individual width in pixels.
- **Horizontal Element Alignment:**
Determines the horizontal placement of the element within the tile when the width of the element is less than the width of the tile (left-aligned, centered, right-aligned).
- **Height:**
Analogous to width: The full tile height can be used or an individual height can be defined.
- **Vertical Element Alignment:**
Determines the vertical placement within the tile when the element height is less than the tile height.
- **Margin (px):**
Specifies the distances to the edge of the tile – top, bottom, left, and right.

3.2.4 Flow layout

The flow layout defines a responsive layout in which the element width adapts to the available device width. The view elements are stacked in tiles and arranged in a column (or flowing). This layout is particularly suitable for display on smartphones.

Configure flow layout for the view:

The flowing layout can be set to "Flow" in the view properties under the "Layout" group by setting the layout type to "Flow".

Position element in the flowing layout:

Elements can be moved up or down in the stack by drag and drop and thus repositioned.

Define element size and spacing:

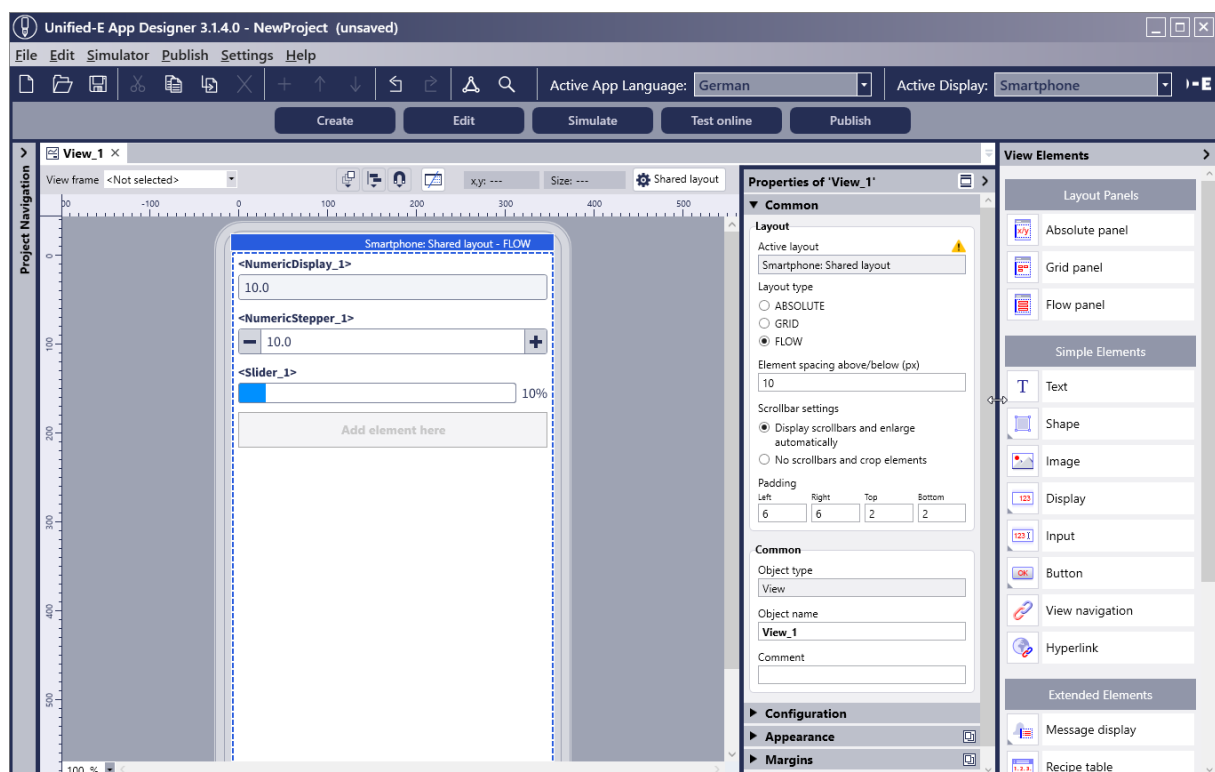
The flowing layout can technically be recreated with a grid layout that uses only one column and multiple rows with automatic height.

Similar to the grid within a tile, the element size can be set using the Size palette:

- **Width:**
You can choose to use the full display width for the element or set an individual width in pixels.
- **Horizontal Element Orientation:**
Determines the horizontal placement of the element within the display width when the width of the element is less than the display width (left-aligned, centered, right-aligned).

- **Height:**
The automatically determined height can be used based on the element content or an individual height can be defined.
- **Margin (px):**
Specifies the distances to the edge of the tile – top, bottom, left, and right.

Example:



The example above shows how the flow layout can be used to quickly create clear views for smartphones. The elements automatically take over the full display width of the operator device.

3.3 Display-specific Layout of Elements

In the previous chapters, it was assumed that the elements of a view are uniformly laid out and that this layout applies to all display sizes.

However, if an extended responsive layout behavior is desired – i.e. a layout that differs depending on the display size – so-called individual layouts must be defined for the corresponding display objects.

The following applies: For each individual layout within a view, the layout type must be defined again, and positioning and sizing of all elements must be carried out completely. A view thus has its own layout object for each display, for which an individual layout has been

defined. This layout object manages the position, size, and visibility of all elements, regardless of the shared layout.

This means that one and the same view element must be explicitly placed both in the shared layout and in each individual layout in which it is to be visible. Changes in the shared layout do not affect existing individual layouts.

The following element properties can be set display-specific for an individual layout:

- Element Position
- Element Size
- Visibility – certain elements or areas may be hidden in smaller displays

3.3.1 Shared vs. Individual Layout

Shared layout:

Each view has a shared layout. In most cases, the views only have this layout, so elements are only laid out once in the shared layout. The shared layout can be assigned to all displays at any time.

Individual layout:

If several displays are defined in the HMI project (see Chapter 15.2), it may be useful to create an individual layout for a specific display for certain views. As described in the next chapter, such a layout can be created and managed specifically per display.

Use case:

An HMI dashboard is supposed to display the areas in several columns (in a grid layout) on the tablet, and one below the other in a flowing layout on the smartphone.

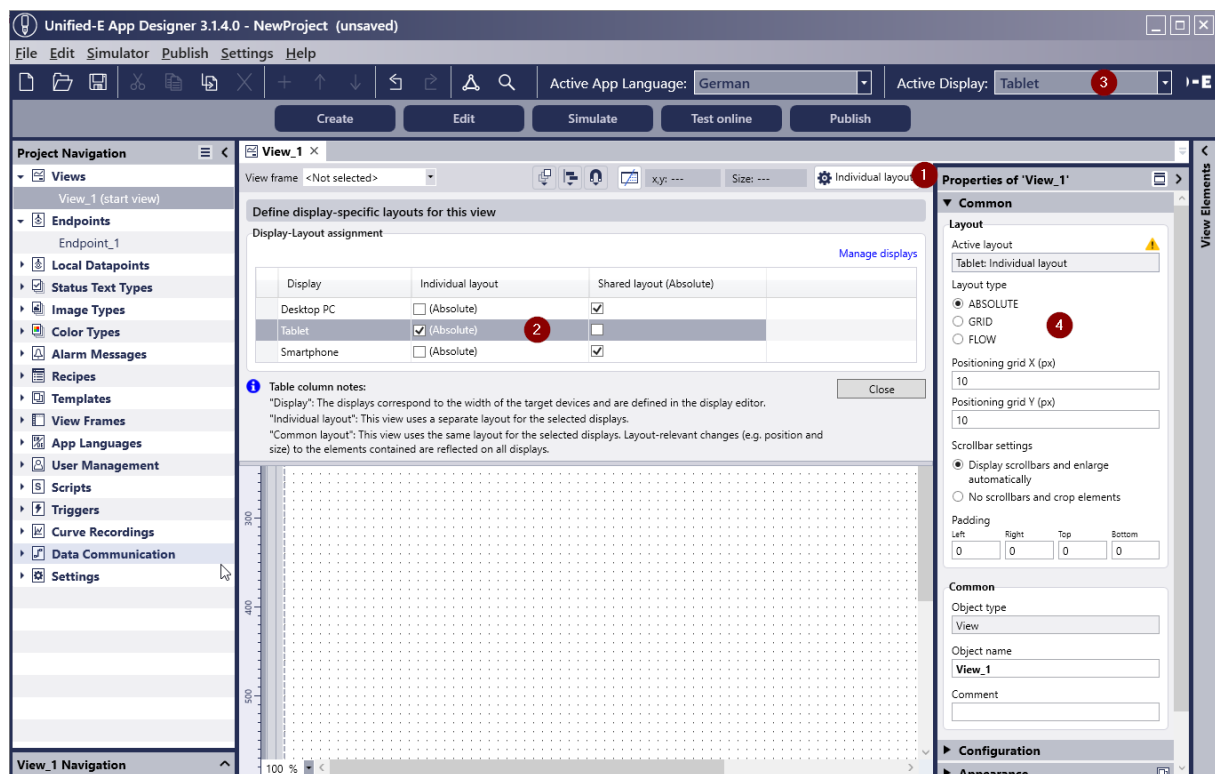
Solution:

- Two display objects are needed, each for tablet and smartphone.
- The dashboard view is configured to use an individual layout for smartphone – the tablet is linked to the shared layout. The details are described in the next chapter.
- At runtime on the control panel, either the shared layout or the individual layout for the smartphone is applied, depending on the configured width of the display object.

3.3.2 Create an Individual Layout

Assumption: There have already been at least two displays as described in the chapter 15.2.

In the following, the figure is used to describe the steps with which to edit the individual layout.



Steps to create the individual layout (numbering according to the figure):

1. Open the "Set display layouts of the view" window:
Click on the "Individual layout" or "Shared layout" button to open the corresponding window. In the table included, it can be seen for each display whether an individual layout or the shared layout is used.
2. Activate "Individual layout":
For the desired display, select the checkbox in the "Individual layout" column. Once this is set, all view elements for this display must be re-laid out. The layout type of the view for this layout can also be adjusted in the view properties. Once activated, the window can be closed by clicking the "Close" button.
3. Toggle active display:
In the application toolbar, under "Active display", select the display for which you have just activated an individual layout.
4. Set layout type for view and re-layout element:
The layout type of the view must now be set again for this display – this only affects the individual layout. The elements for the active display can then be positioned and sized for this specific display. The shared layout remains unchanged.

Show or hide an element in the layout:

An element can be shown or hidden in the element navigation by clicking on the symbol "Use in active layout" in the first column (see also chapter 1.7.3).

Toggle Active Display:

Switching the active display via the application toolbar has the following effects on the view editor (or on all open view editors):

- The view is displayed in the selected display.
- The view is displayed in the assigned layout of the display (individually or together).

If an individual layout is activated for the active display, then all changes to the layout properties take place in the individual layout of the active display.

Layout selection at runtime when using individual layouts:

The current display width of the operator device (or the window width of the client or simulator window) is relevant for the layout selection.

The following rule applies:

1. Display object mapping: Based on the actual width of the device (or window), the display object whose defined width is identical or slightly larger than the current width is selected.
2. Layout selection: The view uses the layout associated with the discovered display object—either the shared layout or an individual layout, if one has been defined.

3.4 Use Images and Fonts

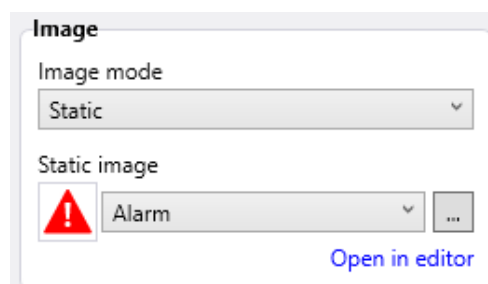
When designing a view, various elements and objects are also linked to images and fonts. These objects can be referred to as "resources" and must be included in the HMI project before they can be linked in the elements and objects.

Images and fonts are fully embedded in the HMI project file, the original files on the file system are not required for publishing or runtime!

Use images:

All images are managed in the "Images" editor, which can be opened via the Project Navigation under "Settings" → "Images". A new, blank project already contains some standard images. Both PNG and SVG files (vector graphics) can be managed centrally here. A detailed description of the Images editor can be found in chapter 15.3.

To link a static image to an element or object, select the desired image from the images stored in the Image editor in the Properties.



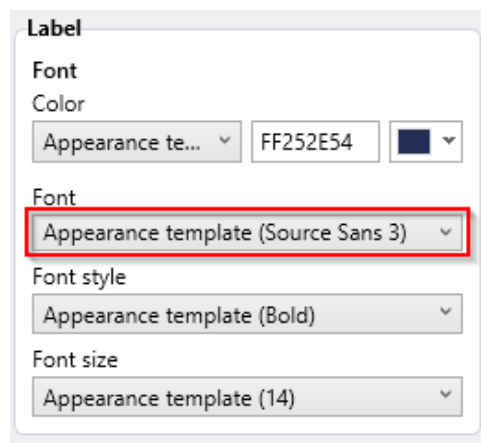
Use fonts:

All fonts are managed in the “Fonts” editor, which can be opened via the Project Navigation under "Settings" → "Fonts". This is where you add all the fonts that you want to use in view elements for display. Font files as well as fonts installed in the Windows system can be selected.

The Fonts editor is explained in detail in Chapter 15.4.

All fonts managed in the Fonts editor are embedded in the HMI project. Therefore, no fonts need to be installed separately on the operator device at runtime.

The default font, which can be overwritten, if necessary, is also defined in the Fonts editor.



3.5 Edit Theme with the Appearance Template

The appearance template ensures that view elements are already preconfigured according to your predefined design (corporate identity) when instantiating.

When you create a view element, all display properties are first linked to the appearance template. This means that even later changes to the appearance template automatically affect all properties that are still associated with the appearance template – they take over the current value from the appearance template.

The “Appearance Template” editor can be opened via the Project Navigation under "Settings" → "Appearance Template" and is described in Chapter 15.5 in detail.

The element properties of the following palettes can be preconfigured via the appearance template:

- “Appearance” palette
- “Alignment” palette
- “Margins” palette
- “Effects” palette

The "Edit appearance template" icon in the palette header allows you to quickly switch to the properties of the assigned appearance template.

As long as a property is linked to an appearance template, it cannot be edited in the properties pane – the input field is "read-only" and displays the value currently defined in the appearance template.

3.6 Create and Use Types

3.6.1 Types at a Glance

Types are used in view elements to determine the appearance - such as the border color - depending on the value of a datapoint.

Each type specification therefore also requires to configure a datapoint which is used for the comparison.

Clarification of the term "Type":

A type defines how an element is represented when a certain datapoint value is present (e.g. color, text or image).

There are three "type" categories:

- **Status text type:**
Determine which multilingual text is displayed at a given datapoint value or range of values.
- **Image type:**
Specify which image from the Image editor is displayed at a given datapoint value.
- **Color type:**
Determine which color is used for a given datapoint value or range of values.

The three type categories are described in detail below.

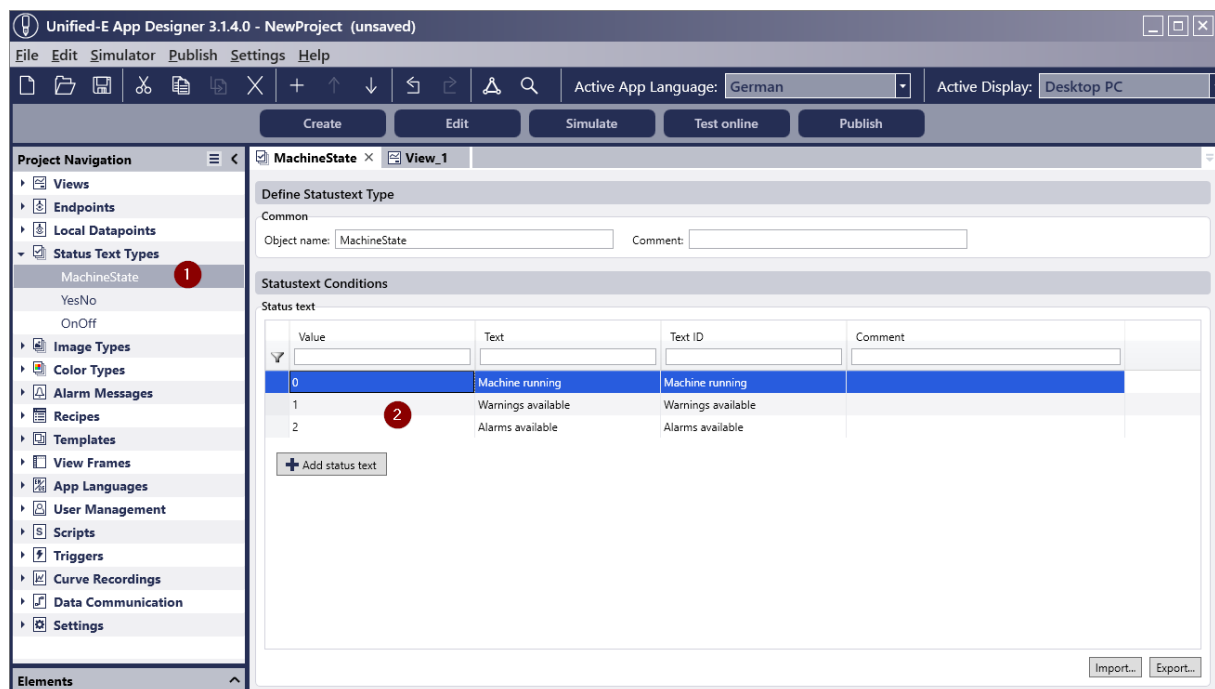
Create a new type:

In the Project Navigation, select the desired type main entry (e.g. "Color types"). In the context menu, a new type can then be created using the command "Add <Type>".

3.6.2 Status Text Types

3.6.2.1 Create a status text type

A status text type determines which text is displayed at a given datapoint value. The Status Text Type editor lists the corresponding texts for all values.



Configuration steps (numbering according to figure):

1. Select desired status text type:
In the Project Navigation, the Status Text Type editor is opened by double-clicking. A new status text type can be created in the context menu under "Status text types".
2. Add status text for value:
In the editor, a new entry for a specific value can be created using the "Add status text" button. The properties of the status text can be edited in the following columns.

Table columns:

- **Value:** Contains either an integer or a string. If this value matches the datapoint value, the associated text from the Text column is displayed.
- **Text:** Specifies the text to be displayed in the currently set app language (see Chapter 12).
- **Text ID:** Contains a language-independent identifier (Text ID) that is used in the Language editor for multilingual HMI apps.
- **Comment:** Optional free text for the internal description of the status text object.

Import/export status texts:

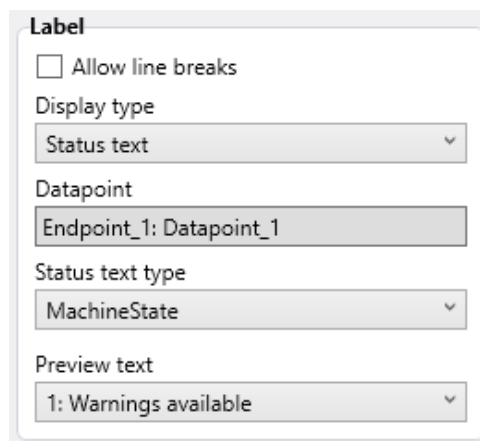
To edit a larger number of status texts, it can be useful to export the entries, revise them in Excel and then import them again. The corresponding buttons for export and import are located below the table.

The process for the export and import process is the same as for the datapoints (see Chapter 2.5).

3.6.2.2 Use status text type

The status text type is to be used for the desired view element as follows.

1. Open properties:
Example "Numeric Display": Open the "Configuration" palette in the properties and scroll to the "Label" group.
2. Set display type:
Set display type to "Status text".
3. Set datapoint:
In the "Datapoint" field, select the datapoint whose value is to be used for the comparison.
4. Set status text type:
Select the desired status text type in the "Status text type" field.
5. Preview text (optional)
The preview text for the editor can then be changed in the "Preview text" field.



The screenshot shows a configuration window titled "Label". It contains the following fields and settings:

- ☐ Allow line breaks
- Display type: Status text (selected in a dropdown menu)
- Datapoint: Endpoint_1: Datapoint_1 (selected in a text field)
- Status text type: MachineState (selected in a dropdown menu)
- Preview text: 1: Warnings available (selected in a dropdown menu)

If no status text entry can be found for the current datapoint value, an error text appears at runtime.

3.6.3 Image Types

3.6.3.1 Create image types

Image types can be created and opened via the Project Navigation. You can find them there under the entry "Image types".

In the Image Type editor, the associated images are listed for display for each defined value.

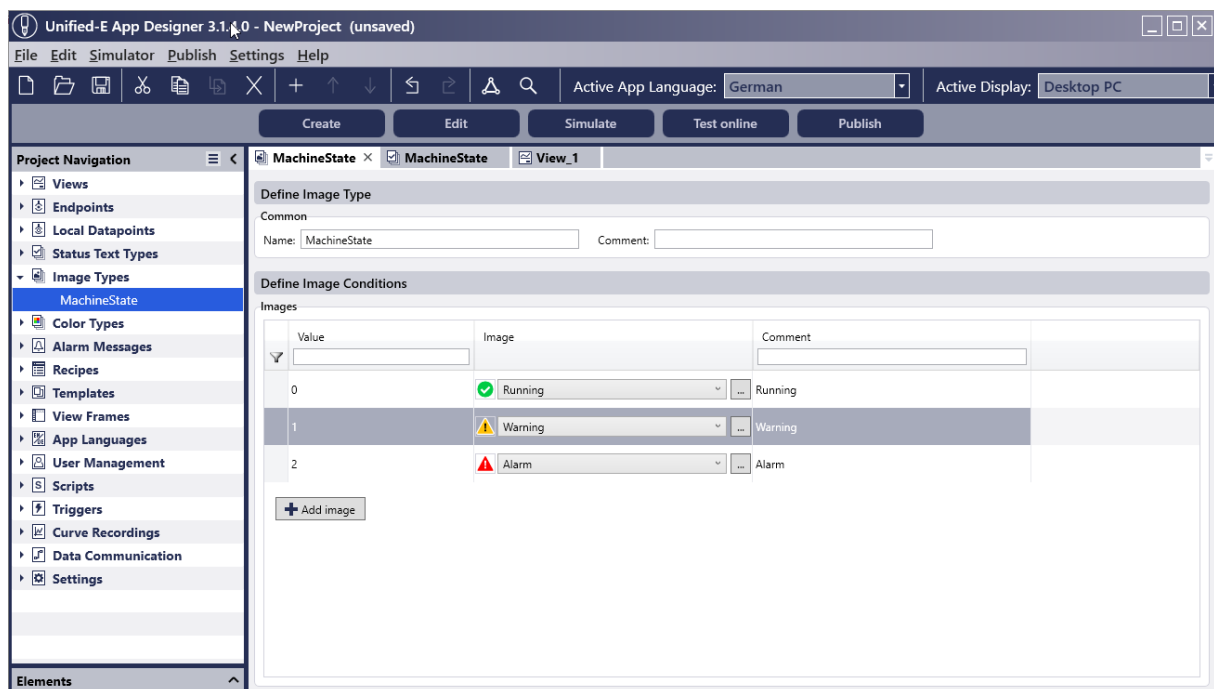


Image Type editor columns:

- **Value:** Contains either an integer or a string. If this value matches the datapoint value, the associated text from the Text column is displayed.
- **Image:** Defines the image as the value. All images of the Images editor are selectable. If necessary, new images can be added to the Image editor (see Chapter 15.3).
- **Comment:** Optional free text for the internal description of the image entry.

3.6.3.2 Use image type

An image type can be used in a view element as follows:

1. Open properties:
Example "Image": Open the "Configuration" palette in the properties and scroll to the "Label" group.
2. Set Picture Mode:
Set the Picture Mode to "Dynamic Image".
3. Select datapoint:
In the "Datapoint" field, select the datapoint whose value is to be used for the comparison.
4. Select image type:
Select the desired image type in the "Image type" field.
5. Preview image (optional):
The preview image for display in the editor can be adjusted in the "Preview image" field.

Image

Image mode

Dynamic

Datapoint

Endpoint_1: Datapoint_1

Image type

MachineState

Preview image

1: Warning

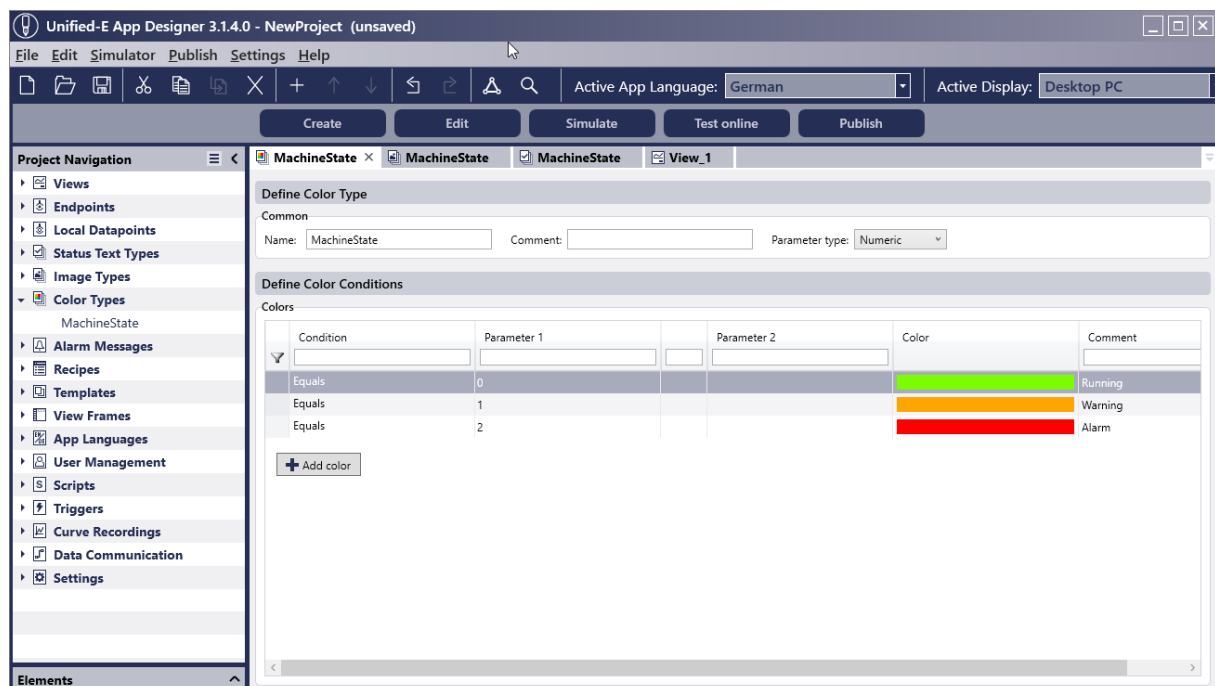
If no image entry can be found for the current datapoint value, then no image will be displayed at runtime.

3.6.4 Color Types

3.6.4.1 Create a color type

Color types can be created and opened via the Project Navigation. They are there under the entry "Color types".

The Color Type editor lists the corresponding colors for display for each defined value or range of values.



In the editor, conditions can be defined that determine which color is displayed at which datapoint value. The mapping is done by numerical or textual conditions, e.g.: If value between "Parameter 1" and "Parameter 2", then use the color "Red".

Therefore, the parameter type above the table must be explicitly set to "Numeric" or "Text".

Columns in the Color Type editor:

- **Condition, Parameter 1, Parameter 2:** Defines the condition that the datapoint value must meet. Depending on the selected condition, parameter 1 and possibly also parameter 2 are used (see Chapter 4.6 for conditions).
- **Color:** Contains the selected color.
- **Comment:** Optional free text for internal description of the color type.

3.6.4.2 Use color type

A color type can be used in a view element as follows:

1. **Open properties:Example "Shape - Rectangle":** Open the "View" palette in the properties and scroll to the "Shape" group → "Fill color".
2. **Activate dynamic fill color:**
Select "Dynamic" in the main selection.
3. **Select datapoint:**In the "Datapoint" field, select the datapoint whose value is to be used for the comparison.
4. **Select color type:**Select the desired color type in the "Color type" field.
5. **Preview color (optional):**
Select the color to display in the editor from the colors of the selected color type.

Shape

Fill color

Dynamic

Datapoint: Endpoint_1: Datapoint_1

Color type: MachineState

Preview: Equals 1

Contour color

Appearance template FF7B8299

Contour options

Appearance template

Stroke thickness (px)

2

Corner radius (px)

Top left Top right Bottom right Bottom left

0 0 0 0

If no color can be determined for the current datapoint value at runtime, "Transparent" is automatically applied.

3.7 Tools for Editing

3.7.1 Positioning Grid in Absolute Layout

In the absolute layout, the positioning grid can be defined in the view properties. The positioning grid helps you align elements in the absolute layout exactly. When moving, enlarging or inserting controls via drag and drop, they are automatically snapped to the nearest grid points. The distances of the grid in the X and Y directions can be individually adjusted.

Automatic alignment only takes place during manual actions such as moving, enlarging or dragging and dropping. Direct entry of position values is not affected.

Show or hide grids:

The grid can be shown or hidden via the context menu of the editor with the menu item "Show grid".

To turn Snapping on or off:

Snapping can be enabled or disabled from the editor's context menu under the Snap to Grid menu.

3.7.2 Group

Apart from layout panel elements that serve as containers for view elements (see Chapter 5.1), several elements within a container or view can be grouped together. Grouped elements can only be selected and moved as a unit in the graphical editor. When clicking on a group item, the entire group is automatically selected.

If the group size is changed, the elements it contains adjust proportionally. When the group rotates, all contained elements within the group are also rotated and repositioned accordingly.

After grouping, the properties of individual group members can only be accessed via the element navigation by selecting the respective element.

The group function is an engineering feature for more efficient design and has no effect on the runtime.

Create a group:

1. Select all desired elements.
2. Open the context menu of one of the selected elements and select the "Group" menu item.

Ungroup:

1. Select the group by clicking a group item in the editor.

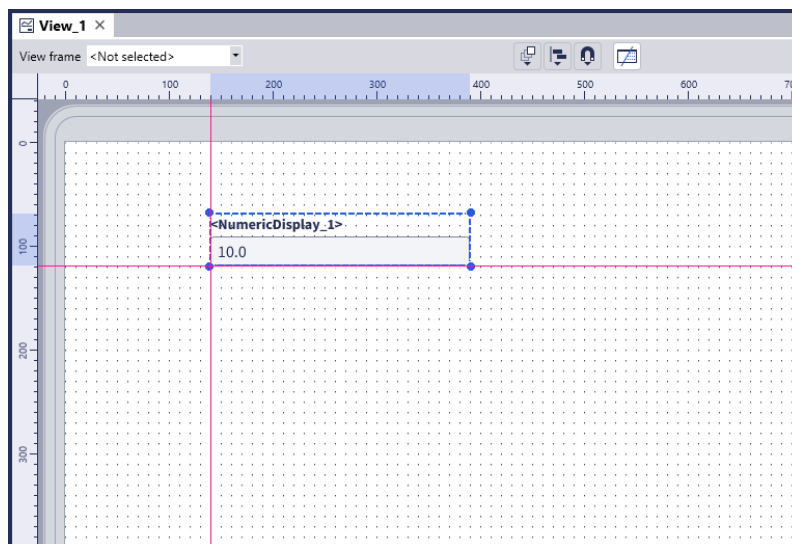
2. Open the context menu of one of the selected elements and select the menu item "Ungroup".

Rotate group:

A selected group can be rotated using the properties of the selected group ("Position" palette). All contained elements are rotated accordingly.

3.7.3 Guides

Guides (snap lines) are valid globally and are used in the absolute layout to ensure a consistent layout through uniform spacing in different views. They perform a similar function to the grid and support the precise placement of elements via drag and drop ("snapping").



Create a guide:

1. Drag and drop a new guide from the ruler that appears to the desired position.
2. Guides that have already been created can be moved at any time without changing the positions of existing elements.

Delete an guide:

Drag the guide back to the ruler area with the mouse to remove it.

Show or hide guides:

Use the context menu of the editor and select the menu item "Display guides" to make guides visible or hidden.

Turn Snapping on or off:

Automatic snapping can be turned on or off via the "Align to guides" menu item in the context menu of the editor.

Lock guides:

To prevent accidental moves, guides can be locked using the context menu. Locked guides are no longer selectable – clicks or drag-and-drop operations instead affect the view or the elements it contains.

3.7.4 Arrange Elements

Each element within a view has its own level, called the z-index. This determines the stacking order of the elements – i.e. which element is displayed in the foreground or background. The "Arrange" function, which is available in the context menu and via the arrow buttons in the editor toolbar, can be used to adjust the z-index in a targeted manner. For example, elements can be moved forward or backward to determine the desired display order.

Arrange functions via context menu (or via editor toolbar):

- Bring to front:
The element is placed on the top level, which is above all other elements.
- Bring forward:
The element is moved up one level – it is then above the element directly above it.
- Send backward:
The element is moved down one level – it is then below the element directly below it.
- Send to back:
The element is placed on the lowest level, behind all other elements.

In addition, arranging is also possible in the element navigation.

3.7.5 Align Elements

Various alignment functions are available for precise alignment of elements within a view. These can be found in the context menu of an element as well as in the editor toolbar.

Align:

- Align left:
All selected elements are aligned to the left edge of the leftmost element.
- Center horizontally:
The elements are aligned along the horizontal center axis.
- Align right:
The right edge of the selected elements aligns uniformly.

- **Align top:**
The top edges of the selected elements are aligned.
- **Center vertically:**
The elements are aligned along the vertical center axis.
- **Align bottom:**
The bottom edges are positioned consistently.

Mirroring (flipping):

- **Flip Horizontal:**
Mirrors the selected element along the vertical axis (left ↔ right). This is only possible within a group or for elements that have a corresponding toggle property. Color gradients are also mirrored accordingly.
- **Flip Vertically:**
Mirrors the element along the horizontal axis (top ↔ bottom) analogous to horizontal tilt.

These features make it easy to align and mirror elements consistently, supporting a clean and consistent layout.

3.7.6 Edit Display Text in the Graphical editor

Display texts can be set directly in the graphical editor without having to open the properties of the element.



There are two ways to get into edit mode:

- Double-click on the display text of the selected element
- Select element, then press ENTER key

3.7.7 Lock an Element in the editor

When working with multiple overlapping elements, it can be useful to lock certain background elements, such as a plant image or a configured Shape element. This prevents these elements from being accidentally moved or edited while adjusting elements above them.

Lock and Unlock:

The lock function is available in the element navigation via the lock symbol. There, elements can be locked or unlocked again.

Lock behavior:

A locked element can no longer be selected in the graphical editor. It cannot be moved or edited in the properties. To make changes, the item must first be unlocked.

3.7.8 Hide Element in editor

In addition to the lock function, elements can also be completely hidden in the graphical editor. This is especially useful when overlaying objects make it difficult to edit underlying elements. Temporarily hiding makes the desired object accessible in the background without having to change the layout.

Show and hide:

The visibility of an element can be controlled in the element navigation via the eye symbol. There, elements can be shown or shown again in a targeted manner.

Hide behavior:

A hidden item is no longer visible in the editor and cannot be selected or edited. The display at runtime on the operator device is not affected by this – hiding it only affects editing in the editor.

3.7.9 Search for Objects

3.7.9.1 Show cross references

The "Show Cross References" function can be used to display all objects that are linked to the currently selected element in the editor. This is especially useful for quickly understanding dependencies and usages—such as which datapoints or images are being used by a control.

Open the cross-reference list:

You can use the context menu of the desired element to call up the menu item "Show cross references". The cross-reference list then appears at the bottom of the editor.

Contents of the list:

The list shows the following information for each linked object:

- Cross-reference type:
You can use "used by" or "used".

- **Object:**
Name of the object to which there is a cross-reference.
- **Object Type:**
The object type describes the object in more detail, e.g. datapoint, Action Button.
- **Path:**
Describes the path to the object.
- **Open Link:**
A button to open the linked object directly in the associated editor.

Automatically follow the selection of the active editor:

The Synchronize icon in the upper right corner above the cross-reference table allows the automatic synchronization of the table content with the selected object of the active editor.

3.7.9.2 Searching for text in the object

The "Search Text in Object" dialog is started via the context menu (or main menu of the application) with the menu item "Search Text in Object". This allows objects to be searched textually, e.g. by object name or display text.

Object name	Object type	Path	Display text	Comment	
Endpoint_1: Datapoint_1	Datapoint	Endpoint_1			Open
Endpoint_1: Datapoint_2	Datapoint	Endpoint_1			Open
Endpoint_1: Datapoint_3	Datapoint	Endpoint_1			Open
LocalDatapoints_1	Endpoint				Open

The search criteria are described in more detail below.

"Search for" input field:

Enter the search term for which you want to search in the project (e.g. an object name, a partial term or a comment text).

Search Options:

- **Search object name:**
All object names are checked for the search term.

- **Search active language:**
Search for text only in the currently selected language (e.g. display text in English if the active app language is set to English).
- **Search all languages:**
All available app languages in the project will be included in the search. This option is useful for multilingual projects.
- **Search Comment:**
The search also includes the comments that are stored in the object properties.

The "Search" button starts the search based on the specified criteria. The results will then be displayed in the list below.

3.8 Reusability with Templates

Templates are used to define a group of view elements in such a way that they can be reused in different contexts. A Template describes the structure and behavior of such a group of elements and can be instantiated as often as desired. The resulting instances are called "Template Elements" in Unified-E.

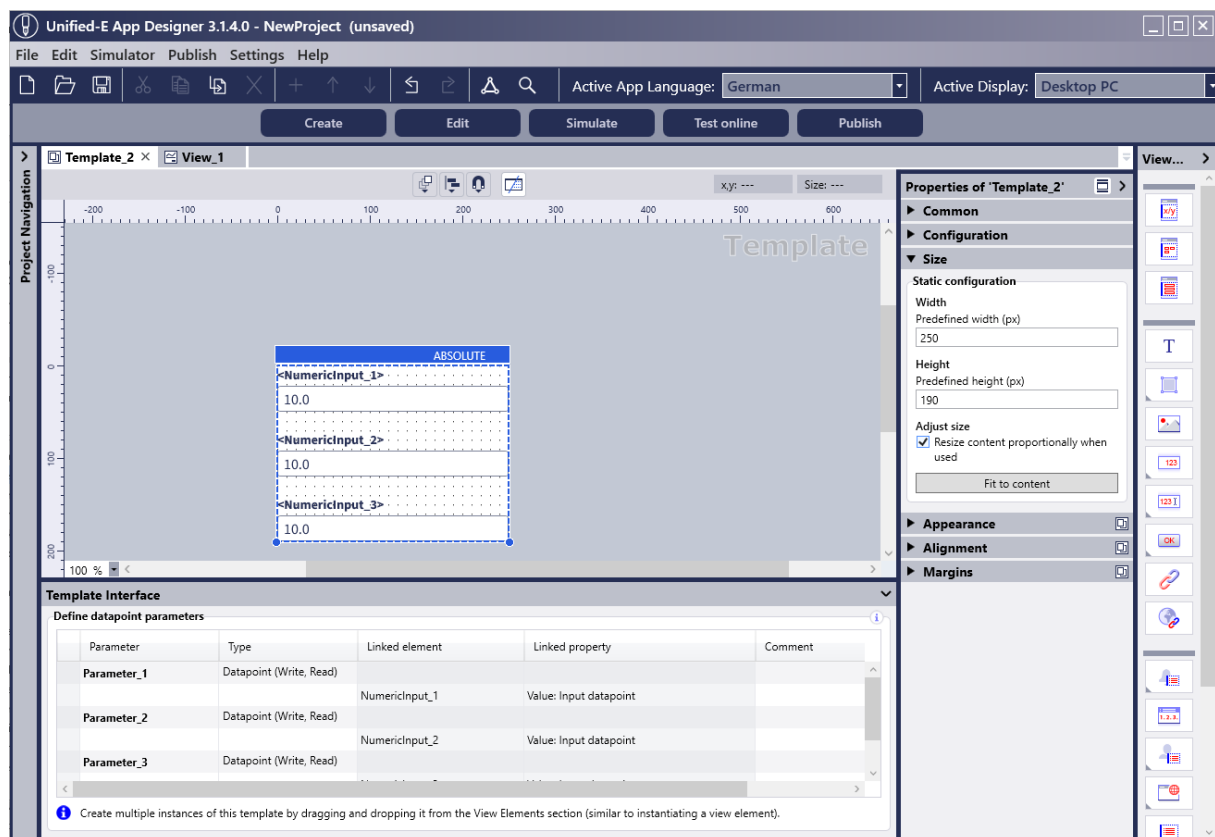
Templates can also be designed to be parameterized. When creating an instance, specific values can be passed via defined template parameters in order to specifically adapt the behavior or appearance of the instance.

The following parameter types are supported for template parameters:

- **Datapoint:**
A datapoint of a view element can be linked to a parameter that is only set at the Template Element (i.e. at the instance).
- **View:**
The view to be opened and the selection mode of the view element "View Navigation" can be defined via parameters.
- **Text:**
The multilingual display text of a view item can be connected to a template parameter that is configured on the instance.

3.8.1 Create a Template

Templates are configured in a graphical editor, similar to views. The template container with its view elements is compatible with the layout panels (see Chapter 5.1) and therefore adopts almost all of their properties. General properties such as "name" or "background color" are not described again at this point.



Create a Template and open it in the editor:

In the Project Navigation, the Templates are located under the "Templates" entry. There, a new Template can be created via the context menu with the menu item "Add Template".

Double-clicking on the desired Template object opens the Template editor.

Determine the layout for the Template:

As with a view, the layout type can be defined in the properties under the "General" palette (see Chapter 3.2.1). After selecting the layout type, view elements can be added via drag and drop.

Note: The layout of the Template is not display-specific configurable – there is exactly one layout for all elements – regardless of the active display.

Set a predefined size:

For the absolute layout, a predefined width and height of the Template can be specified in the "Size" palette. This defines the initial size of the Template Element when it is inserted into an absolute layout.

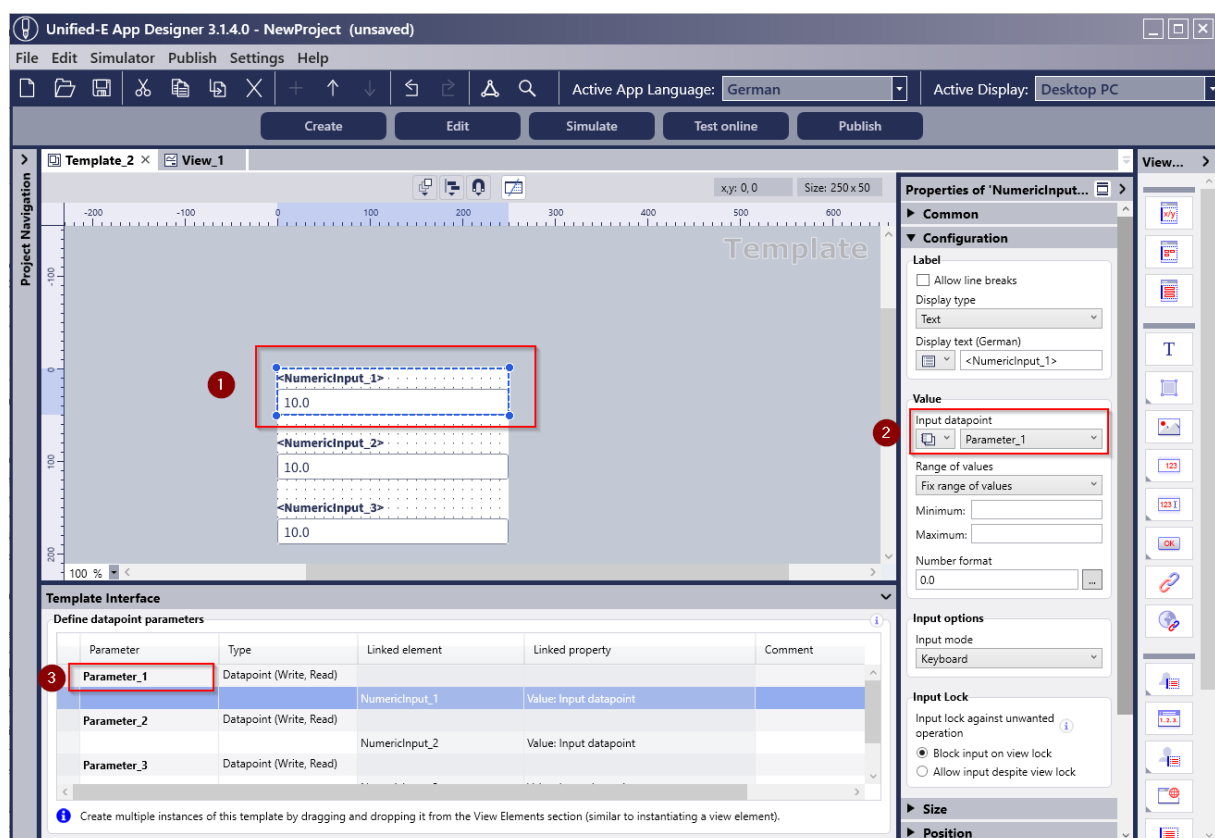
In addition, the option "Resize content proportionally when used" can be activated. If this option is set, the dimensions and positions of all contained elements are automatically adjusted as they are inserted, proportional to the difference between the predefined size of the Template and the actual size of the Template Element.

Automatically adjust predefined size:

After placing the desired view elements, the predefined size can be automatically set to the area actually required. To do this, use the "Fit to content" button in the "Size" palette.

3.8.2 Link View Elements to Template Parameters

Properties of view elements can not only be connected directly to datapoints or view links, but also to template parameters in the case of a Template. If a property is linked to a template parameter, it can be individually configured for the Template Element (i.e. for the instance of the Template).



Properties of view elements can be linked to a Template parameter in just a few steps. This makes the respective property of the Template Element (instance) individually configurable.

Steps (numbering according to the figure):

1. Select view element:
Click on the desired view element to open its properties.
2. Link datapoint property to parameter:
Navigate to the desired datapoint property. Click on the "Template parameters" icon on the left to activate the parameter mode. A drop-down list appears on the right-hand

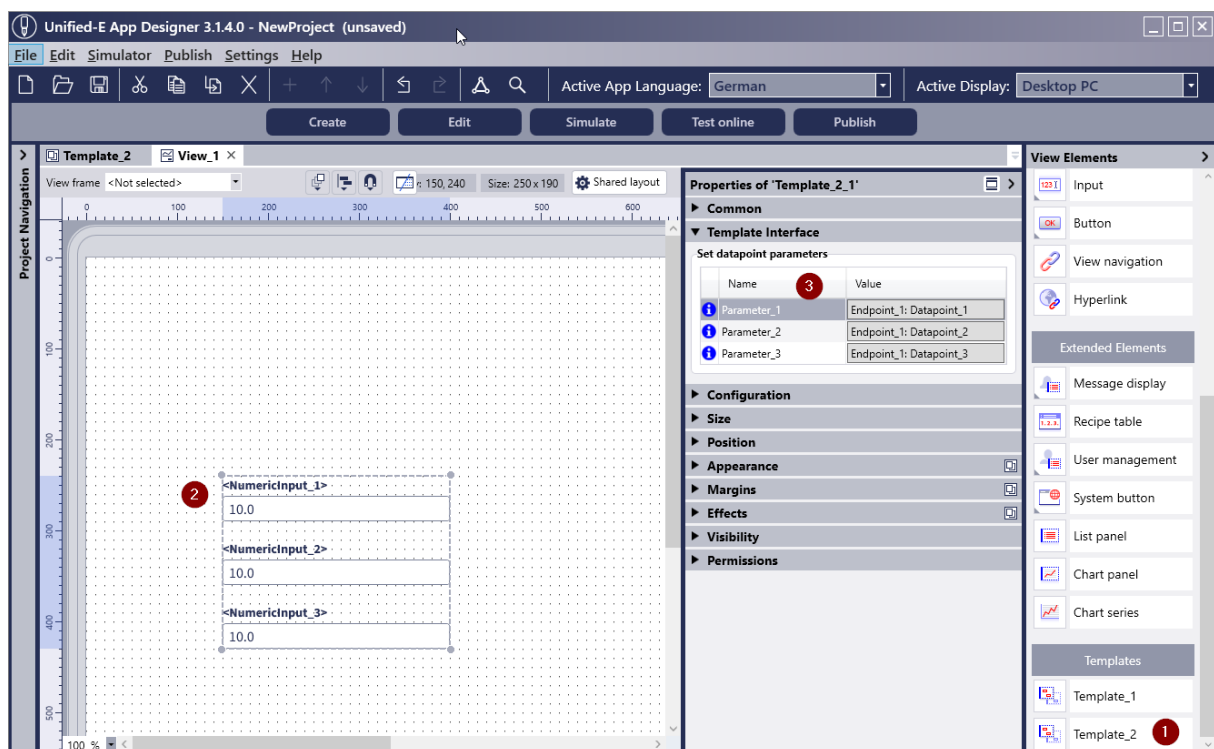
side, in which you can either select "Create new parameter" or assign an existing parameter from the parameter table in the "Template interface" area.

3. Edit parameter name:

In the parameter table, the name of the parameter can be edited directly. For example, a descriptive identifier can be assigned for later use in the Template Element.

3.8.3 Instantiate Template

A Template is instantiated in the same way as a view element via the "View Elements" area. An instance of a Template is also called a "Template Element".



Procedure for instantiating (numbering as in the figure):

1. Navigate to the "Templates" section:
Scroll down in the "Views Elements" section until the "Templates" section is visible.
2. Instantiate Template:
Drag the desired Template onto the Views background like a view element.
3. Set Template parameters:
Open the "Template interface" palette in the properties. There you can set the parameters defined in the Template – for example, by selecting the desired datapoints or setting a text.

3.9 Reusability with View Frame

3.9.1 View Frame Concept

The View Frame is used for the reusability of layout panels that should remain the same in several views – such as header, navigation area or footer. These areas are not defined in the view itself but are part of the View Frame that surrounds the view.

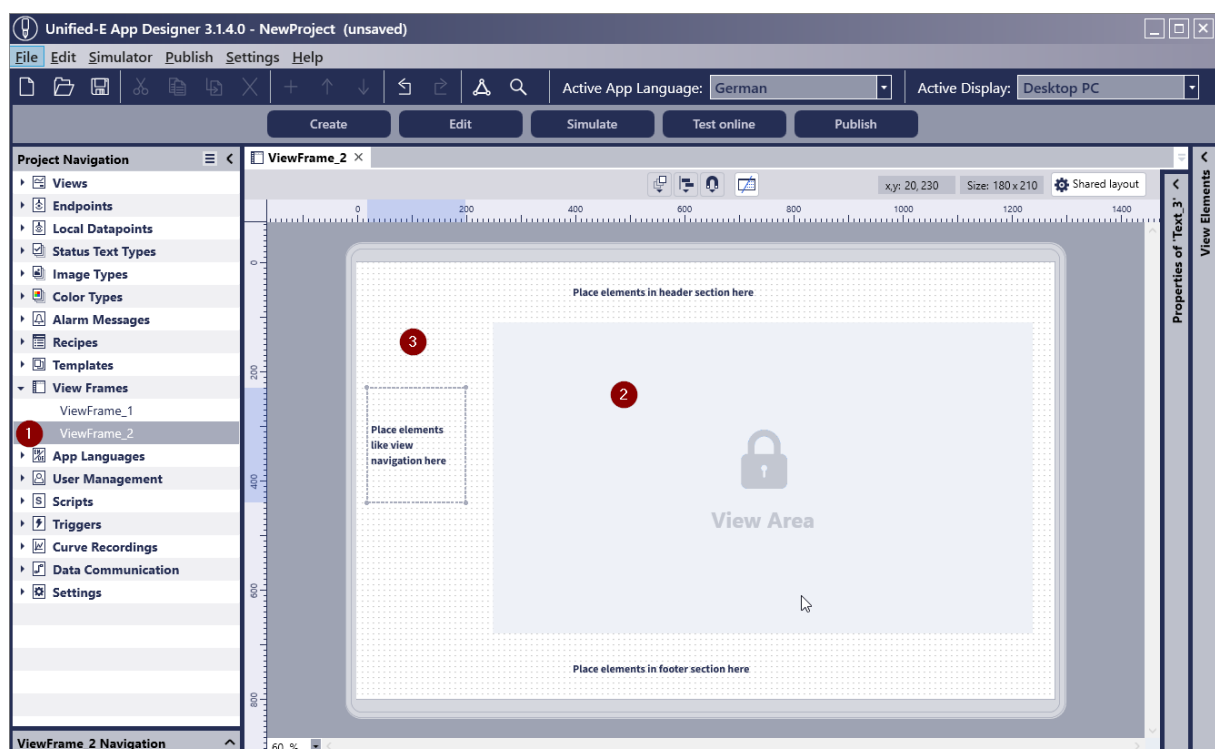
From a UI point of view, when a View Frame is assigned to a view, it becomes an outer container in which the view is embedded. Or to put it another way: a View Frame is "superimposed" on the view from the outside, which contains the shared layout elements. The actual view only fills the intended content area (also called the view area) within the frame.

When switching between views with identical View Frame, the frame remains unchanged – only the interior area, i.e. the concrete view, is replaced. This creates a consistent user interface with minimal maintenance.

The assignment of a View Frame to a view is done in the editor toolbar of the view in the "View Frame" area on the left side (see Chapter 3.1).

3.9.2 Create a View Frame

In order for a View Frame to be associated with a view, it must first be created as described below.

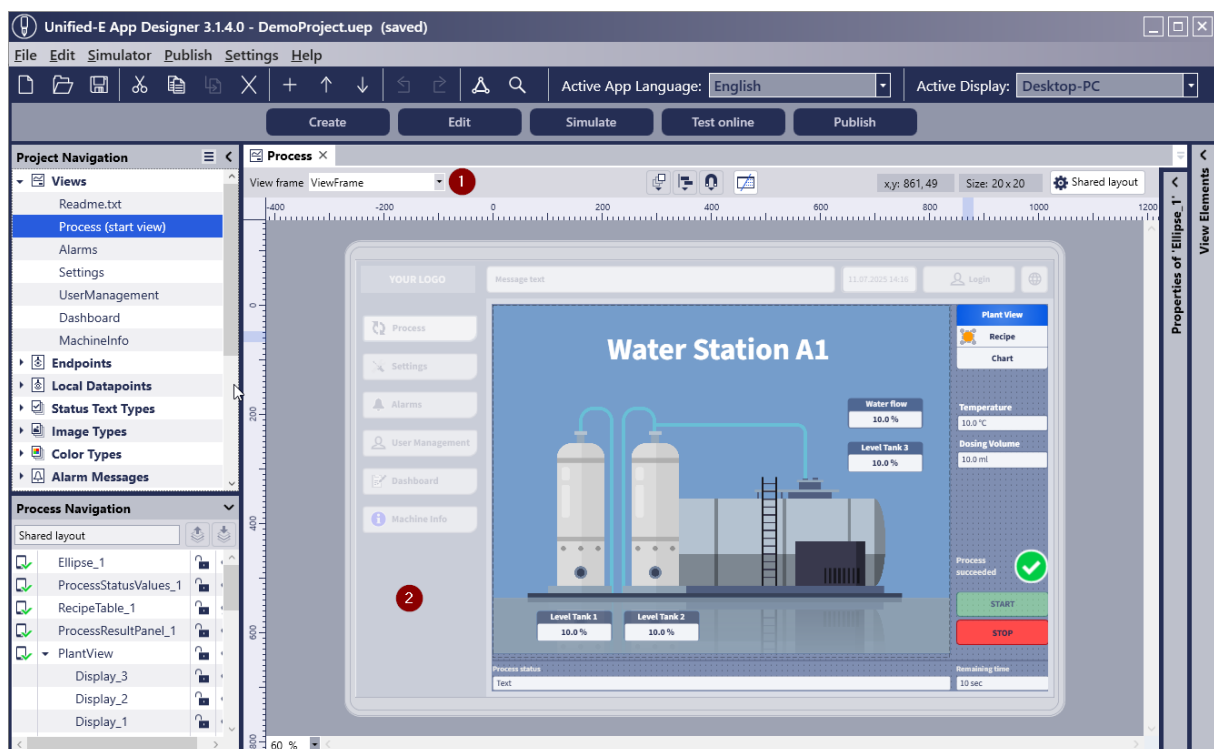


Procedure (numbering according to figure):

1. Open the View Frame editor:
Open the editor via the Project Navigation in the "View Frames" entry. You can use the context menu to create a new View Frame object with "Add View Frame". Double-clicking on the object opens the editor.
2. Define View Area:
The view area is the area in which the assigned view will later be displayed. Adjust the size and position so that there is room all around for the common frame areas. No elements can be instantiated on the view area itself – it is intended exclusively for the embedded view.
3. Define shared areas:
The areas outside the view area are available for reusable layout elements such as header, footer, or lateral navigation. View elements or Templates can be instantiated there. The layout of the common areas is display-specific – just like the views described in the chapter 3.3.

3.9.3 Link View Frame with View

The View Frame is mapped to a view in the "View" editor, as described below.



Procedure (numbering according to figure):

1. Select View Frame:
In the "View Frame" section, select the desired frame. The view is then embedded within this frame and displayed accordingly.

2. Open Associated View Frame:

The border areas outside the view are semi-transparent. Double-clicking on this area opens the assigned View Frame in the editor for editing.

The selected View Frame can be assigned to any number of views. Changes in the View Frame automatically affect all associated views – for example, a header or navigation area only needs to be created and maintained once.

4 General Design Properties

This chapter is about the available general properties of the elements in the View editor, View Frame editor, or Template editor.

These general properties are present in several view elements and are therefore summarized here centrally.

The properties of an element can be set in the Properties Pane and are grouped into property groups. Property groups, on the other hand, are organized into palettes. Often, a property group refers to a specific internal part of the element, e.g. only the label or inner input field.

Part "Body":

Many elements have the part "body". The body has properties for the element background, element border, padding to the inner parts and the orientation of the inner parts. The body therefore has a container function for the other parts of the element.

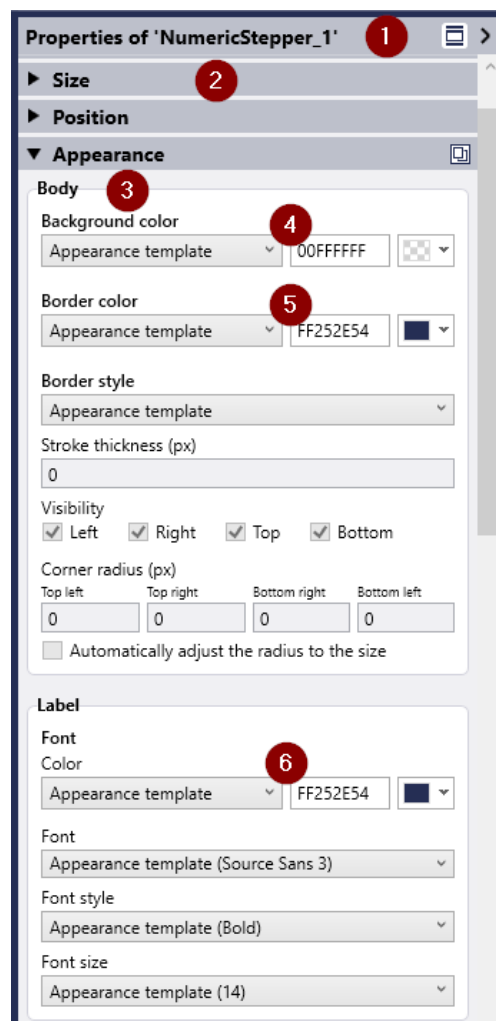
Part "Label":

The label (multilingual text) is represented as a part in many elements. Properties of the font such as font or font color can be configured here.

Part "Image":

Furthermore, many elements have an image as a part, such as buttons in general. The image itself, the image size or the distance to the text can be configured in the properties.

The following example illustrates the structure of the Properties Pane.



Structure of the properties and typical areas (numbering as in the figure):

1. Properties Pane:
The Properties pane includes all the properties of an element. The area can be opened or closed at the top left to get more space for other applications. If the option "Show only one pallet at a time" is set (symbol in the top right), then the previous one is always automatically closed when a pallet is opened.
2. Palette:
Palettes contain the properties of a topic group, such as Appearance.
3. Property group:
Property groups are subgroups within a palette and often refer to a specific part of the element (e.g. Body, Label, Image) but can also group general element properties.
4. Configure Color:
Colors are configured in the Views palette for various facets or parts of an element. The color configuration is described in the chapter 4.2 described in detail below.
5. Configure border:
Many elements or element parts have a border. The border configuration is described in the chapter 4.4 described in detail.

6. Configure font:
Elements or element parts with text displays have properties for configuring the font, which is described in the chapter 4.3 .

4.1 Available Property Palettes

The properties of the selected element or the view itself are organized in palettes.

4.1.1 "General" Palette

This palette always contains the "General" property group. In the case of containers, there is also the "Layout" property group, as in the chapter 3.2.1 can be found.

Group "General":

This group contains the following properties:

- **Object type:** Displays the element type – e.g. "Action Button", "View". For Template Element, the name of the underlying Template is specified here.
- **Object name:** Input field for the unique name within the container
- **Comment:** Optional free text for internal remarks

4.1.2 "Configuration" Palette

This palette contains the specific, functional properties of the Views element. It is often sufficient to edit only these properties if the layout and design have already been defined via the appearance template.

The focus of this palette is on the content and behavior of the element – not its detailed appearance like border color or text color.

The general configuration properties are described in this chapter. The specific properties of the individual view elements are described in the respective subchapter within chapters 5 documented.

Property group "Label":

The label of the element (multilingual text) is configured in this group.

- **Allow line breaks:** If activated, the display text can be entered in multiple lines. The text is also automatically wrapped if there is not enough horizontal space available
 - A line break can be done with SHIFT + RETURN
- **Display type "None"** The label is hidden
- **Display type "Text":** The label text is entered under "Display text"
 - The active app language is taken into account here if multiple app languages are configured in the HMI project

- Display type "Status text": The label text is taken from the assigned status text type – depending on the datapoint value
 - Datapoint: Describes the datapoint that determines the status text
 - Status text type: Select status text type (see Chapter 3.6.2)
 - Preview text: The text to be displayed in the editor
- Display type "Text with datapoint parameters": Here the label text is dynamically created based on one or more datapoints
 - Display text: This is where the (multilingual) text appears. Placeholders ({0}, {1}...) are used for the datapoints. Here, the values 0 and 1 are the datapoint indices in the datapoint table below. With the help of the "{0}" button on the right, placeholders can be easily inserted
 - Datapoint list: One datapoint must be defined for each placeholder (see figure). The formatting of the numeric or date-based datapoint value can be configured by clicking on "...".

Example: Text with datapoint parameters:

	Datapoint parameter
0	Endpoint_1: Datapoint_1 Format: 0.00
1	Endpoint_1: Datapoint_2 Format: <Format not set>

+ Add datapoint parameter

Property group: "Image":

In this group, the image content is configured if the view element contains an image.

- Image mode "None": The image is not displayed in the element
- Image mode "Static ": Select an image integrated into the HMI project
 - With "..." a new image from a file can be integrated into the project
 - Images can be managed in the Images editor (Chapter 15.3)
- Image mode "Dynamic": The image is dynamically displayed at runtime based on a datapoint value and the assigned image type
 - Datapoint: Datapoint that determines the image to be displayed

- Image type: Select image type (see Chapter 3.6.3)
- Preview image: Image that is displayed for preview in the editor

Property group "Input Lock":

As described in chapter 5.4.1 , it is possible to define per view and display whether user input is allowed or only needs to be released after clicking on a lock icon.

For the views elements, this behavior can be customized or overridden using the following options:

- **Block input on view lock:** Default behavior – the element inherits the input lock of the associated view
- **Allow input despite view lock:** The element always remains operable – regardless of the input lock of the view

4.1.3 "User Actions" Palette

For elements that react to mouse or touch interaction (click), e.g. Shape or an Action Button, properties can be found there that configure the action to be performed in the event of a mouse or touch user action.

Example: A datapoint value should be set when clicked.

There are different types of actions. The action to be applied must be set in the "Select action" field – the available properties will be adjusted accordingly. Note that selecting the action type is not always available – e. g. the view element "Checkbox" only supports "Toggle datapoint".

The screenshot shows a configuration window titled 'Actions'. At the top, there is a dropdown menu labeled 'Select action' with 'Set datapoint' selected. Below this, there are three trigger options, each with a checkbox and an information icon (i):

- ☒ Clicked action trigger: This option is selected. It has a 'Datapoint' dropdown menu currently showing '<Not selected>' and an empty 'Value' text input field below it.
- ☐ Pressed action trigger: This option is not selected. It has a 'Datapoint' dropdown menu showing '<Not selected>' and an empty 'Value' text input field below it.
- ☐ Released action trigger: This option is not selected. It has a 'Datapoint' dropdown menu showing '<Not selected>' and an empty 'Value' text input field below it.

Below the trigger options, there is a checkbox for 'Activate alive trigger' which is also not selected. At the bottom of the dialog, there is a section titled 'Action confirmation' with a checkbox 'Execute action only after confirmation in the dialog' which is not selected.

Action: Set Datapoint:

The following properties are available for datapoint setting:

- **Datapoint:** The datapoint to be set must be selected
- **Value :** The value to be set at the datapoint

The "Set datapoint" can be triggered by three events. A distinction is made between three mouse/touch events:

- **Clicked action trigger:** Triggered when both a press and release event has been registered on the item
- **Pressed action trigger:** Triggered when a press event is registered from the mouse or via touch input
 - **Activate alive trigger:** If set, then a alive trigger datapoint can be periodically set to a value. The trigger period can also be configured. This allows the endpoint to cancel an activated action when pressed if the alive trigger datapoint has not been set within the time period (machine safety)
- **Released action trigger:** Triggered when a release event has been registered

"Toggle datapoint" action:

Here, when clicking, the datapoint with the possible values "0" or "1" is alternately switched.

- **Toggle datapoint (0, 1):** The datapoint to be toggled is to be selected.

"Navigate to view" action:

Clicking opens the selected view.

- **Open view:** The view to be opened must be selected.

"Navigate to website" action:

When clicked, the specified website opens in the default browser.

- **Static configuration / Dynamic configuration with datapoints:** Defines whether the URL is fixed or can be read from a datapoint
 - **Open web address:** Enter the fixed URL or the datapoint in the case of dynamic configuration

Action Confirmation:

With the action confirmation you can set that the triggered action is only executed after confirmation in the dialog:

- **Execute action only after confirmation in the dialog:** Dialog appears after clicking when set.

4.1.4 "Size" Palette

In this palette, the element size can be configured depending on the layout type of the container. The properties for configuring the size are described in the chapter 3.2 described in detail.

4.1.5 "Position" Palette

In this palette, the element position can be configured depending on the layout type of the container. These properties for configuring the position are described in the chapter 3.2 described in detail.

4.1.6 "Appearance" Palette

In the "Appearance" palette, the following properties must be set for the element and for the parts (depending on the selected element):

- Colors, e.g. background color, additional color – see Chapter 4.2
- Border, see Chapter 4.4
- Font, see Chapter 4.3

The display properties are preset in the appearance template for each view element.

Appearance for table and table buttons:

In the case of tabular view elements (e.g. Message Table, Recipe Table), the display properties of the table parts are configured in the additional tabs "Table" and "Button". A table consists of the following parts:

- Table body: Table header or column header
- Table content: Cells within the table
- Table background: Background color of the table
- Row: Rows of the table
- Row separator: Divider between two rows in the table

4.1.7 "Alignment" Palette

Here, the alignment or arrangement of a part relative to the element body or another element part is configured.

▼ Alignment

Label

Alignment
Appearance template

Horizontal alignment
Center

Vertical alignment
Center

Image

Image alignment
Appearance template

Horizontal alignment
Center

Vertical alignment
Center

Image position to label
Appearance template

Image position
Left of the text

Possible features:

- Custom / Appearance template: Sets whether the property is linked to the appearance template or whether the value is set individually
- Horizontal Alignment: Describes the horizontal alignment in the body or available area of the part
 - Possible values: Left, Center, Right

- **Vertical Alignment:** Describes the vertical alignment in the body or available area of the part
 - Possible values: Top, Center, Bottom
- **Image Position (optional):** This property is available if the item contains both an image and a label. This describes where the image should be placed
 - Left: Displays the image on the left side of the control, independent of the text alignment
 - Right: Displays the image on the right side of the control, independent of the text alignment
 - Left of Text: Places the image directly to the left of the text
 - Right of Text: Places the image directly to the right of the text
 - Top: Displays the image at the top of the control, independent of the text
 - Bottom: Displays the image at the bottom of the control, independent of the text
 - Above Text: Places the image directly above the text
 - Below Text: Places the image directly below the text

The alignment properties are preset in the appearance template per view element.

4.1.8 "Margins" Palette

In this palette, the distances between the parts of an element, their fixed sizes and the internal distances between the element border and the content are configured.

As an example, the properties of the "Body" and "Image" property groups are described here, as they occur in many view elements.

Property group "Body":

Configures the spacing between the edge of the element and its contents (for example, image or text on a button).

- **Custom / Appearance template:** Determines whether the properties are inherited from the appearance template or individually configured
- **Inner margin (px):** Distance to the edge in pixels
 - Values for "Left", "Right", "Top", "Bottom" can be defined individually

"Image" property group:

Defines the image size in pixels and the distance between the image and the text or content.

- **Custom / Appearance template:** Determines whether the properties are inherited from the appearance template or individually configured

- Width (px): Image width in pixels
- Height (px): Image height in pixels
- Distance (px): Distance between image and text (content) in pixels

▼ Margins

Body

Inner margin (px)

Appearance template

Left: 4 Right: 4 Top: 6 Bottom: 6

Image

Width & height

Appearance template

Width (px): 32

Height (px): 32

Distance (px): 5

4.1.9 “Effects” Palette

This group defines visual effects for the View element. These include transparency settings, an optional body shadow and visual cues for limited input options (only relevant for input elements).

Property group "Transparency":

- Custom / Appearance template: Determines whether the properties are inherited from the appearance template or individually configured
- Transparency: Specifies whether the element is fully or partially transparent
 - Selection via appearance template or individual percentage value (0 = not transparent, 100 = completely transparent)
- Transparency when disabled: Sets the transparency for a deactivated element (see Chapter 4.1.10 to control the activated/deactivated state)
 - For example, if the value is 50, the disabled item will be displayed with 50% transparency (default value)

Property group "Body Shadow":

- Custom / Appearance template: Determines whether the properties are inherited from the appearance template or individually configured
- Shadow: Activates a shadow effect around the body of the element

- The values for X offset, Y offset, and Blur (each in pixels) control the position and softness of the shadow
- **Color:** The color field can be used to select the shadow color

Body shadow

Shadow
Appearance template

X offset (px) Y offset (px) Blur (px)

0 0 0

Color

Property group "Input permission" (only for input elements):

- **Custom / Appearance template:** Determines whether the properties are inherited from the appearance template or individually configured
- **Display when input is not allowed:** Defines the visual appearance when input is locked
 - **Display deactivated:** The element is displayed semi-transparently – as described above in the "Transparency" property group
 - **Represent with hatch:** A semi-transparent hatch is placed over the element. The color of the hatching can be selected via the color field

4.1.10 Visibility" Palette

The properties of this palette can be used to dynamically show or hide the View element at runtime, and to enable or disable it. It is controlled via a linked datapoint. For example, elements can only be displayed or locked when the control is in certain states.

Property group "Visibility":

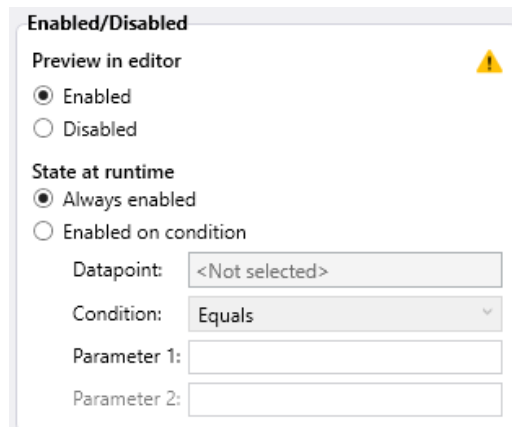
Specifies whether the item is always visible or only appears under certain conditions.

- **Always visible:** The item is always visible regardless of the state of the datapoint
- **Visible when condition is met:** The item is only displayed when a defined condition applies to a datapoint
 - **Datapoint:** The datapoint whose value is being checked.
 - **Condition:** Comparison operator (e.g. Equals, Not Equals, Greater Than, etc., see Chapter 4.6)
 - **Parameter 1 / 2:** Comparative values, depending on the selected condition

Property group "Enabled/disabled":

Controls whether the item is enabled or disabled at run time. Deactivated elements are optically dimmed (semi-transparent) – no input is possible for input elements. The transparency of a deactivated element is defined in the "Effects" palette, as this value can also be controlled via the appearance template (see Chapter 4.1.9).

- **Preview in editor:** Defines whether the element is displayed as enabled or disabled in the editor (has no effect on runtime)
- **Runtime State:** Sets the Enabled/Disabled
 - Always on: The element is always active and operable at runtime
 - Enabled when condition is met: The element is active only if a defined condition is true
 - Datapoint: The datapoint that is being checked
 - Condition: Comparison operator (see Chapter 4.6)
 - Parameter 1 / 2: Comparative values



4.1.11 "Permissions" Palette

This palette can be used to control which user roles are allowed to see or operate a view element. Additional permissions are available for common objects (such as messages). The restrictions take effect at runtime in the HMI app as soon as a user is logged in and has been assigned one or more roles.

If no login is performed or no user is active yet, all authorization checks are considered not met.

This means that an element that is only visible or operable for certain roles will not be displayed or displayed locked in this case.

Visibility:

When on, the item appears only to users who have at least one of the enabled roles. In the table below, it is possible to determine individually for each defined role whether the element is visible or not.

Input: (only relevant for input elements)

When this option is enabled, only users with a shared role are allowed to enter. Users without the appropriate permission can see the item, but cannot operate it.

▼ Permissions

User rights

Visibility
☒ Activate 'Visibility' only for selected user roles

User role	Visible	
UserRole_1	<input type="checkbox"/>	
UserRole_2	<input checked="" type="checkbox"/>	
UserRole_3	<input type="checkbox"/>	

Input
☐ Activate 'Input' only for selected user roles

User role	Input possible	
UserRole_1	<input type="checkbox"/>	
UserRole_2	<input type="checkbox"/>	
UserRole_3	<input type="checkbox"/>	

Editor state
☐ Display input restriction

Preview in the editor:

- **Show Input Restriction:** Simulates the visual representation in the editor when locked

Configure display on input restriction:

If an authorization is missing, an element can be hatched (as in the image below) or partially transparent. The presentation can be described in the chapter 4.1.9 and can also be set via the appearance template.



4.2 Configure Color

4.2.1 Configure Element Color

After instantiating a view element in a view, all colors are set to "Appearance template" by default. The color is automatically taken from the appearance template.

Color "Appearance template":

The element always adopts the current color from the assigned appearance template. If the color in the appearance template is changed, the appearance of the element updates automatically.

Color "Custom":

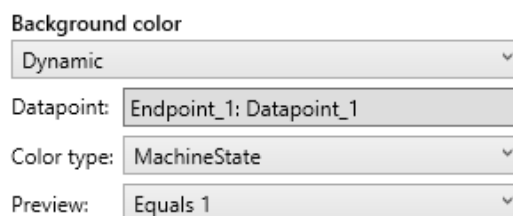
If the color should be detached from the appearance template and a custom color is to be used, an individual color can be defined using the color picker (see the following chapter). The setting is set to "Custom", which means that the element no longer reacts to color changes in the appearance template.



Color "Dynamic":

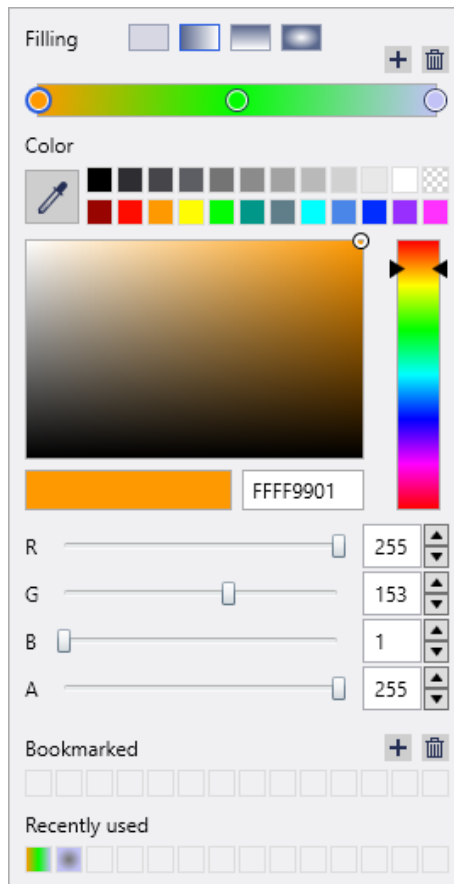
In this mode, the color is controlled at runtime by a datapoint value. Depending on the value of the datapoint, a color from the selected color type is displayed (see Chapter 3.6.4).

- **Datapoint:** Describes the datapoint that determines the color.
- **Color type:** Selection of the color type with multiple color value-dependent assignments
- **Preview:** Color that appears in the editor to provide guidance



4.2.2 Color Picker

The color picker is used to define the color for areas, borders, or texts. It supports both simple full colors and gradients with multiple color stops.



Fill Type - Full Color or Gradient:

In the upper area, the filling type can be selected:

- Far left: Full color (Solid)
 - A single color is used, with no gradient
 - This color can be defined directly from the underlying color range
- To the right: Gradient fills
 - The following gradient types are available: Horizontal gradient, Vertical gradient and Radial gradient
 - Multiple color stops can be defined for each gradient type (see below)

Edit gradient stops:

When selecting a gradient, a gradient with any number of color stops can be defined in the upper ribbon:

- The plus symbol (+) can be used to add a new color stop
- Each stop can be selected and customized by clicking on it
- The position of the stop in the gradient is adjusted by dragging it
- The trash can icon can be used to remove the currently active stop

Define color:

The color of a color stop (or full color) is set in the middle area:

- Selection via color swatches, eyedropper or by entering a HEX value
- In the large color field, the hue and brightness can be adjusted
- To the right of it is a vertical color slider for color selection
- The RGBA values (red, green, blue, alpha) can be entered numerically or adjusted using sliders
- Alpha (A): Controls transparency (0 = fully transparent, 255 = opaque)

Watch list and history:

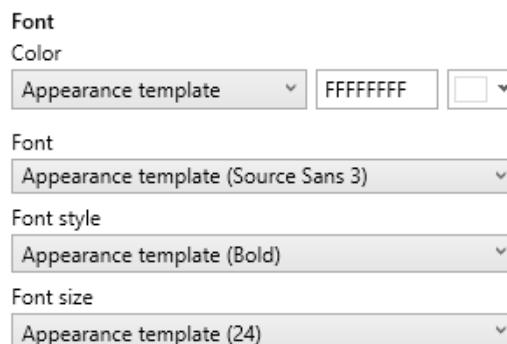
- Watchlist: Frequently used colors can be permanently saved via the plus symbol at the bottom
- Recent: Shows the most recently selected colors for quick reuse


Note: Only one full color can be used for the font color. Color gradients are only possible for borders or backgrounds.

4.3 Configure Font

For view elements with text display (e.g. buttons, text elements or labels), the font can be individually configured. The figure shows the available properties for the "Label" part.

All font properties are predefined in the assigned appearance template. To apply these values, simply select "Appearance template" in the drop-down boxes.



Font
Color
Appearance template ▼ FFFFFFFF 

Font
Appearance template (Source Sans 3) ▼

Font style
Appearance template (Bold) ▼

Font size
Appearance template (24) ▼

Properties:

- Color: Font color of the element
 - Selection as in the chapter 4.2 described
 - Only full colors possible (no color gradients)

- **Font:** Font family for the displayed text
 - Additional fonts can be created via the Fonts editor (Chapter 15.4 **Error! Reference source not found.**)
- **Font style:** Font style of the text
 - Possible values: Normal, Bold, Italic
- **Font size:** Font size in pixels (conversion to points: 12 pt = 16 px)

4.4 Configure Border

A border can be defined for all view elements. For certain parts of an element – such as an "input field" – a separate border can also be configured.

The border configuration includes the color, style, line width, visibility of individual pages, and corner radii.

The screenshot shows the 'Border' configuration panel. It includes a 'Border color' section with a dropdown menu set to 'Appearance template', a text input field containing the hex code 'FF7B8299', and a color swatch. Below this is the 'Border style' section with a dropdown menu also set to 'Appearance template'. The 'Stroke thickness (px)' section has a text input field with the value '2'. The 'Visibility' section contains four checkboxes, all of which are checked: 'Left', 'Right', 'Top', and 'Bottom'. The 'Corner radius (px)' section has four text input fields for 'Top left', 'Top right', 'Bottom right', and 'Bottom left', each containing the value '6'. At the bottom, there is an unchecked checkbox labeled 'Automatically adjust the radius to the size'.

Properties:

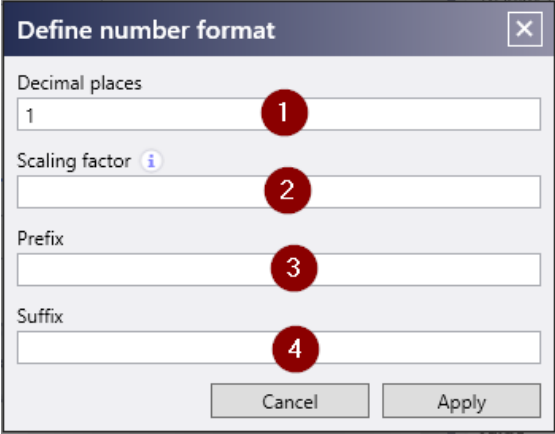
- **Frame color:** Color of Frame
 - Choice of color, see Chapter 4.2
- **Border style:** Adopts style from the appearance template or allows individual definition
 - This includes visibility, line width and corner radius
- **Line width (px):** Thickness of the border in logical pixels (DIPs)
 - Example: Value "2" results in a fine border
- **Visibility:** Determines which sides of the border are displayed
 - Options: "Left", "Right", "Top", "Bottom"
 - Multiple pages can be displayed in combination
- **Corner radius (px):** Rounding of the individual corners of the border
 - Values per corner: "Top left", "Top right", "Bottom right", "Bottom left"

- The values define the radius in pixels
- **Automatically adjust radius to the size:** When enabled, the corner radius is automatically adjusted to the size of the element

4.5 Configure Number Format

For some view elements, a number format must be defined for the representation or input of a numeric value.

4.5.1 Configuration in the Dialog



The dialog box titled "Define number format" contains four input fields, each marked with a red circle and a number from 1 to 4. The fields are: "Decimal places" (containing the value "1"), "Scaling factor" (with an information icon), "Prefix", and "Suffix". At the bottom are "Cancel" and "Apply" buttons.

The number format is configured in the dialog as follows (numbering according to the figure):

1. **Decimal places:** Specifies how many decimal places the value is displayed with
2. **Scaling factor:** Specifies the factor by which the value of the datapoint is multiplied before it is plotted. For more complex calculations, script datapoints can be used
3. **Prefix:** Text that appears before the numeric value
4. **Suffix:** Text displayed after the numeric value

4.5.2 Textual Configuration of Number Format

Alternatively, the number format can also be defined directly in the properties. The format must correspond to the following structure:

Format without scaling factor:

<Preceding text><number of decimal places> < ending>

Number of decimal places:

- "0" for no decimal place

- "0.0" at one decimal place
- "0.00" with two decimal places
- and so on.

Format with scale factor:

<Preceding text>{<Number of decimal places>:F<Scale factor>}< ending>

Number of decimal places: analogous as described above.

Examples of format at value = "2,473":

- Format "0.00": output is "2.47"
- Format "0.0 °C": Output is "2.5 °C"
- Format "{0.0:F10}°C": output is "24.7 °C"

4.6 Conditions for Datapoint Comparison

In various properties, the appearance of an element can be dynamically controlled via datapoint comparisons. Typical use cases are the visibility or the activation or deactivation or deactivation of an element depending on the current datapoint value.

It is controlled by conditions that compare either numeric or textual values.

Building a condition:

- Each condition consists of:
- a comparison type (e.g. "Equal" or "Between"), see below
- One or two comparative values parameter 1 and parameter 2 (depending on the condition)
- the datapoint value to be monitored

Available conditions (comparison types):

- **Equal:** The datapoint value must be exactly the same as the reference value
- **Not Equal:** The datapoint value must not be the same as the reference value
- **Greater:** The datapoint value must be greater than the reference value
- **Less:** The datapoint value must be less than the reference value
- **Greater or equal:** The datapoint value must be greater than or equal to the reference value
- **Less or equal:** The datapoint value must be less than or equal to the reference value
- **Between:** The datapoint value must be within a range of values (including the two thresholds)

- **Not between:** The datapoint value is outside the defined value range (excluding the two thresholds)

5 View Elements

View elements are graphical building blocks within a view that are used to display information or receive user input. Many elements are linked to datapoints and allow direct interaction with the controller or process.

The view elements can be inserted (instantiated) directly from the "Views Elements" gallery on the right side of the editor into a view or layout panel using drag-and-drop.

In the following subchapters, all available view elements are described in detail. This also includes special elements such as the view itself, templates, and Template Elements (instances of a template). These are listed in the subchapter 5.4 listed.

Structure of a view element:

Each view element consists of several parts, each of which can be configured. The following parts are represented in many view elements:

- **Body:** Outer border and background color. The body serves as a container and contains all other parts
- **Label:** Display text—for example, the text of a button
- **Image:** Image that is displayed - for example, an icon on a button.

General features:

Many view elements are assigned general, common properties, which are described in detail in Chapter 4. These properties are organized into palettes and include:

- **Common:** Object name, comment, layout type for containers
- **Configuration:** Display text and image
- **Size and Position:** Layout Properties
- **Appearance:** Borders, Colors, Font
- **Alignment:** Orientation of certain element parts
- **Spacing:** Sizes and spacing of certain element parts
- **Effects:** Shadows and Transparency
- **Visibility:** Visible/invisible and enabled/disabled configuration
- **Permissions:** User rights based on user roles

If properties refer to a specific part, the corresponding property group within a palette bears the name of that part, such as "Symbol", "Label" or "Image".

In the following sections, the general properties of the respective view elements are mentioned only if they have a special meaning for that element.

5.1 Layout Panel Elements

Layout panels are container elements that can contain other view elements. While groups (see Chapter 3.7.2) represent a loose grouping of selected view elements, layout panels are independent view elements with real container function.

Like a view, a layout panel can optionally display a scrollbar if the view elements contained in it are outside the currently visible area.

For each layout type (see also chapter 3.2.1), a specific layout panel is available for each of them:

- Absolute Panel: The elements contained are positioned absolutely
- Grid Panel: The elements are arranged in a grid of rows and columns (tiles)
- Flow Panel: Elements are displayed below each other in a column

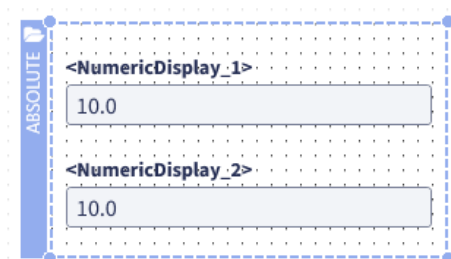
Parts of the layout panel:

Each layout panel has the following parts:

- Body: Background with border, contains the other parts
- Content/Work area: Area for placing and arranging the included view elements
- Display text (optional): Title in the header area of the layout panel.
- Input Lock (optional): Input elements in the layout panel can be configured so that their values are only written to the linked datapoints by clicking on the "Apply" button and not right after losing the focus (see Chapter 5.1.1)

Select elements in the layout panel:

Layout panels behave like other view elements. When a layout panel is selected with the mouse, its properties appear in the Properties pane. A drag-and-drop operation on the layout panel moves it within the view or its container.



The included child elements can only be selected after the layout panel has been opened for editing. The following options are available to select child elements:

- Double-click on the desired child element
- Click on the "Open" symbol (top left of the selection box of the layout panel, see figure), then select the child element with a simple click
- Selection via the element navigation (see Chapter 1.7.3)

5.1.1 Common Properties of the Layout Panels

In addition to the general properties (see Chapter 4) that apply to all view elements, a title and an "Apply" button can optionally be configured for all layout panels as follows.

Configure title in header:

The title of the header can be set under "Configuration: Display text". The header area can be separated from the content/work area by a horizontal line. This can be set in the "Appearance: Body" property section with the "Label border" option.

Property group "Configuration: Send input values":

Here you can configure a common "Apply" button for the layout panel. The values entered in the input elements are only transferred to the respective datapoints after clicking on "Apply", and an additional trigger datapoint can be set after transferring. If the "Apply" button is activated, the input elements do not necessarily show the current datapoint value.

This function is useful when multiple input values are to be applied consistently in one step (transactionally) or when you want to prevent inputs from being accidentally transmitted to the controller immediately.

- **Send input values via "Apply" button:** If activated, the values from the contained input elements are not transferred to the datapoints until the user clicks the "Apply" button. If this option is deactivated, every input will be applied as soon as you leave the input field
- **Send trigger (optional):** Select a datapoint to use as a trigger. If a trigger datapoint is specified, it will be described with the trigger value defined below when you click on "Apply". This can be used, for example, to trigger a write operation in a PLC
- **Trigger value (optional):** Specifies the value that will be written to the send trigger datapoint when "Apply" is clicked (if a trigger datapoint is specified)
- **Input field initialization when opening the view:** Determines how input fields behave when opening view:
 - **Empty after transferring the value:** Input fields are emptied after clicking on "Apply"
 - **Update once and after transferring the value:** The input fields are filled with the current datapoint value when first opened and updated again after transferring. However, there is no cyclic polling every second

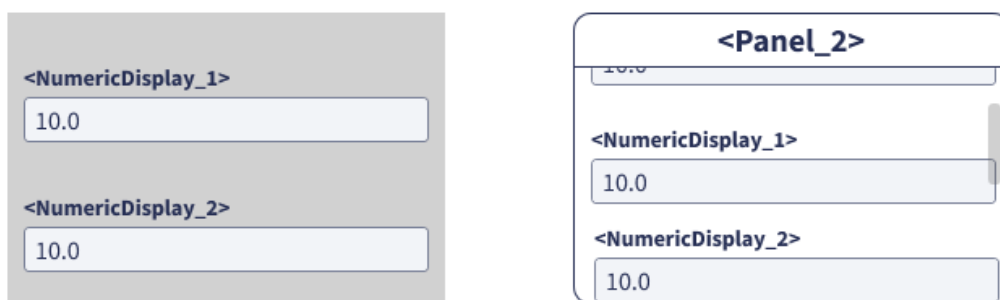
- **Always show current value:** If activated, the current value of the linked datapoint is always displayed in the input element. The display of this value can be configured in the respective input element as soon as the "Apply" button has been activated for the container

5.1.2 Absolute Panel

In the "Absolute Panel" layout panel, the included view elements with a fixed position (X/Y coordinates) and fixed size are placed. Positioning is done in the same way as the "Absolute" layout type for views (see Chapter 3.2.2).

The Absolute Panel is particularly suitable for scenarios where exact placement is required – e.g. the precise arrangement of display or control elements on a background image (e.g. plant plan, machine picture or process figure). It is also possible to overlay several view elements, which is not provided for grid or Flow Panels.

Examples:



The figure above shows two Absolute Panels at runtime.

Property group "Common: Layout":

This property group is used to configure the layout behavior for the included view elements.

- **Positioning grid X (px):** Horizontal distance of the grid (in pixels) that elements align with when placed with the mouse
- **Positioning grid Y (px):** Vertical distance of the grid (in pixels) that elements align with when placed with the mouse
- **Scrollbar settings:**
 - **Display scrollbars and enlarge automatically:** Scrollbars appear when an element leaves the visible area
 - **No scrollbars and crop elements:** Elements outside the visible area are not displayed (cut off)

5.1.3 Grid Panel

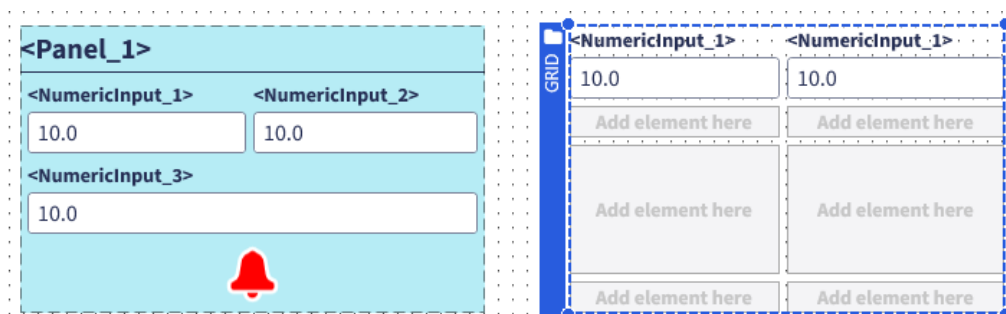
The Grid Panel is a layout container in which view elements are arranged in a clearly structured grid of rows and columns. The elements are placed in so-called tiles, which automatically align themselves according to the configured grid structure. The layout behavior corresponds to that of the view with the layout type "Grid" (see Chapter 3.2.3), but the Grid Panel can be used as an embedded element within another view.

Within the Grid Panel, columns and rows can be flexibly configured, for example by specifying fixed widths/heights or percentage ratios. Elements can be placed across several columns or rows (so-called "span").

If the elements contained exceed the available space vertically, a vertical scrollbar can be displayed automatically. A horizontal scroll bar, on the other hand, is not provided in the Grid Panel – this is always fixed in width.

The Grid Panel is particularly suitable for structured layouts where a uniform and tidy appearance is desired – e.g. for arranging several input or display elements in table form or in a classic tile structure.

Examples:



The image above shows two Grid Panels. New elements can be placed in the "Add element here" placeholders. If an element is to occupy more than one column or row, this can be set in the element properties under the "Size" palette.

Property group "Common: Layout":

In this property group, the grid layout of the container is configured – i.e. the number of rows and columns as well as their size behavior:

- **Number of columns / Number of rows:** Specifies how many columns or rows should be defined in the grid
- **Column widths / Row heights:** The size behavior can be configured individually for each column or row. The following types are available:
 - **Automatic:** The size automatically adjusts to the content of the respective column/row
 - **Fixed pixel length (px):** The size is set to a fixed pixel value

- Fill: The remaining space available is distributed proportionally. The value specifies how many parts of the remaining space this column/row should occupy (e.g. "1" = single, "2" = twice as much as one with "1")
- Element spacing left/right (px): Defines the horizontal padding between tiles in the grid
- Element spacing above/below (px): Defines the vertical padding between tiles in the grid
- Scrollbar settings: Determines how to handle overflowing content in the panel:
 - Display scrollbars and enlarge automatically: A vertical scrollbar appears when the content exceeds the height of the area
 - No scrollbars and crop elements: Elements outside the visible area are not displayed (cut off)

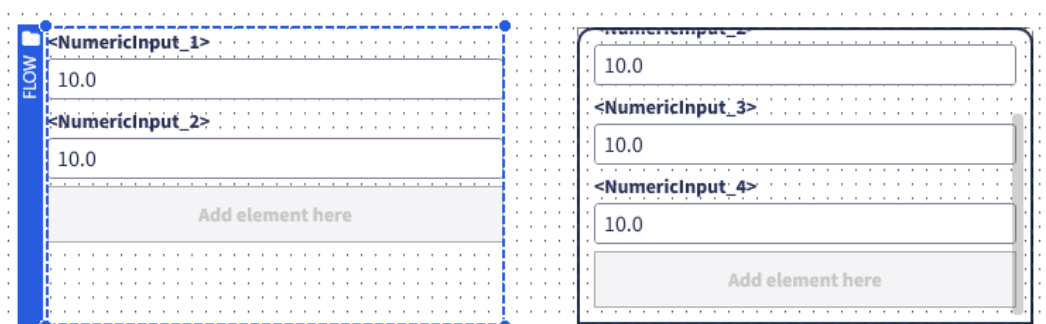
5.1.4 Flow Panel

The Flow Panel is a container element in which the contained view elements are automatically stacked below each other – comparable to a vertical layout flow. The elements are inserted directly below the preceding element, regardless of their size. A fixed grid structure as in the Grid Panel is not provided here.

This layout is particularly suitable for simple forms, configuration areas, or lists of input and display elements where a vertical arrangement is desired. The order results from the insertion order of the elements.

If there is a lack of space, a vertical scrollbar is automatically displayed – a horizontal scrollbar is not provided in this layout type.

Examples:



Property group "Common: Layout":

- Element spacing above/below (px): Specifies how many pixels of distance between each view element should be maintained vertically in the Flow Panel. This distance is applied above and below each element
- Scrollbar settings: Determines how to handle overflowing content in the panel:

- Display scrollbars and enlarge automatically: A vertical scrollbar appears when the content exceeds the height of the area
- No scrollbars and crop elements: Elements outside the visible area are not displayed (cut off)

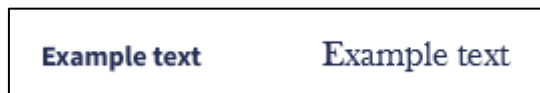
5.2 Simple Elements

Simple elements are basic view elements that typically represent individual functions or content in a visualization. They are lightweight, versatile and cover typical requirements such as Text Display, Image, Numeric Input or View Navigation. These elements can be placed in layout panels and individually designed.

5.2.1 Text

The "Text" view element can be used to display texts.

Examples:



Parts:

The following parts can be configured:

- Body
- Label

Configure Text:

The parts can be configured with the general properties – see Chapter 4 -configure. A user action cannot be configured on the text element.

5.2.2 Shape Elements

A Shape can be used to design simple geometric objects that can also react to click events. The "Shape" group includes four display elements for basic Shapes:

- Rectangle: Element for rectangles
- Ellipse: Element for circles or ellipses
- Polygon: Polygon with freely configurable vertices
- Line: Simple line, (without click action)

Examples:



Parts:

In contrast to other view elements, Shapes are not divided into several parts. All properties are directly related to the main Shape element.

5.2.2.1 Specific properties

In addition to the general properties (see Chapter 4), the following specific features are available:

Property group "Configuration: Shape":

This property group is available for Shapes with fill (not Lines) and includes the setting of the Paint mode:

- **Paint style:**
 - Filling and contour: The Shape is made up of fill and a border
 - Filling: The mold is only made of filling
 - Contour: Only the border is visible, user actions (click) are not configurable here

Additional properties of the Polygon:

- **Closed path:** Specifies whether the Polygon's outline is closed (the last point is connected to the first point) or open
- **Number of vertices:** Defines the number of vertices. When a value is entered manually, the points are automatically arranged symmetrically and can then be moved graphically

Property group "User Actions: Actions":

This defines which datapoints are set when the Shape is clicked (see Chapter 4.1.3).

Note: In the absolute layout, a click area could be defined above a large Image by placing a transparent Shape with a configured click action over the image.

5.2.2.2 Edit Polygon graphically

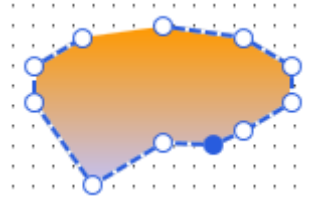
Start editing mode on Polygon:

Select the desired Polygon object with a mouse click. The edit mode can be activated by double-clicking on the object or by clicking on the pencil button above the selection box.

In edit mode, the individual Polygon points are visible, selectable and can be changed in their position.

Move a Polygon point:

A point can be moved by drag and drop. Alternatively, the selected point can also be positioned pixel-by-pixel with the arrow keys of the keyboard.



Add a new Polygon point graphically:

A new Polygon point can be inserted by clicking on a line between two points.

5.2.2.3 Edit Line graphically

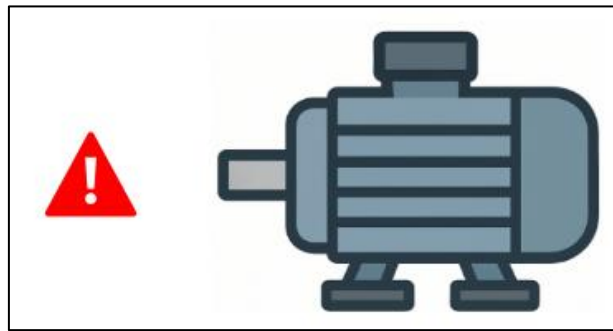
The startpoint and endpoint of a Line can be edited in edit mode in the same way as Polygon points. After starting the edit mode, the endpoints can be moved with the mouse or with the arrow keys.

5.2.3 Image

The "Image" view element can be used to display pixel-based images or SVG vector graphics. Typical use cases are:

- Showing a symbol: The Image part can be dynamically shown or hidden, or it can also be displayed via an image type (see Chapter 3.6.3) and a datapoint can be dynamically modified.
- Showing a fixed plant image: The Image element can be linked with a plant image from the Images editor (see Chapter 15.3). On the image, for example, buttons, input fields or click areas can be overlaid with the help of Shape elements.

Examples:



Parts:

The following parts can be configured:

- Body: Background color or transparent
- Image: Selected image from the Image editor (or dynamically configured)

5.2.3.1 Specific properties

In addition to the common properties (see Chapter 4), the following specific features are available:

Property group "Configuration: Image":

Here the image is to be configured - as in the chapter 4.1.2.

Property group "Configuration: Image alignment":

The following options are available for alignment:

- **Fill element with image:** The interior of the body is completely filled with the image – the image is stretched or compressed accordingly
 - The option "Use image aspect ratio" must be deactivated for this, see below
- **Image alignment:** The image retains its aspect ratio. If the aspect ratio of the image element differs from the original, the image inside the body will be aligned accordingly
 - Horizontal alignment: Left, Center or Right
 - Vertical alignment: Top, Center or Bottom

Property group "Configuration: Flip image content":

Here you can specify whether the image should be flipped for display:

- **Flip horizontally:** The image is tilted from left to right
- **Flip vertically:** The image is tilted from top to bottom

Property group "Size: Static configuration":

This is where the image size is set. In addition, you can force the aspect ratio of the image to be retained when enlarging with the mouse in the editor.

The following options are available:

- **Use original image size (%)**: The desired size as a percentage of the original size is specified. The size cannot be changed with the mouse with this option.
- **Individual element size**: width and height can be entered directly in pixels
 - **Use image aspect ratio**: When checked, changing the width or height automatically adjusts the other size to preserve the aspect ratio.
 - **Reset**: This button resets the size to the original image size.

5.2.3.2 Layout Image graphically

Like other elements, the Image view element can be placed graphically in the editor and resized.

If the Image is to be explicitly stretched, the "Use image aspect ratio" option must be deactivated in the "Size" palette.

If the element is enlarged with the mouse so that it protrudes into the negative area, the image (and also the gradient) is automatically flipped.

5.2.4 Display Elements

A Display Element can be used to display values of the current datapoint, e.g. the current temperature within a machine component. In addition, the display can also display a status indicator (color indicator) or an image (see "Parts" below).

The "Display" group includes the following specific display view elements:

- **Numeric Display**: Displays the numeric datapoint value in the field
- **Text Display**: Displays the datapoint value of type "Text" in the field
- **Date/Time Display**: Displays the current time or the time based on a datapoint of type "Text"
- **Bar Display**: Displays the numeric datapoint value (e.g. percentage value) as a bar

Parts:

A display element consists of several parts that can be shown or hidden in the Properties pane.



List of parts (numbering according to figure):

1. Body: Background (green in illustration)
2. Display text: Optional label to value – can be displayed at the top or right of the value
3. Value: Displayed datapoint value
4. Image: Optional image – displayed to the left or right of the value
5. Status display: Optional color indicator – always displayed to the right of the value

Special case "Bar Display":

In the Bar Display, the "value" part is replaced by the "bar" and "bar value" parts.

5.2.4.1 Common display properties

In addition to the general properties (see Chapter 4) that apply to all view elements, the following common properties are available for all display elements:

Property group "Configuration: Label":

The label can be configured as in the chapter 4.1.2 .

Property group "Configuration: Image":

The image can be configured as in the chapter 4.1.2.

Property group "Configuration: Status display":

The status display (color indicator) can be configured as follows:

- Without status display: The "Status display" part is not visible
- Static: The color selected in the color picker is displayed in the status display.
- Dynamic: The color for the status display is controlled at runtime via a datapoint value. Depending on the value of the datapoint, a color from the selected color type is displayed (see Chapter 3.6.4).
 - Datapoint: Describes the datapoint that determines the color.
 - Color type: Selection of the color type with multiple color value-dependent assignments
 - Preview: Color that appears in the editor to provide guidance

Property group "Alignment: Label":

The following display-specific properties can be found here:

- Orientation: Specifies whether the label should appear above the value (bar) or to the left of the value (bar)

- Label on top: Label appears above the value
- Label on left: Label appears to the left of the value
- Label width (%): Must be configured when selecting "Label on the left". Specifies the width in percentage to reserve the designation in the body

Property group "Alignment: Image":

The following display-specific properties can be found here:

- Image position:
 - Left: The image will be aligned to the left to the label/value
 - Right: The image will be aligned to the right to the label/value

5.2.4.2 Numeric Display

This view element displays numeric datapoint values.

Examples:



Property group "Configuration: Value":

Here the value to be displayed must be configured:

- Display options: The "Only use image/status display " option hides the "Value" part – no datapoint needs to be configured for the value display
- Datapoint: The datapoint whose value is to be represented in the "Value" part
- Number format: Here you have to configure the number format
 - Static configuration: The number format is set via the "..." button in the dialog (see Chapter 4.5)
 - Dynamic configuration: A datapoint of type "Text" must be selected to provide the number format (see Chapter for format description 4.5.2)

5.2.4.3 Text Display

This view element displays the text of a datapoint value.

Examples:



Property group "Configuration: Value":

Here the value to be displayed must be configured:

- **Text from datapoint:** This option takes the text from the datapoint value
 - **Datapoint:** Select the datapoint that provides the text
- **Status text type and datapoint:** This option determines the text based on the selected datapoint and status text type
 - **Datapoint:** Describes the datapoint that determines the status text
 - **Status text type:** Select status text type (see Chapter 3.6.2)
 - **Preview Text:** The text to be displayed in the editor

5.2.4.4 Date/Time Display

This view element can be used to display date/time values.

Examples:



Property group "Configuration: Value":

Here you must configure the date/time value to be displayed.

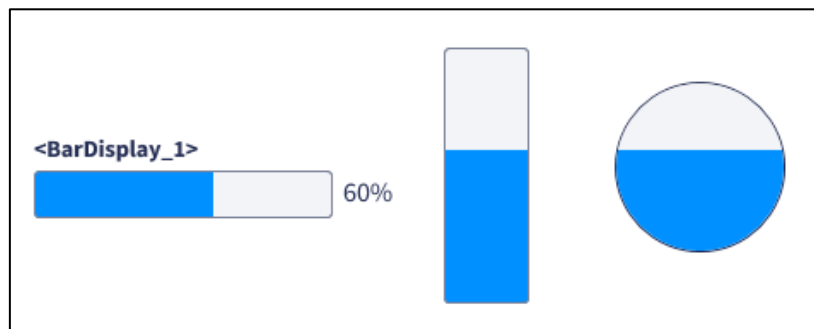
- **Display options:** Specifies whether to display the local date or a value from the selected datapoint
 - **Local date/time:** The current date or time is displayed
 - **Date/time from datapoint:** Select the datapoint and set the time zone of the date value
 - **Datapoint:** Datapoint that provides the date/time value. The date must be transmitted as a string in a standardized format such as "yyyy-MM-ddTHH:mm:ss" (ISO 8601 standard). Example value: 2025-06-25T14:30:00. Alternatively, the date can be delivered in the format "MM/dd/yyyy HH:mm:ss"
 - **Time zone of datapoint value:** Specifies how the datapoint value is interpreted. If the "Universal time zone (UTC)" option is selected, an automatic conversion to the local time zone of the operator panel takes place
- **Date/time format:** Specifies which values are to be output from the date/time

- Date: Only the date is displayed
- Time: Only the time is displayed (format: HH:mm)
- Date and time: The date and time are displayed and separated by a space
- Time with seconds: Specifies whether the seconds should also be displayed (format: HH:mm:ss)

5.2.4.5 Bar Display

With the Bar Display, numerical values can be visualized as a colored bar. This view element is particularly useful for displaying percentages, fill levels or other scaled measured values. In addition to the bar, the current numerical value (bar value) can also be displayed as an option.

Examples:



Note: The middle and right Bar Display has been configured with a rotation angle of -90 °C under "Position" to display the bar vertically.

Property group "Configuration: Value":

The value is configurable as follows:

- **Datapoint:** The datapoint determined for the bar length and used as the bar value (if not hidden)
- **Range of values:** Sets the range for the possible value, affecting the bar length
 - Fixed range of values / Range of values from datapoint: Specifies whether the value range should be set with fixed values or via additional minimum and maximum datapoints
 - Minimum: Sets the minimum (e.g. "0" for 0%)
 - Maximum: Sets the maximum (e.g. "100" for 100%)
- **Number format:** Here you must configure the number format
 - Static configuration: The number format is set via the "..." button in the dialog (see Chapter 4.5)

- Dynamic configuration: A datapoint of type "Text" must be selected to provide the number format (see Chapter for format description 4.5.2)
- **Show bar value:** Specifies whether the bar value should be displayed to the right of the bar

5.2.5 Input Elements

Input elements allow users to enter or change values directly from the HMI app. The entered values are transmitted to the configured datapoint. Depending on your use cases, different input types are available – from simple Text Input to convenient selection or Dropdown List or Slider elements.

The "Input" group includes the following specific input view elements:

- **Numeric Input:** Allows you to enter a numeric value using the keyboard or, on mobile devices, by scanning a barcode
- **Numeric Stepper:** Allows you to change a numeric value via plus/minus buttons
- **Text Input:** Allows you to enter a text value using the keyboard or the camera for barcode scanning on a mobile device
- **Dropdown List:** Allows you to select a predefined value from a list
- **Date/Time Input:** Allows you to enter a date or time using a date picker field
- **Slider:** Allows you to select a numeric value via a slider

Input datapoint:

Each input element is linked to an input datapoint that is written as it is entered. The input datapoint must be configured in the datapoints table with "Write, Read" access.

Parts:

An input element consists of several configurable parts.

- **Body:** The background or container for the parts "Label" and "Input field"
- **Label:** Optional label to value – can be displayed at the top or right of the value
- **Input field:** Datapoint value for the representation of the current value and for the input
- **Stepper buttons (only for "Numeric Stepper"):** buttons for quickly increasing or decreasing the value
- **'Scan' button (only for "Numeric Input" and "Text Input"):** button to scan the value
- **Drop-down symbol (only for "Dropdown List" and "Date/Time Input"):** Button to open the selector element

Special case of "Sliders":

In the Slider, the "Value" part has been replaced by the "Bar" and "Bar value" parts.

5.2.5.1 Common input properties

In addition to the general properties (see Chapter 4) that apply to all view elements, the following common properties are available for all input elements:

Property group "Alignment: Label":

The following input-specific properties can be found here:

- **Orientation:** Specifies whether the label should appear above the input field (bar) or to the left of the input field (bar)
 - Label on top: Name appears above the input field
 - Label on left: Label appears to the left of the input field
 - Label width (%): Must be configured when selecting "Label on the left". Specifies the percentage width of the label in the body

5.2.5.2 Numeric Input

With this view element, a numerical value can be entered via the keyboard or taken over via barcode or QR code scan on mobile devices. In addition, minimum and maximum values can be set to limit the input to a defined range of values.

Examples:



The diagram illustrates two configurations of the `<NumericInput_1>` view element. In the first configuration, the label `<NumericInput_1>` is positioned above the input field, which contains the value `60.0`. In the second configuration, the label `60.0` is positioned to the left of the input field, which also contains the value `60.0`.

Property group "Configuration: Value":

Here the value to be displayed for the input field is configured:

- **Input datapoint:** The datapoint that will be associated with the input field
- **Range of values:** Sets the minimum and maximum allowed value input
 - Fixed Value Range / Datapoints for Value Range: Specifies whether the value range should be set with fixed values or via additional minimum and maximum datapoints
 - Minimum: Sets the minimum allowed for input
 - Maximum: Sets the maximum allowed input
- **Number format:** Here you must configure the number format for the representation of the input value. The number format is set via the "..." button in the dialog (see Chapter 4.5)

Property group "Configuration: Input options":

Specifies whether the input should be made exclusively via scanning (Android, iOS only) or via the keyboard.

- **Input mode:** Defines the type of input
 - **Keyboard:** Input is done via the keyboard
 - **Barcode scanner (iOS, Android only):** Input is made by scanning the barcode or QR code with the camera. For Windows operator devices, this setting is ignored

5.2.5.3 Numeric Stepper

With the Numeric Stepper, values can be quickly increased or decreased via "+"/"-" buttons – ideal for touch input or operation with the mouse. In addition, direct input in the input field is possible.

Examples:



Property group "Configuration: Value":

The configuration of the value is analogous to that of the "Numeric Input" view element. In addition, the following value must be set:

- **Step value:** Specifies the amount by which the value is increased or decreased when the "+"/"-" buttons are clicked

5.2.5.4 Text Input

With this view element, a text value can be entered using the keyboard. The input can optionally also be configured as a password field, whereby the entered text is replaced by wildcard characters.

Examples:



Property group "Configuration: Value":

Here, the input field is linked to a datapoint.

- **Input datapoint:** The datapoint that will be associated with the input field
- **Maximum length:** Maximum number of characters that the entered text can contain

Property group "Configuration: Input options":

Specifies whether the input should be made exclusively via scanning (Android, iOS only) or via the keyboard.

- **Input mode:** Defines the type of input
 - **Keyboard:** Input is done via the keyboard
 - **Barcode scanner (iOS, Android only):** Input is made by scanning the barcode or QR code with the camera. For Windows operator devices, this setting is ignored
- **Password entry, text is displayed masked:** Determines whether a password should be entered

5.2.5.5 Dropdown List

With the "Dropdown List" view element, input is made via a list of predefined values. The available entries are based on a status text type and are displayed to the user in a dropdown menu. After selecting an entry, the corresponding value is transmitted to the configured datapoint. Alternatively, the entries can be obtained dynamically at runtime via a table datapoint.

Examples:



Property group "Configuration: Value":

This is where the possible entries and the datapoint associated for the input are configured.

- **Input type: Selection list (status texts):** The possible entries are defined by the status text type, which is to be set in the "Status text type" field
- **Input type: Selection list (dynamic):** The possible entries (list contents) are supplied by a table datapoint, which is to be set in the "List content (table)" field
- **Input datapoint:** Datapoint that is described in the list selection
 - **For status texts:** The value of the status text is written to the input datapoint
 - **For "dynamic":** The value of the first column is always written from the selected row of the table datapoint to the input datapoint. If the supplied table

of the datapoint has more than one column, the value of the second column is used for display, otherwise the value of the first column

5.2.5.6 Date/Time Input

The "Date/Time Input" input element can be used to enter a date or time.

Examples:



Property group "Configuration: Value":

- **Input datapoint:** Datapoint that is described as you type. The datapoint must be of type "Text". The selected date is written to the datapoint in the format "MM/dd/yyyy HH:mm:ss"
- **Time zone of datapoint value:** Specifies how the datapoint value is interpreted. If the "Universal time zone (UTC)" option is selected, the display automatically converts to the local time zone of the operator device. When writing, the local input value is converted to UTC.
- **Date/time format:** Specifies whether to capture the date or time as you type
 - Date: Input element for date only
 - Time: Input element for time only

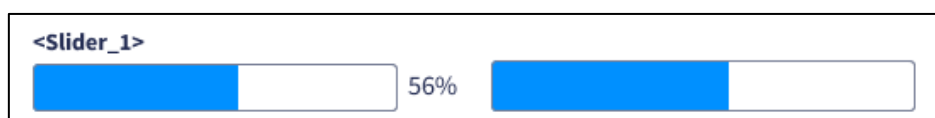
Note: If you want to enter both the date and time, you can create two instances of the Date/Time Input element, both of which are linked to the same input datapoint.

5.2.5.7 Slider

The "Slider" input element can be used to select a numeric value by moving a bar with the mouse or using a touch gesture. The selected value is transmitted to an input datapoint.

The element is similar in appearance to the Bar Display, but expands it with the possibility of direct value input through user interaction. It is particularly suitable for intuitive setting of percentages, thresholds, or other scaled values.

Examples:



Property group "Configuration: Value":

Here the value to be displayed for the input is configured:

- **Input datapoint:** The datapoint that is associated with the bar input
- **Range of values:** Sets the range for the possible value, affecting the bar length
 - **Fix range of values / Range of values from datapoint:** Specifies whether the value range should be set with fixed values or via additional minimum and maximum datapoints
 - **Minimum:** Sets the minimum (e.g. "0" for 0%)
 - **Maximum:** Sets the maximum (e.g. "100" for 100%)
- **Number format:** The number format is set via the "..." button in the dialog (see Chapter 4.5)
- **Show bar value:** Determines whether the current bar value is displayed or hidden to the right of the bar

5.2.6 Button Elements

Button elements allow users to trigger actions or control states via the visualization. Depending on the element type, they can be used either to trigger an action once or to directly control a datapoint in the form of a state (e.g. on/off).

The button elements differ in terms of their behavior and typical use cases – from classic action triggers to State Switches for user interfaces.

The "Button" group includes the following specific view elements:

- **Action Button:** Triggers an action. The press-and-release behavior can be configured separately (execute script, set datapoint)
- **State Button:** Represents a specific state (e.g., On/Off). When pressed, a defined value is written to the datapoint. Suitable for e.g. tab controls or for switching views
- **Switch:** Represents a binary state (on/off) and automatically toggles between values 0 and 1 when pressed
- **Checkbox:** Functionally like Switch view element, but with typical checkbox representation
- **Radio Button:** Corresponds to an option button. The button is considered active (pressed) if the datapoint has a certain value. Ideal for choosing from multiple options

Parts of Buttons:

A button element consists of several parts.

- **Body:** The background or container for the other parts
- **Label:** Optional text for the button

- Image (only for "Action Button" and "State Button"): Optional image (icon) for the button - the image position is configurable
- Symbol (only for "Checkbox" and "Radio Button"): Box that shows the on/off status
- Switch: (only for "Switch"): Inner circular element to turn on/off

5.2.6.1 Action Button

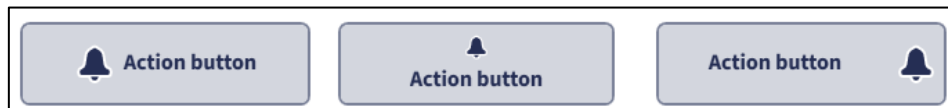
The "Action Button" view element can be used to trigger a user action. The press-and-release behavior can be configured separately to set different control commands or datapoint values.

The Action Button is particularly suitable for short-term control actions such as starting or stopping a process, triggering a pulse or activating a temporary state. It behaves like a button that is only active as long as it is pressed – or triggers different actions when pressed and released.

For safety-critical applications, such as crane control, a survival trigger can also be activated:

While the button remains pressed, a trigger datapoint is described with a specific value at configurable intervals. This can be used on the controller to automatically end the action if the trigger is not received again within the expected amount of time. This allows security-relevant requirements to be implemented, for which permanent user interaction must be proven.

Examples:



Presentation Conditions:

An Action Button has two visual states:

- "Normal": Displayed when the button is not pressed
- "Pressed": Displayed as long as the button is actively pressed

The appearance (colors, borders, fonts, etc.) can be configured individually for each of these states in the "Appearance" properties palette. The preferred state in Engineering mode (editor) can also be selected in the "Appearance" palette, "Common" tab.

Property group "User actions: Actions":

This defines which datapoints are to be set when the button is pressed and released (see Chapter 4.1.3 – "Set datapoint" action type).

Property group "Alignment: Image":

In addition to the general properties, the image position relative to the label can be configured here – see Chapter 4.1.7.

5.2.6.2 State Button

The State Button is a view element for displaying and controlling a binary or defined state. It looks very similar to an Action Button, but it permanently indicates its current state, such as text, color, or borders.

When clicking, you can choose to:

- between the values 0 and 1 (switching function), or
- a fixed value (e.g. "2") can be written to the linked datapoint (e.g. for tab buttons where the second range is displayed at value 2).

This makes the status button suitable for both classic on/off switches and multiple selections, as is common with tabs or mode switches.

Examples:



States:

A State Button has two visual states:

- "Switched on": Displayed when the linked datapoint under user actions has the value 1 (or a defined datapoint value)
- "Switched off": Displayed when the datapoint has the value 0 (or does not match the defined datapoint value)

The appearance (colors, borders, fonts, etc.) can be configured individually for each of these states in the "Appearance" properties palette. The preferred state in Engineering mode (editor) can also be selected in the "Appearance" palette, "Common" tab.

Property group "Configuration: Label":

In addition to the information described in chapter 4.1.2 properties, you can define whether the same display text should be used for both states or whether separate text should be displayed for "on" and "off".

Property group "User actions: Actions":

Here you can specify which datapoint should be switched when you click (0 ↔ 1).

See Chapter 4.1.3– either the action type "Set datapoint" or "Toggle datapoint" can be used.

5.2.6.3 Switch

The Switch is a view element for displaying and directly controlling a binary state – typically "on" or "off". When pressed, the linked datapoint is automatically toggled between the values 0 and 1.

The Switch is particularly suitable for simple on/off operations, such as activating or deactivating functions or switching machine parts on and off. The current switching position is visually represented and corresponds to the current value of the datapoint.

Examples:



States:

The Switch has two visual states:

- "Switched off": Displayed if the datapoint has a value of 0
- "Switched on": Displayed if the datapoint has a value of 1

The appearance (colors, borders, fonts, etc.) can be configured individually for each of these states in the "Appearance" properties palette. The preferred state in Engineering mode (editor) can also be selected in the "Appearance" palette, "Common" tab.

Property group "Configuration: Label":

In addition to the information described in chapter 4.1.2 properties, you can define whether the same display text should be used for both states or whether separate text should be displayed for "on" and "off".

Property group "User actions: Actions":

Here you can specify which datapoint should be switched when you click (0 ↔ 1).

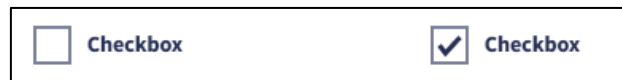
See Chapter 4.1.3– "Toggle datapoint" action type.

5.2.6.4 Checkbox

The Checkbox is a view element for controlling a binary state – like the Switch. It uses the well-known Checkbox display, where a checkmark is visible when the state is "on".

When clicked, the linked datapoint is toggled between the values 0 and 1. The visual representation shows the current state of the datapoint and is particularly suitable for simple on/off settings or user options.

Examples:



Presentation Conditions:

A Checkbox has two visual states:

- Turned off: Displayed if the datapoint has a value of 0.
- Turned on: Displayed if the datapoint has a value of 1.

The display (color, border, font, etc.) can be configured individually for both states in the "View" properties palette. In addition, it is possible to set whether the checkbox symbol is displayed to the left or right of the display text.

Property group "User actions: Actions":

Here you can specify which datapoint should be switched when you click (0 ↔ 1).

See Chapter 4.1.3– "Toggle datapoint" action type.

Property group "Alignment: Symbol":

In addition to the general properties, the symbol position relative to the label can be configured here – see Chapter 4.1.7.

5.2.6.5 Radio Button

The Radio Button is a view element for selecting a certain state from several possible options.

When clicked, a predefined value is written to the linked datapoint. The Radio Button is considered "on" if the current value of the datapoint corresponds exactly to this target value. This allows you to configure multiple Radio Buttons so that only one of them is active at a time, making it ideal for selection groups such as operating modes or view options.

Examples:



Presentation Conditions:

A Radio Button has two visual states:

- Switched off: Displayed if the datapoint does not correspond to the configured target value.
- Switched on: Displayed when the datapoint matches the target value.

The display (color, border, font, etc.) can be configured individually for both states in the "Appearance" properties palette. In addition, it is possible to set whether the circle symbol is displayed to the left or right of the display text.

Property group "User actions: Actions":

This determines which value should be written to the datapoint when clicked. The Radio Button is considered "active" if the datapoint has this value.

See Chapter 4.1.3– "Set datapoint" action type.

Property group "Alignment: Symbol":

In addition to the general properties, the symbol position relative to the label can be configured here – see Chapter 4.1.7.

5.2.7 View Navigation

The view element "View Navigation" can be used to open another view within the visualization. This element allows you to navigate between different user interfaces, such as detail pages, settings, or subpages, in a multi-level visualization.

The display can be either as a classic button or as an underlined text (hyperlink) – depending on the desired style of the user interface. When clicked, the target view configured in the properties opens.

Examples:



Appearance states:

The View Navigation has two states, each of which can be configured individually:

- "Default": Used by default.
- "Selected": Used when the selection mode "Select, if opened" is activated and the currently open view matches the navigation element under "Linked View". For example, the currently open view can be easily highlighted in the navigation menu.

The display (color, border, font, etc.) can be configured individually for both states in the "Appearance" properties palette.

Property group "Configuration: Link":

In this property group, the view to be opened is configured. In addition, it is defined how the display state (normal/selected) is determined.

- **Linked view:** View to open when clicked
- **Ignore current view when navigating back:** Determines whether the view is placed on the view stack when navigating, so that the view can be navigated back with "Back". In this way, subordinate views can be specifically excluded from navigating back
- **Selection mode:** Determines when which display state is displayed:
 - Never selected: The "Default" state is always applied
 - Always selected: The "Selected" state is always used
 - Select, if opened: The "Selected" state is displayed if the currently opened view corresponds to the linked view

Properties group "Alignment: Image":

Here, in addition to the general properties, the image position relative to the label can be configured – see Chapter 4.1.7.

5.2.7.1 Create a reusable navigation panel

In many HMI apps, there is a navigation area in the left panel. This can be implemented as follows:

"View Navigation" elements in Template:

Create a Template (see Chapter 3.8.1) and insert all desired View Navigation elements there. For all View Navigations, the selection mode must be set to "Select, if opened". Instantiate the template in all views in the left pane.

Later changes to the template are automatically applied to all instances.

Alternatively, the reusable navigation can also be used with the View Frame (see Chapter 3.9).

5.2.7.2 Navigation with primary and secondary navigation panels

Example: In the left area, a primary navigation has to be displayed, which has an influence on a secondary navigation bar in the lower area.

Procedure:

1. Create a View Frame per primary View Navigation
 - a. The selection mode must be set to "Never select" for all primary View Navigations, and to "Always select" for active View Navigations
 - b. Insert the corresponding secondary navigations in the lower area – there the selection mode is set to "Select when opened".
2. Link all views to the desired View Frame – the currently open view should be reflected in the secondary navigation

5.2.8 Hyperlink

The "Hyperlink" view element allows you to open an external website or other online destination from the device's default web browser. Clicking on the item opens the configured URL in the external browser.

By default, the display is as underlined text – alternatively, a button display can also be selected.

This element is ideal for referring to further information, manuals, online services or support pages from the visualization. The linked documentation can also be stored offline and downloaded via a URL in the format "file:///C:/..." referenced.

Examples:



Property group "Configuration: Web address":

Here the destination address is defined that should be opened when clicked.

- **Static configuration / Dynamic configuration with datapoints:** Determines whether the target address is to be read dynamically from a datapoint or is already specified in engineering.
- **Address (URL):** The full address to open in the browser. Examples: file:///C:/Dokumente/Manual.html, https://www.unified-e.com

Property group "Alignment: Image":

Here, in addition to the general properties, the image position relative to the label can be configured – see Chapter 4.1.7.

5.3 Extended Elements

Advanced elements are composite or low-level view elements that perform more complex tasks or are deeply linked to the functionality of the visualization environment. They offer integrated logic, access specific system functions or use special type definitions (e.g. messages or recipe types). As a rule, they are less freely designable than simple elements, but offer ready-made functions for recurring tasks in industrial HMI applications.

5.3.1 Message Display Elements

Message displays visualize the current machine or system status in the form of critical alarms, warnings or information messages. In addition, the "Archive Message Table" view element can also be used to display past messages.

To use the message displays, messages, message classes or message groups must be configured in advance in the respective editors. These editors can be found in the Project Navigation under the "Messages" entry (see Chapter 6). Message indicators can be configured to include only messages from certain classes or groups.

The following message indicators are available:

- **Message Table:** Displays pending messages (trigger condition met) and allows acknowledgment. The display is tabular or smartphone-optimized as a tile view
- **Archive Message Table:** Displays past messages and allows time-based filtering
- **Message Symbol:** Displays an icon when at least one message is pending
- **Message Counter:** Displays the number of current messages in a circle icon
- **Message Text:** Displays the message text of the currently pending message

5.3.1.1 Common message display properties

In addition to the general properties (see Chapter 4) that apply to all view items, the following common properties are available for all message display items:

Property group "Configuration: Select messages":

This property group defines which messages are to be included in the message display. The selection can be made either by message classes or message groups. Filtering directly affects the messages displayed in the various message items.

- **Message classes:**
 - **All Message Classes:** Messages from all message classes are displayed (default).

- Multiple Message Classes: Allows you to select several predefined classes (e.g. "System Error", "Alarm", "Warning", "Info").
- A message class: A single message class can be specifically selected via the drop-down menu.
- Message groups:
 - All Message Groups: Messages from all defined groups are displayed.
 - Multiple Message Groups: Allows you to select multiple specific groups.
 - A Message Group: The drop-down menu can be used to define a single message group.

These two filters – message classes and message groups – are each set independently of each other and act together as a combined filter. A selection must be made for both areas (e.g. all, single or multiple classes/groups). Only messages that meet both criteria will be included in the ad.

5.3.1.2 Message Table

The Message Table is a view element for displaying currently pending, not yet acknowledged alarms or messages. It is used to monitor the machine or plant status and supports the user in quickly recognizing critical situations and reacting in a targeted manner.

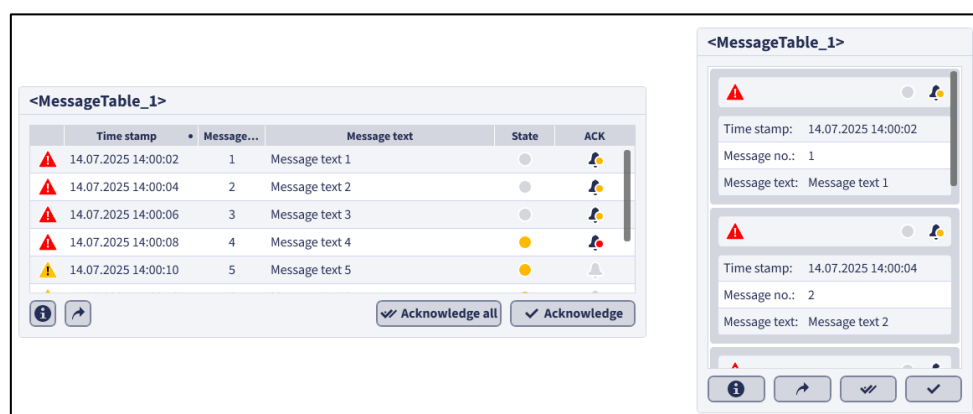
The messages themselves are not configured in this element, but centrally via the editors for messages, message classes and message groups, which are available in the Project Navigation under the entry "Messages" (see Chapter6). The Message Table then automatically displays the pending messages according to the filters you have set.

The display can be either in tabular form or as a tile, whereby the latter is particularly suitable for mobile devices. In the tabular view, the displayed columns can be individually adjusted.

In addition, the Message Table supports interactive functions:

If a message is selected, a help dialog with further information can be automatically displayed or jumped to a linked view that has been defined for the respective message.

Examples:



Property group "Configuration: Select columns":

This property group specifies which columns should be displayed in the Message Table and how they are formatted. The appearance of each column can be customized.

- **Column type** (column selectable via checkbox in first column):
 - **Message Class Icon:** Displays the icon of the corresponding message class
 - **Message class:** Text name of the message class (e.g. "Alarm", "Warning")
 - **Priority:** Priority of the associated message class
 - **Timestamp:** Time when the message was triggered
 - **Registration number:** Internal number of the message
 - **Message text:** Main text of the message
 - **Status:** Shows whether the message is active or already acknowledged
 - **Acknowledgment:** Timestamp of the acknowledgment (if acknowledged)
 - **Acknowledgment User:** Username who acknowledged the message
 - **Message Group:** Assigned Message Group
 - **Message context:** Optional context (e.g. current batch, see Chapter 6.1.1)

Other column properties:

- **Display text (header):** Heading that appears in the table for each column. This can be individually adapted
- **Horizontal Alignment (Head):** Specifies how the column header is textually aligned – e.g. "Center" or "Left"
- **Horizontal Alignment Content:** Controls the orientation of the content within the cell
- **Width Type:**
 - **Fixed pixel size:** Column has a fixed width (in pixels)
 - **Fill:** Column dynamically fills the remaining available space
- **Width Value:**
 - **Fixed Width:** Specifies the width of the column, in pixels.
 - **For "Fill":** The remaining space available is distributed proportionally. The value determines how many parts of the remaining space this column should take up (e.g. "1" = single, "2" = twice as much as one with "1")

Property group "Configuration: Select buttons"

This table specifies which buttons should be displayed within the Message Table and how they are displayed in each case – separately for the table mode and the tile mode.

Column description of the configuration table:

- **Button type:** Type of button. The following types are available:
 - To the view: When clicked, opens the view associated with the message (if configured when the message is displayed).
 - Info: Displays a configured help text for the message in a dialog (if configured when messageed).
 - Acknowledge: Acknowledges the selected message, if the corresponding message class is confirmed).
 - Acknowledge all: Acknowledges all currently visible, acknowledgeable messages.
- **Display text:** The text that appears on the button. This can be adapted depending on the language.
- **Content (Table Mode):** Determines how the button is displayed in tabular mode:
 - Icon Only: Only the icon is displayed.
 - Text Only: Only the display text is displayed.
 - Text and symbol: Both are displayed.
- **Content (Tile Mode):** Appropriate display of the button in tile layout, as it is used on smartphones or small displays. The same display options are also available here.

Property group "Configuration: Display options":

Specifies whether the Message Table should display the entries in rows or in tiles.

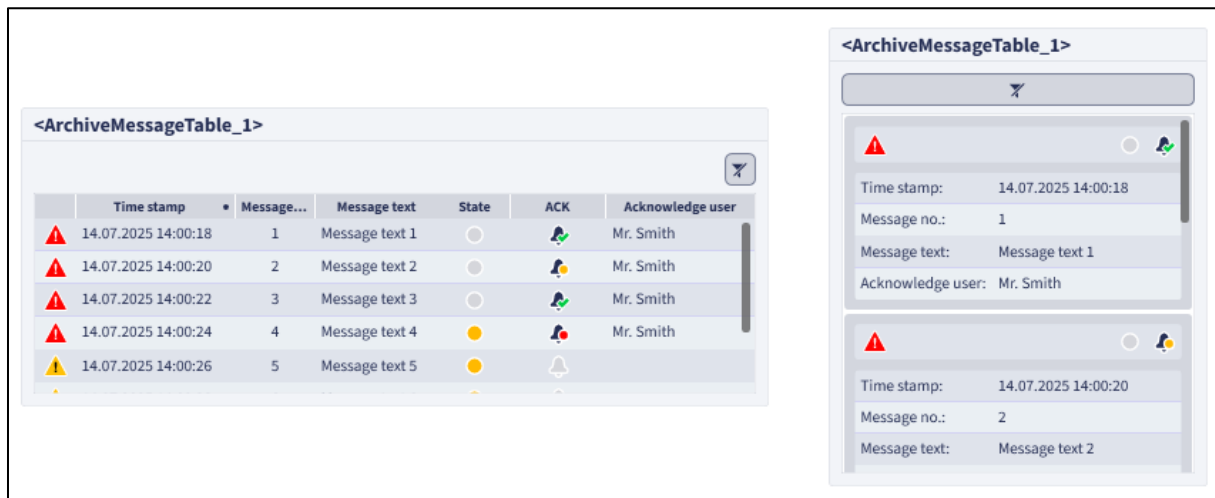
- **Table:** The message entries are displayed in rows in a table with a table header
- **Tiles (optimized for smaller display width):** The list entries are displayed in tiles below each other – e.g. for smartphones
- **Automatic:** Depending on the width (at runtime) of the Message Table, it is displayed as a table or in tile mode. This option is to be selected if the Recipe Table is configured in a view that is laid out for the display (see 3.3)
 - Minimum Table Width (px): Specifies the width at which Table Mode is used

5.3.1.3 Archive Message Table

The Archive Message Table is used to display past messages that occurred at an earlier time and were acknowledged or automatically reset. In contrast to the normal Message Table, which only shows currently pending messages, this element represents a message history and is suitable for logging and tracking events in operation.

The display and configuration of the Archive Message Table is largely identical to the normal Message Table. Here, too, the columns to be displayed as well as optional buttons can be defined individually. In addition, the element has a filter button that can be used to filter specifically according to certain time periods or other criteria.

Examples:



Property group "Configuration: Select columns":

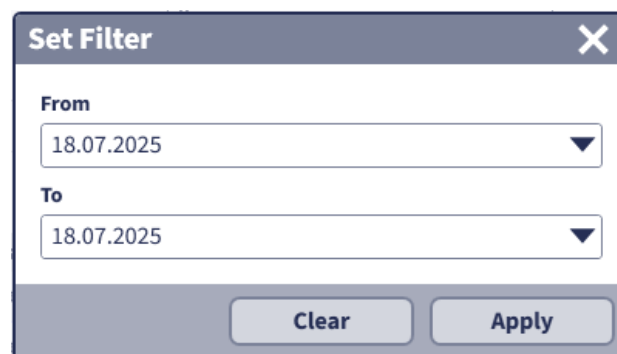
The configuration is analogous to the column configuration for the Message Table (see Chapter **Error! Reference source not found.**).

Property group "Configuration: Select buttons"

The configuration is analogous to the column selection of the Message Table (see Chapter **Error! Reference source not found.**). However, in the case of the Archive Message Table, only the "To view" and "Info" buttons are available. In addition, the "Filter" button can be activated, which is always displayed above the table.

"Set filters" dialog:

If you click on the "Filter" button, a dialog for filtering the time period appears. The display of the dialog is based on the properties of the appearance template (see Chapter 15.5**Error! Reference source not found.**)



5.3.1.4 Message Symbol

The message icon is a compact view element for displaying active messages. It shows an icon (e.g. alarm bell) when there is at least one active message and is particularly suitable for status bars or header areas.

When clicked, a defined view can be opened, such as a message page or detail view.

Examples:



Property group "Configuration: Symbol":

- **Show icon on active messages:** Defines which icon is displayed when there is at least one active message
 - **Show Collective Symbol:** Always displays the same symbol, regardless of the registration class. The icon is any image from the Images editor (e.g. "Alarm bell dark blue")
 - **Show Message Class Icon:** Displays the icon of the corresponding message class in the event of an active message (see Chapter 6.1.2). An icon should be defined in the Message Class editor for all selected message classes. If several messages are active, then the message class with the highest priority is used
- **What to do when no messages are active**
 - **Show inactive symbol:** Shows an alternative symbol as soon as there are no more active messages (e.g. "Alarm bell light grey")
 - **Hide Element:** The entire icon is hidden when no message is active

Property group "User Actions: Actions":

- **Open View:** Allows you to link to a target view to open when you click the icon. Typically, a message view or a detail page is used here.
- **Don't include open view when navigating with back button:** If enabled, the view open when clicked is not placed on the navigation stack. So when the "Back" button is pressed, the view is skipped. This is especially useful for temporary or secondary views

5.3.1.5 Message Counter

The Message Counter is a compact view element for displaying the number of currently active messages. The number is displayed in the center of a round icon. It is particularly suitable for clearly displaying critical plant states, e.g. in the header or on a dashboard page. The icon can be linked to an action, such as clicking to open a message page.

Examples:



"Symbol" property group:

- Background color:
 - Use Color as Configured in Appearance: Uses the background color configured in the View palette of the View element
 - Use Message Class Color: Uses the color of the upcoming message class. If multiple messages are active, the color of the message class with the highest priority is used
- What to do when no messages are active:
 - Show element: The circle is also displayed when there are 0 messages (the number is hidden or displayed as "0", depending on the display)
 - Hide element: The Message Counter is completely hidden if there is no message pending

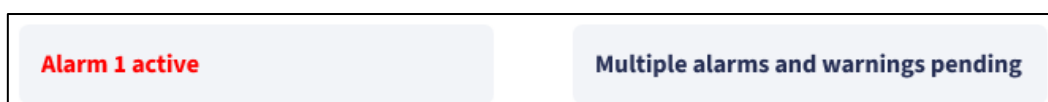
Properties group "User Actions: Actions":

- Open View: Allows you to open a defined view (e.g. a detail or message page) by clicking on the Message Counter
- Do not consider open view when navigating with back button: The open target view is excluded from the history list of the navigation, so the back button leads to the previous view

5.3.1.6 Message Text

The Message Text is a view element for displaying pending messages as plain text. It is particularly suitable for compact status lines or situation-dependent information within a visualization

Examples:



Property group "Configuration: Message text":

- Text color:
 - Use color as configured in Display: The text color depends on the selected display in the Display editor

- Use Message Class Color: The text color of the message class of the displayed message is used. If several messages are active, then the message class with the highest priority is used
- Text when multiple messages are active: Used when multiple messages are active at the same time
- What to do when no messages are active:
 - Show Inactive Text: Displays custom text when there are no pending messages (e.g. "No message active").
 - Hide element: The element is completely hidden if there are missing messages.
- Allow line break: Enables automatic wrapping for longer texts

Properties group "User Actions: Actions":

- Open View: Allows you to open a defined view (e.g. a detail or message page) by clicking on the Message Counter
- Do not consider open view when navigating with back button: The open target view is excluded from the history list of the navigation, so the back button leads to the previous view

5.3.2 Recipe Table

The "Recipe Table" view element is used for managing recipe datasets in the HMI visualization. Users can create, edit, delete, and specifically activate a specific dataset – the setpoint values are transferred to the PLC or other endpoints.

For the Recipe Table to work, at least one configured recipe type must be assigned to it (see Chapter 8). This defines the data structure and the basic behavior of the Recipe Table – e.g. whether a folder-Management is used to determine whether datasets are versioned or whether temporary adjustments are allowed before activation. In the chapter 8.1 all the important terms relating to the topic of recipes are described in detail.

Together with the recipe types, the Recipe Table supports a wide range of functions:

- Visualization of the production process:
Optionally, current sensor values can be displayed and compared to setpoint values. Parameter groups, individual parameters, or entire process steps can be highlighted in color, making the current production state directly visible in the table.
- Versioning & traceability:
Changes to datasets can optionally be versioned. Previous versions remain accessible and can be restored or used as a basis for new recipes.
- Permission control:
User roles define who is allowed to create, edit, activate, or delete datasets. Operators can – if permitted – adjust specific parameters before activation without modifying the original dataset.

- **Dynamic visibility:**
Parameters can be shown or hidden dynamically depending on other parameters or datapoints (e.g., PLC variables) – without the need for multiple recipe types.
- **Step-based process representation:**
Production processes with several similar phases can be represented as steps within the table, with the active step highlighted.
- **Parameter adjustments before activation:**
Operators can adjust permitted parameters within a defined range before activating a dataset. These adjustments are not saved to the record itself but are only applied during activation (loading).

Many of these functions are configured in the associated recipe type rather than in the Recipe Table itself and are described in detail in Chapter 8.

5.3.2.1 Structure of the Recipe Table

The "Recipe Table" view element can be configured very flexibly. The image figure also shows all the optional elements that are visible for all activated functions (see Base-Settings of the recipe type in the chapter 8.6).

The table consists of two main views:

- The "All datasets" view is used for administration, e.g. for editing or activating a dataset (left-hand table in the figure).
- The "Active dataset" view shows the currently set parameters of the activated dataset (right-hand table in the figure).

Switching between the two views is done via the view switcher (see below).

<RecipeTable_1>

Recipe type: <RecipeType_1> Active dataset: Default\Dataset_1\V1

Select dataset

Folder: Default Dataset: Dataset_1

Version	Modified on	Modified by	Comment
✓ V1	14.07.2025 14:10:49	---	---
V0	14.07.2025 14:10:02	---	---

<RecipeTable_1>

Active dataset: Default\Dataset_1\V1

Parameter values

<ParameterGroup_1> <ParameterGroup_2>

Name	Unit	Def. setpoint	Act. value
<Parameter_1>		0.0	
<Parameter_2>		0.0	
<Parameter_3>		0.0	

Elements or areas (numbering according to figure):

1. Table title (optional):
Optional title that can be configured in the properties under "Label".
2. "Active dataset" info area:
Displays the name of the currently activated dataset – if available.
3. View selector: Toggles between the main views "All datasets" and "Active dataset".
4. Dataset list:
Displays all datasets of the currently selected folder in list form. The selection of a dataset determines which dataset subsequent actions (e.g., activate, edit) refer to.
5. Button bar:
This is where the available action buttons are configured. Buttons without text are aligned to the left, those with text to the right.
6. Dropdown "Recipe type" (optional):
Only visible if multiple recipe types are linked to this table. The recipe type selected here determines the context for the other display elements.
7. Dropdown "Folder" (optional):
Visible if folder management has been activated in the base settings of the recipe type. The dataset list always refers to the folder selected in this drop-down box.
8. Dropdown "Dataset" (optional):
Visible if versioning has been activated in the base settings of the recipe type. Here, the dataset without version information is selected; the individual versions are offered for selection in the dataset list.
9. Parameter list:
Displays the parameters of the currently selected or activated dataset. The content depends on the current view ("All datasets" or "Active dataset").
10. Group tab (optional):
Displays the parameter groups defined in the recipe type as tab tabs. By selecting a tab, only the corresponding group is displayed in the parameter list. The group tab can be hidden if only one group has been defined.
11. Step tab (optional):
Steps can only be selected if this function has been activated in the base settings (see also chapter 8.1.6). This tab can also be hidden if necessary, e.g. if only one step per group is defined.

All of the elements mentioned are described in detail in the following subchapters and can be configured via their respective properties – e.g. colors and visibility in the "Appearance" palette.

5.3.2.1.1 States of "All datasets" view

If no dataset has yet been activated, the "All datasets" view is visible by default. In this "Display all datasets" state, a dataset can be selected from the list and edited or activated using buttons, for example.

The "All datasets" view has the following states or subviews:

- **Display all datasets:**
Displays the dataset list for selecting a dataset.
- **Display dataset:**
This state is achieved by clicking on the "Show dataset" button. The parameter list of the selected dataset is displayed, but only for display (read-only).
- **Edit dataset:**
Activated by clicking on the "Edit" button. The parameter list appears in edit mode; Changes can be made and saved.
- **Adjust dataset (optional):**
This state is activated by clicking on the "Adjust dataset" button. In the parameter list, released parameters can be adjusted for activation. The original dataset remains unchanged. This state can only be achieved if customization has been enabled before activation in the base settings of the recipe type.

For each of these states, the list columns and buttons to be displayed can be individually configured (see Chapter 5.3.2.3.1).

The following figure shows the "Adjust dataset" state, in which the selected dataset can be specifically adjusted before activation.

<RecipeTable_1>

Active dataset: Default\Dataset_1\V1

Adjust dataset ☐ All datasets ☒ Active dataset

<ParameterGroup_1> **<ParameterGroup_2>**

1

Name	Unit	Def. setpoint	Adjustable	Adj. setpoint
<Parameter_1>		0.0	Yes	0.0
<Parameter_2>		0.0	No	---
<Parameter_3>		0.0	No	---

Cancel **Apply**

5.3.2.1.2 States of "Active Dataset" view

The "Active dataset" view of the Recipe Table shows the parameters of the currently activated dataset.

The "Active dataset" view has the following states or subviews:

- **Display active dataset:**
 Displays the parameter list with the values of the activated dataset. If configured, process status highlights are also displayed (see image below).
- **Edit dataset:**
 Activated by clicking on the "Edit" button. The parameter list of the active dataset appears in edit mode; Changes can be made and saved.
- **Adjust dataset (optional):**
 This state is activated by clicking on the "Adjust dataset" button. In the parameter list, adjustable parameters can be adjusted before activation. The original dataset remains unchanged. This state is only available if "Support parameter adjustments" has been enabled in the base settings of the recipe type.

For each of these states, the list columns and buttons to be displayed can be individually configured (see Chapter 5.3.2.3.1).

The following figure shows the "Display active dataset" state, in which the activated dataset can be specifically adjusted.

<RecipeTable_3>

Active dataset: Dataset_1\VO

Parameter values ○ All datasets ● Active dataset

Name	Unit	Def. setpoint	Act. value
<Parameter_1>		0.0	
<Parameter_2>		0.0	
<Parameter_3>		0.0	

Edit

5.3.2.2 Main use cases of the Recipe Table

The "Recipe Table" supports two different usage models, depending on how recipe values are processed and stored in the process.

5.3.2.2.1 Protected editing in the Recipe Table

In this model, editing and saving of recipe values is carried out exclusively in the Recipe Table. Changes to the parameters are made in a targeted manner, then saved and optionally versioned. The load datapoints are set in the HMI app only when the dataset is activated.

Advantages:

- All changes are made in a controlled manner at the Recipe Table as a central location
- Traceable versions of the datasets are created after each edit
- Ideal for requirements with documentation requirements or traceability (e.g. FDA-compliant processes)
- The production process is based on clearly defined, reproducible setpoint values

This model offers maximum transparency and is particularly suitable for regulated environments or quality-critical applications (GMP, FDA).

5.3.2.2.2 Set recipe setpoints decentrally (recipe screen)

In this model, the load datapoints belonging to the recipe parameters are also set outside the Recipe Table – for example, in input elements in a plant screen or in an individually designed view. The operator can change the setpoint parameters directly on the spot without switching to the Recipe Table, editing the dataset, and reactivating it.

The currently setpoint values are retained in the datapoints and can be permanently stored in the corresponding recipe dataset at a later point in time in the Recipe Table using the "Save current setpoint values" button.

Advantages:

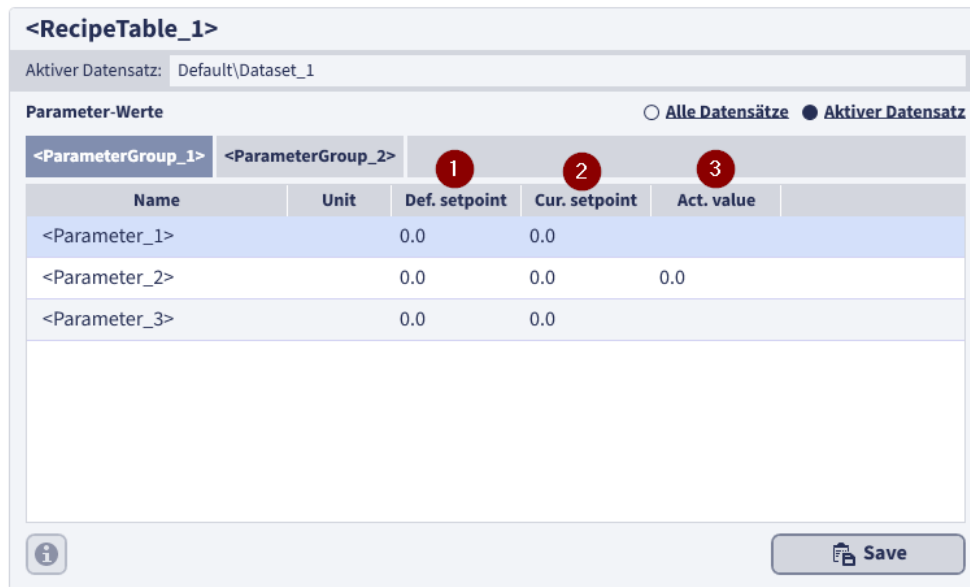
- Direct and flexible input at any point in the visualization
- Ideal for simple, intuitive operating concepts
- Process changes can be made incrementally without being saved immediately
- The current setpoint values (which are in the load datapoints) can be saved with the Recipe Table

Cons/Limitations:

- Less controllability of permissions: There is no central control over the datapoint change
- The "Adjust before activating" function is not useful in this model
- Reduced traceability – no automatic versioning or explicit activation of the changed values

Example: Saving changed process values:

After entering recipe parameters via input fields (which are linked with load datapoints) – for example in views with plant images – the current values can be saved directly in the "Active dataset" Recipe Table view. To do this, use the "Save current setpoints" button.



Name	Unit	1 Def. setpoint	2 Cur. setpoint	3 Act. value
<Parameter_1>		0.0	0.0	
<Parameter_2>		0.0	0.0	0.0
<Parameter_3>		0.0	0.0	

Column description (numbering according to figure):

1. Def. setpoint (defined setpoint): Shows the setpoint stored in the dataset
2. Act. setpoint: The current value of the store datapoint. Displays the current value of the linked store datapoint. This may have been changed after activating the dataset in another view via an input element
3. Actual value: Displays the current value of a linked sensor (optional)

When you click on the "Save current setpoints" button (in the example above, display text "Save"), a new dataset is saved. The existing defined target values are replaced by the currently set values of the store datapoints.

5.3.2.3 Specific properties

In addition to the general properties (see Chapter 4) that apply to all view elements, the specific properties of the Recipe Table are described here.

The Recipe Table contains the following parts, which can be configured in terms of appearance (e.g. color, font):

- Body: Background of the "Recipe table" element
- Label: Title of the Recipe Table
- Element label: Identifiers within the Recipe Table for e.g. collapsible boxes
- Dropdown List: Dropdown list selector
- Dropdown symbol: arrow symbol for collapsible boxes
- Navigation bar: View switcher
- Register: Tabs for Parameter Group or Steps

- Process highlighting: General Color for Process Highlighting
- Buttons: All action buttons below the table
- Button area: Bottom area that contains all buttons
- Symbols in Buttons: Icon within Buttons
- Active dataset: Header area with information of the active dataset
- Table
 - Table body: Table header or column header
 - Table Content: Cells within the table
 - Table background: Background color of the table
 - Row: Rows of the table
 - Row separator: Divider between two rows in the table

5.3.2.3.1 Configure columns and buttons

Columns and buttons can be configured separately for each view ("All datasets", "Active dataset") and state. Before setting properties for a particular view and state, the view and state must first be selected for the preview and configuration (see figure):


Select view for preview and configuration

'All datasets' view

☐ Display all datasets

☐ Display dataset


☐ Edit dataset

☒ Adjust dataset 

'Active dataset' view

☒ Display active dataset

☐ Edit active dataset

☐ Adjust active dataset 

Possible columns of the dataset list:

- Name / Version: Dataset name (or version if versioning enabled)
- Label (optional): Shows the additional dataset label (e.g. product name). Only available if the "Dataset label" feature has been activated in the base settings of the recipe type
- Modified on: Shows when the dataset was last modified
- Modified By: Displays the login name of the user who modified the dataset
- Comment: Shows the comment that was optionally entered when saving

Possible columns of the parameter list:

Depending on the view (and state) you set, not all columns are selectable.

- **Parameter name:** The multilingual display text of the parameter, as defined in the recipe type
- **Unit:** The unit of the parameter (e.g. "m", "°C", "bar")
- **Defined setpoint:** The setpoint stored in the dataset that is loaded when activation is activated
- **Adjustable parameter:** Specifies whether the parameter has been enabled for customization. Only visible if the "Adjust before activation" function is activated in the recipe type
- **Adjusted setpoint:** The setpoint entered during adjustment, if the parameter has been enabled for adjustment
- **Adjust minimum:** The configured minimum value limit for this parameter when adjusting. Only visible if "Adjust before activation" is activated
- **Adjust maximum:** The configured maximum value limit for this parameter when adjusting. Only visible if "Adjust before activation" is activated
- **Current setpoint:** The current value of the linked load datapoint (see Chapter 8.2)
- **Actual value:** The current value of the linked actuals datapoint (if any) – for example, a sensor value
- **Minimum:** The minimum value defined in the recipe type for this parameter
- **Maximum:** The maximum value defined in the recipe type for this parameter

Possible buttons:

The following buttons are available - depending on the view and status:

- **Display info:** Opens a dialog with additional information
 - For the parameter list: Shows the configured info text for the respective parameter
 - For dataset list: Displays the comment that was saved when the dataset was saved
- **Adjust dataset:** Switches to the "Adjust dataset" state to adjust adjustable parameters before activation
- **Edit dataset:** Switches to the "Edit dataset" state to change parameters and save the changes
- **Create dataset:** Opens a dialog for creating a new dataset
- **Delete dataset:** Deletes the currently selected dataset
- **Manage folders:** Opens a dialog for creating or deleting dataset folders. Only available if folder management has been activated in the base settings of the recipe type

- **Display dataset:** Displays the parameters of the selected dataset read-only (switches to "Display dataset" state)
- **Activate Dataset:** Activates the currently selected or adjusted dataset
- **Back:** Returns to the main "All datasets" or "Active dataset" view
- **Takeover actual value (Teach):** Transfers the current actual value to the "Def. setpoint" column – for example, to transfer the current axis position
- **Cancel:** Cancels the current operation and returns to the main view
- **Save dataset:** Saves the changes made to the dataset
- **Save current setpoints:** Takes the current values of the linked load datapoints as new setpoints and stores them in the dataset. In this way, setpoints can also be set in other views or directly via view elements outside the Recipe Table. In this case, the "Recipe Table" is only used to create, save and activate datasets – but usually not to enter the target values directly

Property group "Configure views":

Here, the desired columns and buttons are configured for all views and states of the Recipe Table.

Properties of a column:

- **Select column:** Determines whether the column is displayed
- **Column type:** Column description (read-only)
- **Display text (header):** Multilingual text in the column header
- **Horizontal alignment:** Alignment of the column title in the spade head
- **Horizontal alignment (Content):** Alignment of the cell contents in the column
- **Width type:** Specifies whether the column has a fixed width or is based on the available width, see below
- **Width value:**
 - **Fixed pixel width:** Specifies the width of the column, in pixels.
 - **Fill:** The remaining space available is distributed proportionally. The value determines how many parts of the remaining space this column should take up (e.g. "1" = single, "2" = twice as much as one with "1")

Properties of a button:

- **Select button:** Determines whether the button is displayed
- **Button type:** Button description (read-only)
- **Display text:** Multilingual text that appears on the button
- **Content (Table mode):** Determines whether only the text, only the symbol, or both are displayed in Table mode.

- **Content (Tile mode):** Determines whether only the text, only the symbol, or both are displayed in tile mode.

5.3.2.3.2 Configure further display options

The following properties are defined in the "Configuration: Display options" property group:

Display mode:

Specifies whether the Recipe Table should display the entries in rows or in tiles.

- **Table:** The list entries (dataset list or parameter list) are displayed in rows in a table with a table header
- **Tiles (optimized for smaller display width):** The list entries are displayed in tiles below each other – e.g. for smartphones
- **Automatic:** Depending on the width of the Recipe Table, it is displayed either as a table or in tile mode. This option must be selected for the Recipe Table if the layout should adapt to the specific display – depending on its width. (see 3.3)
 - **Minimum width for table (px):** Specifies the width at which Table Mode is used

The following figure shows the Recipe Table in tile mode.

The screenshot shows a configuration window titled "<RecipeTable_3>". At the top, there is a field for "Active dataset:" with the value "Dataset_1\V20". Below this, there are two radio buttons: "All datasets" (unselected) and "Active dataset" (selected). Under the heading "Parameter values", there are two tabs: "Group1" (selected) and "Group2". The main area displays two parameter tiles. The first tile is for "Action" and includes fields for "Name:", "Unit:", "Def. setpoint:" (set to "Mixing"), "Cur. setpoint:", and "Act. value:". The second tile is for "Temperature" and includes fields for "Name:", "Unit:" (set to "°C"), "Def. setpoint:" (set to "10.00"), and "Cur. setpoint:". At the bottom, there are two buttons: an information button (i) and an edit button (pencil icon).

Other settings:

Further view options must be defined here.

- Show folder selection list even when folder management is deactivated: Displays the folder drop-down box even if Folder Management is disabled in the recipe type
- Always switch to the "Active dataset" view after activation: Automatically switches to the "Active dataset" view after successful activation
- Automatically select group and step registers when process highlighting is active: When process highlighting is active, the parameter group and step are automatically preselected and displayed
- Show group tab even if there is only one parameter group: Displays the group tab even if only a single parameter group is defined in the recipe type
- Show step tab even if only one step is supported: Displays the step tab even if there is only one step in the dataset

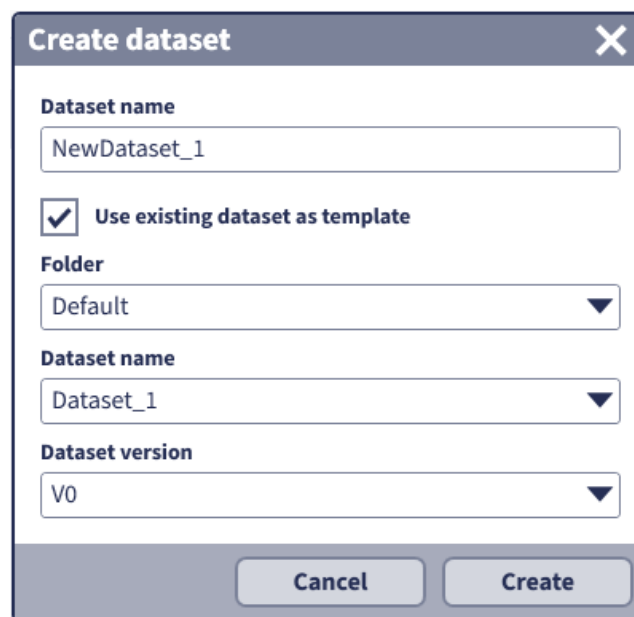
5.3.2.4 Integrated input dialogs

Some buttons in the Recipe Table open a dialog after clicking. In addition to simple confirmation dialogs – for example, to confirm the deletion of a selected dataset – there are also more extensive input dialogs with input fields, which are described below.

The display of the dialogs (e.g. background color of input fields) is based on the properties of the controls used in the appearance template (see Chapter 15.5 **Error! Reference source not found.**).

Create a new dataset:

Clicking on the "Create dataset" button opens a dialog in which a new dataset can be created. The new dataset is created either with the default values defined in the recipe type or on the basis of a selected dataset (see figure).



The "Create dataset" dialog box features a title bar with a close button (X). It contains the following fields and controls:

- Dataset name:** A text input field containing "NewDataset_1".
- Use existing dataset as template:** A checked checkbox.
- Folder:** A dropdown menu showing "Default".
- Dataset name:** A dropdown menu showing "Dataset_1".
- Dataset version:** A dropdown menu showing "V0".
- Buttons:** "Cancel" and "Create" buttons at the bottom right.

Create a new folder:

By clicking on "Manage folders", a selection dialog appears for selecting the folder action. After selecting "Create", the "Create folder" dialog appears for creating a new folder (see figure).



The "Folder Management" dialog box has a title bar with a close button (X) and contains three buttons:

- Create folder
- Edit folder permission
- Remove folder



The "Create folder" dialog box features a title bar with a close button (X) and contains the following fields and controls:

- Folder name:** A text input field containing "new_folder_1".
- Permission:** A dropdown menu showing "No restriction".
- Buttons:** "Cancel" and "Apply" buttons at the bottom right.

Edit folder permission:

This input dialog allows you to change the permission and is identical to the layout of the "Create folder" dialog.

Remove folder:

The currently selected folder in the Recipe Table is deleted.

5.3.3 User Management Elements

User management elements allow interaction with user and rights management directly in the visualization. This includes features such as logging in, logging out, user viewing, or

dynamically showing and hiding content based on user roles or rights. The elements make use of the user, role, and rights logic that is available in the "User Management" editor area (see Chapter 7).

All elements here refer to local user management, i.e. the users are stored locally on the HMI operator device (in the Unified-E Client). This also applies to Gateway communication via the Unified-E App Manager.

Users can already be predefined in the HMI project or explicitly created or edited in the "User Table" view element at runtime.

The following user management elements are available:

- **User Table:** Used to manage users directly in the HMI app on the HMI device. Allows users to be created, edited, and deleted at runtime
- **Authentication Button:** Shows the current login status both textually and via an icon image. In addition, the actions "Login", "Logout", "Change user" and "Change password" can be carried out
- **User Display:** Shows the name of the currently logged in user, if a user is logged in. When clicked, the login actions can also be carried out as with the Authentication Button
- **User Role Display:** Shows the user role of the currently logged in user, if a user is logged in. When clicked, the registration actions are also available here

Disable local user management:

If local user management is deactivated (e.g. explicitly in the HMI project, see Chapter 7.1.3) or in the case of app registration via the App Manager, then the view elements of the user management are deactivated.

5.3.3.1 User Table

The User Table shows all users registered in the HMI app and allows users to be added, deleted, or edited at runtime. It can be used by authorized users to create new users directly on the HMI device or to manage existing users.

Examples:



Login Name	User role	Log out (min)	Type	Comment
User1	Administrator	10	Predefined	Comment
User2	Operator	---	Defined in runtime	Comment
User3	Engineer	30	Defined in runtime	Comment
User4	Administrator	10	Defined in runtime	Comment
User5	Administrator	10	Defined in runtime	Comment

+ New Delete Edit

<UserTable_1>

Login Name: User1
User role: Administrator
Log out (min): 10
Type: Predefined
Comment: Comment

Login Name: User2
User role: Operator
Log out (min): ---
Type: Defined in runtime

+ Delete Edit

5.3.3.1.1 Specific properties

In addition to the general properties (see Chapter 4) that apply to all view elements, the specific properties of the User Table are described here.

Property group "Configuration: Columns":

This property group specifies which columns should be displayed in the User Table and how they are formatted. The appearance of each column can be customized.

- **Column type** (column selectable via checkbox in first column):
 - Login name: Shows the unique username (this column is mandatory and cannot be disabled)
 - User role: Displayed role of the user (e.g. "Administrator")
 - Log out (min): Defined automatic logout time in minutes when inactive
 - Type: Source or classification of the user (e.g. local or external). Possible values are "predefined" or "defined at runtime"
 - Last login: Time of last successful login
 - Last logout: Time of the last logout
 - Created on: Date and time of user creation
 - Comment: Optional free text field for description or internal information

Other column properties:

- **Display text (header):** Heading that appears in the table for each column. This can be customized regardless of the column type
- **Horizontal Alignment (header):** Specifies how the column header is aligned textually—for example, "Left," "Center," or "Right"
- **Horizontal Alignment Content:** Controls the alignment of the content within the cells for this column
- **Width type:**

- Fixed pixel length: Column has a fixed width in pixels
- Fill: Column takes up the remaining available space (only useful for one column)
- Width value:
 - Fixed width: Specifies the width of the column, in pixels.
 - For "Fill": The remaining space available is distributed proportionally. The value determines how many parts of the remaining space this column should take up (e.g. "1" = single, "2" = twice as much as one with "1")

Property group "Configuration: Buttons":

This table specifies which buttons should be displayed within the User Table and how they are displayed in each case – separately for table mode and tile mode.

Column description of the configuration table:

- Button type: Type of button. The following types are available:
 - New: Opens the dialog for creating a new user
 - Delete: Deletes the currently selected user after confirmation
 - Edit: Opens the edit dialog for the selected user
- Display text: The text that appears on the button. This can be adapted and defined in several languages
- Content (table mode): Determines how the button is displayed when tabular
 - Symbol only: Only the symbol is displayed
 - Text only: Only the display text is displayed
 - Text and symbol: Both text and symbol are displayed
- Content (tile mode): Corresponding display of the button in the tile layout (e.g. on smartphones):
 - Icon Only: Only the icon is displayed
 - Text only: Only the display text is displayed
 - Text and symbol: Both are displayed

Property group "Configuration: Display options":

Specifies whether the Message Table should display the entries in rows or in tiles.

- Table: The list entries are displayed in rows in a table with a table header
- Tiles (optimized for smaller display width): The list entries are displayed in tiles below each other – e.g. for smartphones

- **Automatic:** Depending on the width (at runtime) of the User Table, it is displayed as a table or in tile mode. This option is to be selected if the Recipe Table is configured in a view that is laid out for the display (see 3.3)
 - Minimum width for table (px): Specifies the width at which Table Mode is used

5.3.3.1.2 Integrated input dialogs

All buttons in the User Table open a dialog after clicking. In addition to the simple confirmation dialog when deleting a user, an input dialog appears when clicking on "New" and "Delete".

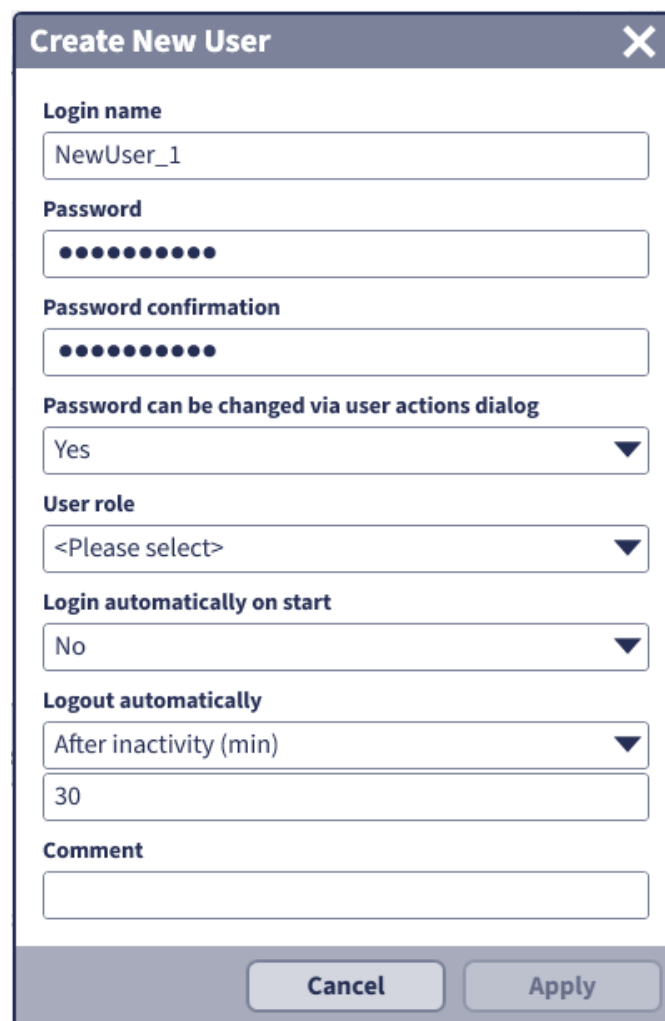
The display of the dialogs (e.g. background color of input fields) is based on the properties of the controls used in the appearance template (see Chapter **Error! Reference source not found.**).

Create a new user:

The "Create new user" dialog is displayed in the HMI app at runtime when an authorized user selects the "New" button from the User Table. In this dialog, a new user can be created on the HMI device.

Input fields in the dialog:

- **Login name:** Username used to log in. Must be unambiguous
- **Password:** Initial password for the user. Entered and saved at runtime
- **Password confirmation:** Repeat the password to avoid input errors
- **Password can be changed via user actions dialog:** Specifies whether the user is allowed to change his password later
- **User role:** Select a role from the user roles predefined in the project: Determines the available views and permissions
- **Log in automatically on start:** If enabled, this user will be automatically logged in on the current device when the app starts (default user)
- **Log out automatically:** Determines whether the user is automatically logged out after a certain period of inactivity. If the option is activated, an input field for the time in minutes will appear
- **Comment:** Optional free text field for hints or additional information about the user



The image shows a 'Create New User' dialog box with a close button (X) in the top right corner. It contains the following fields and controls:

- Login name:** A text input field containing 'NewUser_1'.
- Password:** A password input field with 10 dots.
- Password confirmation:** A password input field with 10 dots.
- Password can be changed via user actions dialog:** A dropdown menu with 'Yes' selected.
- User role:** A dropdown menu with '<Please select>' selected.
- Login automatically on start:** A dropdown menu with 'No' selected.
- Logout automatically:** A dropdown menu with 'After inactivity (min)' selected, followed by a text input field containing '30'.
- Comment:** A text input field.
- Buttons:** 'Cancel' and 'Apply' buttons at the bottom.

Edit selected user:

This dialog is displayed when the Edit button is selected in the User Table. It is used to edit an existing user. The structure is the same as the "Create new user" dialog, with the following exceptions:

- **Login Name:** The username is displayed, but cannot be changed

All other fields (e.g. password, role, comment) can be adjusted as when you create them. Changes are saved after confirmation with Apply and are immediately active.

5.3.3.2 Authentication Button

The Authentication Button shows the current login status and enables central user actions within the HMI app. The display optionally combines text and symbolic images and can be adapted to the design specifications.

When the button is clicked, the user management distinguishes between two states:

- Not logged in: The "Please log in" dialog appears, where you must enter your username and password to log in.
- Logged in: The "Select User Action" dialog appears, where the following actions are available:
 - Change password
 - Change user
 - Log off

Examples:



5.3.3.2.1 Specific properties

In addition to the general properties (see Chapter 4) that apply to all view elements, the specific properties of the Authentication Button are described here.

Property group "Configuration: Label":

This property group defines what text is displayed on the button, depending on whether a user is logged in or not.

- **Allow line breaks:** Enables automatic line break if the text is longer than the button
- **If logged out:** Determines which text is displayed in the logged out state
 - No text: No text is displayed
 - Display text (English): Freely definable text for the element (e.g. "Login"). The text can be maintained in several languages
- **If logged in:** Determines the text display when the user is active
 - Username: The currently logged in username is displayed
 - Username and user role: Displays username and assigned role
 - No text: No text is displayed
 - Display text (English): Freely definable text for the registered state

Property group: "Configuration: Symbol":

This group determines whether and which icon is displayed depending on the sign-in status.

- **If logged out**
 - Without symbol: No symbol is displayed

- Show symbol: Activates the display of an icon for the logged out state- The symbol is an image from the Images editor (see Chapter 15.3)
- If logged in
 - Without symbol: No icon is displayed
 - Show symbol: Enables the display of an icon for the logged in state. Symbol selection analogous to the logged-out state

Property group "Configuration: Preview in editor":

Here you can define the Authentication Button state in the graphical editor. This property does not affect the runtime of the HMI app.

Property group "User Actions: Click Actions":

This property group determines what action is performed when the Authentication Button is clicked, depending on whether a user is currently logged in or not.

- Click behavior, if logged out: Defines the behavior of the button when the user is not logged in
 - No action: One click has no effect
 - Logout: Opens the login dialog to enter your username and password
- Click behavior, if logged in: Defines the behavior when logged in user
 - No action: One click has no effect
 - Show user action selector: Opens the dialog with the following choices:
 - Change password
 - Change user
 - Log off
 - Logout: Logs out the currently logged in user directly, without prompting

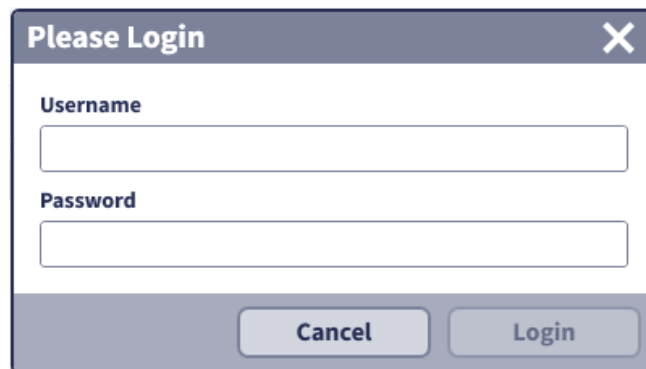
5.3.3.2.2 Integrated input dialogs

The dialogs associated with the Authentication Button are described below. The display of the dialogs (e.g. background color of input fields) is based on the properties of the controls used in the appearance template (see Chapter 15.5).

User Login:

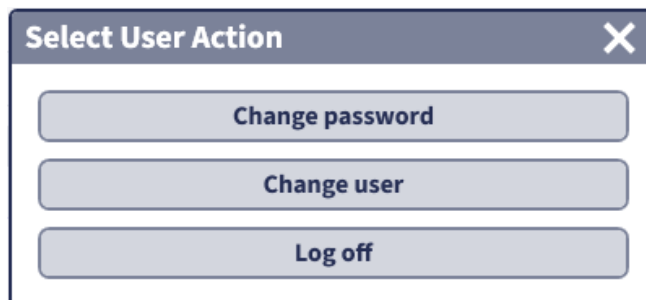
This dialog appears for the following actions:

- Clicking on the Authentication Button in the unlogged state
- Click on the Authentication Button in the logged in state and select the "Switch user" action in the "Select user action" dialog

A dialog box titled "Please Login" with a close button (X) in the top right corner. It contains two input fields: "Username" and "Password". At the bottom, there are two buttons: "Cancel" and "Login".

Select user action:

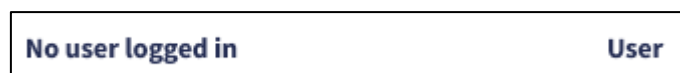
In this dialog, the specific action can be selected when clicking the Authentication Button in the logged in state.

A dialog box titled "Select User Action" with a close button (X) in the top right corner. It contains three buttons stacked vertically: "Change password", "Change user", and "Log off".

5.3.3.3 User Display

The User Display shows the name of the currently logged in user. If no user is logged in, the element remains empty or optionally shows freely definable text. Clicking on the element – as with the Authentication Button – can open a dialog with user actions, e.g. to log out or switch users. The display updates automatically every time you log in or log out.

Examples:

A rectangular box representing the User Display. It contains the text "No user logged in" on the left and the word "User" on the right.

Property group "Label":

This property group controls what text is displayed in the User Display element, depending on the sign-in status.

- **Allow line breaks:** Enables automatic line breaks if the text is longer than the available width
- **When logged out:** Determines what is displayed when no user is logged in
 - **No text:** No text is displayed

- Display text: Free text such as "No user logged in". The text can be stored in several languages
- When logged in: The username of the currently logged in user is automatically displayed. The text is not configurable, but results from the registration

Other property groups:

The properties in the "User Actions: Click Actions" and "Configuration: Preview in editor" property groups can be configured as with the Authentication Button (Chapter 5.3.3.2.1).

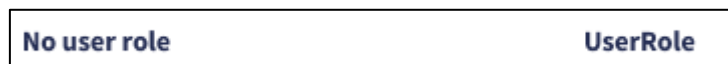
Integrated input dialogs:

The dialogs that are opened when clicked are identical to the input dialogs of the Authentication Button (see Chapter 5.3.3.2.2).

5.3.3.4 User Role Display

The User Roles display shows the role of the currently logged in user. If no user is logged in, the element remains empty or optionally shows freely definable text. Clicking on the element – as with the Authentication Button – can open a dialog with user actions, e.g. to log out or switch users. The display is automatically updated as soon as the user's status changes.

Examples:



Property group "Configuration: Label":

This property group controls what text is displayed in the User Role Display element, depending on the sign-in status.

- Allow line breaks: Enables automatic line breaks if the text is longer than the available width
- If logged out: Determines what is displayed when no user is logged in
 - No text: No text is displayed
 - Display text (English): Free text such as "No user role available". The text can be stored in several languages
- If logged in: The user role of the currently logged in user is automatically displayed. The text is not configurable, but results from the registration

Other property groups:

The properties in the "User Actions: Click Actions" and "Configuration: Preview in editor" property groups can be configured the same way as the Authentication Button (Chapter 5.3.3.2.1).

Integrated input dialogs:

The dialogs that are opened when clicked are identical to the input dialogs of the Authentication Button (see Chapter 5.3.3.2.2).

5.3.4 System Button Elements

System buttons are view elements that perform predefined actions when clicked. They look similar to regular buttons, but they don't have the ability to configure a custom action. The range of functions varies depending on the platform, as some system functions are only available on Windows.

The following system buttons are available:

- **Language Switcher:**
The active app language can be switched here at runtime.
- **Exit Button:**
Exits the Unified-E Client – on Windows, the Windows application. This feature is only supported for Windows.
- **Shutdown Button:**
Clicking will shut down the computer. Again, this feature is only available for the Windows platform.

5.3.4.1 Language Switcher

The Language Switcher is a special system button that allows users to switch between the languages configured in the HMI project at runtime. The display of this button can be customized. Adding app languages is described in the chapter 12 described in detail.

Examples:



Property group "Configuration: Label":

This property group determines whether and how a label is displayed on the button.

- **Allow line breaks:** Enable this option if you want the text within the button to wrap automatically on multiple lines
- **Display label:** Specifies what type of text you want to display on the button:
 - Name of the current language: e.g. "German" or "English"
 - Abbreviation of the current language: e.g. "DE" or "EN"
 - No text: Only an symbol (if enabled) is displayed

- Display text: You can specify a multilingual text that will be displayed

Property group "Configuration: Symbol":

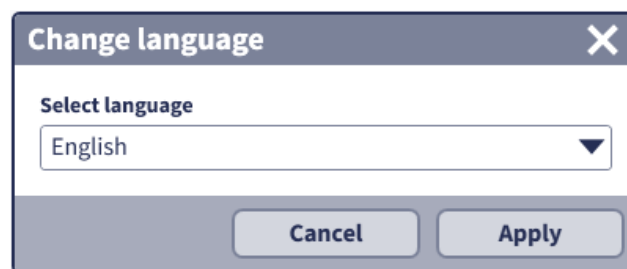
This property group controls the display of a symbol next to the text.

- **Without symbol:** No symbol is displayed
- **Show symbol:** Activates the display of an symbol - The symbol is an image from the Images editor (see Chapter 15.3)

Language selection dialog:

The language selection dialog appears when you click on the Language Switcher and allows you to set an app language configured in the HMI project.

The appearance of the dialog (e.g. background color of input fields) is based on the properties of the view elements which are defined in the Appearance Template editor (see Chapter 15.5).

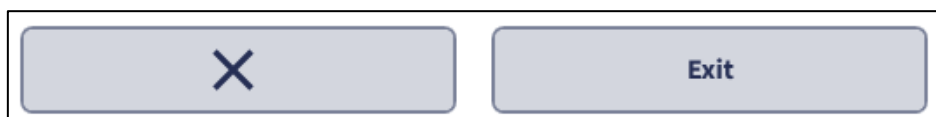


5.3.4.2 Exit Button

The Exit Button is a system button that terminates the Unified-E Client within a view. On the Windows platform, the corresponding Windows application is closed. This feature is not available on other platforms.

This button is particularly suitable for touchscreen operator devices where there is no physical "close" button available in full-screen mode.

Examples:



Property group "Configuration: Label":

This property group determines whether and how a label is displayed on the button.

- **Allow line breaks:** Enable this option if you want the text within the button to wrap automatically on multiple lines

- **Display label:** Here you can specify whether or which text should be displayed:
 - **No text:** The button is displayed without text
 - **Display text:** You can specify a multilingual text that will be displayed

Property group "Configuration: Symbol":

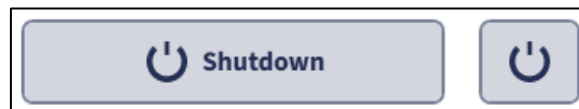
This determines whether or which symbol is displayed on the button.

- **Without symbol:** No icon is displayed
- **Show symbol:** Activates the display of an icon - The icon is an image from the Images editor (see Chapter 15.3)

5.3.4.3 Shutdown Button

The Shutdown Button shuts down the computer by clicking. This feature is only available on the Windows platform. The configuration of the properties is analogous to the configuration at the Exit Button.

Examples:

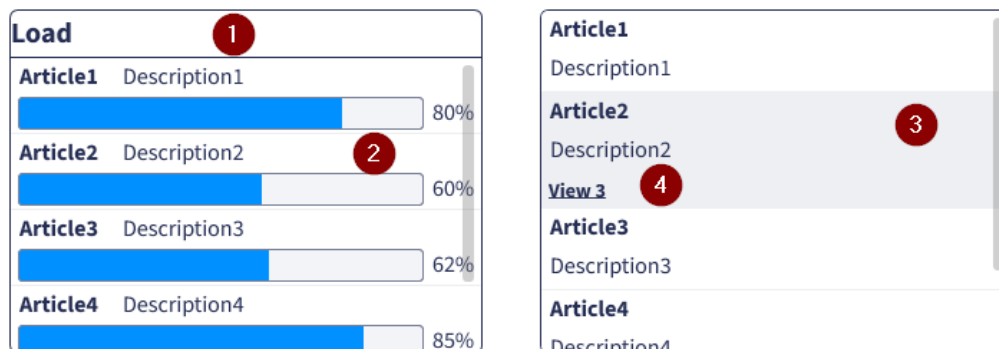


5.3.5 List Panel

The "List Panel" view element is used to display tabular data in list form. It enables the visualization of table values from a table datapoint, for example the result of an SQL query or a data structure provided by the WebHttp JSON adapter.

This element is particularly suitable when structured data rows are to be displayed in a compact and dynamically generated representation. The display is not made by a predefined number of display elements, but by an automatically repeated template for list entries.

Examples:



Element description (numbering according to figure):

1. Label (optional): Optional label for the List Panel.
2. List entry (unselected): Displays a list entry (tile) that corresponds to a row of the table from the table datapoint.
3. List entry (selected): Displays a list entry after it has been selected.
4. Details View Navigation (optional): When selected, a View Navigation can optionally be displayed. The details view can evaluate datapoints that were set when selecting the list entry and thus display selection-dependent content (see below). This part is referred to in the "Details View Navigation" property group.

5.3.5.1.1 Definition of terms

Table datapoint:

A datapoint whose value contains a structured table with multiple rows and columns (see Chapter 2.2.5). Each row of this table is represented as a separate list item.

List entry:

Corresponds to a single row in the table value. The entry can be displayed as a classic line or as a tile, depending on the layout chosen. The same template is used for each list item at runtime.

List entry template:

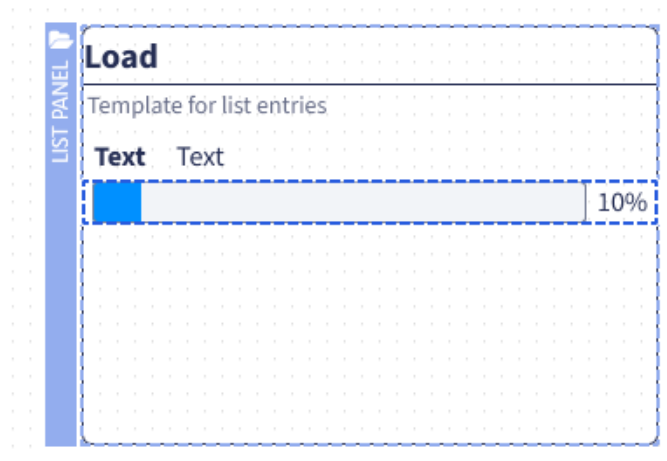
A structure defined graphically in the editor that defines the layout and appearance of an individual list item. This template contains display elements, each representing a column value. The template acts like a container in which layout types such as "Grid" or "Flow" can be used (see Chapter 3.2.1).

Display element per column (cell):

To display a value from a specific column (cell), a view element from the group "Display" is used, e.g. "Numeric Display" or "Bar Display". In contrast to the usual data binding via a specific datapoint, the assignment here is done via the column index (starting at 0).

5.3.5.1.2 Configure List Entry Template

Unlike most view elements, the "List Panel" view element is not displayed in the graphical editor as it appears later at runtime. In the editor, only the template for a single list entry – i.e. a row of the table datapoint – is configured. At runtime, several entries are then displayed. The configured template is automatically reused for each row in the table value.



Minimal workflow when configuring the template:

1. Select table datapoint:
In the "Configuration: List" property group, select the table datapoint that will serve as the data source at runtime.
2. Set Layout type for list entry template:
In the "Common: Layout" property group, the layout for the display elements is defined, each of which represents a column value (cell) of the table row. The layout types "Flow" and "Grid" are available.
3. Add display elements for column values:
By default, a "Text Display" display element is already included. Other view elements from the "Display" group can be added. All elements are positioned on the placeholder (or background) within the template.
4. Use column index instead of a datapoint:
All view elements within the template do not specify a direct datapoint, but a column index (e.g., "0", "1", etc.). The column index refers to the respective column of the table row.

Property group: "Configuration: List content":

This group defines the data source of the list and limits the number of entries that are initially displayed.

- **Table datapoint:** Selection of a datapoint of type "Data table" (e. g., from SQL adapter or WebHttp-JSON adapter) whose rows are displayed as list entries. Each row corresponds to an entry according to the configured template

- **Max number of entries at start:** Specifies the maximum number of entries that will be loaded and displayed at runtime during initialization. This limitation can help optimize performance for large amounts of data. The default value is "25". To load the next rows, the HMI app operator must click on "X more entries available" to load the next list entries

Property group: "Configuration: Selection":

This group defines how user interactions with a list item are processed - for example, to pass a selected value or to open a detail view.

- **Entry ID (Column index):** Specifies the column index (starting at 0) whose cell value is used as the ID when selecting a list item. This value is written to the target datapoint configured below and can be used for further processing or navigation
- **Datapoint for selection transfer (optional):** Target datapoint into which the selected value (according to the column index defined above) is transferred when an entry is selected
- **Details view (optional):** Select a view that opens when you click on a list item. For example, this view can display detailed information about the selected row. To transfer the selected value, use the datapoint defined above for selection transfer. The link for the detailed navigation is displayed in the list entry as soon as it has been selected

Note: The SQL adapter supports so-called placeholder datapoints (see the SQL adapter manual). This allows you to define detailed datapoints for the selected list entry, for example, which always have exactly the value of the selection. To do this, the placeholder datapoint must be used for both the detail datapoints in the address in the WHERE condition and must be selected as the datapoint under "Datapoint for selection transfer".

Datapoint for selection transfer in SQL queries:

The SQL adapter supports so-called placeholder datapoints (see the SQL adapter manual). This allows you to define datapoints that always reflect the value of the currently selected list entry.

The same placeholder datapoint must be used in two places:

- in the addressing of the detail datapoints (e.g., in the WHERE clause of the SQL query)
- in the property "Datapoint for selection transfer" of the simple list

5.3.6 Chart Panel

The "Chart Panel" view element can be used to graphically visualize the values of table datapoints. Each data series (e.g. Line) is assigned a table datapoint and serves as the data source. For the time chart, you can also select one of the configured chart recordings as a datapoint (see Chapter 10.1).

Unified-E supports four chart types:

- **Time Chart:**
Shows the change in one or more quantities over time. This is particularly useful if measured values are to be visualized over time. The time value of the X-axis can be absolute or starting from the start of the process in seconds.
- **Line Chart:**
Used to show trends across different categories. The X-axis shows discrete categories, while the Y-axis indicates the values. Each data series is displayed as a line of dots. The categories are already fixed in the HMI project, while the Y-axis values come from the table datapoints of the individual data series.
- **Pie Chart:**
Illustrates portions of a whole. Each category is represented as a circular segment, the size of which corresponds to the relative share of the total. The legend gives both absolute values and percentages. The categories are already defined in the HMI project, the individual values come from the table datapoint of the single data series.
- **Column chart:**
Displays values per category as vertical bars. Each data series is visualized by its own bar group, which allows values of several datapoints to be directly compared across different categories. The categories are already fixed in the HMI project, while the Y-axis values come from the table datapoints of the individual data series.

Chart Panel vs. Chart Data Series:

A Chart Panel visualizes the data of one or more data series, which are provided in tabular form – depending on the chart type. The values (e.g. measured values) of a data series are used for the Y-axis, whose properties are configurable.

A Chart Data Series is – as described in chapter 5.3.7 – linked to a table datapoint. The datapoint is read out cyclically (every second) so that the chart is updated regularly.

The "Chart Panel" view element thus acts as a container for "Chart Data Series" objects, which can be added or removed in engineering.

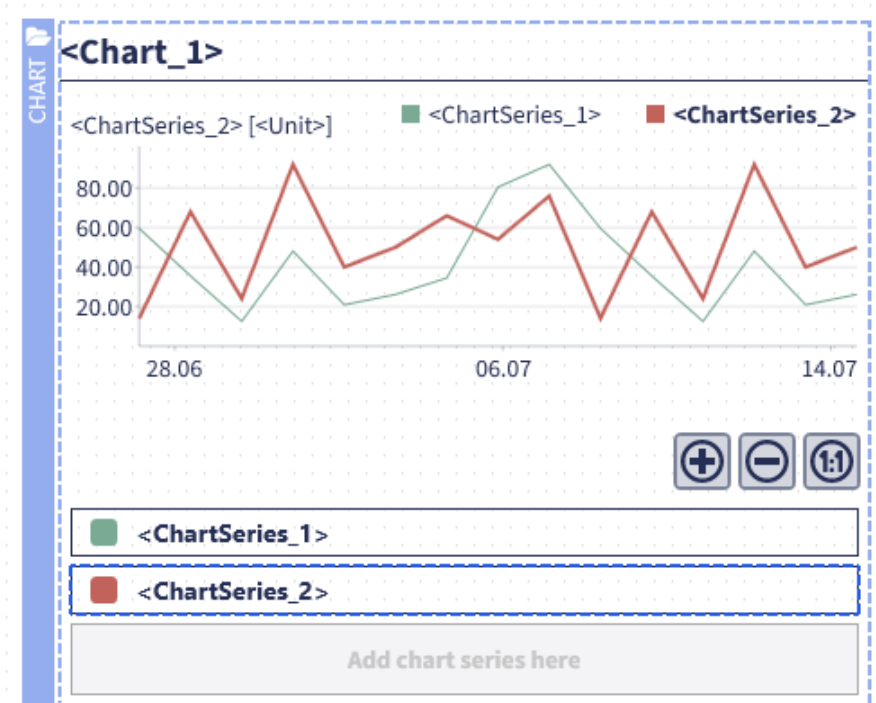
Y-axis chart types:

In the Time Chart, a data series can be selected in the legend. The Y-axis is then adjusted according to this selection. The Line Chart and the Column chart use the same Y-axis for all data series. The pie chart always uses exactly one data series; their values are used to represent the circle segments.

5.3.6.1 Add chart data series

Chart data series can be added by dragging the "Chart Data Series" element from the Views Elements area to the placeholder in the chart ("Insert Chart Data Series Here"). Alternatively, a chart data series can also be pasted into the placeholder via the clipboard.

Note: Placeholders are only visible if the editor option "Show layout aids" has not been deactivated – see Chapter 3.1.



5.3.6.1.1 Configure common Chart Panel properties

In addition to the general properties (see Chapter 4), which applies to all view elements, there is the following property group for a Chart Panel view element, depending on the chart type set.

Property group "Configuration: Chart":

Here, the desired chart type is set. Depending on the chart type you set, not all property groups are available.

For the time chart, you can still configure in this property group:

- **Time axis:** Determines the type of time axis on the X axis:
 - **Absolute date/time values:** The timeline shows real timestamps (e.g., 7/3/2025 2:30 PM)
 - **Numeric time values:** The time axis starts at 0 seconds and displays the time relative to the start time of the process
- **Hide zoom buttons:** If enabled, no zoom buttons will appear in the time graph

Property group "Configuration: Common Y-axis":

This property group is available on Bar and Line Charts and defines the properties of the Y-axis. This Y-axis is shared across all configured data series, regardless of which one is selected in the legend.

- **Static configuration / Dynamic configuration with datapoints:** Determines whether the Y-axis is defined with fixed values in the HMI project or whether the axis properties are controlled by datapoints at runtime
 - Static configuration: The properties below are defined in the HMI project
 - Dynamic configuration with datapoints: Axis properties are dynamically determined at runtime using linked datapoints
- **Display text:** Multilingual text of the Y-axis displayed on the graph (e.g. "Temperature", "RPM")
- **Unit:** The unit of measurement represented in the axis title (for example, "°C", "bar", "%")
- **Number format:** Formatting of the numerical values along the Y-axis (see also chapter 4.5). Example:
 - 0.00 for two decimal places
 - 0 for integers
 - User-defined via the "..." button
- **Minimum:** The smallest value to be displayed on the Y-axis. If all values of the data series are greater than this value, the axis is started at this minimum. If no value is specified or the data series contain smaller values, the minimum adjusts automatically
- **Maximum:** The largest value to be displayed on the Y-axis. If all values of the data series are less than this value, the axis is bounded at this maximum. If no value is specified or if the data series contain larger values, the maximum adjusts automatically

Property group "Configuration: Categories":

For the chart types "Column Chart" and "Line Chart", the individual categories can be seen in the X-axis, and the value (e.g. measured value) can be seen in the Y-axis. The categories can already be predefined in the HMI project during the engineering process or they can alternatively be supplied with the values from the table datapoint.

Categories are also used for pie charts, each pie segment is assigned to a category.

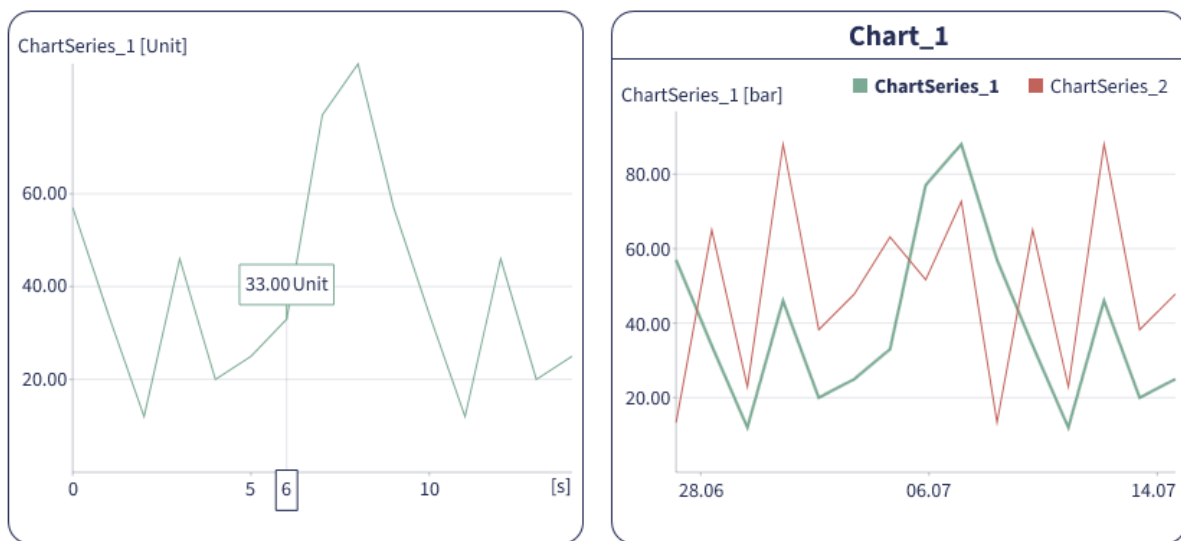
- **Static Configuration / Dynamic configuration with datapoints:** Determines whether the categories are fixed (multilingual display text) or are dynamically determined by the table datapoint at runtime of the HMI app
 - Static configuration: Headings fixed in the categories table (see below)
 - Dynamic configuration with datapoints: Categories must be delivered in the table datapoints of the chart data series (first table column is interpreted as a category column)
- **Categories table:** Only available in case of static configuration
 - Name: Unique name of the category (only for engineering)

- Display Text: Category name used in the chart at run time
- Color (Pie chart only): Specifies the color for the corresponding pie segment

5.3.6.1.2 Configure Time Chart properties

The Time Chart displays recording values (e.g. measured values) over time. The X axis can be represented with absolute values (e.g. in seconds) or with date/time values.

Examples:



Runtime controls (see image above):

- Value selector: Clicking on a curve point displays the original value and the associated time
- "+" / "-" / "1:1" buttons: After clicking with the value selector, you can zoom in on the surroundings of the selected point or deactivate the zoom again with the "1:1" button

Property group "Configuration: Chart":

This is where the X-axis is configured.

- Static configuration / Dynamic configuration with datapoints: Specifies whether the X-axis is defined with fixed values in the HMI project or whether the axis properties are controlled by datapoints at runtime
 - Static configuration: The properties below are defined in the HMI project
 - Dynamic configuration with datapoints: Axis properties are dynamically determined at runtime using linked datapoints
- Min. range (hours): The minimum time period to be displayed. For a timeline with absolute date/time values, the period must be entered in hours

- **Max range (hours):** The maximum time period to be displayed. For a timeline with absolute date/time values, the period must be entered in hours

5.3.6.1.3 Configure Line Chart properties

The Line Chart is a chart with X and Y axes. The categories are displayed on the X-axis, and the corresponding values from the individual data series are displayed on the Y-axis.

The common Y-axis and the categories for the X-axis labeling are described in Chapter 5.3.6.1.1.

Example:



For each line, a new "Chart Data Series" view element must be added to the Line Chart.

Provide chart data:

The data for a chart data series is provided by the linked table datapoint.

Basically, the table contains one row per category value. The first line corresponds to the first category, the second line to the second category, and so on. The first category is shown on the far left of the diagram, the last on the far right.

- If categories are fixed in the HMI project:
 - 1st column: Value of the category
- If the categories are dynamically defined (see Categories property):

- Column 1: Category display text for the X-axis (this must be identical in all data series)
- Column 2: Value of the category

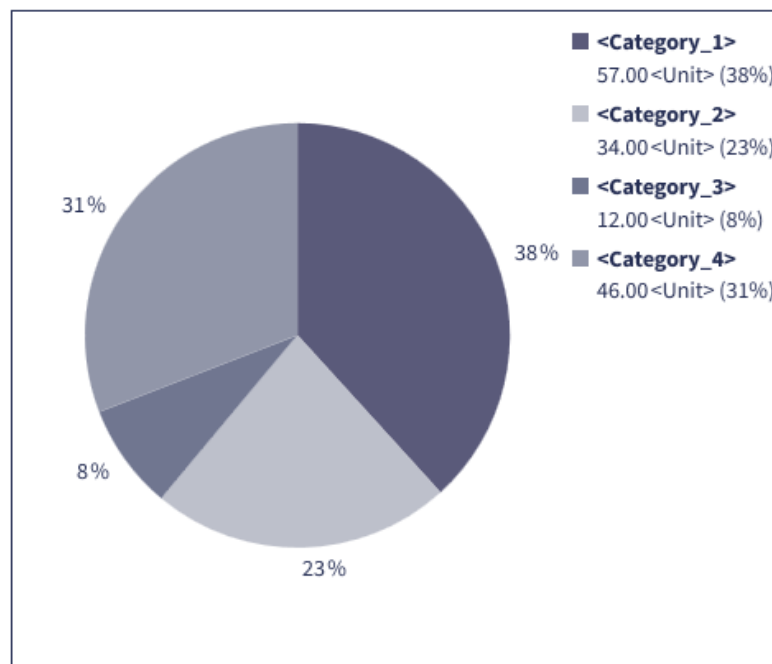
5.3.6.1.4 Configure Pie Chart properties

The pie chart represents the proportions of multiple values as segments of a pie. Each data value of a data series corresponds to a pie segment, the size of which is proportional to the total value of all segments. The sum of all values thus results in 100% of the circle.

For configuration, exactly one Chart Data Series is added to the chart. The categories are used to label the segments, the values determine their size.

The configuration of the categories for the segment labels is described in the Chapter 5.3.6.1.1.

Example:



Provide chart data:

The Pie Chart always uses exactly one fixed Chart Data Series. The chart data is supplied by the linked table datapoint of the "Chart Data Series" element. Each table row corresponds to a pie segment.

- If categories are fixed in the HMI project:
 - Column 1: Value of the category
- If the categories are dynamically defined (see Categories property):

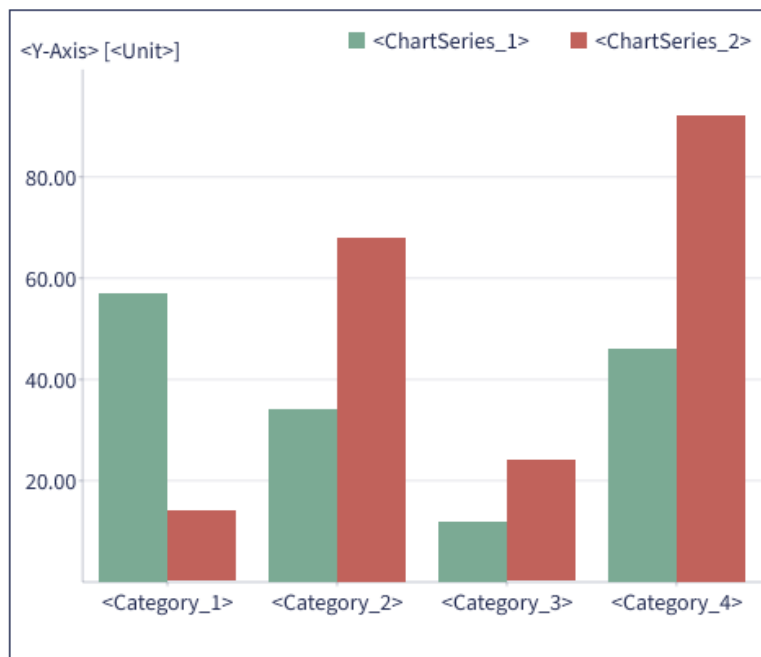
- Column 1: Display text for category
- Column 2: Value of the category
- Column 3: Circle segment color: The expected color format is "RRGGBB", e.g. "FF0000" for red

5.3.6.1.5 Configure Column Chart properties

The Column Chart represents values in the form of vertical bars. The categories are displayed on the X-axis, the corresponding values on the Y-axis. The representation thus corresponds to the line diagram in its axis orientation, only the graphical representation of the data differs.

The configuration of the common Y-axis and the categories on the X-axis is described in chapter 5.3.6.1.1.

Example:



Provide chart data:

For each data series, a new "Chart Data Series" element must be added to the Column chart. The data is provided via the linked table datapoint.

The table contains one row per category value. The first row corresponds to the first category (on the left side of the diagram), the second row to the second category, and so on.

- If categories are fixed in the HMI project:
 - Column 1: Value of the category
- If the categories are dynamically defined (see Categories property):

- Column 1: Display text of category
- Column 2: Value of the category

5.3.7 Chart Data Series

The "Chart Data Series" element is added to the Chart view element to represent a data series in the chart. This is where the table datapoint is assigned, which is used to represent the data series or, in the case of a pie chart, the segments. The chart data series is displayed depending on the parent chart type (e.g., line, columns, pie chart).

With pie charts, exactly one "chart data series" is fixed and does not have to be added manually.

Select a Chart Data Series:

In the case of the time chart, a specific chart data series can be selected at runtime via the legend. The Y-axis is then adjusted according to the current selection.

In the graphical editor, the chart data series is selected by double-clicking on the "Chart data series" element in order to be able to configure its properties in the Properties pane – see also Chapter 5.3.6.

Row limitation for the table datapoint:

If a table datapoint contains more than 5000 rows (e.g. in the case of a chart recording over time – see Chapter 10.1), only the first 5000 rows are delivered by default from the endpoint for performance reasons. At chart recordings, the most recent 5000 entries are taken into account if there are more than 5000 entries.

If the HMI app communicates with the App Manager in a firewall-friendly manner (Pro license), the maximum number of entries transferred is 1000.

General property group "Configuration: Table of values":

Here you have to select the table datapoint. This provides the values for the data series that are used to represent it in the chart. The structure of the table is different depending on the chart type of the chart (chart data series container). The table structure is described in detail above for the specific chart types.

5.3.7.1 Specific data series properties for time chart

The following property groups are available in the Chart Data Series in the Time Chart.

Property group "Configuration: Table of values":

- **Recording or datapoint (Table):** Here you can either use a configured chart recording (Chapter 10.1) or a table datapoint of an endpoint

- **Time zone of time values:** Select the universal time zone if the time values are delivered as UTC or if curve recordings of the Unified-E App Manager are used as the source. The delivered time values are then converted to the local time zone of the operator device for display

Property group "Configuration: Display options":

- **Static configuration / Dynamic configuration with datapoints:** Determines whether the color is defined in the HMI project or controlled by a datapoint at runtime
 - **Static configuration:** The color is determined in the HMI project
 - **Dynamic configuration with datapoints:** The color is determined at runtime via a configured datapoint with data type "Text". The expected color format is "RRGGBB", e.g. "FF0000" for red
- **Color:** Determines the color of the line in the time chart. The type of color assignment depends on the mode selected above

Property group "Configuration: Y-axis":

Unlike the line and Column chart, the Y-axis in the time chart is configured per chart data series because it does not use a common Y-axis. The configuration is analogous to the common Y-axis (see Chapter 5.3.6.1.1).

Property group "Visibility: Visibility":

The data series can be shown or hidden in the chart at runtime. The configuration is analogous to the visibility control for view elements (see Chapter 4.1.10).

5.3.7.2 Specific chart data series properties for Line and Column Chart

For Line and Column charts, the color and visibility can be configured for each chart data series – analogous to the configuration in the time chart (see Chapter 5.3.7.1):

- The color of the line or bar can either be defined in the project or controlled at runtime via a datapoint with a color value in the format "RRGGBB" (e.g. "FF0000" for red).
- The visibility of the data series can be dynamically controlled at runtime, analogous to the visibility configuration for view elements (see Chapter 4.1.10).

5.3.7.3 Specific Data Series Properties for Pie Chart

The pie chart always has exactly one chart data series, which is configured as follows:

Property group "Configuration: Display options":

This is where general display options are configured for displaying the values:

- **Static configuration / Dynamic configuration with datapoints:** Determines whether the properties are defined with fixed values in the HMI project or controlled by datapoints at runtime
 - Static configuration: The properties below are defined in the HMI project
 - Dynamic configuration with datapoints: Display options are determined at runtime based on linked datapoints
- **Unit:** The unit of measurement that is displayed with the value (for example, "°C", "bar", "%") when the segment is labeled. The unit appears after the numerical values
- **Number format:** Formatting of the numerical values for the segment labels (see also chapter 4.5). Example:
 - 0.00 for two decimal places
 - 0 for integers
 - User-defined via the "..." button

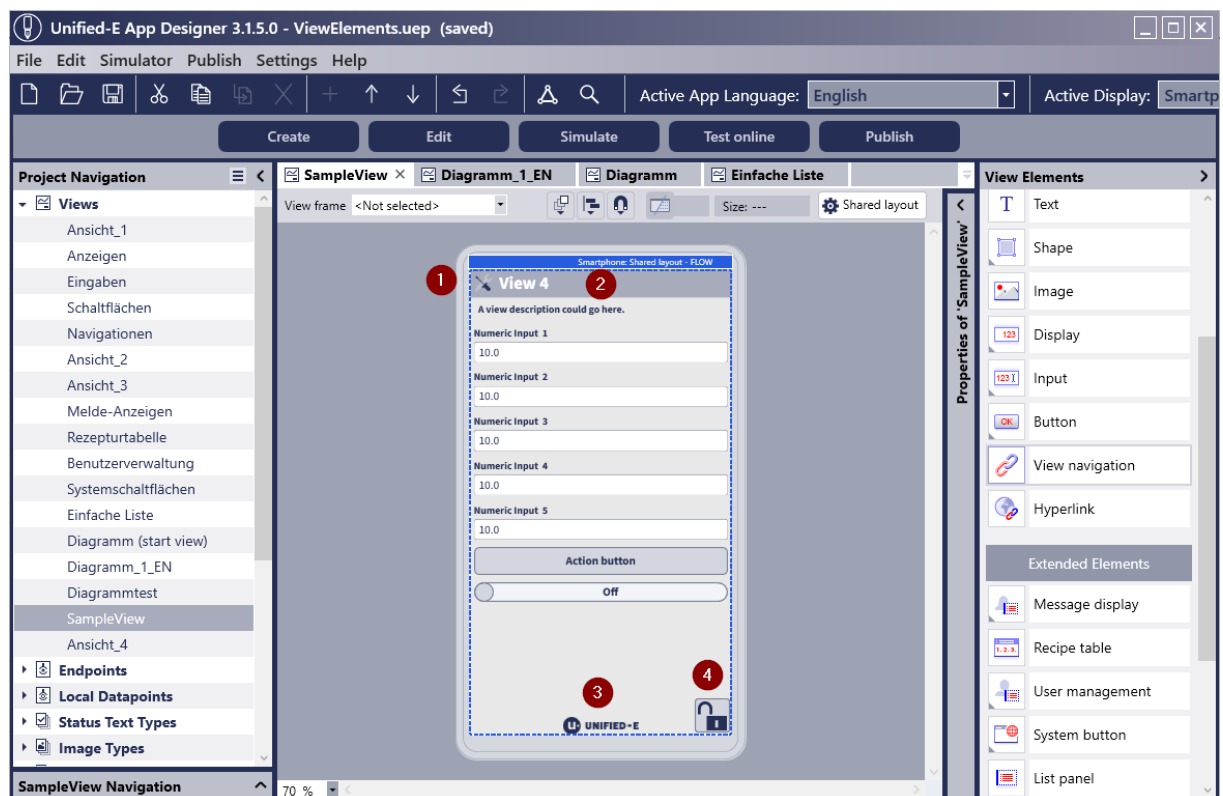
5.4 Special Elements

This chapter describes elements that are not arbitrarily instantiated like view elements via the "View Elements" area, but are nevertheless configurable in the graphical editor in terms of appearance and behavior.

5.4.1 View

The view element as a container for view elements or Template Elements is described in detail in Chapter 3.

The specific view properties are discussed in more detail here and can be opened in the Properties pane by clicking in the view background.



The view itself consists of the following parts:

- View title (optional, see 1 and 2 in figure):
The title is displayed at the top (image and label) and can be shown or hidden.
- Body: Describes the main area that contains the contained view elements.
- Background (optional, see 3 in figure): Hintergrundfarbe
- Input Lock (optional, see Figure 4):
For certain displays, the input can be configured so that input fields are only enabled after the input has been unlocked using the Input Lock element.

Property group "Common: Layout":

The "Layout" property group defines the layout behavior of the view elements placed there.

- Active Layout: Shows the currently active display and whether it is the shared or a individual layout
- Layout type: Describes the layout behavior of the elements (see Chapter 3.2.1)
- Positioning grid: See Chapter 3.7.1
- Scrollbar settings: Defines the behavior when the available area is smaller than the content
 - Display scrollbars and enlarge automatically: Scroll bars appear when needed
 - Hide scrollbars and crop elements: No scrollbar appears; protruding elements are cut off on the right and bottom

- **Padding:** Describes the border to the container

Property group "Configuration: Display options":

This property group controls the general display behavior of the view.

- **Hide view title:** If this option is activated, the title of the view (label with image) will not be displayed in the header area. By default, this option is activated.
- **Disable periodic update of datapoint values:** If this option is activated, the values of the datapoints in the view will no longer be updated periodically. An update then only occurs during user interactions, such as when the user operates a control or triggers an action. This can improve performance when there is no need for continuous display of values.
- **Image:** An image can be selected from the Images editor, which is displayed in the header area next to the label - as long as the view title is visible.

Property group "Configuration: Background image":

This property group allows you to insert a background image into the view. The image is displayed behind all other content elements in the view.

- **Image:** An image can be selected from the project or re-uploaded.
- **Horizontal Orientation:** Specifies how the image is positioned horizontally within the view. Possible values: Left, Center, Right
- **Vertical Alignment:** Determines how the image is positioned vertically within the view. Possible values: Top, Center, Bottom
- **Image Width (%):** Specifies the width of the image as a percentage of the view width. The height of the image is automatically calculated according to the aspect ratio.

Property group "Configuration: Input lock":

This property group can be used to determine whether and how inputs are allowed on a view depending on the active display. This serves to protect against unintentional operating actions – for example, due to accidental scrolling on a touch display.

A separate input lock can be defined for each configured display. Which setting applies at runtime depends on the width of the active target device, which determines the active display and thus the associated configuration (see Chapter 15.2).

Possible options per display:

- **No block, entries are possible immediately:** Entries are allowed without restrictions.
- **Allow input after unlocking:** After the first click or touch on an input element, a lock icon appears in the bottom right of the view. Only by clicking on this symbol is the input permanently released. This option provides protection against unintentional inputs without permanently restricting usability.

- **Inputs always locked:** The input elements of the view are completely disabled. No interactions are possible, even after unlocking.

Note: The Input Lock protects against unwanted input, e.g. when scrolling. If "Allow input after unlocking" is used, a lock symbol appears after the first click on an input element. Only by clicking on this symbol will the input be released.

Property group "Appearance: Preview in editor":

Here you can specify how the "Input Lock" part is displayed in the editor in order to check the configured display while editing.

The following preview options are available:

- **Invisible:** The input lock is not displayed in the editor
- **Released:** The input lock is displayed with the lock open (input released)
- **Locked:** The input lock is displayed with the lock closed (input locked)

5.4.2 View Frame

The View Frame element makes it possible not to configure identical, recurring view regions multiple times. Common areas such as the header or the navigation are defined once and superimposed on the desired views as "frames". Linking a view to a View Frame is described in the chapter 3.9 described in detail.

5.4.2.1 Terms

Container "View Frame":

The View Frame acts as a container: at runtime, it contains both the linked view and the elements placed in the frame (area outside the "View area" element). When a view is opened, the assigned View Frame is automatically loaded, which displays the corresponding view in the "View area" contained therein.

Placeholder "View Area":

The View Frame is configured in a similar way to a normal view. The "Views Area" placeholder element is an integral part and can be customized in terms of size and border. At runtime, the linked view is displayed in this area. Common elements of the frame (e.g., logos, navigation bars) are placed outside the view area, forming the actual "frame" around the embedded view.

5.4.2.2 Properties

Configure View Frame:

Clicking on the background of the View Frame allows editing the properties of the View Frame. Analogous to the configuration of a normal View, the layout for the contained elements can also be defined here (see Chapter 5.4.1).

Configure the "Views Area" placeholder element:

The View Area determines the size and position that are available to the embedded view. The placeholder element can be selected like a normal element to edit its properties in the Properties pane.

It consists only of a body in which the view is displayed as "content". The configuration is limited to general properties such as size, position, background color and border.

5.4.3 Template

Templates are a kind of reusable areas and can be configured in the same way as the view elements of the "Layout Panels" group (see Chapter 5.1).

Working with Templates is described in the chapter 3.8 described in detail.

Define layout type:

While a dedicated layout panel (e.g. 'Absolute Panel') is available for each layout type, the properties of the selected layout type can be individually configured for Templates – just as for views.

Important exception: The layout type configured in the Template applies regardless of the active display. This means that the same layout is used for the elements contained on all displays – in contrast to the layout behavior of views (see Chapter 3.3).

Configure properties:

The properties of a Template can be edited by clicking on the background of the Template in the Properties pane.

The display properties can be predefined in the Appearance Template editor – analogous to the layout panels.

5.4.4 Template Element

A Template Element is an instance of a Template. This means that there can be several instances of the same Template – so-called Template Elements – that can be individually configured via predefined parameters within the Template.

Templates and Template Elements are detailed in the chapter 3.8 described.

Template Element as container:

When a Template Element is selected in a view, its display properties can be edited. These refer to the body that contains the actual instance of the Template. The general characteristics of the body are described in the chapter 4 described.

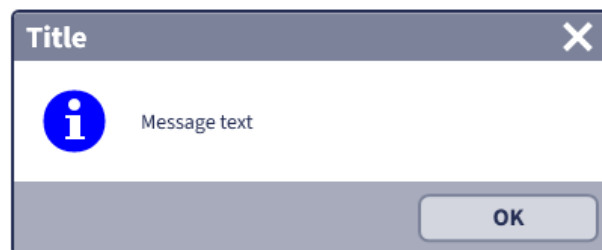
5.4.5 Dialog

Certain view elements display a system dialog at runtime when a click action is performed – for example, on an integrated button as in the Recipe Table. These dialogs do not have to be created explicitly; their content and functions are predefined.

At runtime, Unified-E has two types of dialogs:

- Confirmation dialog:
A simple confirmation dialog, e.g. with the "Yes" and "No" buttons.
- Input Dialog:
A dialog for entering values, e.g. for entering a record name for a new recipe dataset.

Example:



Adjust the display of the dialogs:

The display properties for dialogs are not configured via the properties of a selected view element – as is the case with view elements – but centrally for all dialogs in the Appearance Template editor.

All elements contained in the dialog (e.g. checkboxes, numeric value input) are displayed according to the configuration in the Appearance Template editor.

In addition, the appearance Template editor contains a special "Dialog" element entry. The following areas of the dialog box can be individually configured there:

- Body: Background of the main window.
- Title: Color, background, and font for the title text.
- Footer: Background area for buttons.
- Buttons: Display of the buttons in the dialog.
- Symbol: Symbol next to the message text (e.g. Info, Error, Warning).

6 Alarms and Messages

Alarms and messages must be defined separately so that view elements in the "Message Display" category (e.g. Message Table) can display messages.

This chapter describes the editors used to configure the message logic – that is, the configuration that determine when messages are triggered and which messages appear. The visual representation of this logic is handled by suitable Message Display view elements (see Chapter 5.3.1).

The editors for configuring messages can be found in the Project Navigation under the entry "Alarm Messages".

Message Events:

A message event describes the conditions under which a particular message text should be displayed or logged – for example, when a datapoint value is set to 1.

The logging, alarming or display of a message event can be done in several ways:

- **Save to SQLite database:**
Messages are stored in an SQLite database (see Chapter 6.1.4). Past message events can be displayed via the "Archive Message Table" view element in the HMI app.
- **Save to CSV file(s):**
Message events can be saved in one or more CSV files – see Chapter 6.1.5).
- **Send email:**
A message can be sent by email to the configured recipients (only possible via App Manager for Gateway communication).
- **Push notification to mobile device**
A message can be sent as a push notification to a smartphone. When the message arrives, a ringtone sounds on the device.
- **Message Displays in the HMI app:**
Currently pending messages can be displayed in a Message Table within the HMI app. In addition, other display elements are available, e.g. a Message Counter, which displays the number of active messages – for example, filtered by message class (see Chapter 5.3.1).

Message classes:

Message classes are assigned to the message events and define the higher-level behavior – in particular, whether or when the event must be acknowledged.

In addition, visual properties such as color or symbol can be assigned to a message class.

The message class also determines which event phases – i.e. arrived, gone, or acknowledgment – should be logged.

Message groups:

Message groups are used for the logical assignment of message events. They typically represent a module, a machine component or a functional unit.

A message event can optionally be assigned to a message group. In the Message Table, it is thus directly visible in which area of the plant an event occurred – if a message group has been defined.

Centralized logging:

Logging is performed in the App Manager for Gateway communication, so datapoints with 'Per Operator device' runtime are not usable.

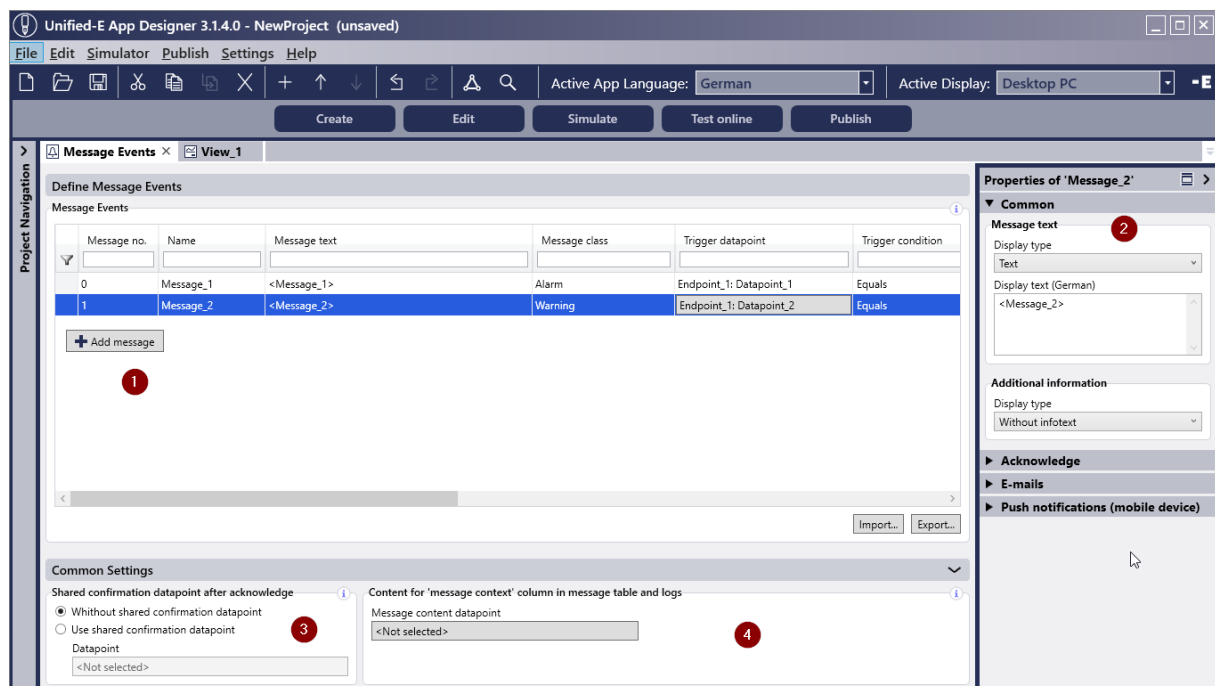
6.1.1 Manage Alarm Message Events

A message event defines the conditions under which a message is triggered – for example, when a datapoint reaches a certain value.

The configuration is done in the Message Events editor, which is opened in the Project Navigation under the "Messages" entry by double-clicking on "Message Events".

For all required messages, message events must be entered in the message event table.

6.1.1.1 Message event editor at a glance



Areas of the Message Event editor (numbering as shown in the figure):

1. Message Events Table: All message events are configured and managed in this table.
2. Properties pane: The message event selected in the table can be configured there with extended properties.

3. Shared confirmation datapoint after acknowledgement: Optionally, a common confirmation datapoint can be defined here – see Chapter 6.1.1.5.
4. Content for “message context” column in Message Table and logging: If required for message logging, an additional datapoint value can be configured for logging here – see Chapter 6.1.1.6.

Export/import message events:

To process a larger number of message events, it can be useful to export the entries, revise them in Excel and then import them again. The corresponding buttons for export and import are located below the table.

The process for the export and import process is the same as for the datapoints (see Chapter 2.5).

6.1.1.2 Configure trigger condition

The message event is triggered as soon as the defined trigger condition – consisting of condition, datapoint, parameter 1 and, if applicable, parameter 2 – is met. The evaluation is carried out periodically (according to the adjustable evaluation rate), whereby it is checked whether the condition for the trigger datapoint is newly fulfilled or no longer applies.

The following conditions can be configured:

- **Bit trigger:** Message is triggered when the defined bit (starting at 0) is set in the integer datapoint value
- **Between:** Message is triggered if the value is between parameter 1 and parameter 2 (including boundaries)
- **Not between:** Message is triggered if the value is outside the range between parameter 1 and parameter 2
- **Equal:** Message is triggered if the value is exactly equal to parameter 1
- **Not Equal:** Message is triggered if the value is not equal to parameter 1
- **Greater:** Message is triggered if the value is greater than parameter 1
- **Less:** Message is triggered if the value is less than parameter 1
- **Greater or equal:** Message is triggered if the value is greater than or equal to parameter 1
- **Less or equal:** Message is triggered if the value is less than or equal to parameter 1
- **Value changed:** Message is triggered as soon as the value of the datapoint changes compared to the previous state (regardless of the concrete value)

6.1.1.3 Available message event phases

The following phases of a message event can be recorded during logging:

- **Arrived:** Triggered once the trigger condition is met

- **Gone:** Triggered once the trigger condition is no longer met
- **Acknowledgment:** Triggered as soon as an acknowledgeable message event has been acknowledged by the user or the system (plant or machine)

6.1.1.4 Configure message event properties

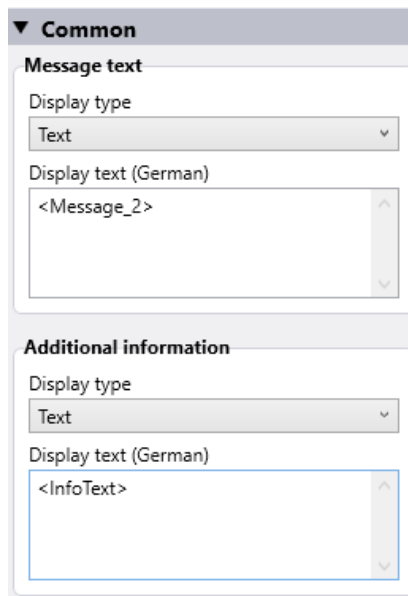
Message events are created and configured in the Message Event table. After an event has been added via "Add message", the properties can be adjusted directly in the table as well as in the Properties pane.

6.1.1.4.1 Columns of the message events table:

- **Message no.** Unique number for identification during logging. By assigning number bands per machine module or severity, the number can be given additional meaning
- **Name:** Unique technical name of the message
- **Message text:** Multilingual message text, which can also be entered in multiple lines in the Properties pane. A new line can be inserted with SHIFT + RETURN. The message text can be configured with datapoint placeholders in the Properties pane, as described below
- **Message class:** Assigned message class. This is managed in the Message Class editor (see Chapter 6.1.2)
- **Trigger datapoint, Trigger condition, Param. 1, Param 2:** Define the trigger logic, as described in the chapter 6.1.1.2 described
- **Set after acknowledgment:** Optional confirmation datapoint that is triggered after acknowledgment – depending on the acknowledgment behavior of the message class. 6.1.1.5
- **Acknowledgment bit:** Defines the bit to be set (starting at 0) in the confirmation datapoint
- **Message group:** Optional assignment to a message group. Message groups are maintained in the Message Group editor (see Chapter 6.1.3)
- **Linked view:** Optionally defined view for targeted troubleshooting. In the Message Table, this can be opened via a "Open view" button. In the case of push messages, this target view is automatically opened when tapping the arrived push notification
- **Evaluation rate (s):** Time interval in seconds in which the trigger condition is checked. Less important messages can be evaluated with a higher evaluation rate
- **Comment:** Free text field for internal comments

6.1.1.4.2 Palette "Common"

In this palette, the message text and an optional info text can be defined. The info text can be displayed at runtime via a button in the "Message Table" view element as a help dialog.



Property group "Message text":

- Display type "Text": Multi-line message text that must be entered in the display text field
- Display type "Text with datapoint parameters": As described in chapter 4.1.2, the message text can be provided with placeholders (e.g. {0}, {1}) in order to integrate dynamic values such as actual states into the text

Property group "Infotext":

An additional help text can be defined here, which is displayed via a button in the "Message Table" view element.

- Without info text: No info text is stored for the selected message event
- Text: The info text can be entered in several languages in the "Display text" field

6.1.1.4.3 Palette "Acknowledgement"

Property group "Acknowledgement user rights":

If pending messages with acknowledgment confirmation are to be acknowledged only by authorized users, then the authorized user roles must be selected in the list (analogous to chapter 4.1.11).

6.1.1.4.4 Palette "Emails"

This palette is used to configure the sending of emails when a message event arrives. This feature is only available for Gateway communication. The email recipients are defined in the Unified-E App Manager based on a user role and are not configured in the HMI project.

Property group "Send emails":

Specifies at which event phases emails should be sent:

- **Arrived event:** The email is sent as soon as the trigger condition has been met again
- **Gone event:** The email is sent as soon as the trigger condition is no longer met
- **Acknowledgment event:** The email will be sent as soon as the pending message has been acknowledged

Property group "Email user rights":

Specifies the user roles for which the email should be sent. Accordingly, only registered email recipients with the appropriate role will receive a message (analogous to chapter 4.1.11).

6.1.1.4.5 Palette "Push notifications (mobile)"

This palette configures the sending of push notifications to mobile devices when a message event arrives. This feature is only available for Gateway communication. The operator device must be registered as a mobile device in the Unified-E App Manager. If user management with user roles is activated, a user role must also be assigned.

Property group "Send push notifications":

Specifies at which event phases push notifications are to be sent:

- **Arrived event:** The push notification is sent as soon as the trigger condition has been fulfilled again
- **Gone event:** The push notification is sent as soon as the trigger condition is no longer met
- **Acknowledgment event:** The push notification is sent as soon as the pending message has been acknowledged

Property group "Push notification user rights":

Specifies for which user roles the push notification should be sent. Accordingly, only registered mobile devices with a matching role will receive a message (analogous to the 4.1.11).

6.1.1.5 Acknowledging messages

Many messages require a conscious acknowledgement by the user – especially safety-relevant alarms or operating states that require a reaction. Whether and when a alarm message must be acknowledged is defined by the assigned message class (see Chapter 6.1.2).

Acknowledging a message means that the user has actively taken note of the notice. Only then is the message marked as "acknowledged". The user acknowledges in the HMI app

using the "Acknowledge" buttons in the "Message Table" or "Archive Message Table" view elements.

A message that can be acknowledged is no longer considered as active and "disappears" from the display if it has been acknowledged and the trigger condition is no longer met. Only then is it removed from the Message Table, for example, or no longer counted in the Message Counter.

Confirmation Datapoint:

If a confirmation datapoint is assigned to a message event, the defined bit (according to the "Acknowledgment bit" column) is set in the corresponding datapoint after the user has acknowledged the message.

Set a common shared datapoint:

In many cases, a shared confirmation datapoint is sufficient for all acknowledgeable messages. This is used if no individual confirmation datapoint has been defined for the individual message event (column "Set after acknowledgement").

In the "Shared confirmation datapoint after acknowledgment" property group in the editor, the behavior can be set as follows:

- **Without a shared confirmation datapoint:** No datapoint is set if no confirmation datapoint is specified in the message itself.
- **Use shared confirmation datapoint:** The datapoint configured in the "Datapoint" field is set to 1 after acknowledgment, unless an individual confirmation datapoint is stored in the message itself.

Possible acknowledgment types (defined in the properties of the message class):

- **Without acknowledgment:** The message event does not require acknowledgment. The message will automatically disappear as soon as the Gone event occurs
- **Acknowledgment after Came event:** The message is displayed as "pending" after it occurs and remains active until it has been acknowledged manually – regardless of whether the status (trigger condition) still exists.
- **Acknowledgment After Gone Event:** The message can only be acknowledged after the trigger condition is no longer met. Only then are the "Acknowledge" buttons active.
- **Automatically acknowledge after Came event for logging:** The message is automatically acknowledged as soon as it occurs. This is used exclusively for logging and displaying in archives – without the user having to actively acknowledge. Automatic acknowledgment sets the confirmation datapoint (if defined).
 - **Example of an information message for the archive:**
If an information message is to be recorded exclusively in the archive, then the PLC can interpret the confirmation datapoint as "logging recorded" information and the trigger datapoint can be reset by the PLC. This ensures that the information message has not been lost.

6.1.1.6 Configure the message context

The message context is an additional value that is collected for logging when a message event occurs. This context can be displayed in the Message Table in the HMI app, for example, or written down in a CSV file.

Typical examples of the message context are:

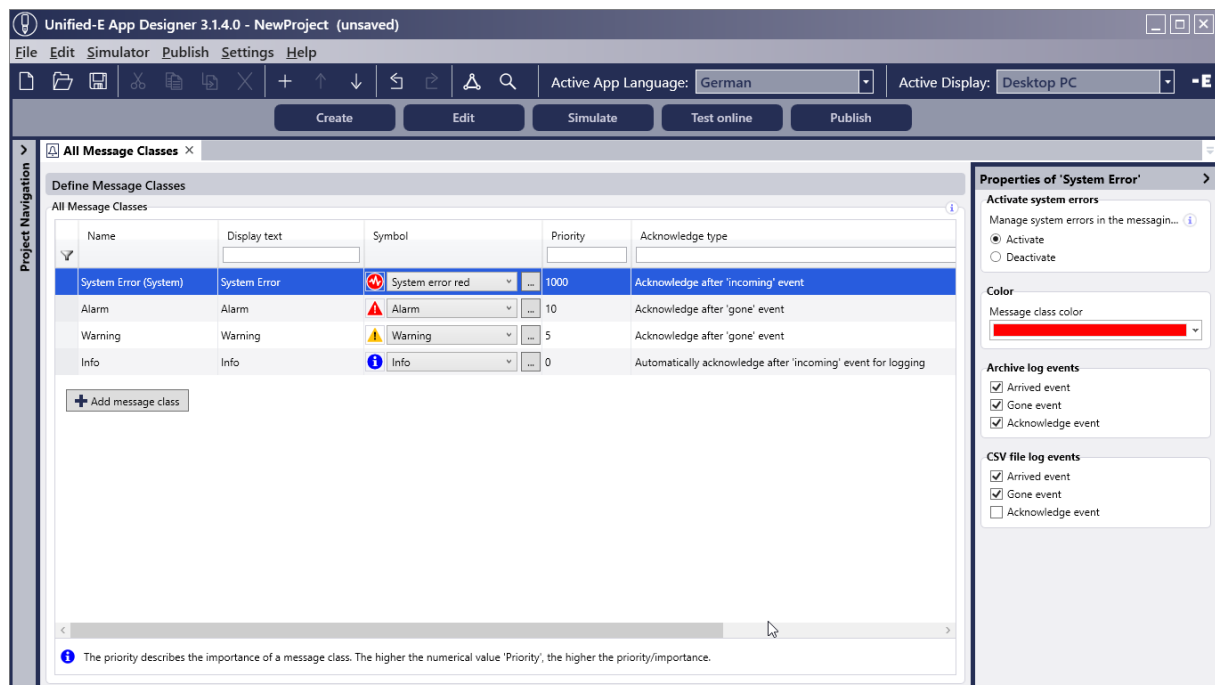
- Current work order number
- Current recipe name
- Current batch

The message context is transferred to the message system via a datapoint. This datapoint must be selected in the "Content for message context column in Message Table and logging" property group.

6.1.2 Manage Message Classes

Message classes describe the basic properties of the assigned message events.

The configuration is done in the Message Class editor, which is opened in the Project Navigation under the entry "Messages" by double-clicking on "Message Classes".



Unified-E App Designer 3.1.4.0 - NewProject (unsaved)

File Edit Simulator Publish Settings Help

Active App Language: German Active Display: Desktop PC

Create Edit Simulate Test online Publish

Project Navigation

All Message Classes

Define Message Classes

Name	Display text	Symbol	Priority	Acknowledge type
System Error (System)	System Error	System error red	1000	Acknowledge after 'incoming' event
Alarm	Alarm	Alarm	10	Acknowledge after 'gone' event
Warning	Warning	Warning	5	Acknowledge after 'gone' event
Info	Info	Info	0	Automatically acknowledge after 'incoming' event for logging

+ Add message class

Properties of 'System Error'

Activate system errors

Manage system errors in the messagin...

☒ Activate

☐ Deactivate

Color

Message class color

Archive log events

☒ Arrived event

☒ Gone event

☒ Acknowledge event

CSV file log events

☒ Arrived event

☒ Gone event

☐ Acknowledge event

The priority describes the importance of a message class. The higher the numerical value 'Priority', the higher the priority/importance.

6.1.2.1 System error message class

In addition to the individual message classes, there is a predefined "System error" message class that cannot be deleted. The properties are configurable as for the individual message classes (except acknowledgment type).

Internal messages, e.g. endpoint connection errors, recording errors, are assigned to the "System error" message class and are not configured manually.

Activate system errors:

In the Properties pane, system errors can be enabled or disabled in the "Activate system errors" property group in the message system. If system errors are activated, they also appear in the log or in a "Message Table" view element (provided the system error class is included in the filter there).

6.1.2.2 Message class properties

Message classes are created in the table using the "Add message class" button. The configuration of the basic properties is done directly in the table. Advanced settings can be adjusted in the Properties pane.

Message classes are primarily used when configuring the "Message display" view elements for filtering to include only messages of specific message classes.

6.1.2.2.1 Columns of the message class table

- **Name:** Unique object name of the message class
- **Display Text:** Multilingual text used for display and logging
- **Symbol:** Representative symbol for the message class. This can be used in view elements such as "Message Symbol" or "Message Table"
- **Priority:** Describes the urgency of the message class. Describes the urgency of the message class. Higher values mean a higher priority in display and sorting.
- **Acknowledgement type:** Specifies the acknowledgment behavior of the assigned messages (see Chapter 6.1.1.5)
- **Comment:** Freely definable, internal text for the message class

6.1.2.2.2 Properties pane

In the Properties pane, advanced properties are offered.

Property group "Color":

A color picker can be used to set the class color. This can be used, for example, as the text color for the message in the Message Table.

Property group "Archive message events":

This property group specifies which event phases of a message event should be logged in the message archive (SQLite database).

- **Arrived event:** An entry is made as soon as the trigger condition has been fulfilled again
- **Gone event:** An entry occurs as soon as the trigger condition is no longer met
- **Acknowledgment event:** An entry is made as soon as the pending message has been acknowledged

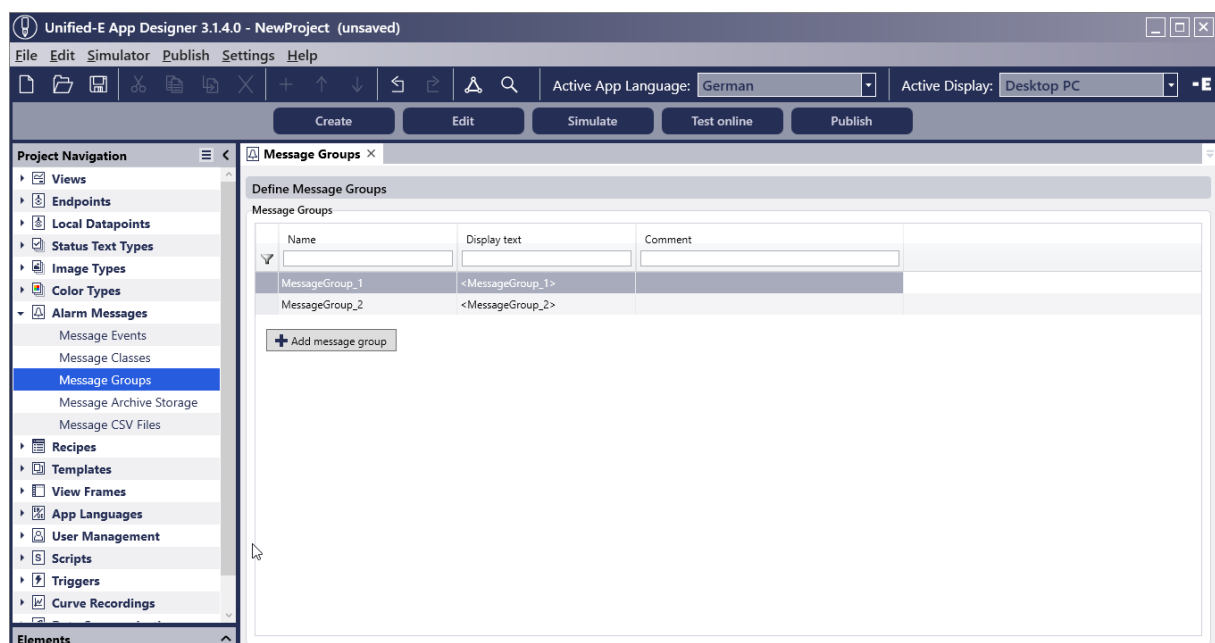
Property group "CSV file message events":

In this property group, the event phases to be recorded are to be entered in the same way as the "Archive message events" property group.

6.1.3 Manage Message Groups

A message group can optionally be assigned to a message event. For example, a message group describes a system module or a logical unit within the machine. In the Message Table and in the logging, it can thus be made directly clear from which area of the system a message originates.

The message groups are configured in the Message group editor, which is opened in the Project Navigation under the entry "Messages" by double-clicking on "Message groups".



A message group is created using the "Add message group" button. The properties of a message group are configured directly in the table:

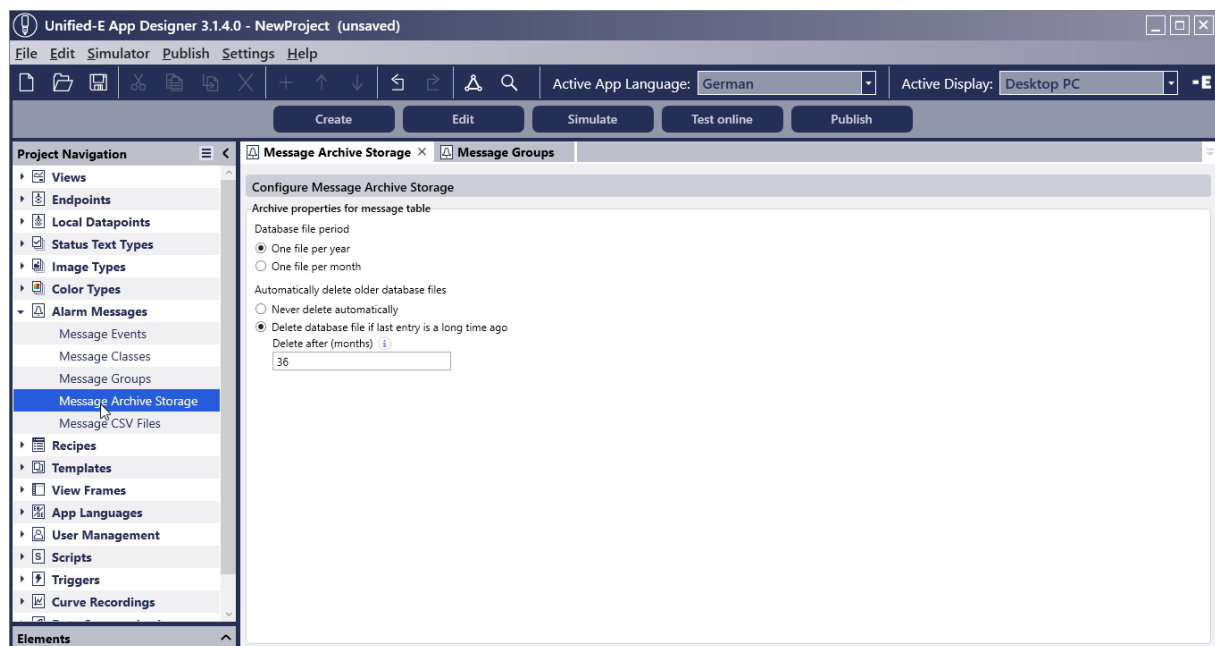
- **Name:** Object name of the message group
- **Display text:** Multilingual display text as used in the protocol

- Comment: Free text for internal use

6.1.4 Configure the Message Archive Storage

Messages can be stored in the message archive for later listing and traceability. Internally, storage is done in SQLite databases. Messages stored in the archive can be viewed in the HMI app using the "Archive Message Table" view element (see Chapter 5.3.1) by time period.

Archiving is configured in the "Message Archive Storage" editor.



Properties:

- **Database file period:** Specifies whether a file should be created per month or per year:
 - **One file per year:** An annual database file is created. Annual database file. Recommended if less than 100,000 entries per year are expected
 - **One file per month:** Monthly database file. Recommended if more than 100,000 entries per year are to be expected
- **Automatically delete older database files:** This function is intended to prevent the archive storage from getting larger and larger
 - **Delete after (months):** Specifies the number of months after which the file is deleted. The count begins at the end of the defined file period, not at the date the file was created. The file will only be deleted if automatic deletion is enabled

Backup SQLite files:

At runtime, the "MessageArchives" folder with the SQLite files is accessible as follows:

- Unified-E App Manager:
Select the installed app, then under the tab "Messages" the button "Message archive..." opens the Windows Explorer of the folder.
- Unified-E Client (for Direct communication):
In the context menu of the registered app, select the menu item "Open data folder" opens the Windows Explorer of the folder.

6.1.5 Configure Message CSV Files

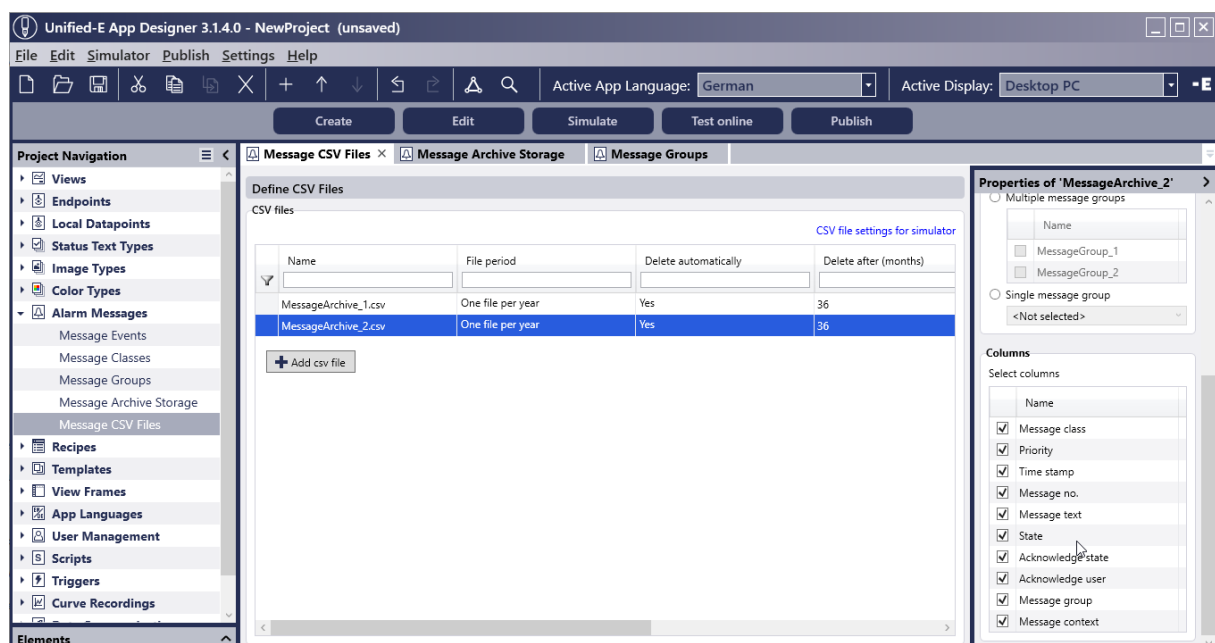
In addition to the message archive in SQLite databases, messages can also be archived in CSV files. This method is particularly suitable for simple export for further processing or archiving in external systems.

The configuration is done in the "Message CSV files" editor, which is available in the Project Navigation under the "Messages" entry.

Note: You can define message classes that include only message events for the CSV file logging. For example, process results or batch results could be logged as a row.

6.1.5.1 The Message CSV files editor at a glance

The "Add CSV file" button can be used to create a new archive file.



In the CSV files table, the following properties are defined for each entry:

- Name: File name of the CSV file (e.g. MeldeArchiv_1.csv)
- File period: Determines whether a file is created per year or per month

- **Delete automatically:** Specifies whether the file is automatically deleted after a certain amount of time
- **Delete after (months):** Editable only if not automatically deleted
 - Number of months after the end of the file period after which the file will be deleted
 - The count starts from the end of the defined file period, not from the creation date. The file will only be deleted if "Delete automatically" is enabled

6.1.5.2 Properties pane of the CSV file

In the Properties pane, further settings can be configured for the selected CSV file.

Property group "Select messages":

In this group, a filter is defined to determine which messages should be included in CSV logging, depending on the assigned message class and message group.

Which event phases (arrived, gone, acknowledgement) are logged can be configured in the corresponding message class (see Chapter 6.1.2.2.2).

- **Message classes:** Determines which message classes a message must be assigned to in order for it to be logged into the CSV file:
 - All message classes: All messages regardless of the message class will be taken into account
 - Multiple message classes: Only the message classes enabled in the list are taken into account
 - Single message class: Only the selected message class is taken into account
- **Message groups:** Determines from which message groups messages are taken into account – analogous to filtering the message classes:
 - All message groups: No restriction
 - Multiple message groups: Only the selected groups are taken into account
 - A message group: Only messages from the selected group are logged

Property group "Columns":

Here you can specify which columns (properties of the message) are to be logged into the CSV file at runtime.

The following columns or properties are available:

- **Message class:** Assigned message class (e.g. "Alarm", "Warning")
- **Priority:** Numerical urgency of the message
- **Time stamp:** Time of the event phase
- **Message number:** Unique message number
- **Message text:** Multilingual text that is displayed when the message is made

- **Status:** Current status of the message (e.g. pending, acknowledged)
- **Acknowledgement state:** Indicates whether and when an acknowledgment has been made
- **Acknowledgment User:** Username of the person who acknowledged the message
- **Message group:** Assigned message group (e.g. Machine module)
- **Message context:** Context information at runtime (e.g. work order number, tool)

6.1.5.3 CSV files in the file system

At runtime, the folder "CsvFiles" with the CSV files is accessible as follows:

- **Unified-E App Manager:**
Select the installed app, then under the tab "Messages" the button "Message archive..." click.
- **Unified-E Client (for Direct communication):**
In the context menu of the registered app, select the menu item "Open data folder".

CSV export is already active in the App Simulator. The directory can be opened during a simulation in the simulator settings (Chapter 15.6).

7 User Management

User management in Unified-E specifically controls access and input rights within the HMI application. User roles are defined and users are assigned to them. Based on these roles, certain views, elements, or functions in the visualization can be selectively shown, hidden, or locked.

The authorization check is carried out at runtime in the HMI app – e.g. a "Switch" view element can only be operated if a user with the corresponding role is logged in.

Important terms of user management are user roles, users, and permissions, as described below.

This chapter explains how to preconfigure user management in the Unified-E App Designer. At runtime, view elements such as the "User Table" for managing active users or a "Authentication Button" for login or switching users are available (see Chapter 5.3.3).

Roles:

Roles form the basis for the authorization check. They define which functions or content a user is allowed to see or use.

User:

For each user role, users can be defined with a username and password. These users are available to log in to the HMI app with the "Authentication Button" view element. Users can be predefined in the HMI project or created later at runtime.

Permissions:

For many objects (e.g. view elements or messages), it is possible to define in the properties whether an authorization check should be carried out for a certain function. This determines for which roles a certain function (e.g. visibility, input, acknowledgment) is allowed.

Depending on the currently logged in user, the function is released (e.g. element is visible) or remains locked (see also chapter 4.1.11).

Local user management:

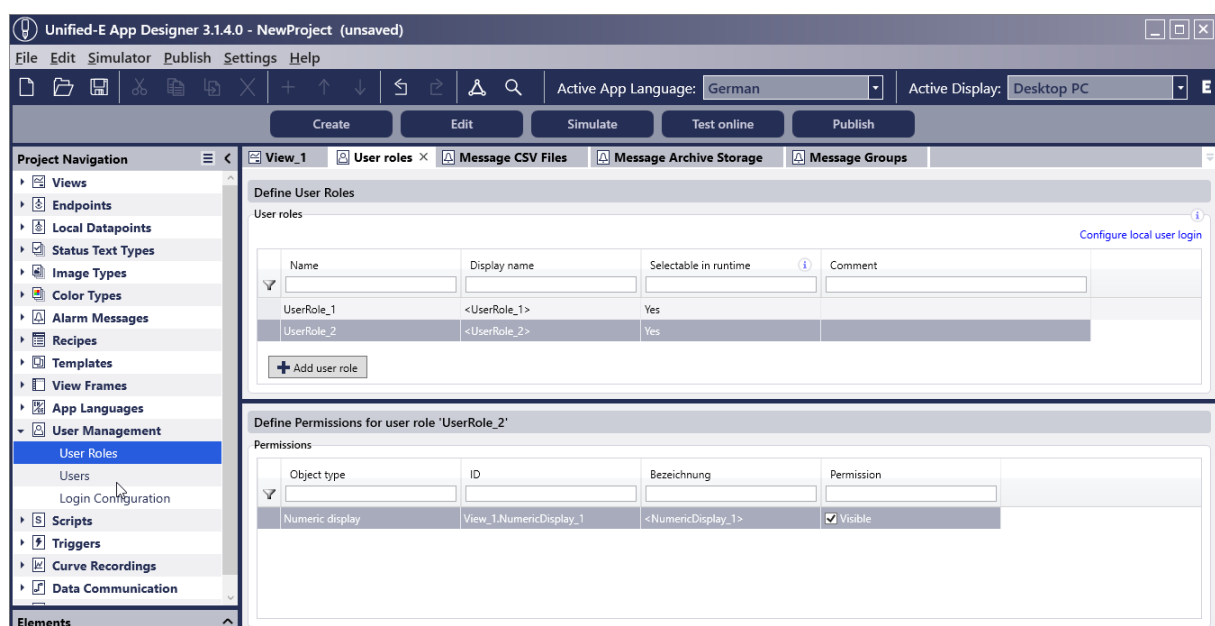
"Local user management" refers to the explicit login/logout on the local operator device and the management of users in the "User Table" view element. The registered users are also stored locally on the operator device during Gateway communication. Local user management must be explicitly enabled in the login configuration (Chapter 7.1.3) (default case), otherwise configured permissions will not be taken into account in Direct communication.

Regardless of whether local user management is enabled in the HMI project, the permissions in the Unified-E App Manager can be set for the operator device via the selected user role or optionally via the local user management. See also chapter 7.1.3.

7.1.1 Manage User Roles

User roles form the basis for the authorization check in the HMI app. Roles define which users are allowed to see or operate which content – such as the visibility of elements or the ability to make entries or acknowledge messages.

The user roles are configured in the User Role editor, which is opened in the Project Navigation under the entry "User Management" by double-clicking on "User Roles".



7.1.1.1 Define user roles

The "Add user role" button creates a new role. The following properties can be configured in the table:

- **Name:** Technical name of the role (e.g. Benutzerrolle_1)
- **Display Name:** Name for display in view elements (multilingual)
- **Selectable at runtime:** Specifies whether the role should be available for selection when a new user is created in the HMI app. If "no" is selected, the user role can only be used for predefined users (see Chapter 7.1.2)
- **Comment:** Free text for internal notes or descriptions

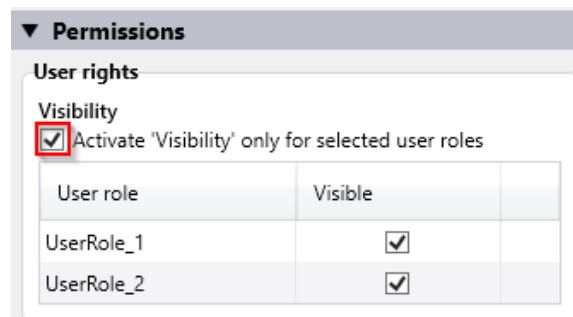
7.1.1.2 Maintaining permissions

At the bottom of the User role editor, the Permissions table is displayed. This lists all objects for which an authorization check is active for the currently selected user role. In addition, the permissions for the selected user role can be managed in the table.

Enable/disable permission on the item for the selected role:

In order for a permission to appear in the table, it must be explicitly activated for the respective object – e.g. by setting the checkbox in the "Permissions" property group (see Chapter 4.1.11).

Only after activation does an entry appear in the Permissions table, in which the permission (user right) for the selected user role can be maintained.



User role	Visible
UserRole_1	<input checked="" type="checkbox"/>
UserRole_2	<input checked="" type="checkbox"/>

The table shows:

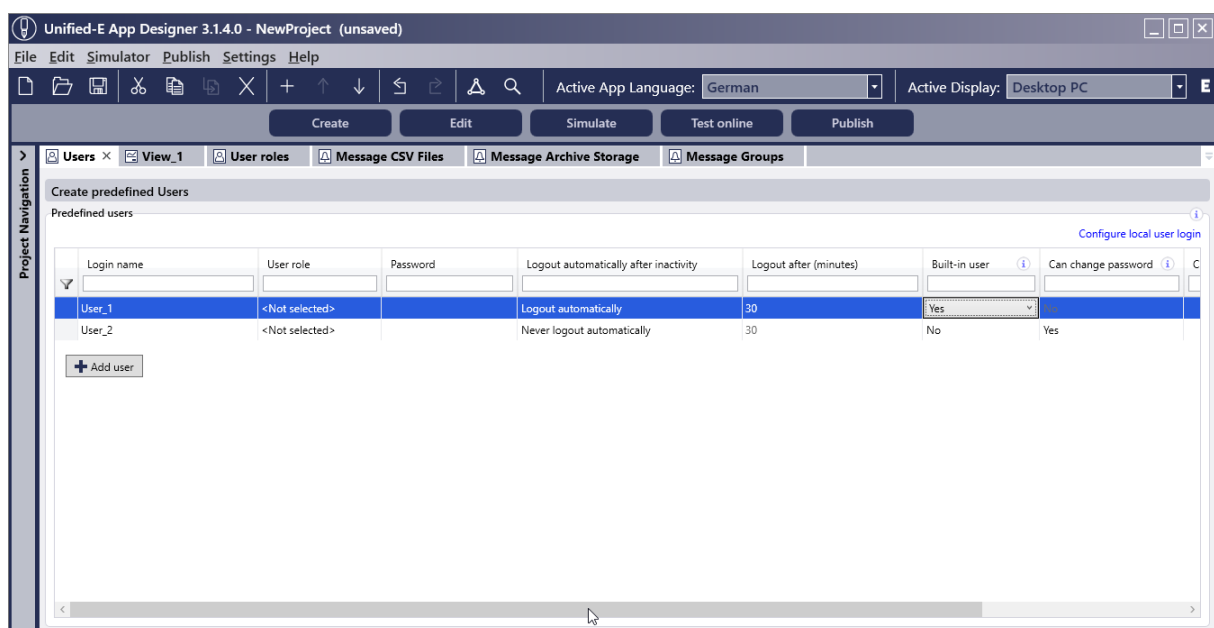
- **Object type:** Type of the element (e.g. "Numeric Display")
- **ID:** Unique object identifier consisting of the path to the view and the element name
- **Label:** Display name of the item (if available)
- **Permission:** The specific authorization (e.g. "visible", "input possible", "acknowledgement possible")
 - This can be activated or deactivated via the checkbox

7.1.2 Define Predefined Users

In the Users editor, users can be predefined for logging in to the HMI app. These users can log on to HMI app with their username and password. A user role is assigned to each user, which is used to check the permissions at runtime.

The predefined users appear – just like users created at runtime – in the "User Table" view element and can be managed there (see Chapter 5.3.3.1).

The configuration is done in the Users editor, which is opened in the Project Navigation under "User Management" by double-clicking on "Users".



Add users:

The "Add user" button creates a new user. The following properties can be defined in the table:

- **Login name:** Login username (unique)
- **User role:** Role that sets the user's permissions.
- **Password:** Password for login. The input is in plain text, but is stored encrypted
- **Logout automatically after inactivity:** Specifies whether the user should be automatically logged out after a period of inactivity
- **Log out after (minutes):** Duration of inactivity (in minutes) after which an automatic logout occurs
- **Built-in user:** This option is set to "No" by default. Built-in users are permanent users, which means that they are created for commissioning and service purposes. Built-in users cannot be edited or deleted at runtime in the "User Table" view element.

- **Can change Password:** Determines whether the logged in user is allowed to change the password at runtime
- **Comment:** Free text for internal notes or descriptions

7.1.3 Login Configuration

7.1.3.1 Configure Login Behavior

Local user login takes place directly on the operator device – even if the device is connected to the endpoints via Gateway communication. The local user administration checks at runtime whether the user logged on to the operator device is authorized to display or operate certain content.

The login dialog is automatically displayed in the HMI app as soon as the user clicks on one of the following view elements in the visualization:

- User Display
- Authentication Button

In the dialog, the user logs in with a username and password. The permissions are checked via the user rights configured in the HMI project on the basis of the user roles.

Local user management can be enabled or disabled in the HMI project. Its behavior differs depending on whether Direct or Gateway communication is used.

Activate local user login in the HMI app (operator device):

This setting allows local user login as described above.

- **Gateway communication:** The local user management (including the login dialog) can be disabled during the registration process Add new operator device in the App Manager and replaced with a fixed user role assignment.
- **Direct communication:** Local user login is mandatory. Initially, protected functions are not active; only by registering are they activated in a targeted manner.

Deactivate local user login in the HMI app (operator device):

Local user management is not available, i.e. local login to the operator device and the User Table are also deactivated.

- **Gateway communication:** It is still recommended to configure user roles and permissions in the HMI project. During the "Add new operator device" registration process, a user role must then be explicitly assigned in the App Manager that applies to all users on the device.
- **Direct communication:** Local user login is mandatory. Initially, protected functions are not active; they are only unlocked through user login.

Working with the App Simulator:

In the simulator settings, you can specify whether the simulation takes place in Gateway communication or Direct communication mode. This setting affects the behavior of the user logon accordingly.

7.1.3.2 Set Login View

If local user login is enabled, then the login view opens when starting the HM app and after logging out. After successful login, the start view (display-specific, if necessary) is opened.

The login view is to be set in the "Login View" property group.

8 Recipes

In industrial automation, the term "recipe" refers to a structured collection of setpoints, parameters, or target values required for a specific production process.

The recipe management in the Unified-E system allows such data to be centrally organized, easily edited, and specifically transferred to machines using recipes.

Recipe type vs. Recipe Table:

In the Unified-E App Designer, the recipe type describes the basic structure and logic of the associated recipe datasets. It determines which parameters (or datapoints) are included, how they are grouped and displayed, and whether dataset versioning is enabled.

At runtime, the individual recipe datasets are managed via the "Recipe Table" view element, which is linked to one or more recipe types. New datasets can be created, existing ones edited, versioned, and activated in the Recipe Table – as described below.

Create and edit recipe type:

New recipe types can be created via the context menu in the Project Navigation under the "Recipes" entry. Double-clicking on the desired recipe type object opens the Recipe type editor.

Central recipe management:

Recipe management is performed in the App Manager during Gateway communication, so datapoints with 'Per operator device' runtime are not usable in the recipe type configuration.

8.1 Important Terms

Since the term "recipe" can be used in different ways, Unified-E distinguishes between several precisely defined terms.

8.1.1 Recipe Type

A recipe type describes the structure of a recipe in the Unified-E system. It defines which recipe parameters belong to the recipe, in which order they are displayed, whether they can be described and how they are to be visualized. Each recipe parameter is linked to a specific datapoint that is used for control communication.

A recipe type thus forms the data model for the associated recipe datasets and defines their logical structure.

Example: A recipe type "Beverage mix" could contain the following parameters:

- Amount of sugar
- Carbon dioxide content
- Filling temperature

8.1.2 Parameter Group and Parameter

The parameters defined in the recipe type are organized into parameter groups. This allows parameters to be logically structured. In the Recipe Table, the parameters are displayed in groups – by selecting the appropriate tab, you can switch between the parameter groups.

In the standard case, exactly one datapoint is assigned to each editable parameter. This is used both when activating (see below) and when saving the recipe dataset directly.

8.1.3 Recipe Dataset

A recipe dataset (or "dataset" for short) is a specific expression of a recipe type – i.e. a complete compilation of values according to the data model defined in the recipe type. As a rule, there are several recipe datasets for a recipe type. These can either be predefined in engineering or created directly in the Recipe Table at runtime.

Example: A dataset for the type "Beverage mix" could be:

- Amount of sugar: 7.2 g/l
- Carbon dioxide content: 4.0 g/l
- Filling temperature: 5 °C

Recipe datasets can be accessed at runtime via the "Recipe Table" view element (see Chapter 5.3.2). This includes, in particular, the editing of existing recipe datasets as well as the activation (download to PLC) of recipe datasets.

8.1.4 Recipe Dataset Version (optional)

If versioning has been enabled in the recipe type, each recipe dataset is provided with a version history. Changes to a dataset then do not lead to an overwrite, but to the creation of a new version. This way, all previous versions remain archived and traceable.

Versioning is particularly useful in regulated industries or for safety-critical processes in which subsequent changes must be documented and traceable. However, it also supports the optimization phase of a production, as previous versions can be reactivated and compared with each other at any time.

8.1.5 Activate Recipe Dataset

When a recipe dataset is activated, the selected recipe dataset – or the desired dataset version if versioning is activated – is downloaded to the controller. The stored values are written according to the datapoint assignment defined in the recipe type. The activated dataset is marked as "activated" in the runtime recipe management and is also visible in the Recipe Table in the "Active dataset" section.

Important: All load datapoints are loaded with the respective parameter value when activated – load datapoints must therefore always be unique – even when using steps – see below.

8.1.6 Steps

Use case: A production process consists of several steps that use similar or even identical parameters – but are to be configured with different values in the recipe dataset.

With the concept of "steps", a parameter group only needs to be defined once. At runtime, individual value inputs for all parameters of this group per step are then possible. The maximum number of steps allowed is set in the parameter group, as an individual load datapoint must be configured for each parameter and step.

The parameter values of all defined steps are transferred (downloaded) to the controller when the recipe dataset is activated. For this reason, load datapoints must always be configured individually per parameter and step.

Recipe

Active dataset: Engineering\MED 123457 | Article_2

Parameter values ○ All datasets ● Active dataset

Heating **Dosing** Finish

1 2 **2**

Name	Unit	Def. setpoint	Act. value
Refill		On	
Speed	ml/sec	0.0 3	
Volume	ml	0.0	
Dosing			
Pressure	bar	0.0	
Volume	ml	0.0	
Speed	ml/sec	0.0	

Edit

Parameter selection in the Recipe Table (numbering as in the figure):

1. Select parameter group:
Each tab in the header represents a defined parameter group.
2. Select step:
Within each parameter group, each step is shown as its own tab. In edit mode, steps can be added or removed.
3. Select parameter:
The parameter list displays all the parameters of the selected parameter group for the selected step, along with their values.

The step identification is numeric and begins with "1". Working with steps is optional and must be specified in the base settings of the recipe type (see Chapter 8.6) can be explicitly activated.

8.1.7 Dataset Folders

Use case: Not all datasets should be editable by all HMI app users. Production staff should only be allowed to load the latest version for production use.

In Unified-E, this use case can be handled using dataset folders and folder permissions. To do so, the Enable folder management option must be explicitly activated in the base settings of the recipe type (see Chapter 8.6). All datasets will then be managed within folders.

8.1.8 Process Highlighting

In the Recipe Table, the tabs for parameter groups, steps, but also individual parameter rows can be highlighted. This allows the production progress or the active process or machine unit to be visualized at runtime. The logic of when a highlight should take place is defined in the recipe type, while the highlight color is freely selectable in the "Recipe Table" view element.

8.1.9 Parameter Adjustment Before Activation

This function makes it possible to adjust certain parameters of a recipe dataset before activating it – without changing or saving the modified dataset. The adjusted values apply only to the current activation process and are not stored in the recipe dataset.

The adjustment is primarily intended for operators who activate a recipe according to the production work order, but are allowed to adjust individual parameters within defined min/max limits – e.g. for fine adjustment depending on environmental conditions. The original dataset remains unchanged.

When activated, the adjusted parameter value is downloaded to the controller – not the setpoint stored in the original dataset. This allows the recipe to be flexibly adapted to situational requirements without the need to edit or version the dataset.

When editing the dataset, a technician or engineer (depending on configured permissions) determines which parameters can be adjusted and in which area. The prerequisite for this is that the function is set in the base settings of the recipe type (Chapter 8.6) via the "Support parameter adjustments" checkbox.

Important: The "Adjust parameters" function should not be confused with editing a recipe dataset. When adjusting, there is no saving – when editing, on the other hand, the dataset file is permanently changed or versioned.

The following figure shows the Recipe Table with the "Parameter adjustment" function activated in edit mode. Additional columns for configuring the adjustment of individual parameters are available to the technician and only concern the input options for the operator.

Recipe

Active dataset: No dataset activated

Edit dataset ● All datasets ○ Active dataset

Engineering\MED 123457\V0 | Article_2:

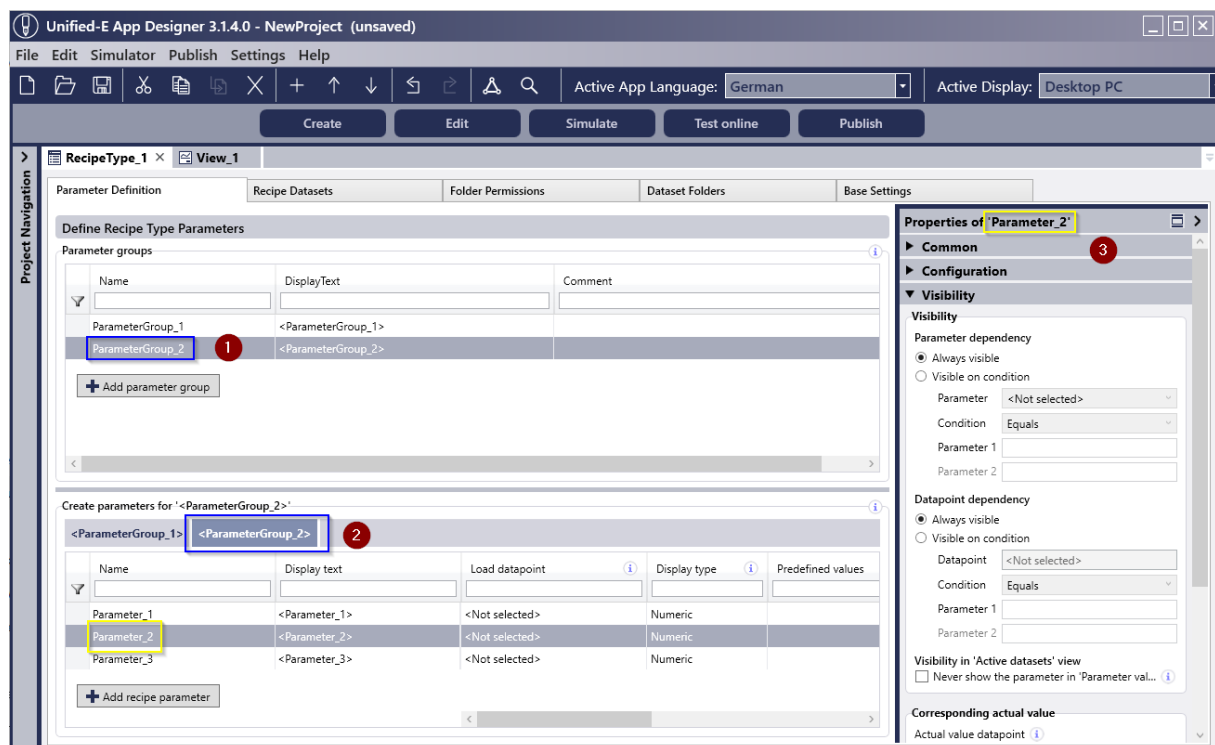
Heating **Dosing** Finish

Name	Unit	Def. setpoint	Adjustable	Adj. min.	Adj. max
Refill		Off	No	---	---
Dosing					
Pressure	bar	0.0	Yes	0.0	100.0
Volume	ml	6.0	Yes ▼	5.0	100.0
Speed	ml/sec	0.0	No	---	---

ⓘ X Cancel Save

8.2 Define Recipe Parameters

All parameters of a recipe type can be managed in the Recipe type editor by selecting the "Parameter Definition" tab.



Areas of the "Parameter Definition" tab (numbering as in the figure):

1. Parameter Group table:
As described above, parameters are organized into parameter groups, which can be created and configured here.
2. Parameter table:
This table manages the parameters of the selected parameter group (selection via tab or via selection in the parameter group table).
3. Properties pane:
The advanced properties of the last selected object (parameter group at the top or parameter at the bottom) are displayed here.

8.2.1 Properties of a Parameter Group

8.2.1.1 Properties in the table

The most important properties are configured in the table:

- Name: Unique group name
- Display text: Multilingual display text as displayed in the Recipe Table
- Comment: Freely definable text for internal description

Other columns for step activation:

If steps are activated for the recipe type, additional columns are available:

- Step type: Describes whether the number of steps per dataset is configurable or fixed

- Number of steps can be changed per dataset: If selected, then the number of steps in this parameter group can be adjusted when editing the dataset (at least 1 step required)
 - Fixed number of steps for all datasets: The number of steps on the dataset is not editable, but predetermined
- **Step count:** Defines – depending on the step type – either the fixed number or the maximum number of steps allowed
- **Steps count datapoint:** Datapoint to which the actual number of steps used is written when a dataset is activated. This value informs the controller or PLC about the number of valid steps

8.2.1.2 Properties in the Properties pane

Property group "Visibility: Process highlighting in Recipe Table":

These properties define the conditions under which the selected parameter group is highlighted in the Recipe Table with an indicator on the tab – for example, to visualize the active process phase.

- **Group selector:** Defines by datapoint and condition when the tab of the parameter group should be highlighted
- **Step selector (only for active steps):** Defines via a datapoint which step tab is highlighted within the active parameter group
 - No highlighting: Step selector tabs are never highlighted
 - Highlight active processing step: The selected datapoint is expected to have the step number of the active processing (starting at 1). This datapoint value is cyclically read out in the "Active dataset" view of the Recipe Table. If the value is less than 1 or greater than the number of steps, no highlighting is made.

8.2.2 Properties of a Parameter

The following parameter properties can be configured by selecting the parameter in the parameter table.

8.2.2.1 Properties in the table

The most important properties are configured directly in the table:

- **Name:** Unique object name within the parameter group
- **Display text:** Multilingual display text, as the parameter is called in the Recipe Table
- **Load datapoint:** Datapoint that is written/loaded with the dataset value when activated. When working with steps, a separate load datapoint must be defined for each possible step.
- **Display type:** Specifies how to interpret the datapoint value:
 - **Text:** The datapoint value is interpreted as text

- Numeric: The datapoint value is formatted for display (depending on the value in the "Decimal places" column). In addition, a minimum and maximum must be defined for the input
- Status text type: The assigned status text type must be selected under "Predefined values". At runtime the Load Datapoint value determines the status text to use
- **Predefined values:** Select the status text type if the display type is set to "Status text type"
- **Default value:** Initial value when creating a new dataset
- **Unit:** Value displayed in the Unit column of the Recipe Table.
- **Decimal places:** Number of decimal places for display type "Numeric"
- **Minimum:** Minimum input value for "Numeric" display type
- **Maximum:** Maximum input value for "Numeric" display type
- **Comment:** Freely definable text for internal description

Loading datapoints with enabled "Steps" feature:

If steps are enabled, separate columns are available in the table for each step for the respective load datapoint. The column "Load datapoint (Step 1)" corresponds to the datapoint for step 1, "Load datapoint (Step 2)" to that for step 2, and so on.

Create parameters for '<ParameterGroup.1>'

<ParameterGroup.1>		<ParameterGroup.2>					
1	2	3	4				
Name	Display text	Load datapoint (Step 1)	Load datapoint (Step 2)	Load datapoint (Step 3)	Load datapoint (Step 4)	Display type	Predefined values
Parameter_1	<Parameter_1>	<Not selected>	<Not selected>	<Not selected>	<Not selected>	Numeric	
Parameter_2	<Parameter_2>	<Not selected>	<Not selected>	<Not selected>	<Not selected>	Numeric	
Parameter_3	<Parameter_3>	<Not selected>	<Not selected>	<Not selected>	<Not selected>	Numeric	

+ Add recipe parameter

8.2.2.2 Properties in the Properties pane

The properties in the Properties pane can be configured after selecting the parameter in the parameter table.

Property group "Common: Info text":

A help text can optionally be configured for the parameter, which can be displayed in the Recipe Table in the dialog.

- **Display type:** Without info text: No info or help text is defined
- **Text:** Multilingual info text can be entered in the "Display text" field

Property group "Configuration: Display options":

By default, parameter objects are linked to a datapoint and are intended for editing. This property group can instead be used to configure a parameter as text display only, which serves as a title or description, for example, within the parameter list in the Recipe Table.

- **Parameter type:** Specifies whether it is an input parameter (default) or a text parameter
 - **Input parameter:** The parameter is used as a regular dataset value, which is stored and loaded
 - **Text only:** The parameter is used exclusively to design the list – for example, as a comment or section category. In this case, only the display text of the parameter is displayed, there is no link to a datapoint and no value input supported
- **Highlight in Recipe Table:** If set, then the parameter name (or text) is highlighted (in bold).
 - **Note:** This emphasis is unrelated to the process highlighting

Property group "Configuration: Load value for unused steps":

These properties only have an effect if the steps feature is activated for the recipe type. It specifies which values should be written to the load datapoints of the unused steps.

For example, the parameter group supports a maximum of 5 steps, but only two steps are configured in the enabled dataset. Steps 3 to 5 remain unused and can still be described with a defined value (e.g. 0) to simplify handling in the controller.

- **Do not set datapoint:** The load datapoint of the unused steps is not written (default)
- **Set constant value:** The value specified in the input field is written to the load datapoints of the unused steps

Property group "Visibility: Visibility in Recipe Table":

This property group defines whether a parameter is displayed in the Recipe Table at runtime, either always or only under certain conditions. Visibility can be configured depending on the value of another parameter or a datapoint.

- **Parameter dependency:** Controls visibility based on another recipe parameter value
 - **Always visible:** No conditional visibility is configured
 - **Visible on condition:** The parameter is displayed only if the defined condition is met based on the selected reference parameter
- **Datapoint dependency:** Controls visibility based on a datapoint value
 - **Always visible:** No conditional visibility is configured
 - **Visible when condition is met:** The parameter is displayed only if the defined condition is met based on the selected datapoint
- **Visibility in "Active dataset" view:** Specifies whether the parameter should also be displayed in "Active dataset" view. This option can be used to hide advanced parameters (e.g. parameters only for experts) so that production employees only see the relevant process parameters

Combination: If any of the above visibility conditions are not met, the parameter will not appear in the parameter list of the Recipe Table.

Property group "Visibility: Corresponding actual value":

This defines the datapoint whose current value should optionally be displayed in the actual value column of the Recipe Table. The actual value is typically a sensor value associated with the parameter – it is not the value of the charging datapoint, which represents the setpoint in the process.

- **Actual value datapoint:** If set, the value is displayed in the "Actual value" column in the Recipe table view element. When teaching (function of the Recipe Table), this actual value is also transferred to the focused input field

Property group "Visibility: Process highlighting in Recipe Table":

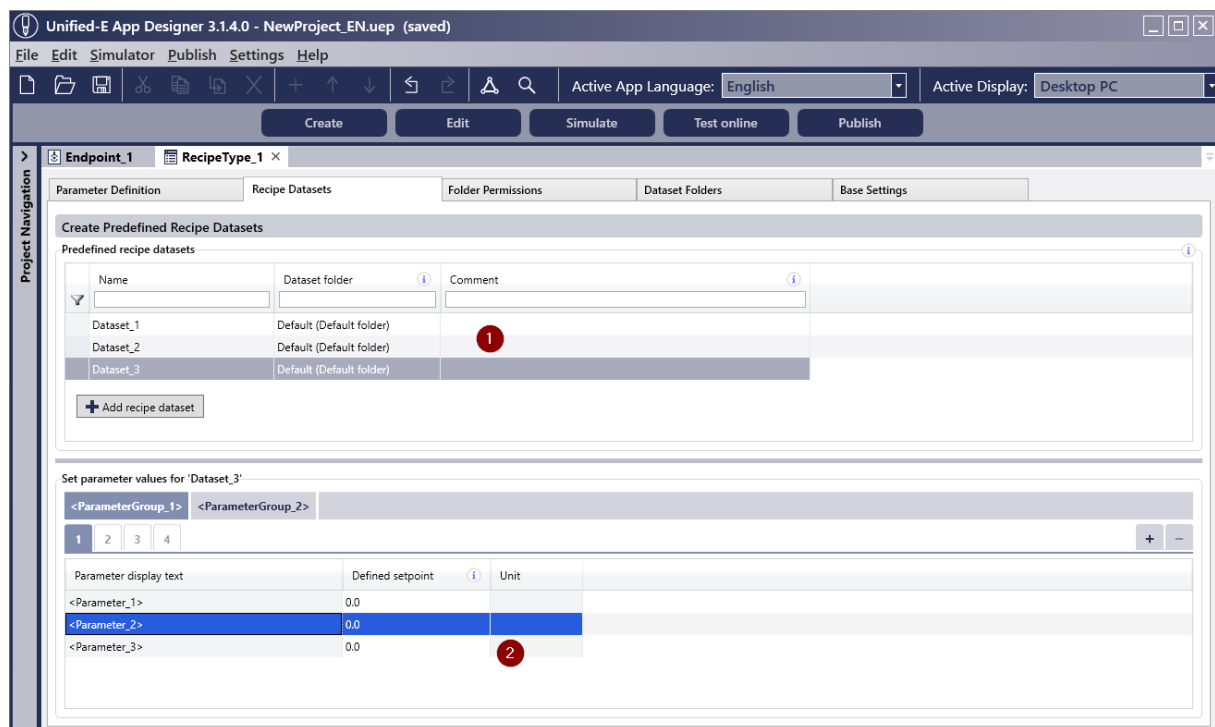
This property group defines the conditions under which the selected parameter is highlighted in the Recipe Table with a visual indicator – for example, to represent the current process phase.

- **Don't highlight:** No conditional highlighting is configured
- **Highlight when condition is met:** The parameter is highlighted when the defined condition is met based on the selected datapoint

8.3 Predefined Recipe Datasets

Recipe datasets can already be predefined during engineering, which can then be selected for activation in the Recipe Table when the HMI app is started for the first time. At runtime, the predefined datasets can be edited or deleted, or new datasets can also be created.

Predefined datasets can be defined in the Recipe type editor under the "Recipe Datasets" tab.



Areas of the "Recipe Datasets" sub-editor (numbering as shown in the figure):

1. Datasets table:
This area is used to create and configure predefined recipe datasets.
2. Parameter values table:
This is where the setpoints of the parameters are set, which are written to the assigned load datapoints when the dataset is activated. The parameters are shown in the tabs of the respective parameter groups. If the steps function is activated, you can set setpoint values for each step and add or remove steps (using the "+" and "-" symbols).

8.3.1 Create and Configure a Recipe Dataset

A new predefined recipe dataset can be created in the Dataset table by clicking on "Add recipe dataset". The configuration is done in the following columns:

- **Name:** Name of the dataset as it appears in the Recipe Table
- **Dataset folder:** If folder management is activated, a predefined folder can be selected here. Otherwise, the default folder will be mapped
- **Comment:** Freely definable text for internal description

If the "Dataset labeling" function is activated (see Chapter 8.6), the following column will also appear:

- **Label:** Additional unique name for the dataset. For example, the dataset name could be used as the article number and the dataset label as the article description.

8.3.2 Set Setpoints for the Parameters

For a selected dataset in the dataset table, the setpoint values of the defined parameters can be defined in the lower area. These values are written to the associated load datapoints when the dataset is activated.

When activating additional functions in the base settings (e.g. step function and parameter adjustment), the parameter-value table can have the following structure:

Set parameter values for 'Dataset_3'

<ParameterGroup_1>		<ParameterGroup_2>				
1	2	3	4			
Parameter display text	Defined setpoint	Adjustable	Adjustment min	Adjustment max	Unit	
<Parameter_1>	0.0	<input checked="" type="checkbox"/>	0.0	100.0		
<Parameter_2>	0.0	<input checked="" type="checkbox"/>	0.0	100.0		
<Parameter_3>	0.0	<input type="checkbox"/>				

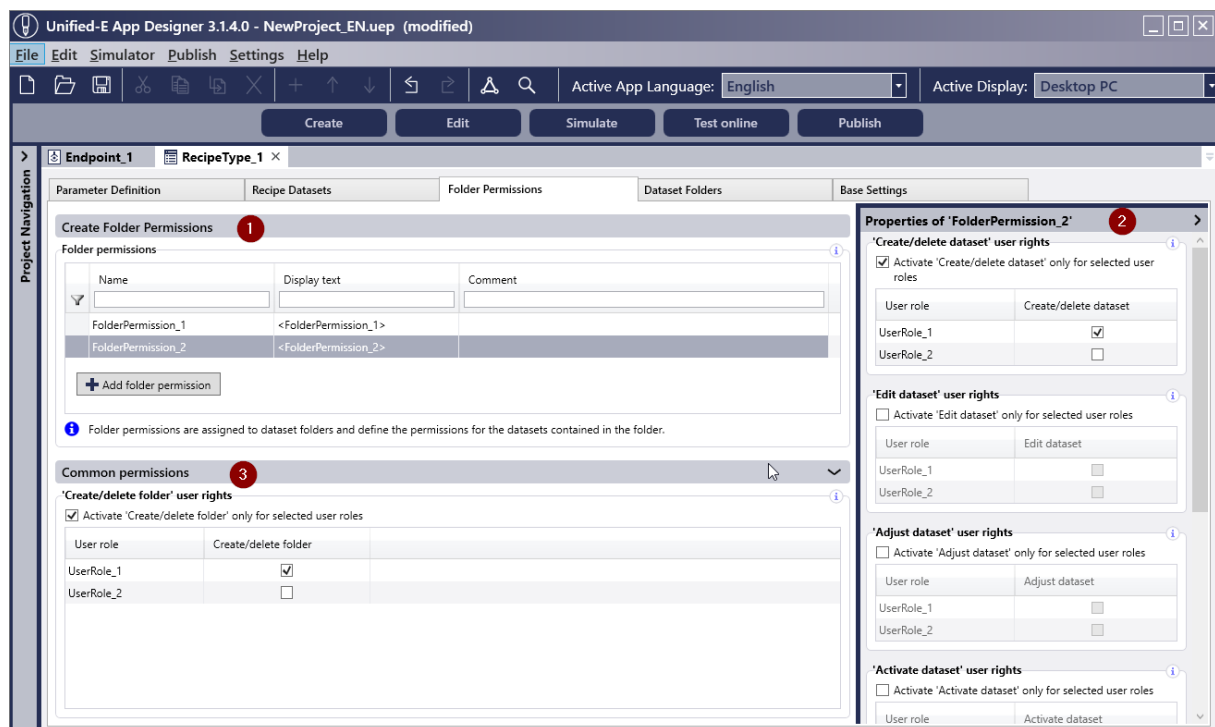
Description of the table areas (numbering as in the figure):

1. Step selection (only if steps function is activated):
Select the step whose parameter is to be configured.
2. Add/Delete step (only if Steps feature is enabled):
The buttons can be used to add a new step or delete an existing one.
3. Configure parameter adjustment (only if the Parameter adjustment function is activated):
 - a. Column "Adjustable": Specifies whether the parameter should be enabled for adjustments before activation.
 - b. Column "Adjustment min": Defines the minimum allowed for an adjustment before activation.
 - c. Column "Adjustment max": Defines the maximum allowed for an adjustment before activation.

8.4 Folder Permissions

If the "Folder management" option is enabled in the base settings of the recipe type (Chapter 8.6), recipe datasets are organized in folders. These folders can be assigned folder permissions to restrict editing of datasets to specific users or user roles, for example.

If a folder permission object is assigned to a folder, the permissions configured in it apply to the folder itself as well as to all recipe datasets contained therein.



Areas of the "Folder Permissions" sub-editor (numbering as shown in the figure):

1. Folder permissions table: In the table, folder permissions can be created and selected for configuration.
2. Properties pane:
This is where the detailed permissions of the selected folder permission object are configured.
3. Common permissions:
In this area, the global "Manage folders" permission is set. Only users with the appropriate permission are allowed to create or delete folders.

8.4.1 Properties of a Folder Permission

8.4.1.1 Properties in the table

The following properties can be configured directly in the Folder permissions table:

- **Name:** Unique object name
- **Display Text:** Multilingual text as used in the Recipe table at runtime
- **Comment:** Freely definable text for internal description

8.4.1.2 Properties in the Properties pane:

By selecting a folder permission object in the upper table, the associated permissions can be configured in the Properties pane.

The following property groups define which user role is allowed to perform a particular function in connection with recipe datasets in the folder. The configuration is analogous to chapter 4.1.11.

Property group "Create/delete dataset" user rights:

Authorized users can create or delete datasets in the folder.

Property group "Edit dataset" User rights:

Authorized users can edit datasets in the folder.

Property group "Adjust dataset" user rights:

Authorized users can adjust datasets in the folder. This permission is only relevant if dataset adjustments are supported. Adjusting the parameter values before activating them is described in the Chapter 8.1.9.

Property group "Activate dataset" user rights:

Authorized users can activate datasets in the folder, i.e. download/transfer the defined setpoint values to the load datapoints.

Property group "Activate older version" user rights:

Authorized users can activate older datasets in the folder. If the user only has the 'Activate Dataset' permission, then only the most recent datasets can be selected. This permission is only relevant if dataset versioning is supported.

Property group "Select folder" user rights:

This permission affects the folder selection list in the Recipe Table. Only folders with 'Select folder' permission are visible

8.4.2 Common Permissions

In this area, the global, common permission is configured.

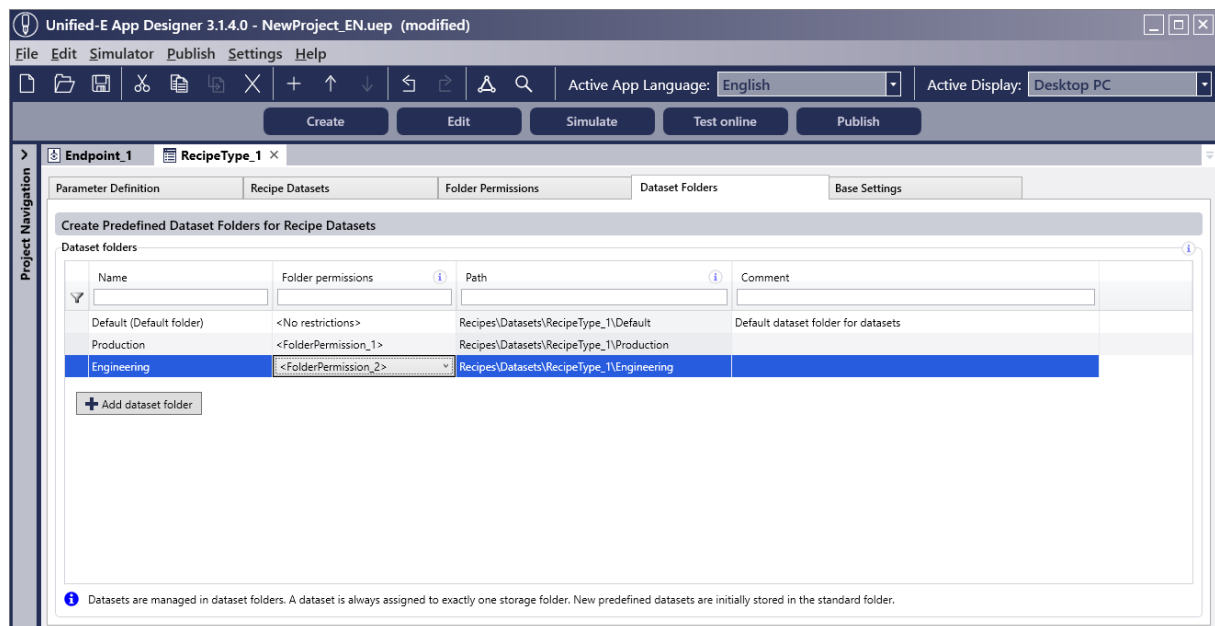
Property group "Manage folder user rights":

Authorized users are allowed to create, delete, or assign permissions to folders. This permission is only relevant if folder management is enabled.

8.5 Dataset Folder

If the "Folder management" function is activated in the base settings of the recipe type (Chapter 8.6), recipe datasets are organized into dataset folders.

Predefined folders are created in the Recipe type editor in the "Dataset Folders" sub-editor. At runtime, additional folders can be created in the Recipe Table or existing folders can be edited.

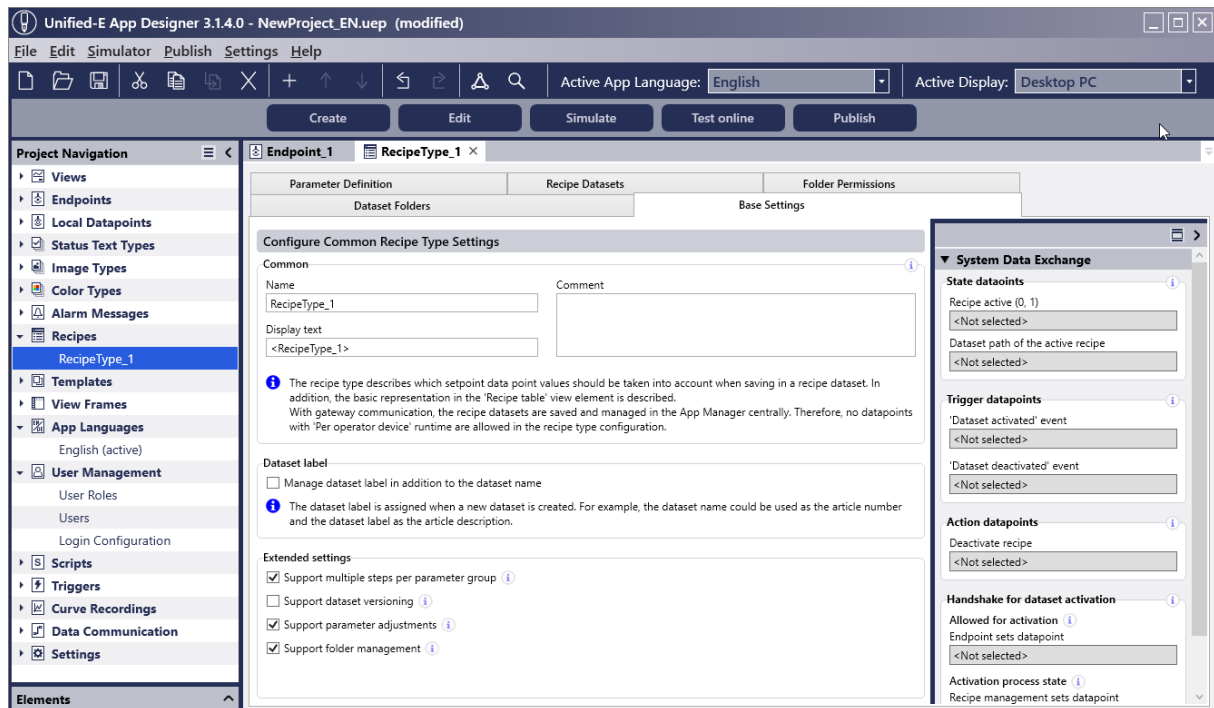


Columns of the Dataset-folders table:

- **Name:** the folder name – corresponds to the corresponding folder name in the file system
- **Folder permission:** If set, the linked permissions apply to the logged in user. Otherwise, there are no restrictions
- **Path:** The relative path to the contained datasets (see Chapter 8.7)

8.6 Configure Base Settings

The "Base Settings" sub-editor is used to configure general properties of the recipe type and to activate advanced functions such as versioning, parameter adjustment or folder management. In addition, the state, trigger, and action datapoints for the dataset activation process can be defined here.



8.6.1 Configure Common Recipe Type Settings

Property group "Common":

- **Name:** Internal name of the recipe type. This name is used in configuration and management.
- **Display text:** Multilingual text that appears as a label in the Recipe Table.
- **Comment:** Optional free text for the internal description of the recipe type.

Property group "Dataset label":

If the "Manage dataset label in addition to the dataset name" option is activated, each dataset can be assigned a separate label in addition to the name – e.g. to separate the article number and article description.

Property group "Extended settings":

- **Support multiple steps per parameter group:** Activates the "Steps" feature. Parameter groups can thus contain several similarly structured steps (e.g. "Mixing steps 1 - 3") - see Chapter 8.1.6
- **Support dataset versioning:** Enables per-dataset version management. Changes create new versions, existing ones remain stored in a traceable way – see Chapter 8.1.4
- **Support parameter adjustments:** Allows you to temporarily adjust certain parameters before activating a dataset. These adjustments are not saved – see Chapter 8.1.9

- **Support folder management:** Enables the management of recipe datasets in folders with user rights (see Chapter 8.4 and 8.5)

8.6.2 Configure System Data Exchange in the Properties pane

This is where the datapoints for the recipe activation process are defined – if necessary:

Property group "State datapoints":

The condition datapoints are set by the recipe management at runtime after an action has been performed.

- **Recipe active (0/1):** Will be set to 1 by the recipe management as soon as a recipe is active.
- **Dataset path of active recipe:** Sets the path of the activated dataset by Recipe Management after activation, such as <Folder Name>\<Dataset Name> <Version>

Property group "Trigger datapoints":

Trigger datapoints are set to 1 by recipe management when an event occurs. The HMI app or even an endpoint can be notified of an activation or deactivation and then reset the datapoint to 0.

- **Dataset activated event:** Will be set to 1 once a dataset has been activated
- **Dataset deactivated event.** Will be set to 1 once the dataset has been deactivated

Property group "Action Datapoints":

If the action datapoint is set to 1 by the HMI app (or by the endpoint), then the corresponding action is executed by the recipe management. After the end of the action, the datapoint is reset to 0 by the recipe management.

- **Deactivate recipe:** If the recipe management registers a 1, then the active dataset is deactivated and the datapoint is reset to 0

Property group "Handshake for dataset activation":

The handshake datapoints ensure that datasets are only activated when the endpoint is ready for them. In addition, the endpoint can ensure that it does not start unwanted process processing during an 'Activate Dataset' operation.

- **Allowed for activation:** Endpoint releases Activation
- **Activation process state:** Maintained by recipe management
 - This datapoint is set to 1 by the recipe management before starting the 'Activate Dataset' operation and is reset to 0 after the activation is complete (when all datapoint values have been set).
 - For example, the endpoint can use the datapoint to block the start of the process during activation.

- Possible values: 0 = Activation operation not active, 1 = Activation operation active

Property group "Process highlighting in Recipe Table":

This process highlight appears in the 'Active Dataset' display on the Recipe Table. The highlight color can be configured in the properties of the Recipe Table under "Appearance".

- **No highlighting:** No conditional highlighting is configured
- **Highlight on condition:** The dataset is highlighted when the defined condition is met based on the selected datapoint

8.7 Recipe Datasets in the File System

At runtime, the recipe datasets are managed in XML files – in versioning, one file is used per version.

Structure of the file name:

<Dataset Name>.<Version Number>.xml.

The XML files are located at:

Recipes\Datasets\<Folder Name>\<Recipe Type Name>\

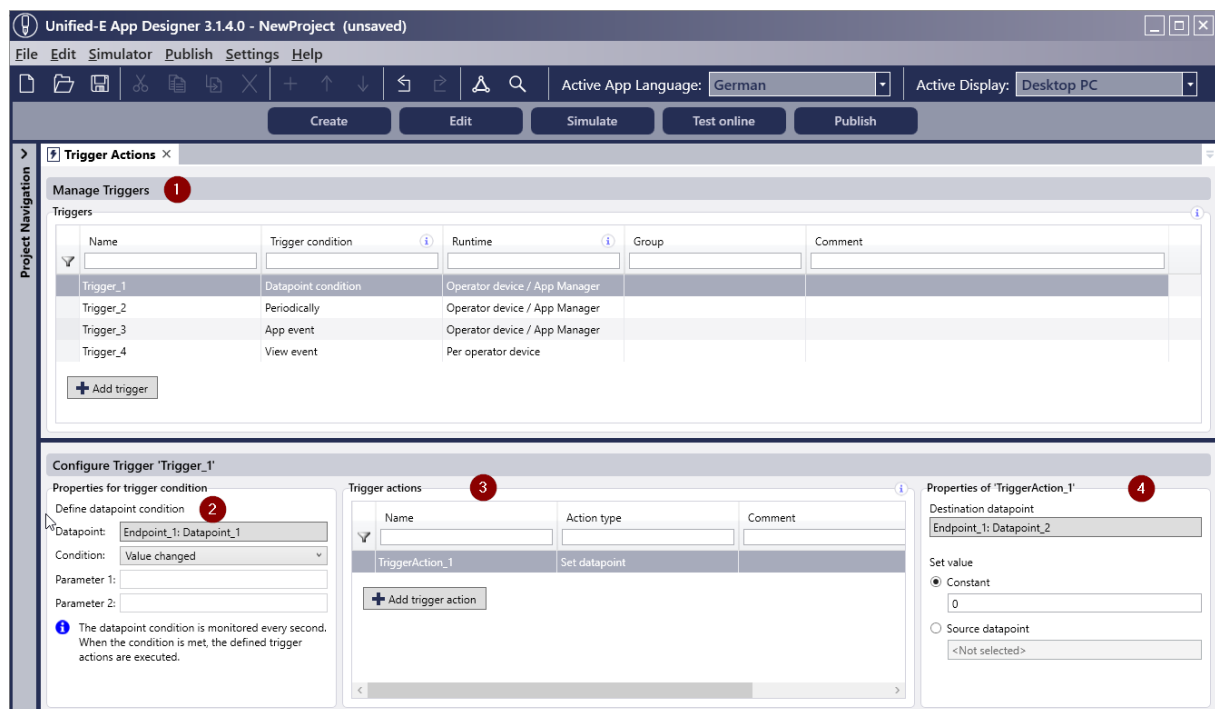
At runtime, the "Recipes" data folder is accessible as follows:

- **Unified-E App Manager:**
Select the installed app, then click the "Open folder" button under the "Recipes" tab.
- **Unified-E Client (Direct communication only):**
In the context menu of the registered app, select the menu item "Open data folder".

9 Trigger Actions

Trigger objects can be defined to automatically trigger predefined trigger actions when certain conditions occur. These actions are executed at runtime either in the App Manager or locally on the operator device. Typical use cases include setting datapoints, automatically opening views, or executing a script.

The Trigger actions editor can be opened via the Project Navigation under the "Triggers" entry by double-clicking on "Trigger Actions".



The editor includes the following areas (numbering according to the figure):

1. Triggers table:
In this table, all triggers are managed created and basic properties are set.
2. Properties for trigger condition:
For the selected Trigger object, the trigger condition and additional properties are configured here.
3. Trigger actions table:
The table contains all the defined actions to be performed when the trigger condition arrives.
4. Trigger action property group:
The selected trigger action can be configured here.

9.1 Manage Triggers

In the trigger table, basic properties of the selected trigger object are configured:

- **Name:** Unique name of the trigger
- **Trigger condition:** Type of condition that triggers the action when it occurs (e.g., datapoint condition, app event, periodic) – see Chapter 9.2
- **Runtime:** Specifies whether the trigger is executed in the App Manager or per operator device in the Unified-E Client – see Chapter 9.4
- **Group:** Optional grouping for structural organization
- **Comment:** Description of purpose or behavior

With the button "Add trigger" a new Trigger object can be created.

9.2 Trigger Conditions

Depending on the selected trigger condition, the corresponding parameters can be configured in the "Properties for trigger condition" property group. Once the condition is met, the trigger event is raised – that is, all defined trigger actions are executed.

Trigger condition "Datapoint condition":

The trigger event occurs when the defined condition applies to the selected datapoint.

The condition is checked every second. Once it is fulfilled, the associated trigger actions are automatically executed. This configuration is suitable, for example, to react to control signals or status changes (see Chapter 4.6 for datapoint conditions).

Trigger condition "Periodic":

The trigger is triggered automatically at regular intervals.

- **Interval (ms):** Time interval in milliseconds. The minimum allowed value is 500 ms.

Trigger condition "App event":

The trigger is raised when the app (application) is started or closed.

- **Gateway communication:** App refers to the App Manager
- **Direct communication:** App refers to the HMI app running in the Unified-E Client
- **App Event:** Specifies the app event in more detail
 - **On App/App Manager:** Trigger is triggered on startup
 - **On App/App Manager exit:** Trigger is triggered on exit

Trigger condition "View event":

The trigger is raised when a particular view is opened or closed.

- **Linked view:** View to be monitored
- **View event:** Specifies whether the trigger should be triggered when the view is opened or closed
 - **On view open:** Triggered on open
 - **On view close:** Triggered on close

9.3 Manage Trigger Actions

The actions of the selected trigger are managed in the Trigger Actions Table. The following properties are available:

- **Name:** Unique name of the trigger action
- **Action type:** Determines the type of action that will be taken when the trigger is triggered. The details are defined in the associated property group of the trigger action.
 - Possible actions: Set datapoint, execute script, navigate view (see below)
- **Comment:** Description or purpose of the action

Action type "Set datapoint":

This action sets a target datapoint to a specific value.

- **Target datapoint:** Datapoint to be written
- **Set value:** Sets the value to be written
 - Constant: A fixed value that is specified directly in the text field
 - Source datapoint: The value is read from the selected source datapoint and written to the target datapoint

Action type "Execute script":

This action executes the write script of the associated script datapoint. The value 0 is passed as the input variable writeValue to the write script.

- **Script datapoint:** Datapoint at which the write script is stored. (Access must be set to "Write" or "Write, Read")

"Navigate view" action type:

When executed, the configured view opens. This type of action is only available if the trigger condition is set to "Datapoint condition" and the runtime is set to "Per operator device".

- **Navigate to view:** View to open when running
- **Ignore current view when navigating back:** If set, the view will not be included in the list for back navigation (e.g. on mobile device)

9.4 Runtime of the Trigger

The Runtime property determines in which application the trigger should be monitored and executed.

'Operator device / App Manager':

The trigger is configured to run in the App Manager, so no "Per operator device" runtime datapoints can be used for the trigger configuration. In the case of Direct communication, the trigger is always executed in the operator device (HMI app).

Per operator device':

The trigger is executed in the operator device, so all local datapoints (including script datapoints) can be used.

10 Curve Recordings

With the "Curve Recordings" module, Unified-E offers a simple way to systematically record measured values of datapoints. This allows both the graphical representation of historical data in the form of charts and the export of measured values in CSV files for further processing.

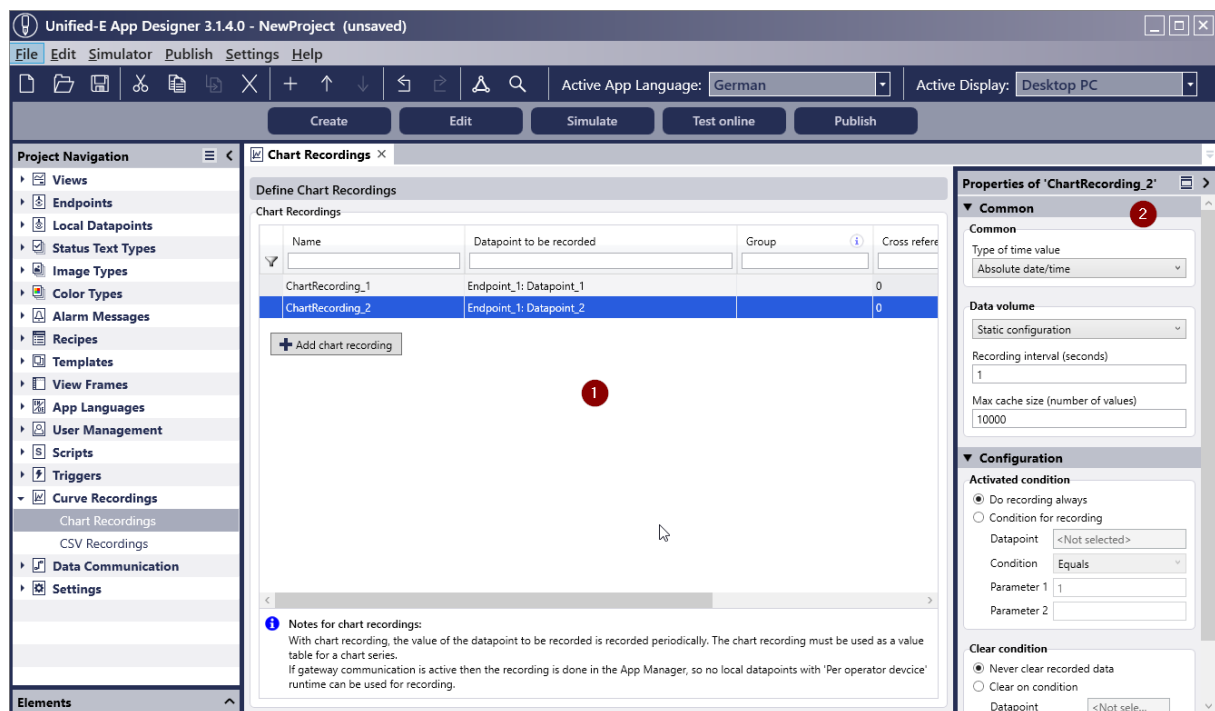
The recordings are stored in files – directly in the operator device for Direct communication, or centrally in the Unified-E App Manager when using Gateway communication.

Central Curve Recording:

When using Gateway communication, the recording takes place in the Unified-E App Manager. Therefore, no datapoints with a "Per operator device" runtime can be used for curve recordings.

10.1 Chart Recordings

These recordings are intended for visualization in a time chart and can be linked directly to a chart data series of the "Chart" view element (see Chapter 5.3.6). This allows historical data such as temperatures, pressure curves or consumption values to be displayed clearly. The values are stored in a ring buffer (max. 30,000 values). The recording can be started, stopped or deleted via datapoints. For example, when starting a new process, the recording can be reset. The time values can be saved as an offset in seconds from the start of the recording or as an absolute time value.



The editor contains the following areas (numbering as in the figure):

1. Chart Recordings table:
This is where new chart records are created and selected for configuration.
2. Properties pane:
The Properties pane configures advanced properties of the selected recording.

Use Chart Recording:

The chart record is represented as a datapoint of type "Table" and can be selected as a datapoint in a view element such as "Time Chart" (via Chart Data Series – see Chapter 5.3.7) or "List Panel".

10.1.1 Properties in the Table

The most important properties are configured directly in the table:

- **Name:** Unique name, used for the selection in the chart data series and at the same time corresponds to the file name of the recording file (see Chapter 6.1.5.3)
- **Datapoint to be recorded:** The datapoint whose value is to be recorded periodically
- **Group:** Optional value for grouping and filtering in the table
- **Cross references:** Shows the number of usages of the recording. Clicking on the cross-reference symbol opens the cross references section with all usages
- **Comment:** Freely definable text for description

10.1.2 Properties in the Properties Pane

Property group "Common:Common":

- **Type of time value:** Specifies how the time to the read datapoint value should be recorded
 - **Absolute date/time:** The timestamp is stored in UTC format with date and time
 - **Numerical time value(s):** The time is stored in seconds from the start of the recording – ideal for visualizing production processes

Property group "Common: Data volume":

The recording values are stored in the ring buffer together with the time value. This can be configured as follows:

- **Static Configuration / Dynamic Configuration with datapoints:** Determines whether the configuration values are defined in the editor or dynamically via datapoints at runtime
- **Recording interval (s):** Interval in seconds at which the value of the datapoint (according to the table column "Datapoint to be recorded") is to be recorded
- **Maximum cache size (number of values):** Maximum number of stored values in the ring buffer (maximum 30,000)
 - **Note:** A maximum of 5000 values can be displayed in the time chart at once

Property group "Configuration: Activated condition":

With this condition, a datapoint can be used to control whether recording is activated.

- **Do recording always:** Recording is always active
- **Condition for recording:** Recording occurs only when the condition is met based on a defined datapoint

Property group "Configuration: Clear condition":

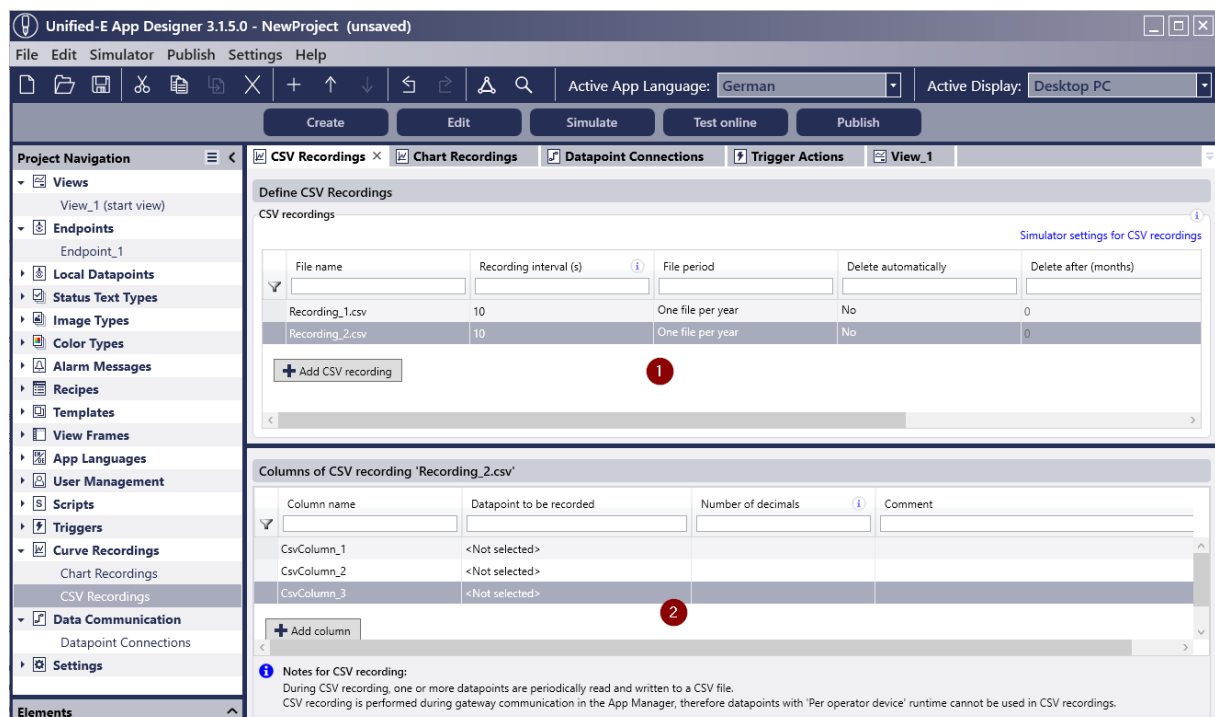
Specifies how the deletion of the recording values in the ring buffer in the HMI app or from the endpoint is triggered.

- **Never clear recorded data:** There is no automatic deletion
- **Clear on condition:** The recording is deleted when a given datapoint takes on the defined delete value
 - **Datapoint:** The datapoint to be monitored.
 - **Clear Value:** Value at which the recording should be deleted
 - **Acknowledgement value:** Value that the recording module writes back to the datapoint after deletion (for acknowledgment)

10.2 CSV Recordings

This type of recording periodically writes values to a locally stored CSV file. The files can be used for later analysis, long-term archiving or for import into external systems such as Excel or databases. New records are appended to the CSV file in a performant way and the number of records is theoretically unlimited (depending on the hardware memory).

The CSV Recordings editor can be opened via the Project Navigation under the "Curve Recordings" entry by double-clicking on "CSV Records".



Editor areas (numbering as in the figure):

1. CSV Recordings table:
This table defines new CSV records (files)
2. Columns table: Specifies which datapoints are to be stored per record (columns of the CSV file)

10.2.1 Define CSV Recordings

The top section of the editor contains the CSV Recordings table, where individual recordings can be configured.

Columns in the "CSV records" table:

- **File name:** Name of the CSV file, e.g. Recording_1.csv. The file name must not contain any special characters

- **Recording Interval(s):** Time interval in seconds in which the values are to be written
- **File period:** Determines how often a new file is created (e.g. "One file per year", "One file per month")
- **Delete automatically:** Specifies whether to automatically delete old files
- **Delete after (months):** If automatic deletion is active, this value determines how many months the file will be deleted

10.2.2 Define Columns for CSV Recording

If a CSV recording is selected, the Columns table appears at the bottom of the editor. Here the datapoints are to be defined that are to be written to the file.

Columns in the "CSV Recording Columns" table:

- **Column name:** Name of the column in the CSV file. The name is written as a column header
- **Datapoint to be recorded:** Datapoint whose value should be written to the corresponding column
- **Number of decimals:** Number of digits to which the value should be rounded
- **Comment:** Free text describing the column (optional, only visible in the editor)

10.3 Recording Files in the File System

The curve recordings are stored in the data folder of the respective app. Depending on the recording type, the files are located in the following subfolders:

- CSV file: Subfolder "CsvRecordings"
- Chart recordings (XML file): Subfolder "Recordings"

Open Data Folder:

- **Unified-E App Manager (for Gateway communication):**
Select the installed app, then click the "Open folder" button under the "Curve recordings" tab in the corresponding section for chart recordings or CSV recordings.
- **Unified-E Client (for Direct communication):**
In the context menu of the registered app, select the menu item "Open data folder".

The curve recordings are already active in the App Simulator. The corresponding data directory can be opened during or after a simulation in the simulator settings (Chapter 15.6).

11 Datapoint Connections

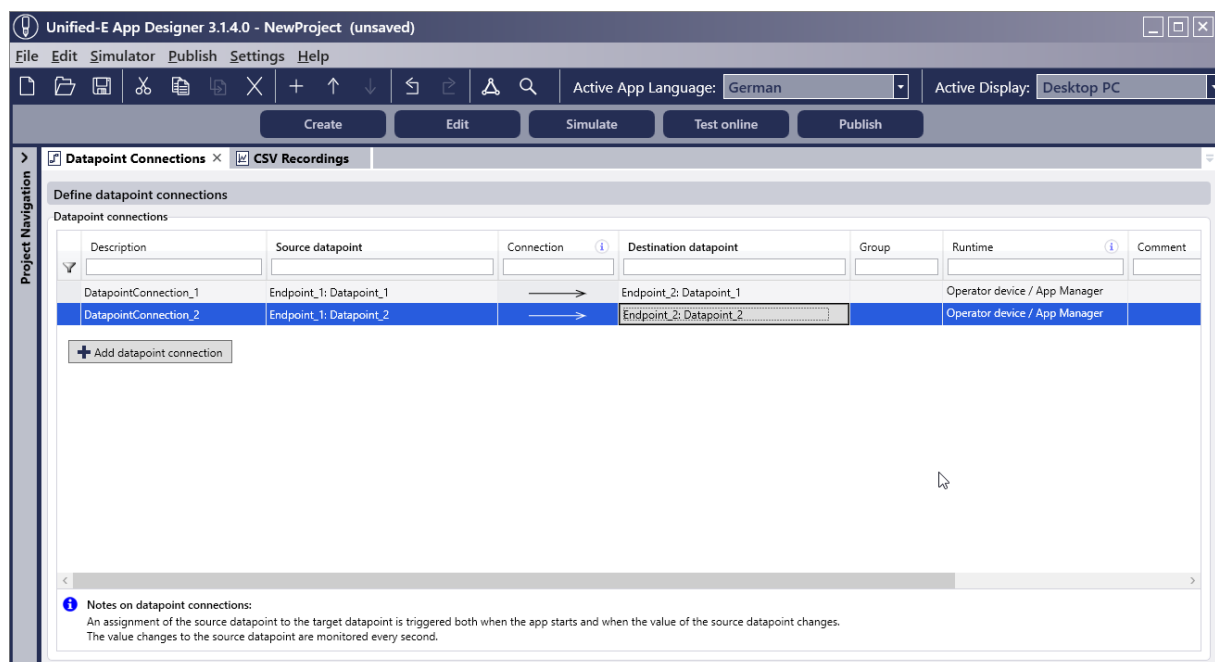
With a datapoint connection, datapoints from different endpoints (e.g. PLC controllers) can be synchronized with each other.

Example: The datapoint "DataPoint_1" should always be synchronized with the value of "DataPoint_2" from another controller. This can be implemented in Unified-E with a datapoint connection, which is active in the Unified-E App Manager as a HMMI server even if no operator devices are connected.

In a datapoint connection, the source datapoint is queried cyclically every second. In the event of a value change, the new value is written to the target datapoint.

To create datapoint connections:

The Datapoint Connections editor can be opened in the Project Navigation under the entry "Data Communication" by double-clicking on "Datapoint Connections". In the editor, new datapoint connections can be created using the "Add datapoint connection" button and configured in the table.



Properties of a datapoint connection:

The properties of a datapoint connection are configured in the datapoint connections table:

- **Description:** Unique name of the datapoint connection
- **Source datapoint:** Datapoint whose value is to be used for synchronization
- **Target datapoint:** Datapoint to be described when the source datapoint is changed.
- **Group:** Used for filtering or sorting in the table
- **Runtime:** Specifies in which application the datapoint connection should be executed:

- Operator device / App Manager: The datapoint connection is optimized for the App Manager, so no datapoints with a "per operator device" runtime can be used. In the case of Direct communication, the datapoint connection is carried out in the operator device.
- Per operator device: The datapoint connection is carried out in the operator device; all datapoints are therefore usable
- **Comment:** Freely definable text for internal use

12 Multilingual HMI Apps

HMI apps can be configured in multiple languages. This can be used to create user interfaces for series machines that are used internationally. For language switching at runtime, the "Language Switcher" view element (under "System button") is available.

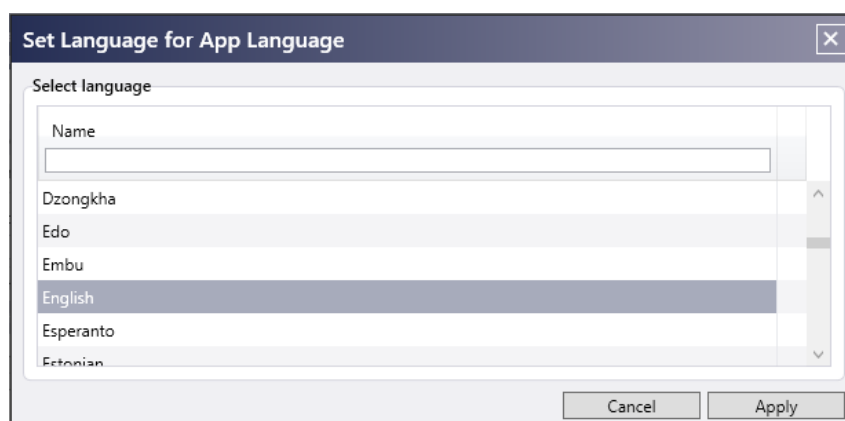
Display texts can be set directly in the properties in the currently set "Active language" (see Chapter 12.2).

In addition, a Language editor is available to manage the display texts or system texts of a specific language in tabular form. There are extensive export and import functions that are very helpful for working with the translation team (see Chapter 12.4).

12.1 Add New Language

When creating a new HMI project, an App Language object is automatically created and adopted based on the current language from Windows.

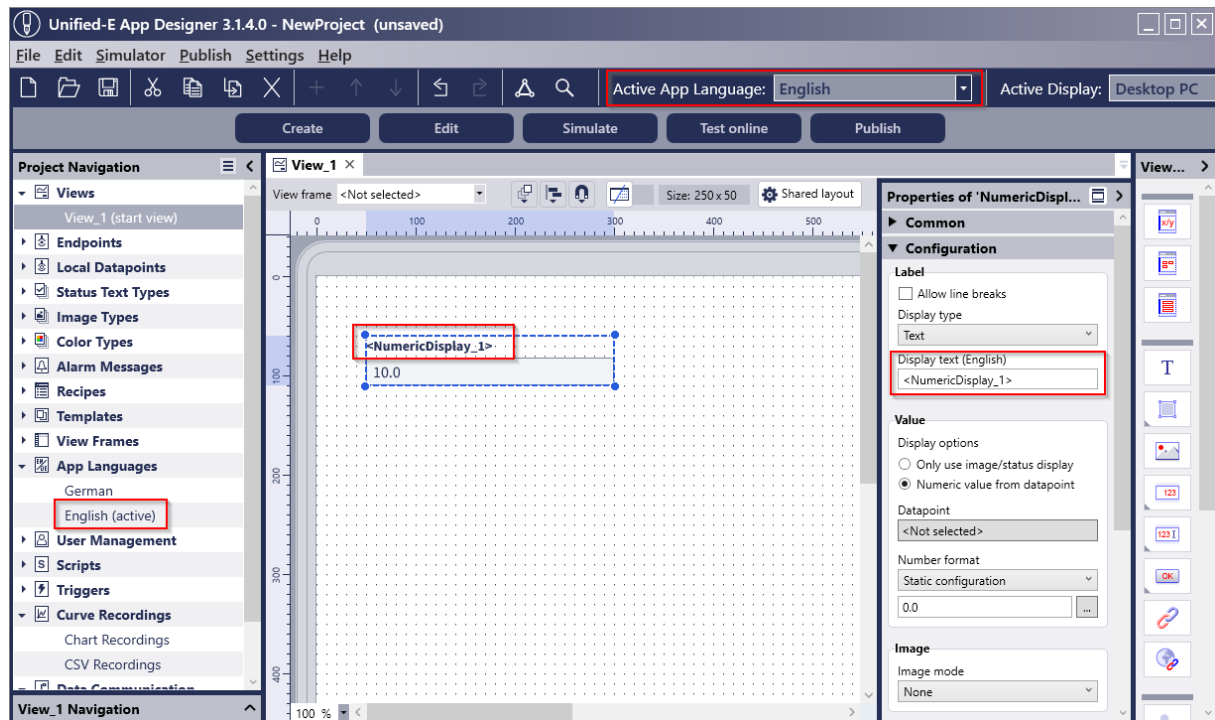
If you want to add another language to the HMI app, you can do so via the Project Navigation under the "App Languages" entry. In the context menu, the menu item "Add App Language" is available for this purpose. When adding, the following dialog appears:



In the dialog, select the desired language that should be linked to the new language object (e.g. English). Display texts must then also be defined for the new app language – as described in the next chapter.

12.2 Set Display Text in Active App Language

The (multilingual) display texts can be set directly in the Properties pane of the view element (or object). The value of the display text always refers to the active app language.



Set active app language (see figure above):

The active app language can be set in the "Active app language" section of the application toolbar by selecting the desired language object. Alternatively, the active app language can be set in the context menu of the desired language object in the Project Navigation (menu item "Set as active language").

Enter display text (see figure above):

To edit the display text (in the figure, the name of a Numeric Display), the view element must be selected. The display text can then be edited in the Properties pane. Alternatively, you can edit the display text directly in the graphical editor:

1. Select Display element with Click
2. Press the RETURN key. This will start the embedded editing
3. In the temporary input field above the text, the display text can be edited

App simulation in active app language:

The currently set active app language is also taken into account in the App Simulator. When starting up in the simulator, the active app language is initially used.

12.3 App Language at Runtime

User texts vs. system texts:

All texts that are defined in the "Display text" input in the Properties pane, as described above, are referred to as "User texts" in Unified-E.

Some view elements display a dialog at runtime for certain actions, which is displayed with text content. These texts contained in the dialog are called "System texts" and cannot be configured in the Properties pane of the view element, but must be explicitly defined in the corresponding language editor (see Chapter 12.4) can be overwritten.

System texts are delivered in the Unified-E App Designer for the languages German and English. For other languages, the necessary system texts must be translated in the Language editor (see Chapter 12.4).

App language in the operator panel (runtime):

Initially, the HMI app is started with the language of the operating system if a language object is defined for it, or alternatively "English" or the first (possibly only) language is used.

For language switching, the view element "Language Switcher" (see Chapter 5.3.4) – typically in the header of all views or in a specific "Settings" view.

When using local user management, the last user language used is used after a user logs on.

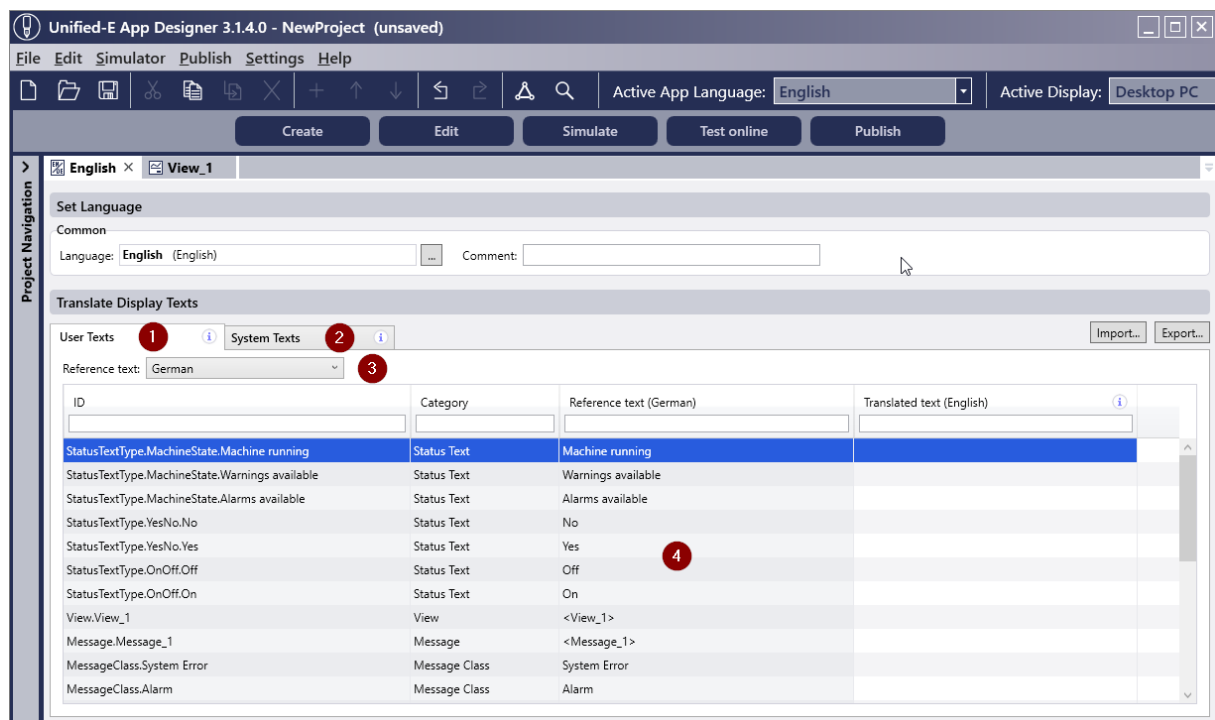
12.4 Manage Text in the Language Editor

The "Language" editor manages all configured texts of an app language and allows you to edit the display texts (or translated texts) in tabular form. Import/export for easy collaboration with translators is easy – see Chapter 12.4.2.

The language editor can be opened in the Project Navigation by double-clicking on the desired language object.

12.4.1 Language Editor Overview

The Language editor is optimized in such a way that the texts can be translated manually as efficiently as possible in the table based on a reference text.



Areas in the Language editor (numbering according to the figure):

1. **User texts:**
Under this tab, the user texts, i.e. the display texts individually configured in the element properties, are listed and editable.
2. **System texts:**
Under this tab, the system texts, i.e. global (non-object-specific configured texts), are to be adapted or translated. System texts can be internal error messages, but also the texts of the dialogs of view elements (e.g. "Create new user" dialog of the view element "User Table").
3. **Reference text:**
This is where the language object is selected, whose texts are to be displayed in the language table in the column "Reference text" for simple translation
4. **Language table:**
In the language table, the texts can be adapted or translated

Columns of the language table:

- **ID:** The ID of the text:
 - For "User Texts": Describes the object path to the object that contains the display text or multilingual text. Note: The menu item "Edit - Show cross references" can be used to open the corresponding object via the cross-reference list
 - For "System Texts": The ID is fixed by Unified-E
- **Category:** Category the text belongs to:
 - For "User texts": Shows the object type to which the text belongs

- For "System Texts": Shows the module or view element to which the text is assigned
- **Reference text:** Displays the corresponding text based on the selected reference text in the "Reference text" area:
- **Translated text:** Input field to adapt or translate the speech text
 - For "User texts": No text is displayed if the entry is empty
 - For "System texts": If the entry is empty, the text is displayed in the basic system text language (see below)

System text base language:

If no adapted system text is available, the following system text is used from the available system texts (currently "German", "English"):

- System text "German": If the operating system language is "German".
- System text "English": Used for all other cases.

Context menu functions in the table:

The following functions are available in the context menu of the language table:

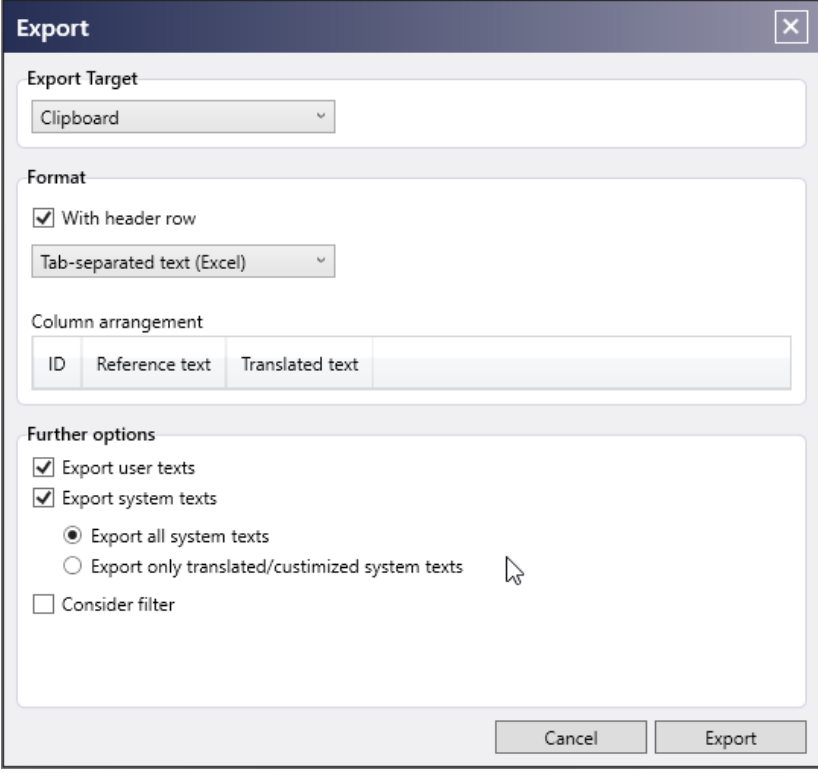
- Take over reference texts: All texts in the "Reference text" column will be transferred to "Translated text"
- Clear translated texts: Deletes all texts in the "Translated text" column

12.4.2 Import and Export Texts

To import and export, there are the buttons "Import" and "Export". This allows translated texts to be imported and exported in a similar way to the import/export of datapoints (see Chapter 2.5).

Here, too, it is recommended to use an Excel file for data exchange, which is easily possible by importing/exporting via the clipboard.

Additional options are available when exporting – see the "Further options" group in the figure:



The image shows a software dialog box titled "Export". It contains several sections: "Export Target" with a dropdown menu set to "Clipboard"; "Format" with a checked checkbox "With header row" and a dropdown menu set to "Tab-separated text (Excel)"; "Column arrangement" with a table showing columns "ID", "Reference text", and "Translated text"; and "Further options" with checkboxes for "Export user texts" and "Export system texts", radio buttons for "Export all system texts" (selected) and "Export only translated/customized system texts", and a checkbox for "Consider filter". At the bottom right are "Cancel" and "Export" buttons.

ID	Reference text	Translated text
----	----------------	-----------------

- **Export user texts:** Exports all texts from the "User Texts" tab of the Language editor
- **Export system texts:** Exports texts of the "System texts" tab
 - **Export all system texts:** Also exports the "original system texts" that have not been adapted – i.e. lines without an entry in the column "Translated/adapted text"
 - **Export only translated/customized system texts:** Only exports rows with entries in the "Translated/customized text" column
- **Consider filters:** Determines whether the table filters should be considered for export

Use case: Translator to translate HMI app texts into "Spanish"

The following procedure describes how the collaboration with the translator could look like.

Assumption: Translator speaks "English" and "Spanish", translator delivers the texts in "Spanish". Texts are already available for the "English" language.

1. Open language editor for "Spanish" (create app language, if not yet done)
2. Select the language "English" as the "Reference text"
3. Click the "Export" button
4. Configure the Export dialog:
 - a. Export target is "Clipboard"
 - b. The other options specify that user texts and all system texts are exported
 - c. Click the "Export" button

5. Open Excel and create a new document
 - a. Select the first cell there and click "Paste" (all texts should be pasted from clipboard)
 - b. Check the inserted rows
 - c. Save the Excel document and send it to the translator
6. Translator fills the column "Translated text" in the Excel document
7. Import revised Excel via Language editor
 - a. Open the Excel document: Copy the three columns "ID", "Reference text" and "Translated text" including the header to the clipboard
 - b. Open language editor for "Spanish"
 - c. Click "Import..." button
 - d. With the help of the ID column, all translated texts are transferred to the "Translated text" column

13 Test the HMI App

The configured HMI app can be tested directly in the Unified-E App Designer – without an operator device and optionally with a simulated endpoint connection. If a connection is established to a physical endpoint, the term "Online" is used in the Unified-E context – in contrast to a simulated endpoint connection.

13.1 Test Endpoint Configuration

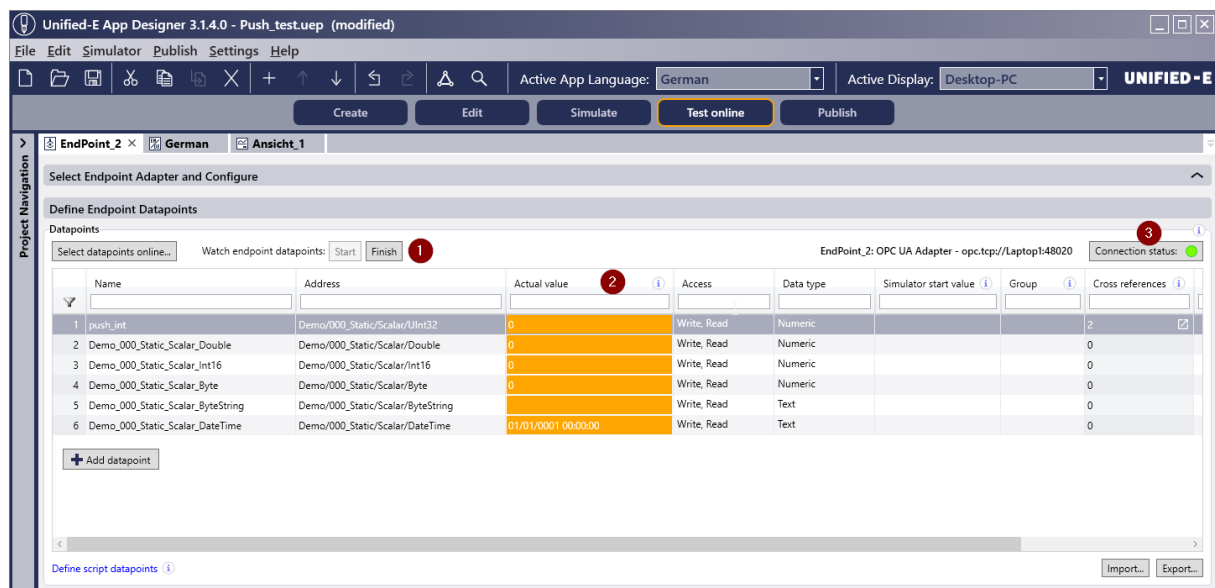
It is highly recommended to test the HMI app in the Unified-E App Designer before going live, checking the endpoint and datapoint configurations.

Test connection:

A connection test for an endpoint based on the configured parameters is performed in the Endpoint editor. In the "Select Endpoint Adapter and Configure" area, the "Test connection..." button is available for this purpose. A detailed diagnosis of connection errors can be found in the chapter 2.2.2.

Test datapoint addresses:

After a successful connection test, the datapoint addresses can be checked for correctness. To do this, click the "Start" button in the "Watch endpoint datapoints" section of the Datapoints table.

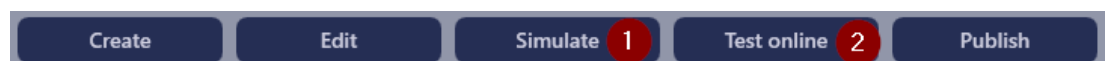


Important elements for testing the datapoint addresses (numbering according to the figure):

1. **Watch endpoint datapoints:** Watching can be started or stopped here using the buttons. When it starts, all endpoints are automatically enabled for watching – not just the endpoint of the current editor.
2. **Actual value:** This column appears as soon as watching is started or an HMI app is simulated. The current value is displayed and automatically updated when it changes.
3. **Connection status:** Clicking on this button opens the Diagnosis dialog with detailed error information – see also chapter 2.2.2.2

13.2 Test the HMI App in the App Simulator

The HMI app is tested in a separate App Simulator window, which is started with the display size of the active display (adjustable via the application toolbar). With a multilingual HMI app, the active app language is adopted when the simulator is started. The start view is displayed, which can be configured in the “Displays” editor (see Chapter 15.2).



The App Simulator is started in the Workflow bar (numbering according to the figure):

1. **Simulate:** Launches the App Simulator with simulated endpoints – see below
2. **Test online:** Launches the App Simulator with connected endpoints – see below

Testing with simulated endpoints:

The HMI app is launched by clicking on the "Simulate" button in the Workflow bar with simulated endpoints. The datapoint values are initialized with the simulator seed value set at

startup. In addition, the values in the "Actual value" column of the datapoints table can be manually changed (or simulated) to check view behavior.

Testing with connected endpoints:

The HMI app is started by clicking on the "Test online" button. A connection to the configured endpoints is established. The current datapoint values are visible in the "Actual value" column of the datapoints table. For safety reasons, these values cannot be changed for connected endpoints.

App Simulator Window Width & Display-Specific Layouts:

The width of the App Simulator window can be changed during the display. Depending on the width and layout configuration, elements can be repositioned or shown or hidden according to the active layout – see Chapter 3.3.

14 Publish the HMI App on the Operator Device

As soon as the configuration of the HMI app is completed, it is to be transferred to the target operator device. This process is called "Publishing" in Unified-E.

The publishing process for the currently open app project is started via the Workflow bar with the Publish button. This will open the publish dialog.

Generate app package file:

For the "Publish" operation, an app package file is generated based on the configuration of the app project. This file contains the display data of the HMI app in a compact form. The subsequent installation or registration of the file takes place depending on the selected type of communication, as described in the following subchapters.

14.1 Publish for Direct Communication

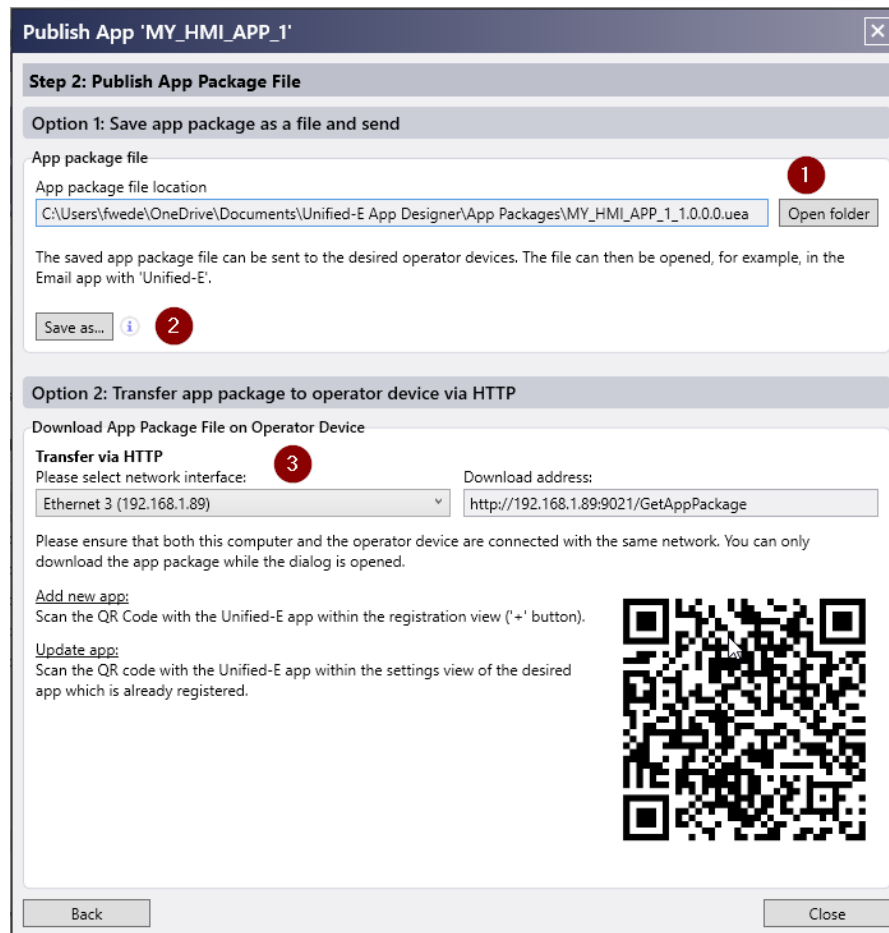
In Direct communication, the operator panel communicates directly with the endpoints without using the Unified-E App Manager as an HMI server.

The generated app package file can be added to the operator device using the Unified-E Client. To do this, select the "Add new App" option in the client and then "Direct communication". For more details, see the Unified-E Client manual.

Publish dialog: Step 1:

In the Publish dialog, select the "Direct communication" option and click "Next". An app package file is created and step 2 of the dialog is displayed.

Publish dialog: Step 2:



Elements of the dialog (numbering according to the figure):

1. "Open folder" button:
Opens the folder in Windows Explorer that contains the created app package file – for example, to copy to a USB stick.
2. "Save As" button:
Allows you to additionally save the app package file to a desired location.
3. Download app package file on operator device (iOS, Android only):
If the operator device (e.g. Android smartphone, iPad) is located on the same local network as the App Designer, the HMI app can be installed directly via a QR code. Please make sure that the correct network interface has been selected in the App Designer.
4. Steps on the operator device:
 - a. Open Unified-E app
 - b. Click on "+" on the start screen,
 - c. Click on "Direct communication - Scanning"
 - d. Scan the displayed QR code
 - e. The HMI app (operator app) is registered

14.2 Publish for Gateway Communication

In Gateway communication, the connection between the operator device and endpoints is made via the Unified-E App Manager, which acts as an HMI server. The Unified-E App Manager is installed on a PC and forwards the requests of the operator devices to the respective endpoints.

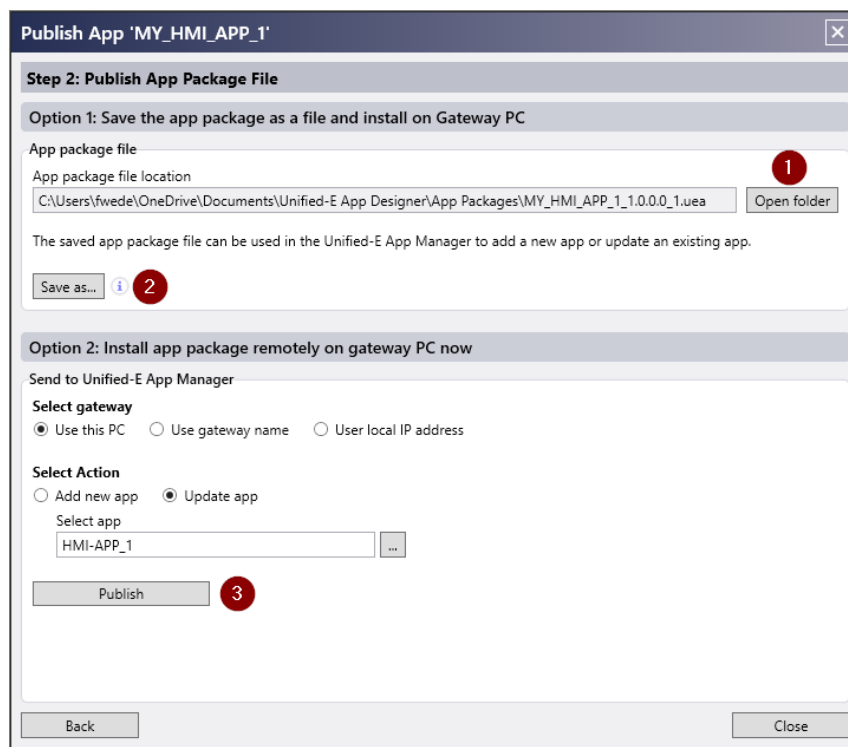
In this case, the app package file must be installed in the Unified-E App Manager. To do this, the app package can be manually selected and installed in the "Installed apps" section by clicking on the "+" button.

Alternatively, the app package can also be installed remotely directly from the App Designer to the desired App Manager during the publishing process, as described in step 2 below.

Publish dialog: Step 1:

In the Publish dialog, select the "Gateway communication" option and click "Next". An app package file is created and step 2 of the dialog is displayed.

Publish dialog: Step 2:



Elements of the dialog (numbering according to the figure):

1. "Open folder" button:
Opens the folder in Windows Explorer that contains the created app package file – for example, to copy to a USB stick.

2. "Save As" button:
Allows you to save the app package file to a desired location.
3. Publish the app package to the selected Unified-E App Manager: The app package can be installed or updated directly on the selected Gateway (Unified-E App Manager):
 - a. Select Gateway:
 - i. Use this PC: The Unified-E App Manager on the local PC is used for publishing
 - ii. Use Gateway name: A remote Gateway is selected by entering the fully qualified Gateway name and password
 - iii. Use local IP address: A Gateway in the local network is addressed via IP address, port and password
 - b. Select action:
 - i. Add a new app: A new HMI app is added in the Unified-E App Manager. The desired operator devices must then be registered in order to use the app
 - ii. Update app: An existing HMI app is updated by entering the app name. Operator devices that have already been registered are automatically supplied with the new version – there is no need to register again

14.3 Start App in the Operator Device

After the HMI app has been successfully added to the operator device, it can be started immediately.

To start an HMI app in the Unified-E Client:

Open the Unified-E app on the operator device. The installed HMI apps are listed on the home screen. Tap the app you want to launch.

The HMI app's user interface is then loaded, and the operator connects to the endpoints – either directly or via the Unified-E App Manager, depending of the chosen communication type.

Further information:

For more detailed information on adding, managing, and launching HMI apps on HMI devices, please refer to the separate Unified-E Client manual and the "Getting Started" section of the official website.

15 Project Settings

The project settings include central or global resources or settings of the HMI project. The corresponding editors can be started in the Project Navigation under the entry "Project Settings".

15.1 App Properties

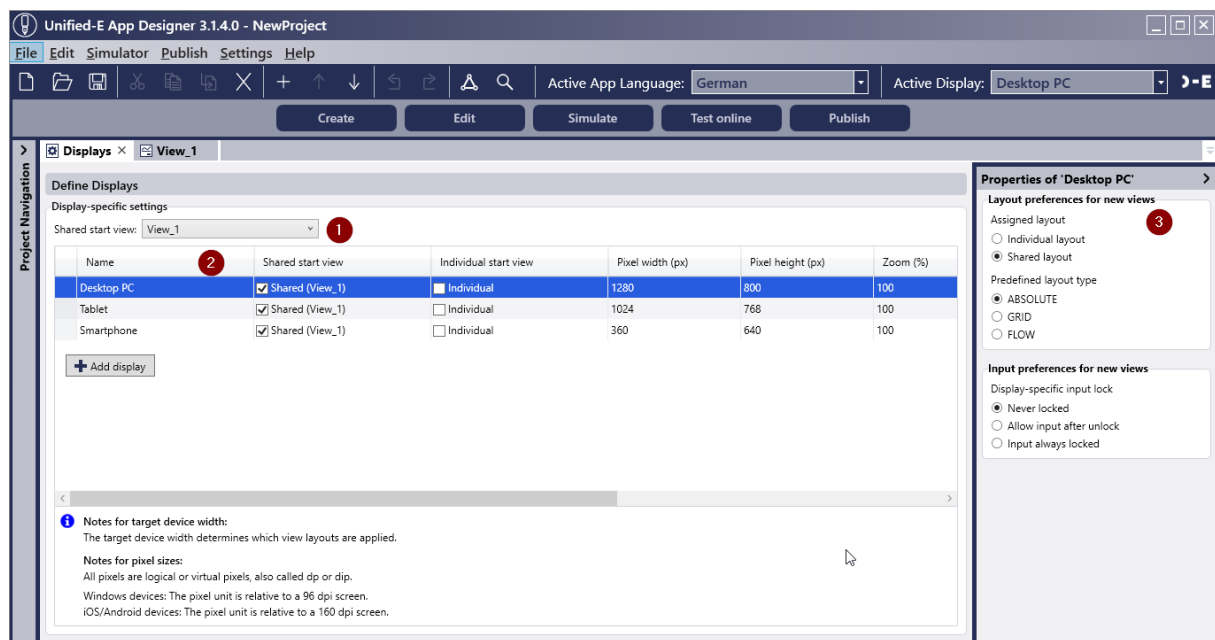
The App Properties editor describes general properties that do not affect the runtime in the operator device, but have an informational character.

- **App name:** Denotes the name of the HMI app. This is read-only and is automatically taken from the file name of the HMI project file.
- **Version:** Freely definable version, which is displayed in the App Manager under the general app information
- **App ID:** This ID should uniquely identify the HMI app and prevent an app package from accidentally overwriting another app when updating. When updating, the app ID of the app package must match that of the app already installed
- **Publisher:** Describes the publisher of the HMI app. This information is displayed in the App Manager under the general app information
- **Description:** Multi-line description of the HMI app, which is displayed in the App Manager under the general app information

15.2 Displays

The Displays Editor defines the displays (screen sizes) of the operator devices on which the views of the HMI app are to be shown. When testing the HMI app, the actively set display is taken into account (see Chapter 13.2).

Properties for new views can be preset per display. These presets can be individually adjusted (overridden) later in the view properties. In addition, the start view is defined in the displays editor on a display-specific basis – i.e. the view that is to be opened initially when the HMI app is started.



Areas of the Display editor (order as shown):

1. Shared start view:
Typically, all displays use the same start view. This can be defined here and must be selected per display.
2. Displays Table: The displays are created and configured in the table. The main features of the displays are described below.
3. Properties pane: This is where additional properties of displays are managed.

Features in the table:

The most important display properties can be configured in the table:

- Name: Unique name of the display
- Shared start view / Individual start view: Determines whether this display uses the shared or an individual start view
- Pixel width (px): Specifies the pixel width of the display. This is taken into account in the layout and simulation (see Chapter 3.3)
- Pixel height (px): Specifies the pixel height of the display. This does not determine the active layout, but is used when starting the App Simulator and in the View editor
- Zoom (%): Zoom value which scales the views on this display
- Target device width (px): Read-only field that shows the target device widths at which this display is active. The target device width determines which "display" is active. Display-specific configurations (e.g. layout or input lock) are applied accordingly
- Comment: Freely definable text

Property group "Layout preferences for new views":

This defines how the layout type properties should be preset for new views (see Chapter 3.3).

Property group "Input preferences for new views":

Here it is defined whether the input should be locked by default for new views (see Chapter 5.4.1).

15.3 Images

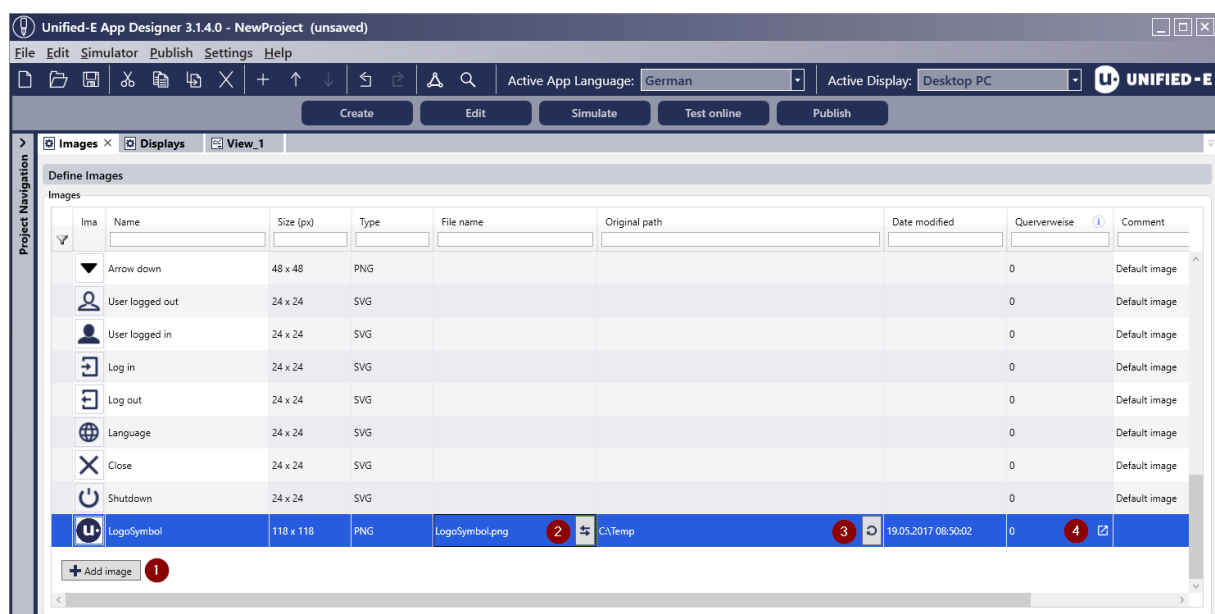
All images that are to be used in views or objects must be listed in the Images Editor. Only then will they be available for selection in image properties (see Chapter 4.1.2). All images listed in the Images Editor are saved in the HMI project file and embedded into the app package.

The following images can be integrated:

- SVG Vector Graphics
- Pixel-based image files such as JPG, PNG, BMP. Internally, these images are stored as PNG files in the HMI project file

15.3.1 Add and Configure Image

Images can be inserted into the image table using the "Add image" button and then configured directly in the table (see below).



Important buttons (numbering according to the illustration):

1. **Add Image:**
Creates a new Image object. In the "Add Image" dialog that then appears, select the corresponding image file.
2. **Select File:**
This button appears as soon as an image object is selected. This assigns an image file to the image object.
3. **Synchronize Image Object:**
Updates the image object based on the original path – for example, if the original image file has been modified.
4. **Cross References:**
Opens the Cross References window, which displays the usages of the image in the project.

Columns of the picture table:

- **Image:** Shows the assigned image as a thumbnail
- **Name:** Unique name of the image
- **Size:** Read-only field, shows the original size in pixels
- **Type:** Displays image type (SVG or PNG)
- **File name:** Displays the original file name of the image file
- **Original path:** Shows the original folder path of the image file
- **Date modified:** Shows the timestamp of the file at the time of mapping
- **Cross references:** Displays the number of usage points in the HMI project
- **Comment:** Freely definable text

15.3.2 Add Images Using the View Editor

In the View editor, new images can be added in the following ways without switching to the Images editor.

Adding images by image file selection:

In the properties of the view element, an image file can be selected in the "Image" property group using the "..." button. This is automatically added to the Images editor and is therefore also available in other view elements.

Adding images via the clipboard:

Images can be copied to the clipboard and then pasted via the context menu using the "Paste" menu item in the active view editor. The following happens:

- A new view element of type "Image" is created and placed at the context menu position

- The image from the clipboard is automatically added to the Images editor
- The "Image" view element is automatically linked to the newly added image in the Images editor

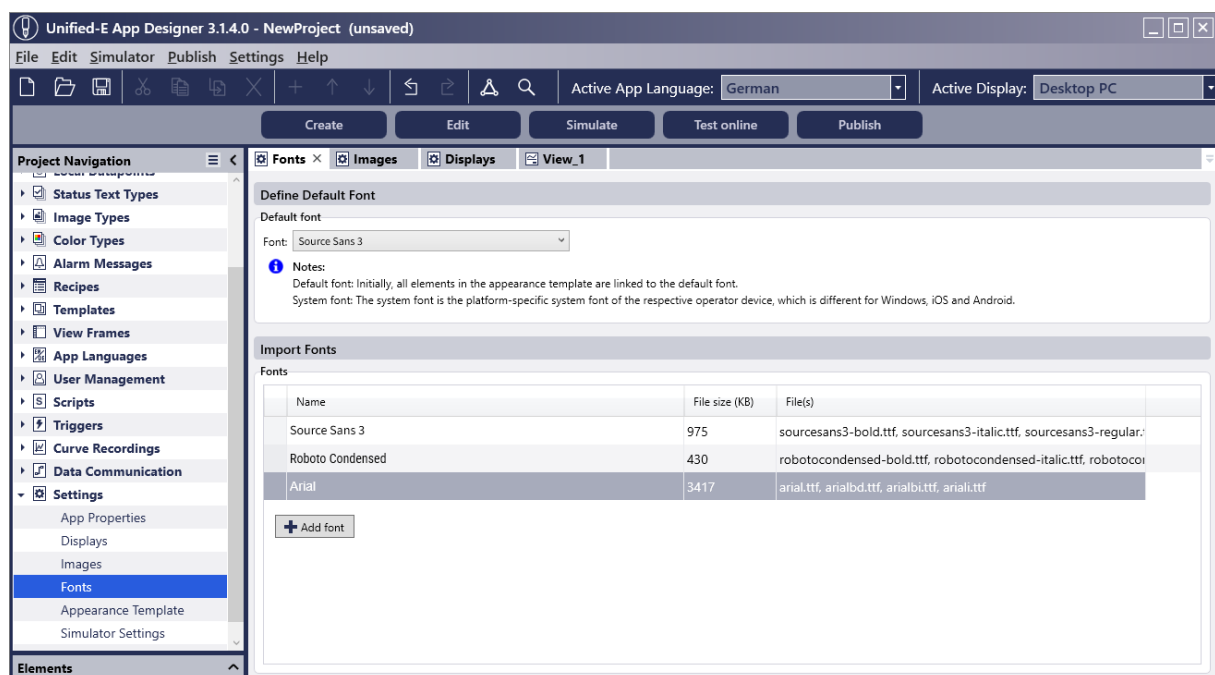
15.4 Fonts

In the Unified-E App Designer, you can use your own fonts without having to install them manually on the target operator device.

Standard fonts are stored directly in the HMI project – analogous to the handling of images. This means that no external file references are necessary. The app package used when deploying the HMI app to the operator automatically includes all embedded fonts.

This ensures that the HMI app is displayed with the intended fonts on all target devices – regardless of which fonts are installed locally on the device.

Fonts are managed in the Fonts editor, which is opened in the Project Navigation under the "Project Settings" entry by double-clicking on "Fonts".



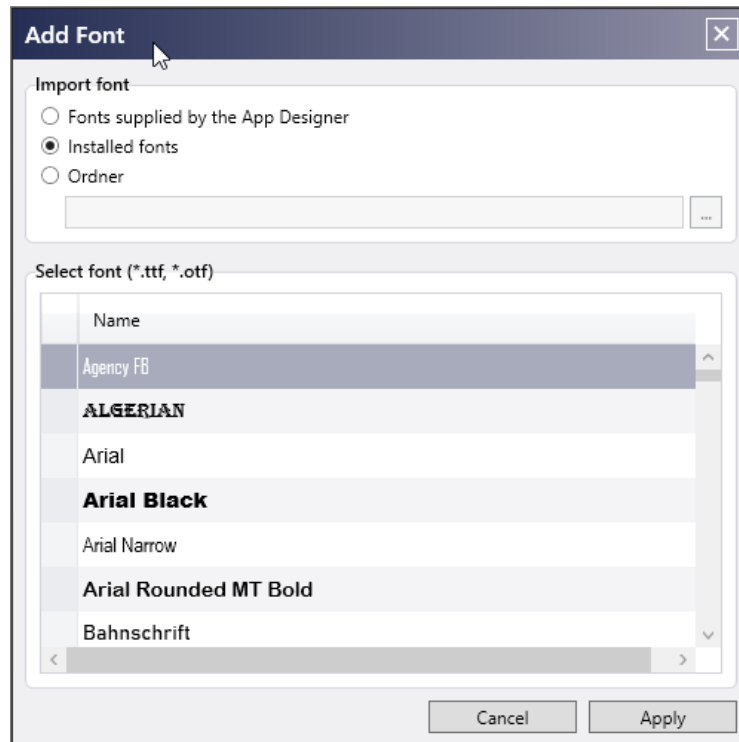
15.4.1 Define the Default Font

At the top of the Fonts editor, select the default font that must be included in the Fonts table.

The default font applies to all view elements, but can be overridden:

- Override in Appearance Template editor: Here, an individual font from the Fonts table can be assigned to a view element. (see Chapter 15.4.1 **Error! Reference source not found.**).
- Overwriting the view element: The default font or the font set for the appearance Template can be overwritten in the "Appearance" palette.

15.4.2 Import Fonts



Options for importing fonts:

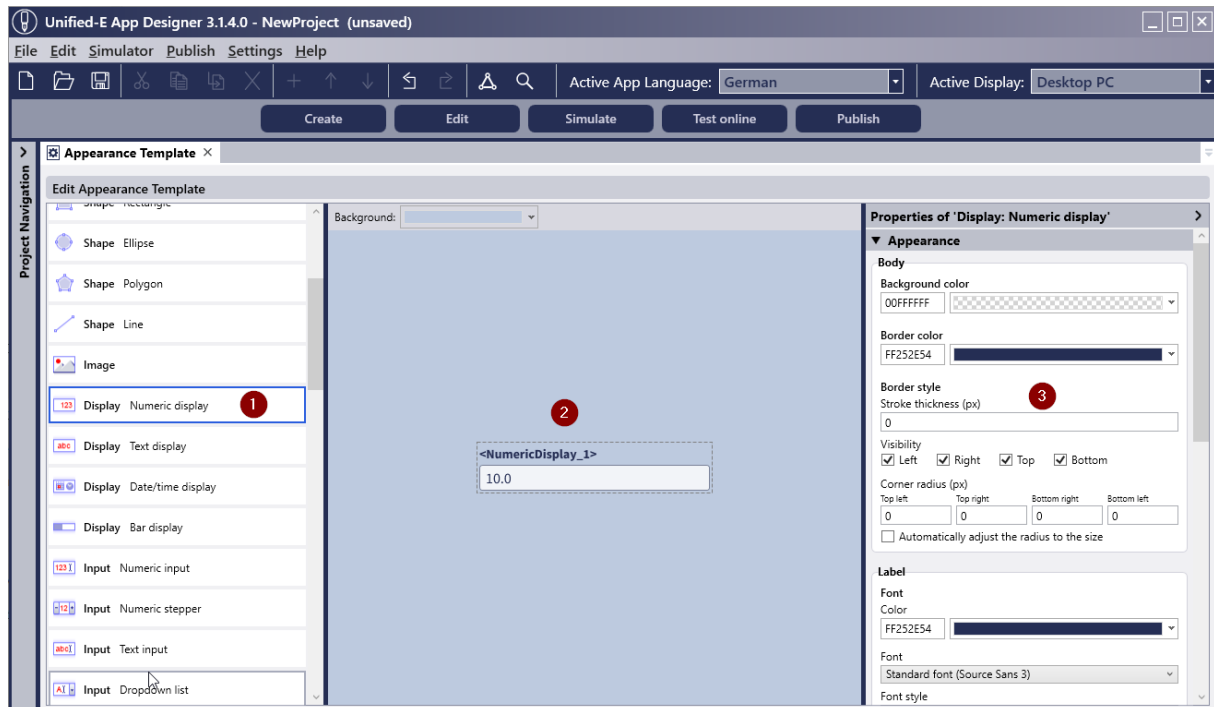
- Fonts supplied by the App Designer: This option offers the fonts built into the App Designer for selection
- Installed fonts: All fonts installed in the Windows system (TTF or OTF fonts are listed for selection
- Folder: This option selects the folder with font files. The fonts found are available for selection

15.5 Appearance Template

The Appearance Template editor is used to define the uniform appearance of the view elements in the project. The goal is to provide a central location where the visual design can be defined and managed according to the desired corporate identity. Changes in the Appearance Template editor automatically affect all affected view elements – as long as the properties in question are still linked to the appearance template.

Open editor:

The Appearance Template editor can be opened in the Project Navigation under Project Settings → Appearance Template.



Structure of the editor (numbering according to figure):

1. Element selection (left): This list shows the available view element types, such as Text, Shapes, Image, Numeric Display, Text Display, and so on. Select the element whose appearance template you want to edit.
2. Preview Area (center):
The center pane displays a preview of the selected View item. It shows the currently configured theme based on the rendering appearance template. Changes to properties are visible directly in the preview. The background color of the preview is customizable at the top with a color picker.
3. Properties panel (right):
On the right, the pre-configurable properties are displayed in different palettes:
 - a. Appearance: background color, border color, border style, border visibility, corner radius
 - b. Alignment: Horizontal and vertical alignment
 - c. Margins: Inner and outer margins
 - d. Effects: Shadow and transparency

Edit Properties:

All properties can be adjusted directly in the Properties panel. The changes apply project-wide to newly created view elements of this type – as well as to existing view elements, as long as their properties are still linked to the appearance template.

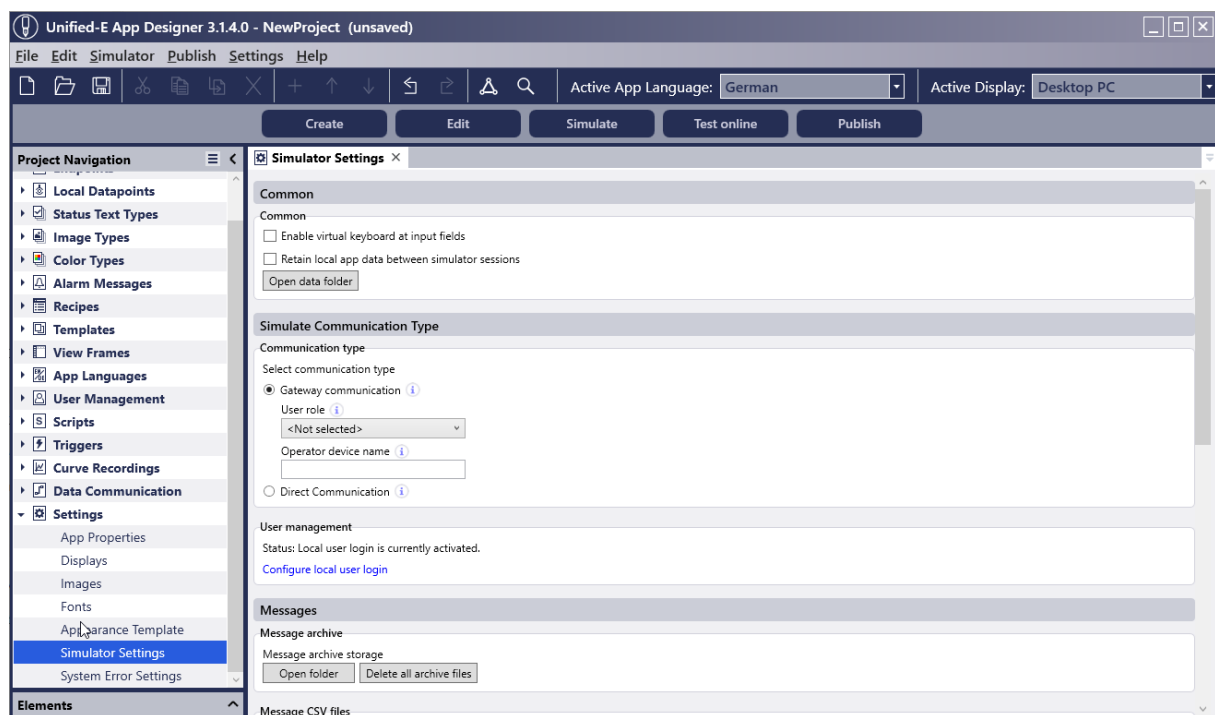
Element properties that are still linked to the appearance template appear as non-editable (grayed out) in the View editor. Instead, they display the current value from the appearance template. This link can be separated in a targeted manner if custom adaptation is necessary.

Switch to the appearance template of the selected element quickly:

In the palette header of the Properties pane in the View editor, there is an icon for quick switching to the appearance template. This allows you to switch directly to the definition of the appearance template for the selected view element.

15.6 Simulator Settings

In the Simulator Settings editor, the behavior of the App Simulator for the HMI app simulation (see Chapter 13.2). In addition, the files recorded by the simulator for messages or curve recordings can be displayed here in Windows Explorer.



Common:

The following options are defined here:

- **Enable virtual keyboard for input fields:** If set, a virtual keyboard will be displayed as soon as an input field is clicked
- **Keep local app data between simulator sessions:** If set, the app data will not be deleted every time the App Simulator is started, but the data from the last session (e.g. added users in the User Table) will be retained

Simulate communication type:

Here, the communication type "Gateway communication" or "Direct communication" can be selected for the app simulation. This affects the permissions at startup or user management if user roles and permissions have been configured.

- **Gateway communication:** The user role is to be selected for the simulated remote device, local user management is disabled here (analogous to registering with the App Manager without local user management)
- **Direct communication:** If local user management is activated, the HMI app starts without permissions, the user must first log in with the Authentication Button. If user management is disabled, the configured user roles are ignored

Messages: Message archive:

Here, the message archive (an SQLite database), which serves as the data source for the Archive Message Table, can be displayed via the "Open folder" button in Windows Explorer. The "Delete all archive files" button can be used to delete all archived files.

Messages: Message CSV files:

Here, the message CSV files generated by the App Simulator can be displayed via the "Open folder" button in Windows Explorer. The "Delete all CSV files" button can be used to delete all message CSV files.

In addition, the following can be set for CSV file logging for the simulator:

- **Language:** Language in which the display texts (e.g. message texts) are to be logged
- **CSV output format:** timestamps, numbers, and delimiters: specifies the exact formatting of the CSV file:
 - Use local culture settings: Formatting is applied according to the operating system's local locale settings.
 - Use culture-independent settings (invariant): The decimal separator is ".", the column separator is ",". The timestamp follows the format yyyy-MM-dd HH:mm:ss.

Recipes:

Here, the recipe datasets can be displayed via the "Open folder" button in Windows Explorer. The "Delete all recipe data" button can be used to delete all recipe data.

Curve Recordings: Chart recordings:

Here, the chart recordings (XML files) can be displayed via the "Open folder" button in Windows Explorer. The "Delete all chart recordings" button can be used to delete all chart recording files.

Curve Recordings: CSV recordings:

Here, the CSV recordings generated by the App Simulator can be viewed via the "Open folder" button in Windows Explorer. The "Delete all CSV files" button can be used to delete all CSV recording files.

In addition, the CSV recording in the simulator can be set to the following:

- **CSV output format:** timestamps, numbers, and delimiters: specifies the exact formatting of the CSV file:
 - Use local culture settings: Formatting is applied according to the operating system's local locale settings.
 - Use culture-independent settings (invariant): The decimal separator is ".", the column separator is ",". The timestamp follows the format yyyy-MM-dd HH:mm:ss.

15.7 System Error Settings

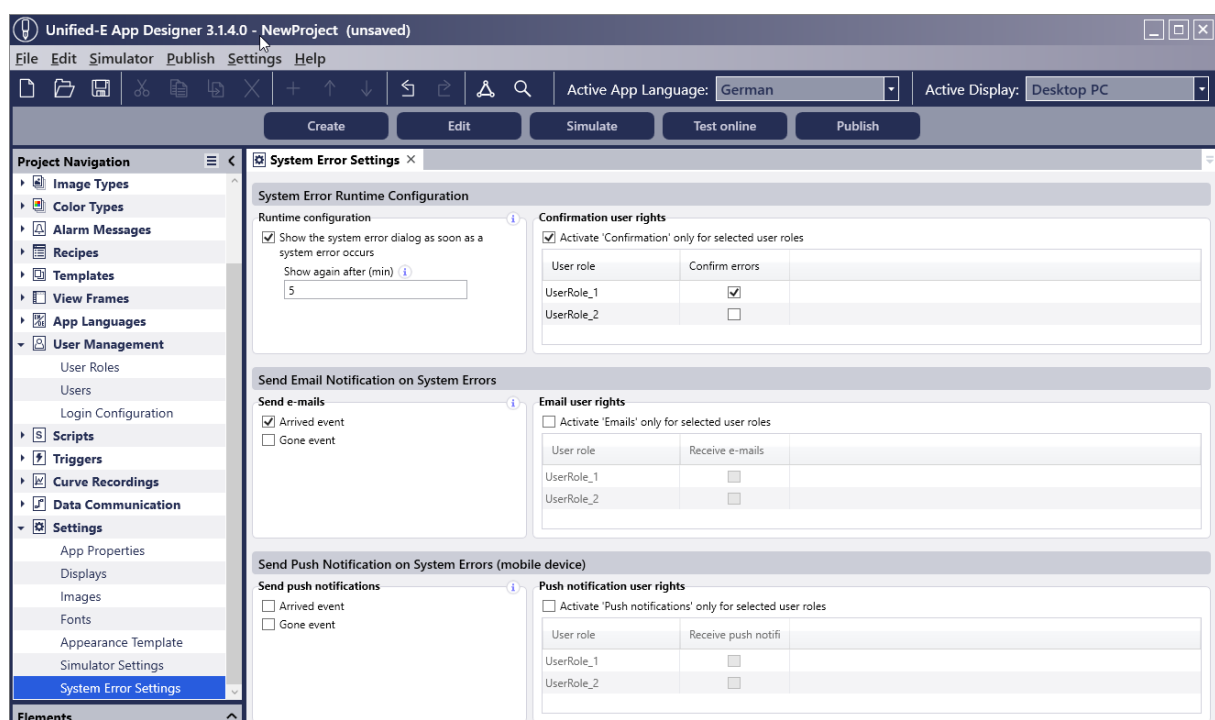
In the System Error Settings editor, the occurrence of system errors can be configured.

What is a system error?

In contrast to user-defined message events, system errors are malfunctions that can occur in the App Manager or operator device during operation.

Examples are: Connection failure to the endpoint, datapoint error reading an endpoint, script error.

In principle, system errors should not be ignored, as this could have fatal consequences for traceability or even for the production process. In the System Error Settings editor, you can configure how and which users should be informed when a system error occurs.



15.7.1 General Behavior in Case of System Errors

This defines the general behavior when system errors occur.

Property group "Runtime configuration":

Under "Runtime configuration", you can set whether a dialog should be displayed in the HMI app when a system error occurs. Optionally, it can be specified after how many minutes the same error may be displayed again.

- **Show System Error Dialog:** Activates the display of a System Error dialog when a system error occurs
- **Show again after (min):** The system error dialog will be displayed again after the entered time has elapsed if the error has not been confirmed by then. The confirmation of system errors can be done in the system error dialog itself (if authorized) or in the App Manager during Gateway communication

Property group "Confirmation of user rights":

This specifies which users are allowed to confirm a system error, so that the system error is displayed again from the Message Table or not again in the system error dialog (see also chapter 4.1.11 for configuring permissions).

15.7.2 Email Notification in Case of System Errors

Here you can specify when and to which user roles an email should be sent (only possible for Gateway communication with App Manager).

Property group "Send emails":

Specifies at which event phases emails should be sent:

- **Arrived event:** The email is sent as soon as the trigger condition has been met again
- **Gone event:** The email is sent as soon as the trigger condition is no longer met

Property group "Email user rights":

Specifies the user roles for which the email should be sent. Accordingly, only registered e-mail recipients with the appropriate role will receive a message (analogous to chapter 4.1.11).

15.7.3 Push Notifications in Case of System Errors

This specifies when and to which user roles a push message should be sent to a mobile device (only possible for Gateway communication with App Manager).

Property group "Send push notifications":

Specifies at which event phases push messages are to be sent:

- **Came event:** The push message is sent as soon as the trigger condition has been fulfilled again
- **Gone event:** The push message is sent as soon as the trigger condition is no longer met

Property group "Push notification user rights":

Specifies for which user roles the push message should be sent. Accordingly, only registered mobile devices with a matching role will receive a message (analogous to the 4.1.11).

16 General Features

16.1 Open Another App Designer Instance

In the main menu under "File: Start new application instance", another App Designer instance can be quickly started. This allows objects to be conveniently copied between two HMI projects – either via the clipboard or via drag & drop.

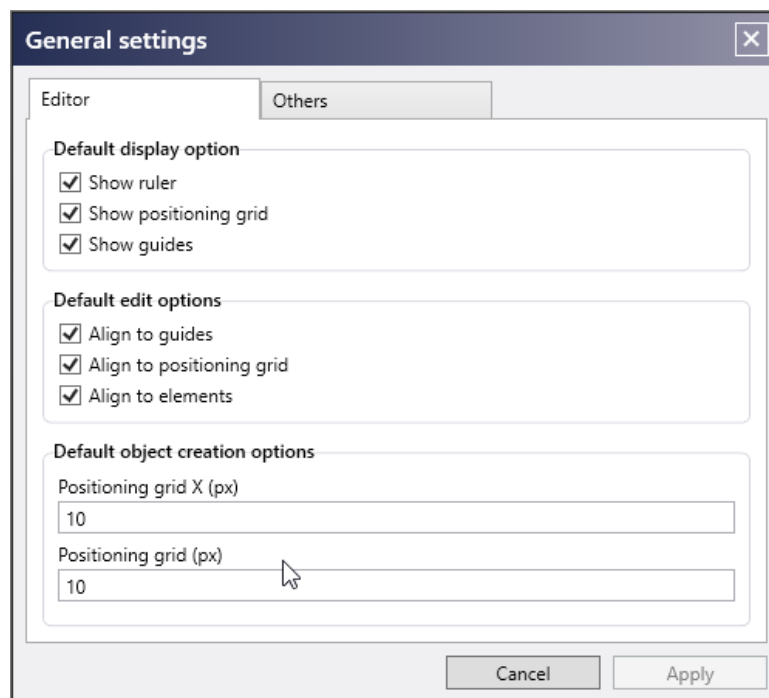
16.2 General Editor Settings

Cross-project preferences, for example for new HMI projects, can be configured in the General Settings dialog. This can be accessed via the main menu of the application under the menu item "Settings" → "General Settings".

The general settings are divided into two tabs: "Editor" and "Other". The following chapters describe the options within the "Editor" tab.

16.2.1 Editor Settings

The options listed here apply only to the graphical area of the View editor. (see 3.1).



Property group "Default display options":

These options determine which visual aids appear by default in new projects:

- **Show ruler:** Shows a ruler at the edge of the canvas at the start of the project
- **Show positioning grid:** By default, activates the grid in the absolute layout
- **Show guides:** Initially displays new guides in the project

Property group "Default editing options":

These settings define the default behavior when dragging and dropping elements into the absolute layout:

- **Snap to guide:** Automatically align elements to existing guides
- **Snap to Grid:** Elements automatically snap to the grid
- **Snap to Element:** Elements automatically align to adjacent objects

Property group "Default for new objects":

Specifies the grid size in pixels that is used for newly created views in the absolute layout:

- **Positioning grid X (px):** Horizontal grid spacing
- **Positioning grid Y (px):** Vertical grid spacing

16.2.2 Others

The following global properties can be configured under the "Other" tab.

Property group "Language":

Here you define the language of the user interface of the Unified-E App Designer: German or English.

Property group "App package":

- **Embed project in app package when publishing:** Determines whether the entire HMI project is embedded in the app package file

Note: The app package file is technically a ZIP file. By renaming the file extension ".uea" to ".zip", the embedded project file can be extracted – for example, if the original project file is no longer available.

Property group "Default storage paths":

Here you define default paths for opening and saving:

- **Projects:** Default directory for opening and saving HMI projects
- **App packages:** Directory in which the generated app package file is initially stored

17 Appendix

17.1 Tips and Tricks

Common header and navigation bar in multiple views:

Many HMI apps use the same header and navigation pane in all views. Such central elements can be implemented easily and in a reusable way using the View Frame concept (see Chapter 3.9).

Define click areas in the image (hotspots):

Example: A plant image should contain interactive click areas.

This can be implemented as follows:

1. Place the plant image with the "Image" view element
2. Place the "Shape" view element (e.g. rectangle) over the system image in the absolute layout
3. Configure the Shape in Fill mode with a transparent background
4. Assign a click action in the "User Actions" palette (e.g. View Navigation, set datapoint)

To configure animation in a view:

Example: Elements on a conveyor belt should move visually.

This is achieved by dynamically controlling the X and Y positions of the view element in question (e.g. image) via datapoints (Palette "Position").

Then, the position value is not fixed, but read from a datapoint at runtime.

These animation datapoints can, for example, be defined as local datapoints with the runtime set to "Per operator device" and updated using a periodic trigger action (see Chapter 9).

Import and export recipe datasets:

Recipes are stored as XML files in the file system and can therefore be easily transferred between HMI devices (see Chapter 8.7).

Remote monitoring with your smartphone:

With the Unified-E App Manager, a system can also be operated via a smartphone.

In the Message Event properties, you can define when a push notification should be sent to the smartphone (see Chapter 6.1.1.4.5).

Direct PLC communication with Android or iOS device:

Unified-E supports Direct communication with many PLC systems, including on Android and iOS devices, without the need for additional servers or hardware (see Chapter 1.3.4).

Setting up M2M communication:

Datapoint synchronization between multiple endpoints (e.g. PLC controllers) can be set up in the Datapoint Connections editor (see Chapter 11).

Copying between HMI projects:

The Unified-E App Designer can be started several times, e.g. via the menu "File: Start new application instance". Objects can thus be copied across HMI projects via the clipboard or via drag & drop.

Working with multiple screens:

The individual editors can be detached from the main window via the tab in the main area and placed on a second screen.

17.2 Support and Further Information

For further information on the use of the Unified-E App Designer, please visit our website at www.unified-e.com. In particular, the "First Steps" section offers a compact introduction with clear examples and frequently asked questions.

If you need technical support or have specific questions about your configuration, you can always contact our support team. To do so, please send an email to:

support@unified-e.com

We will do our best to respond to your request as soon as possible. Please include relevant screenshots or a short description of your project structure in your email – if available – to enable efficient editing.