zenon manual

Batch Control

v.8.20

Contents

1 Welcome to COPA-DATA help

ZENON VIDEO TUTORIALS

You can find practical examples for project configuration with zenon in our YouTube channel (https://www.copadata.com/tutorial_menu). The tutorials are grouped according to topics and give an initial insight into working with different zenon modules. All tutorials are available in English.

GENERAL HELP

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

PROJECT SUPPORT

You can receive support for any real project you may have from our customer service team, which you can contact via email at support@copadata.com.

LICENSES AND MODULES

If you find that you need other modules or licenses, our staff will be happy to help you. Email sales@copadata.com.

2 Batch Control

The module Batch Control offers the possibility to automate batch-orientated manufacturing processes for lot products. The module complies to ANSI/ISA–88.01–1995 also known as ANSI/ISA–S88.

For Batch Control there are two different editors available in the Runtime.

- Matrix editor: (on page 208) For simple, sequentially or parallel running recipes.
- PFC editor: (on page 166) For complex recipes with branches.

Depending on the license, either both editors or only the matrix editor is available to you.

⚠Attention

Notes in the help embedded in the Editor on VBA keywords are not functional in the current version.

NOTE FOR CHANGE FROM 7.00 TO 7.10 OR HIGHER

Before converting a project to a new zenon version all recipes must be completed. Running recipes are not executed any longer after a restart. The restart only functions within the same zenon version.

Attention: Projects with recipes that were created in zenon 7.10 or higher cannot run in zenon 7.0.

EFFECTS OUTSIDE BATCH CONTROL

When using different versions for Editor and Runtime, problems can occur if the Batch Control module is licensed but Batch Control is not used.

Background: Some files are compiled for Runtime as soon as the module is licensed. A batch project that was compiled with 7.0 cannot be executed in 7.10.

Solutions:

- Compile the project with an Editor of version 7.10 or higher.
- Use a Runtime of version 7.0.
- Use a license that does not include the Batch Control module.

BATCH CONTROL AND COMMAND SEQUENCER MODULE

If both the Batch Control (on page 7) module and the **Command Sequencer** module, which both require a license, are licensed at the same time, selection of the module used in the Runtime is carried out by means of the project setting.

To select the preferred module in the Runtime:

- Click on the node of your project in the Editor.
- Go to the **Runtime settings** project properties group.
- Select, for **License module in the Runtime** of the **Preferred module** property, **Command Sequencer**(default) or *Batch Control*.

The selected model is then available in Runtime for further project configuration.

3 Introduction

The module Batch Control consists of three parts:

1. **The engineering environment (on page 15) in the zenon Editor:**
   Here, all units (ISA nomenclature, chapter 4.2.5: units) and unit classes with their phases (ISA - 5.1.2.4: phases) and reactions are created. The phases must have an equivalent in the control (ISA: Equipment Control). This is called technological function (ISA: Process Action). Batch Control reflects the physical model in accordance with ISA 4.2 as flat hierarchical level based on units.
   The other levels of the model such as process cell, area, plant, etc. were deliberately forgone. When creating the batch recipes too, only the lowest level (phases) of the ISA structure model 5.1 and operations were implemented. Additional levels such as unit procedures and procedures are not available.
2. **PFC editor (on page 166) and Matrix editor (on page 208):**
   With the help of these editors, the master recipes are created in zenon Runtime (ISA: Master recipes). The control recipes that can be executed (ISA: Control recipes) are derived from these (see also ISA 5.3.1.). During the process the exact status of the Batch recipe is displayed in the respective editor and you can interfere in the recipe process.
3. **Recipe Execution Engine (on page 255) (REE):**
   The REE is directly integrated in the zenon Runtime and executes a Batch recipe automatically in the background. Via commands such as **Start**, **Pause**, **Stop** etc. the user can control the REE. There are three possible modes: *Automatic*, *Semi-automatic* and *Manual*.

SPECIAL FEATURES OF MODULE BATCH CONTROL:

In contrast to most other zenon modules, a large part of the engineering - the recipe creation - is done in the Runtime and not in the Editor. This entails special features which are dealt with in the respective chapter. So for example changed phases are no longer transferred to an already released master recipe in order to prevent unwanted data changes.

The module is designed in a way which makes it completely independent of the control. This means that the data communication take place via all available zenon drivers with any PLCs or even RTUs. They only execute the process actions. The entire recipe processing is done at the computer in the REE. For changes on the Batch recipe or for new master recipes, no modifications are necessary in the PLC code.

The module follows the strict separation between the procedure of the batch recipe (ISA: Procedural Control Model) and execution of the technological function (ISA: Process Model) as described in ISA–S88, chapter 5.2.1.

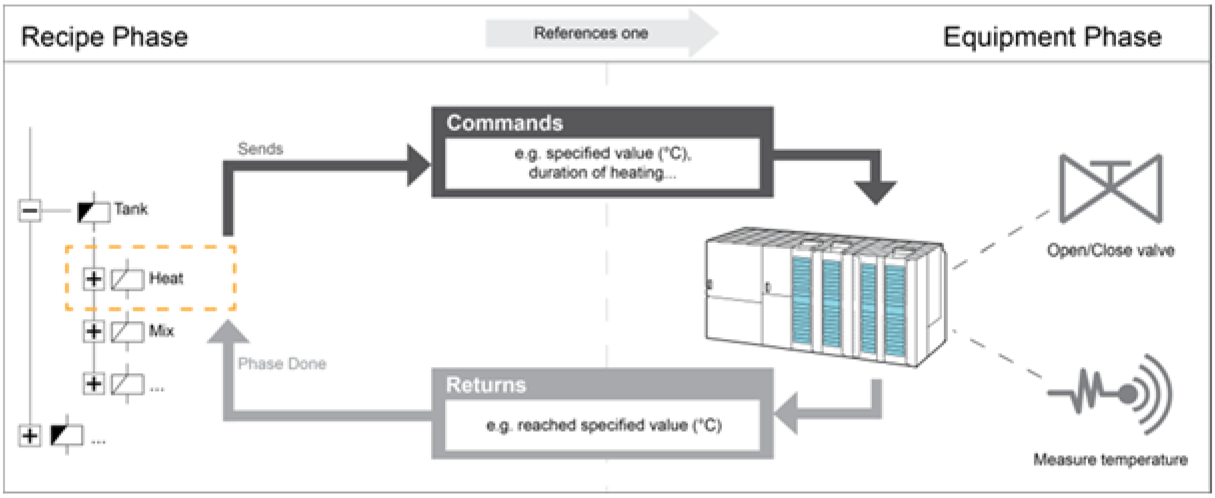
PRINCIPLE STRUCTURE OF THE COMMUNICATION

For the communication with the process standard zenon variables are used. As variable names often have a very technical and thus difficult-to-read label, an additional abstraction level was introduced. It contains the tags which are available in two types:

- Command tags: Transfer set values to the PLC when a phase is executed.
- Return/PLC sync parameter: Read back values from the PLC for evaluations.

With both tag types, values can be both written (e.g. in reactions) and also read (e.g. in conditions).

Schema:



The PLC communicates with the zenon driver. The driver communicates with the zenon Runtime. The Runtime sends the values to the REE where they are processed. The REE internally works asynchronously to the zenon Runtime in a 100 ms cycle.

NETWORK

The module **Batch Control** is fully capable of using a network in terms of Client/Server technology. This means that Batch recipes can be created, duplicated, edited, deleted, etc. on a Client. The whole recipe management remains always on the server. Likewise the whole process control such as **start** recipe, **pause** recipe, **stop** recipe, etc. can be done from the Client. Also mode changes and manual operations such as **jump** are possible.

⚠Attention

Module Batch Control does not support redundancy. There is no synchronization between Standby Server. When the Server breaks down, the executed Batch recipes are not continued on the Standby! Recipes can also be started whilst the configured Server 2 is the primary server.

4 Terminology

The following terms are used in the zenon **Batch Control** module:

Term	Definition
Unit	Physically available machine or equipment part with which phases can be carried out. (*ISA 88: Unit*)
Releasing the unit	Element of module Batch Control which cancels the allocation of a unit in the unit manager. With this the unit can be allocated by another recipe again.
Unit allocation	Element of module Batch Control which causes the allocation of a unit in the unit manager. An allocated unit can only be used by phases with the recipe. With this the unit is locked for phases of other recipes which are executed parallel.
Unit instance	Unit that is based on a unit class. All properties of the unit class are applied to the unit instances and can be amended there.
Unit class	Template for unit. Unit instances are created on the basis of this template. All properties of the unit class are applied to the unit instances and can be amended there
Unit manager	Internal management mechanism which manages the unit allocation for all REE's in the Runtime.
Unit placeholder	Unit that is based on a **unit class**. Is used in the master recipe as a placeholder for actual parts of equipment. When testing and in the control recipe, the placeholder is then replaced with an actual item of equipment.
Action	Used in **Batch Control**: all commands which are used for editing a recipe e.g. insert phase, testing recipes etc.
Begin parallel branch	Element that ensures the separation of the recipe process in two or more branches.
Begin branch	Element that makes it possible to separate a recipe in two or more branches of which only one can be active at a time. Each following branch must start with a transition. The transition defines which branch is executed in the recipe process.
Start element	Element of module **Batch Control** with which every recipe begins.

Term	Definition
Active element	Position in a recipe in the batch control module where the processing is interrupted in a semi-automatic and manual mode and the active elements are put into a pause status. With the "next step" command, the process is resumed from this position.
Batch Control	Tool for creating master recipes and creating and executing control recipes in accordance with ISA-S88.
Batch operation	Automatic and sequential processing of a stack of single operations.
Command TAG	Umbrella term for initial and value parameters.
End parallel branch	Element that combines the separation of the recipe process into two or more branches back into one branch.
End branch	Element which brings together a branch started by a begin branch element.
End element	Element of module **Batch Control** with which every recipe ends.
Phase	Predefined process consisting of input interlocking, command and return/PLC sync parameters, a phase done condition, event reactions, etc... (*ISA 88: Phase*)
Host	Main recipe that contains operation instances.
Initial TAG	Parameter type for default values of the phase. Values are written before checking the input lock.
Instance	Depending on context, short form for **unit instance** or **operation instance**.
Command	Used in **Batch Control**: a command which intervenes in the recipe process e.g start, stop, mode change etc.
Matrix recipe	Recipe in the **Batch Control** module which was created with the Matrix editor.
Parallel branch	Area of module **Batch Control**. A parallel branch starts with a begin parallel branch element and is brought together with an end simultaneous sequence element to one execution branch. Between there are at least two branches which are executed at the same time.

Term	Definition
Parameter	Variable linking for a phase.
PFC recipe	Recipe in the **batch control** module, which was created with the PFC Editor
REE - Recipe Execution Engine	Part of module **Batch Control** for process control of recipes. The engine executes a control recipe and manages the entire process of the recipe.
Recipe	In recipes related data such as machine parameters or format data are summed up. This data can be transferred from the control system to the control and vice versa in one step. We differentiate between standard recipes and RGM recipes. The procedure is defined additionally to data in **Batch Control** Module in a recipe. It is distinguished between Matrix recipe and PFC Recipe.
Return/PLC sync parameter.	Parameter type for values that are read from the PLC.
Jump target	Element of the **Batch Control** module which allows a direct jump to a defined location of a branch.
Control recipe	Part of the **Batch Control** module. Contains the process of a production process on basis of the batch process according to standard ISA S88. A control recipe is always derived from a template recipe and can be implemented once only. (*ISA 88: Control recipe*)
Operation	Recipes can be divided into individual parts within the batch control module. Operation management takes place via a central library. Instances of operations can be added within the recipe. Tags of the applied phase can be edited, the structure can only be edited in the operation template.
Transition	Element of module **Batch Control** which contains a condition. The element is used after phases in order to ensure a defined transition from one phase to another.
Connection line	Part of the connector in the **Batch Control** module: Positions the connection point at the element.
Connection point	Part of the connection element in the **Batch Control** module. Connects two elements to each other (e.g. phase to phase or phase to line). It changes color when the mouse pointer is on it.

Term	Definition
Connection element	A possibility in **Batch Control** to connect elements to one another. It consists of a connection point and a connection line.
Branch	Area of module **Batch Control** which ensures a separation in two or more branches, of which only one can be active at the recipe process. It is an either/or branch. A branch always starts with the Begin branch and ends with the End branch element.
Master recipe	Part of the **Batch Control** module. Contains the process of a production process on basis of the batch process according to standard ISA S88. A recipe consists of the following components: Phases, transitions, parallel branch, etc... (*ISA 88: Master recipe*). Master recipes are used as templates for control recipes.
Value TAG	Parameter type for default values of the phase. Values are written after the input lock has been successfully checked.
Branch	Execution area for the **Batch Control** Module Basic functions, transitions and transfer targets can be placed on it.

5 Configure and apply batch recipes

The engineering and the use of module Batch Control takes place in three main steps:

1. Engineering in the zenon Editor
2. Recipe creation and approval in the Batch Editor (PFC or matrix) in the Runtime
3. Procedure in the Runtime

ENGINEERING

The engineer depicts the existing physical world in the Editor. They define and create units and unit classes (on page 16) and assign phases (on page 28) with tags (on page 32) to them.

RECIPE CREATION

In the Runtime, the recipe creator defines the master recipes on basis of the presets from the Editor. They define the process. In doing so, only the units, unit classes, phases and tags defined in the engineering environment can be used. Unit classes allow the use of unit placeholders. These make it possible to only assign actual units during recipe execution.
Note on parameter change: If the **Parameter value can be changed in master recipe** property has been activated in the **Write set value** properties group, the corresponding parameters can be modified. This is only possible for command tags. You can find this property group in the parameters in the Batch Control module.

RECIPE EXECUTION

The operator executes the recipe in the Runtime. For this he starts a control recipe which is based on the control recipe. Each control recipe can only be started and used once. With this it can be assigned to a lot unambiguously. Recipes run either automatically, semi-automatically or manually.

The operator can no longer change the recipes.
Exception: If the recipe creator sets the **Changeable in control recipe** option (on page 181) to active in the Runtime, the operator can still edit parameters of a recipe in the Runtime.

6 Engineering in the Editor

To use module Batch Control in the Runtime, you must do the following in the Editor:

- Create units/unit classes (on page 16), phases (on page 28) and reactions (on page 47)
- create a screen (on page 70) of type *Batch Control*
- create a screen switch function (on page 83) for the screen of type *Batch Control*

Note: In the editor, copying and inserting of elements throughout all levels is possible.

DETAIL VIEW

In module Batch Control the detail view divided in two:

- The left part comprises the unit tree with units, unit instances and unit classes. The **Units** entry is the starting point for the tree.
  The existing units are displayed at the next level. The phase belonging to each unit follow.
  Each phase has reactions as a sub-point.
  Unit classes and unit instances are structured identically.
- In the right part, a flat list of the units, unit instances and unit classes, phases, tags or reactions is displayed depending on what level of the tree has been selected.

DISTRIBUTED ENGINEERING

If, in distributed engineering, changes are made to the Batch Control module, the complete module is always checked out or checked in.

6.1 Unit placeholders: Concept and how it works

When configuring recipes in the Runtime, units that have not yet been assigned to any actual item of equipment can now also be used. Unit placeholders are used for this. These relate to unit classes (on page 17) and are only linked to actual items of equipment later using unit instances (on page 18).

Unit placeholders are available for recipes and partial recipes. Placeholders of the same unit class can be compiled into groups. This ensures that, in the recipe process, all placeholders that use a certain unit class are also assigned to the same item of equipment.

All placeholders are administered in a list (on page 239).

CREATING AND EDITING UNIT PLACEHOLDERS

The creation and editing of unit placeholders (on page 242):

- Is carried out in the Runtime
- Can be carried out for master recipes and partial recipes
- Is possible as long as the recipes are in edit mode

With partial recipes, the entries are displayed in the list of the host recipe.

APPLY UNIT PLACEHOLDER

Placeholders are linked to actual parts of equipment using unit instances:

- In the test mode of the master recipe
- in the control recipe

The assignment of equipment parts to a unit placeholder can also still be amended in control recipes during execution.

6.1.1 Units - unit classes - unit instances

The basic project configuration starts with the configuration of the units. They refer to the equipment parts that are actually available.
Unit classes and unit instances are now available in order to make configuration more efficient. It is thus possible to define similar units only during the recipe process using **unit placeholders** (on page 16).

- Unit (on page 17):
  Physically available machine or equipment part with which phases can be carried out.
- Unit class (on page 17):
  Template for units.
  Unit instances are created on the basis of this template. All properties of the unit class are applied to the unit instances and can be amended there.

  Unit classes are given a prefix to clearly distinguish them from units:

  - In the tree view in the detail view
  - In the history of changes
  - For linked properties
  - In the XML export (**Class**)
  - With access via the API (**Class**)
- Unit instance (on page 18):
  Unit that is based on a unit class.
  All properties of the unit class are applied to the unit instances and can be amended there.

Units, unit instances and unit classes are displayed in the unit tree.

> **Information**
>
> To create, copy insert or delete elements, you have the toolbar, the context menu and the keyboard available, with the usual Windows keyboard shortcuts.

6.1.1.1 Units

To create a new unit:

1. In the project manager, go to the **Batch Control** node
2. In the detail view, select **Units**.
3. In the context menu, select **New unit**

   A new unit is created.
   Default name: **Unit n**
   n is consecutive numbering.

6.1.1.2 Unit classes

Unit classes are templates for units Unit instances are created on the basis of them. All properties of the unit class are applied to the unit instances and can be amended there.

To create a unit class:

1. In the project manager, go to the **Batch Control** node
2. Select **Unit Classes** in the detail view.
3. Select the **New function** command in the context menu.

   A new unit class is created.
   Default name: **Unit class n**
   n is consecutive numbering.

Note: If a unit class is deleted, the unit instances linked to it are also deleted.

6.1.1.3 Unit instances

Unit instances are actual units that are based on a unit class. Phases of unit instances can be used like those of conventional units. In addition, unit instances can be used to localize recipes that contain phases of unit placeholders - i.e. to assign an actual item of equipment.

Unit instances take on all properties of their unit class. These can be amended here. When created, unit instances get a default name according to the model **Unit class n : Unit class**. The first part is created automatically. In doing so, **n** is a consecutive number. This part can be renamed. The second part always corresponds to the name of the unit class.

The properties of the unit instance can be derived from the properties of the derived (on page 26) unit class.

Note: If a unit class is deleted, the unit instances linked to it are also deleted.

CREATE UNIT INSTANCE

To create a new unit instance:

1. Select the desired unit class in the detail view.
2. Select **New instance** in the context menu.

   A new aggregate instance with a default name is created in the tree of the units. The name can be freely configured.
   Default name: **Unit class n : Unit class**
   n is consecutive numbering.

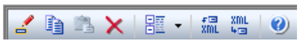
6.1.1.4 Toolbars and context menus

6.1.1.4.1 Batch Control context menu - project tree

Entry	Description
Export all as XML	Exports all units, unit instances and unit classes as an XML file.
Import XML...	Imports elements from an XML file.
Editor profile	Opens the drop-down list with predefined editor profiles.
Help	Opens online help.

6.1.1.4.2 Unit tree - tool bar

The tool bar refers to the complete unit tree with units and unit classes. Depending on the selected element, symbols are available or deactivated.



Symbol	Description
Rename	Opens the name filed of the selected element for renaming. Not available for the main nodes **Units** and **Unit classes**
Copy	Copies the selected unit to the clipboard. Not available for the main nodes **Units** and **Unit classes**.
Paste	Pastes an element script that was copied to the clipboard.
Delete	Deletes selected element after confirmation message. Not available for the main nodes **Units** and **Unit classes**
Expand all	Displays the entire tree structure. By clicking on the arrow you receive a drop-down list in which you can select one of the following commands: ▸ Expand all: expands all nodes ▸ Collapse all: collapses all nodes ▸ Expand selection: expands selected nodes ▸ Collapse selection: collapses selected nodes

Symbol	Description
	A click on the button always expands all elements. **Hint:** Via double click on the superordinate entry elements can also be expanded or collapsed. Default: Units and units classes are expanded with a sublevel
Export all units/unit classes to XML...	Exports all units or units classes as an XML file.
Import units/unit classes from XML...	Imports units or unit classes from an XML file.
Help	Opens online help.

6.1.1.4.3 Units - context menu

Right click the main entry **Unit** in the unit tree to open a context menu:

Entry	Action
New unit	Creates a new unit.
Replace linking in phases...	Opens the dialog to replace linking (on page 67) for linking in phases.
Replace linking in units...	Opens the dialog to replace linking (on page 67) for linking in units.
Paste	Pastes a unit that was copied to the clipboard.
Export all units to XML...	Exports all units as an XML file.
Import units from XML...	Imports units from an XML file.
Help	Opens online help.

6.1.1.4.4 Unit classes - context menu

Right-click the main entry **Unit classes** in the unit tree to open a context menu:

Entry	Action
New unit class	Creates a new unit class

Entry	Action
Replace linking in phases...	Opens the dialog to replace linking (on page 67) for linking in phases.
Replace linking in unit classes...	Opens the dialog to replace linking (on page 67) for linking in unit classes.
Paste	Pastes a unit class that has been saved to the clipboard.
Export all unit classes as XML...	Exports all unit classes as an XML file.
Import unit classes from XML...	Imports unit classes from an XML file.
Help	Opens online help.

6.1.1.4.5 Unit x and unit instance x - context menu

Right click on a created unit in order to open the context menu:

Entry	Action
New phase	Creates a new phase.
Replace linking in phases...	Opens the dialog to replace linking (on page 67) for linking in phases.
Replace linkings in unit/unit instance...	Opens the dialog to replace linking (on page 67) for linking in the selected unit or selected unit instance.
Rename	Allows you to rename the selected unit.
Delete	Deletes the selected unit.
Copy	Copies the selected unit to the clipboard.
Paste	Pastes a unit that was copied to the clipboard.
Export selected as XML...	Exports selected unit as an XML file.
Import phases from XML...	Imports phases from an XML file.
Help	Opens online help.

6.1.1.4.6 Unit class n - context menu

Right-click on a created unit class to open the context menu:

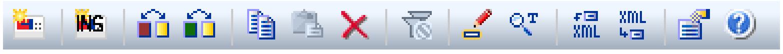
Entry	Action
New phase	Creates a new phase.
New instance	Creates a new unit instance. Default name: **Unit class n : Unit class** **n** is consecutive numbering.
Replace linking in phases...	Opens the dialog to replace linking (on page 67) for linking in phases.
Replace linking in unit classes...	Opens the dialog to replace linking (on page 67) for linking in the selected unit class.
Rename	Allows you to rename the selected unit class.
Delete	Deletes the selected unit class.
Copy	Copies the selected unit class to the clipboard.
Paste	Pastes a unit class that has been saved to the clipboard.
Export selected as XML...	Exports the selected unit class as an XML file.
Import phases from XML...	Imports phases from an XML file.
Help	Opens online help.

6.1.1.4.7 Units - detail view

Entry	Action
New unit	Creates a new unit or unit class in the detail view.
Replace linking in phase	Opens the dialog to replace linking (on page 67) for linking in phases.
Replace linking in units	Opens the dialog to replace linking (on page 67) for linking in units.
Copy	Copies the selected entries to the clipboard.

Entry	Action
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected.
Replace text in selected column...	Opens the dialog to search and replace texts for the selected column.
Export selected as XML...	Exports the selected units as an XML file.
Import units from XML...	Imports units from an XML file.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.1.1.4.8 Unit classes - detail view



Entry	Action
New unit class	Creates a new unit class in the detail view.
New instance	Creates a new unit instance. Default name: **Unit class n : Unit class** **n** is consecutive numbering.
Replace linking in phase	Opens the dialog to replace linking (on page 67) for linking in phases.
Replace linking in unit classes	Opens the dialog to replace linking (on page 67) for linking in unit classes.

Entry	Action
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected.
Replace text in selected column...	Opens the dialog to search and replace texts for the selected column.
Export selected as XML...	Exports the selected unit classes as an XML file.
Import unit classes from XML...	Imports unit classes from an XML file.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.1.1.4.9 Unit and unit class n - detail view

Toolbar and context menu for units and unit classes in the detail view.

Entry	Description
New phase	Creates a new phase in the detail view.
Replace links	Opens the dialog to replace linking (on page 67) for linking in phases.
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".

Entry	Description
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected.
Replace text in selected cell....	Opens the dialog to search and replace texts for the selected column.
Export selected as XML...	Exports the selected phases as an XML file.
Import phases from XML....	Imports phases from an XML file.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.1.1.5 Information in the Runtime

Information on individual units can be called up and displayed in the Runtime using variables. Each element to be allocated initializes the values of the linked variables with *0* or empty text. The information comes from the recipe that allocates the unit when the query takes place. Variables are always only filled in the event of a change.

The variables for the information required in the Runtime is configured in the unit properties in the **Runtime information** group for:

- **Master recipe**: Information on ID, name and description of the master recipe, as well as version (on page 217) and initial version.
- **Control recipe**: Information on ID, name, description and job ID of the control recipe.
- **Execution**: Information on the number of active recipes, as well as execution status and mode, each numerically and as a text.

  Note: The value for **Number of active recipes** is generally *0* or *1*. If a higher number is displayed, then the start of other recipes was forced manually.
- **Phases**: Information on active phases. If the phase is in an operation, the name of the operation is displayed in brackets.
- **Error**: Errors are shown visually in the Runtime and saved in logs. A process error in the unit is shown visually this way and the absolute number of historical procedure errors is also

displayed. The counter is increased by *1* as soon as a procedure error occurs. If the phase is restarted, the historic error goes from the display; it is no longer signalized visually. The logged information is retained however. The same applies to communication failures: Symbols only display active errors; counters also inform you of historic errors.

- **Matrix information**: Display of the active steps in a matrix recipe. It is always the information from the main recipe that is used, even if the trigger object is in an operation. All numerical variables whose data type **>=** *2 byte* can be selected.
  PFC recipes always receive empty values or *0*.

Recommendation: Use the **UDINT** data type for numerical variables. Avoid the data types **INT**, **SINT**, **USINT** and **UINT**.

VALUES FOR EXECUTION STATUS AND EXECUTION MODE

Values for variables of the **Execution status (numeric)** and **Execution mode (numeric)** are in the **Variable allocation** (on page 91)/**General** (on page 98) chapter**.**

INFORMATION IN RECIPES AND UNIT

When restarting after Runtime has been restarted, the respective status is stored with the information in the recipes.
For example: **Paused after normal shutdown** or **Held after incorrect shutdown**.

The execution status is also displayed in the unit information.
The **execution status (numerical and text)** in the unit information contains a number and a text. They correspond to those of the variables in the screen.
In addition, information is contained:

- Whether triggered by a restart
- About objects with a different status
- About objects that delay a status change

Attention: The content of these variables is not compatible between zenon 7.10 and 7.11!

6.1.2 Differences from unit classes in unit instances

Properties of the unit instances and their subelements are derived, i.e. taken from the corresponding properties of the respective unit class. This derivation is displayed by an arrow next to the property. These properties can also be configured locally. The derivation is separated in the process.

SEPARATE DERIVATION

Derivations of properties are separated by:

- Local configuration:
  If a property of a unit instance or an element is amended, the derivation is separated from the unit class.
- Separation by means of a corresponding command in the context menu:
  The context menu of a property can be used to connect this property to the unit class again.

DERIVE PROPERTIES

Separated properties can be derived from their unit class again. To do this, select the **Inherit [property] from unit class**. To derive all separated properties again, select **Inherit all properties from unit class**.

CONTEXT MENU

The properties of the unit instance and its subelements can be separated from its unit class or derived again by means of a context menu.

Entry	Description
Unlink [property] from unit class	Separates the selected property from the unit class from which it is derived. Only available if this property is derived.
Inherit [property] from unit class	Derives the selected property from its original unit class again. Only available if the derivation of this property is separated and the property has been configured locally.
Inherit all properties from unit class	Derives all separated properties from from their original unit class again.
Unlink all properties from unit class	Separates all properties from the unit class from which they are derived.

Information

Substitutions are also carried out for properties if variables are still inherited. If they are modified by the substitution, the derivation is separated.

6.2 Phases

The phase is the execution object of a recipe - and therefore its main component. Each phase in module Batch Control must stand facing a **Technological function** in the control.

Example: You want to heat up a tank.

For this you need:

1. The corresponding equipment: a heating system in the tank.
2. A temperature sensor which measures the actual temperature in the tank. Connect this sensor with a control.
3. A program in the controller that controls the heating until the set temperature is reached. This control program is the **process action** in the PLC. It:
   - has an input tag: a set temperature - which is implemented via a command tag (on page 35) in Batch Control
   - Needs an output tag - also a set temperature - that must be reached and that is implemented via a return/PLC sync tag (on page 36) in Batch Control

To inform the control about the progress of the recipe, you need corresponding status information which is transferred to the control. For this you use reactions (on page 47) and conditions (on page 40) for the response.

A phase therefore consists of:

Type	Description
Command TAG (on page 35)	The set values which should be transferred to the control
Return/PLC sync parameter (on page 36).	The response values which inform the REE about the status of the **process action** in the control. They can be evaluated in conditions and transitions.
Reactions (on page 47)	REE events can be used on the one hand to inform the **process action** in the control about the state of the REE and on the other hand to inform the user about errors (e.g. time outs, invalid tag values). time outs, invalid tag values).
Conditions (on page 40)	Serve to evaluate the return/PLC sync tag: The state of the **process action** in the control is evaluated.
Times (on page 45)	Time critical processes can be monitored with this. If the engineered time is exceeded, an event is triggered on which you can react with a reaction.

ENGINEERING

To create a new phase:

1. Select the unit, the unit class or unit instance for the phase.
2. Select the **New Phase** command in the context menu

   A new phase is shown in the detail view.

   The **Reactions** sub-item is automatically added to the phase.

   > **Information**
   >
   > If several phases are selected in the Editor at once whose formulas are identical but whose tags are different, this is not displayed by color coding the different values.

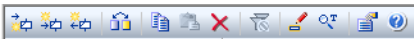
6.2.1 Context menu phase

CONTEXT MENU PHASE X

Right click on a created phase in order to open a context menu for creating the parameters:

Parameter	Description
New initial TAG	Creates a new initial tag (on page 35).
New value TAG	Creates a new value tag (on page 35).
New return/PLC sync TAG	Creates a new return/PLC sync tag.
Replace linking in phase	Opens the dialog to replace linking (on page 67) for linking in phases.
Rename	Makes it possible to change the name of the currently selected phase.
Delete	Deletes the currently selected phase after a confirmation message.
Copy	Copies the selected element to the clipboard.
Paste	Pastes an element script that was copied to the clipboard.
Export selected as XML...	Exports the selected phases as an XML file.
Help	Opens online help.

6.2.2 Detail view phase



Parameter	Description
New initial TAG	Creates a new initial tag (on page 35).
New value TAG	Creates a new value tag (on page 35).
New return/PLC sync TAG	Creates a new return / PLC sync tag in the detail view.
Replace links	Opens the dialog to replace links (on page 67).
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected.
Replace text in selected column...	Opens the dialog to search and replace texts for the selected column.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.2.3 Phases in unit classes and unit instances

If phases are edited, created or deleted in an phase unit class, these changes are also applied automatically in their instances.

AMEND PHASE

Phases can also be edited in instances. If a property is amended, the derivation is removed for the property. Changes to the phases from the class are then no longer applied. Local amendments to the instance can be replaced with derivations from the class again.

<u>**Overview of amendability of properties of a phase in an instance:**</u>

Property	Editing possible
Name in the **General** group	--
All other properties of the **General** group	*X*
All properties of the **Condition transient status** group	*X*
All properties of the **Loss of communication** group	*X*
Active control strategies	--
control strategy tag Control strategy parameters are displayed as a reference to the parameter in the instance.	--

<u>**Key:**</u>

- *X*: possible
- --: not possible

LOCAL PHASES

If a phase is created at the unit instance directly, it is then only present locally. This is displayed via the active checkbox of the **Local** property in the **General** properties group, a separate symbol and a separate background color in the list.

If a phase of a unit class is is renamed and this leads to a naming conflict with a local phase, a prefix **local n** is automatically placed before the local phase. In doing so, **n** is a consecutive number, starting with *0*.

Copied phases are always inserted as a local phase, even if the original function is derived.

Local phases can be renamed and deleted. Derived phases can only be deleted or renamed in the the unit class.

COPY AND PASTE

Phases, reactions, parameters and control strategies can be copied from unit classes and unit instances and pasted to other unit instances.

The following is applicable for pasting:

- The inserted objects are always pasted as a local copy. Linkings to unit classes are separated.
- All properties have the value that they had at the time of copying.

6.2.4 Duration of execution

The execution duration is controlled via two independent properties. Their values must not complement one another.

MAXIMUM EXECUTION DURATION

The **Maximum execution duration** refers to *Phase deactivated* and therefore to the process. It is not connected to the **Minimum execution duration**.

MINIMUM EXECUTION DURATION

Property **Minimum execution duration** defines how long zenon waits after writing the command tag independent of the check of the phase done condition. During the execution the maximum execution duration is checked. An event is triggered if this is exceeded. This can be linked to a reaction. This happens regardless of whether the phase still checks its **Phase done condition** or only waits for the **Minimum execution duration**.

The length of the **minimum execution** duration can exceed the **maximum execution** duration.

EXAMPLE

- There is a phase: **Start mixing**. The confirmation that the mixer runs must not take longer than 5 seconds before a warning of an error is displayed.
  Engineering: Property **Maximum execution duration** gets value *5 seconds* with corresponding reaction.
- The mixer however should run 15 minutes before the next phase is executed.
  Engineering: Property **Minimum execution duration** gets value *15 minutes*.

With this the minimum execution duration is 15 minutes and the maximum execution duration 5 seconds.

6.2.5 Parameter

TAGs are the communication interface to the control. With them all values are transferred to the control and also read back. To not have to work with complex and for the user incomprehensible variable names in module Batch Control, the abstract level is used. Each tag consists - for each phase

- of a unique name and a description. In this, the engineer can give the recipe creator or user a description of what the tag is used for or which effects it has.

You can add any number of tags to a phase. A distinction is made between command tags (on page 35) and return/PLC sync tags (on page 36). Command parameters are further subdivided into initial parameters and value parameters. Each tag can be switched between command tag and return/PLC sync tag at any time.

For the behavior of parameters in control strategies of a unit instance, see the **Parameters in unit classes and unit instances** (on page 39) chapter.

CHANGING VALUES IN THE RUNTIME

Only command tags have a configurable parameter value. This set value can be configured as fixed or as changeable in the recipe.

To configure values as changeable in the Runtime:

1. Activate the checkbox for the **Parameter value can be changed in master recipe** property.
2. Decide whether the values for minimum and maximum will also be able to be changed.
   To do this, activate or deactivate the **Parameter limit value can be changed in the master recipe** property.
3. Decide whether parameters can be changed if a phase is already active.
   To do this, activate or deactivate the **Tag values changeable during phase execution** checkbox.

You can read information on changing the values in the **editing parameters** (on page 181) chapter.

CHANGING PARAMETERS WITH A PHASE ACTIVE

To edit parameter values with a phase active, activate the **Tag values changeable during phase execution** property in the **Edit tag** general properties group.

Parameter values can thus also be edited (on page 45) in the Runtime with a phase active.

Note: If this property is active, the following properties are blocked for editing in the Editor and ignored in the Runtime:

- **Immediately write value changes of parameters**
- **Timeout for value change acknowledgement**

LIMIT VALUES FOR COMMAND TAGS

Values for command tags can be limited with a minimum value and a maximum value. Only values that are within this range can be used in the Runtime.

The minimum and maximum values for command tags can be defined using:

- Properties in the editor:
  The minimum value and maximum value are stipulated with the **Min. value** and **Max. value** properties.
- Linked variables:
  The **Variable for min. value** and **Variable for max. value** properties can be used to define variables that dynamically stipulate the limit values for the minimum value and maximum value in the Runtime.
  The variable values are read and applied when:
  - Loading a master recipe
  - Reloading a master recipe
  - XML import of a master recipe

CONFIGURE VALUES VIA PLACEHOLDER FOR VARIABLES

Parameter values can also be read in using variables. The variables that are available in the Runtime are configured in the Editor using the **Placeholder for tag value variable** property.
In doing so, the following applies:

- Not available for return / PLC sync tag.
- Cannot be edited for tags of phases.
- Cannot be edited in unit instances.

To configure parameter values using placeholders:

1. In the parameter properties, go to the **Placeholder for tag value variable** property.
2. Enter a placeholder in the field.
   In doing so, you can use the die wildcards ***** and **?**.
3. Assign (on page 191) a variable to the parameter when configuring the master recipe. This determines the value of the parameter.
   Only variables that correspond to the model defined in the **Placeholder for tag value variable** property are offered.
   A valid variable must be linked at the latest when approving the master recipe.

Note: If the **Placeholder for tag value variable** property contains a value, the **Tag value** property is deactivated.

6.2.5.1 Detail view tag

Toolbar and context menu provide commands to create and administer variables of parameters.

Parameter	Description
New initial TAG	Creates a new initial parameter in the detail view.
New value TAG	Creates a value parameter in the detail view.
New return / PLC sync TAG	Creates a new return / PLC sync tag in the detail view.
Replace links	Opens the dialog to replace links (on page 67).
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected.
Replace text in selected column	Opens the dialog to search and replace texts for the selected column.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.2.5.2 Command TAGs

Command parameters transfer information and values to the controller. They can be subdivided into:

- *Initial parameters*: Command parameters that are set before the start event. They transfer information that must be stored before setting the input lock in the controller, for example, which control strategy (on page 62) is executed.
- *Value parameter*: Command parameters that are sent after input locking when the phase is executed.

Command tags contain the set values which should be transferred to the control. Initial parameters and value parameter are backed up (on page 276) and written to the controller. You can find the exact description in chapter: Process of a phase in detail (on page 268).

Command tags can also be used in transitions (on page 40), conditions (on page 40) and reactions (on page 47). Initial parameters and value parameters can have the same variable linked. This is taken into account when validating for multiple use of a variable.

Command parameters have a number of properties which can be defined via the property window. In doing so, the following applies:

- Each tag must be linked with a variable.
- The data type of the variable must correspond to the data type of the parameter.
- The set value limits of the parameter must be within the set value limits of the variable.

If this is not the case, error messages are created during the validation.

Hint to property **Parameter value can be changed in master recipe**: With this you define whether the value of the command TAG may be modified by the creator of the master recipe. If e.g. machine tags should not be changeable in the recipe but defined fixedly, you must deactivate this property.

ENGINEERING

To create a new command tag:

1. Select the desired phase
2. Select, in the context menu, the command **New initial parameter** or **New value parameter**
3. a new command tag is created in the detail view

NOTE ON COMPATIBILITY

If Runtime files are created for zenon version 7.10 or older, the initial parameters and value parameters are treated in the same way as command tags again. Command parameters from zenon 7.10 or earlier are all converted to value parameters.

6.2.5.3 Return / PLC sync tag

The return / PLC sync tags contain the return values with which the technological function of the REE communicates its status. Normally the value is set by the control and evaluated by the REE. Return / PLC sync tags can be evaluated in transitions (on page 40) and conditions (on page 40).

They can also be used like this in reactions (on page 47). In doing so, the values configured in the return / PLC sync tag are applied. To do this, they are linked in the reaction properties groups **Allocate tag** and for **Tag write set value** as a target parameter.

ENGINEERING

To create a new return / PLC sync tag:

1. Select the desired phase for which you want to create a new return / PLC sync tag.
2. Select the **New return / PLC sync tag command** in the context menu.
3. A new command tag is created in the detail view.

6.2.5.4 Example for status tag

To be able to communicate with the control, you normally need two status tags:

- one in write direction and
- one in read direction

The variable behind these parameters should have a numeric data type such as USINT or UINT. It is recommended that both tags are executed as a return / PLC sync tag.

> **Information**
>
> This may seem illogic for commands in write direction at first glance but has the following background: When executing the phase, all command tags are always set. With this they are visible in the list of command tags and can therefore be edited accidentally. This makes no sense for a command for the control. The goal is not just to communicate a single value to the control as command but to transmit the state of the phase in the recipe.

Precisely when writing a command tag is when it makes sense to not simply inform the PLC about the writing but the status when all values have been written and the PLC can therefore start to process the technological function.

To do this, it is best to use the reaction to the *finished writing command* result. With reactions to an event, it is also possible to write values to a return / PLC sync tag. It is therefore recommended in this case that return / PLC sync parameter is used for both status parameters.

Here is an example about which values the tags can take on:

STATUS TAG IN WRITE DIRECTION (TO THE PLC): COMMANDS

Value	Name of the event
0	not defined
1	Phase started
2	Writing value tags finished
3	Phase finished: Phase done condition fulfilled and **Minimum execution duration** reached (if engineered)

Value	Name of the event
4	Phase deactivated
5-9	Reserve
10	Status change: Pausing
11	Status change: Resuming
12	Status change: Holding
13	Status change: Restarting
14	Status change: Stopping
15	Status change: Aborting
16-19	Reserve
20	Mode change: Automatic
21	Mode change: Semi-automatic
22	Mode change: Manual
23-29	Reserve
30	Exit Runtime initiated
31	Runtime restart
32	Unit allocation not possible
33	Unit allocation timeout exceeded
34	Input interlocking blocked
35	Input lock timeout exceeded
36	Maximum execution duration exceeded
37	Waiting period following condition exceeded
38	Phase started multiple times

You can find the exact meaning of the events in the Event type (on page 50) chapter.

For each entry in the table you define a corresponding reaction for writing the status value at the phase.
Hint: Use the same tag label for all phases; e.g. *StatusPhase*. Then you only have to engineer the

reaction at one phase and can then transfer it to all phases via copy & paste. You can of course also copy the tags. Do not forget to correct the variable. They must match the respective phase.

TAG IN READ DIRECTION (FROM THE PLC): RETURN VALUES

Value	Description	Linked in property
0 - 1	Not defined	
2	Process action finished	**Phase done condition**
3 - 9	Reserve	
10	Process action paused	**Paused**
11	Reserve	
12	Process action held	**Held**
13	Process action restarted	**Restarted**
14	Process action stopped	**Stopped**
15	Process action aborted	**Aborted**

Link the values with a formula in the respective property.

Hint: You can copy the formula and just change the respective value. If you set this setting at the start of setting the parameters, you can copy the entire phase and with that have these settings for all phases.

6.2.5.5 Parameters in unit classes and unit instances

The following is applicable for parameters in phases of a unit class:

- If a tag is created, deleted or renamed in a phase of the unit class, this change is applied to the linked unit instances.
- If a parameter is newly created in a phase of a unit instance, this is a local tag. It is highlighted in the detail view and in selection lists with its own background color. It has no linking to a higher-level class and is configured locally. It is then shown in the **Local** property with an activated checkbox.
- Many properties of linked parameters can be overwritten locally.
The following cannot be overwritten:
  - **Name**
  - **Type**
  - **Tag data type**

  - **Parameter value can be changed in master recipe**
  - **Parameter limit value can be changed in the master recipe**
  - **Placeholder for tag value variable**
- If a parameter in a unit class is renamed and this name does not correspond to a local parameter, the local parameter is automatically renamed. The uniqueness of the parameter names is thus retained.
  Schematic diagram for renaming: **local <n> <OriginalName>**
  Example: **Parameter** becomes **local 0 Parameter**.
- If a parameter is copied and pasted from an instance, it loses any linkings to a class it may have. It is always local.
- Only the locally-created parameters can be renamed or deleted in a unit instance.
- When compiling for Runtime, each derived parameter takes on the values configured for the unit class for all properties that are linked to a class. In the Runtime, the parameter has precisely the values that were displayed in the Editor.

6.2.6 Conditions

The conditions are used to inform the REE of the status of the **technological function** in the controller. For evaluating the conditions formulas are used which were created with the formula editor (on page 303).

Hint: Use a single status tag (return / PLC sync tag) that takes on different values in order to transmit the status of the **technological function** in the controller to the phase. You can find an example in chapter Example for status parameter (on page 37).

ENGINEERING

To configure a formula for a condition:

1. Click on the corresponding phase.
2. In the properties, select the desired conditions from the **General**, **Condition transient status** or **Conditions for commands** nodes.
3. Click in the field for the value or on button **...** .

   The formula editor is started.
4. Define the formula (on page 303) for the condition.

Note: The counterpart to the conditions are the reactions (on page 47). With them, the execution status is transferred to the technological functions of the controller.

6.2.6.1 General

The following are available as general conditions:

Condition	Description
Input interlocking	Due to an input lock, the phase is only executed in the Runtime when the condition for the input lock is met. The input interlocking is configured with the **Input interlocking** property. The condition that must be met is defined using the formula editor (on page 303). The following is applicable for the formula: ▶ It can consist of one or more command tags and/or return / PLC sync tags of the phase. ▶ Both the value and status of the variables can be used. ▶ It returns *TRUE* or *FALSE* as a result. The condition can be displayed in the Runtime but cannot be changed. The waiting period for the input locking is configured with the help of the **Input interlocking timeout** property. A phase is executed as soon as the following conditions are met: ▶ The phase is active ▶ The unit can be assigned ▶ Another instance of the phase is not already being executed ▶ All necessary variables have received a value ▶ The input lock condition is met, if one has been defined **Note:** You can find more information about input interlockings in chapter Processing a phase in detail (on page 268).
Phase done condition	Defines phase done condition. Once all value tags have written successfully, a phase checks the phase done condition. The phase can only be concluded and the recipe continued if the phase done condition has been met. This is configured using the **Phase done condition** property. The condition to be met is defined using the formula editor.

Condition	Description
	The following is applicable for the formula: - It can consist of one or more command tags and/or return / PLC sync tags of the phase. - Both the value and status of the variables can be used. - It returns *TRUE* or *FALSE* as a result. The formula: The condition can be displayed in the Runtime but cannot be changed. The waiting period for the phase done condition is configured with the **Maximum execution duration** property.

6.2.6.2 Conditions for transient status

If a command, such as pause for instance, is sent to a phase during execution, it does not immediately switch to the corresponding status (*paused*) but to a transient status (*pausing*). In this status, the phase remains until the phase has been confirmed - until the condition for the transient status has been met.

Transfer conditions are defined with parameters that are linked to variables. They determine when the phase switches the status.

The status changes is delayed until:

- Each variable has a value
- The formula is *TRUE*
- There is no variable with an invalid status

If no condition is defined, the status change is always allowed.

All variables for the status change are requested at the advising of the variables in order to receive a value as soon as possible. The value for a variable which define a status change is read on the change to the transient status. A possible pulse must have value *TRUE* within the waiting period in order to be recognized.

Note: When closing the Runtime, it is not waited for the variable for the status change from *stopping* to *stopped* as at this time all variables are already signed off.

CONDITIONS

A formula can be defined for the condition.

The following is applicable for the formula:

- It can consist of one or more command tags and/or return / PLC sync tags of the phase.
- Value and state of the variables can be used.

The following conditions are available for the following status:

Status	Description
Paused	Condition for the change from *pausing* to *paused*.
Held	Condition for the change from *holding* to *held*.
Restarted	Condition for the change from *restarting* to *running*.
Stopped	Condition for the change from *stopping* to *stopped*.
Aborted	Condition for the change from *aborting* to *aborted*.
Escape condition	Condition for leaving a phase If this condition is met, the current execution step is stopped and the phase is exited. You can find details in the Exiting a phase (on page 274) chapter.

6.2.6.3 Conditions for commands

You can use phase commands and recipe commands (on page 251) to trigger commands not just from the user interface, but also through a change to a variable value. Very flexible configuration is thus possible.

The phase commands always relate to the phase in which they are configured, the recipe commands relate to the recipe in which the phase is active.

The formulas are checked in active phases. If the first check for a formula results in *TRUE* or a subsequent check results in a switch from *FALSE* to *TRUE*, the commands are triggered. The triggering happens regardless of whether the command can be executed. If, for example, a phase is sent whilst the recipe is paused, it cannot be executed and is rejected.

If several commands of a condition are triggered, they can mutually overwrite each other.

If a command is triggered due to a formula and executed successfully, the entry is written to the CEL.

Test behavior:

- Restart:
  If a restart is carried out before a start event has been triggered, the formula checking is also not active after the restart until the start event is triggered. If the phase is restarted after the

start event, the checking of the formula is active from the start. It is restarted from the beginning in this case: Initial values can thus trigger commands.

- Communication error:
  The formulas are not checked whilst a communication error is pending. Checking is only started again once the communication problem has been acknowledged. It is restarted from the beginning in this case: Initial values can thus trigger commands.
- Checking sequence and cancel conditions:
  All formulas are always checked.
  That means: If a value change to several formulas results in *TRUE*, commands are triggered fro all these formulas. If this leads to the execution status of the phase changing, checking is stopped at this point.
- Sorting order:
  Formulas for commands are checked after the error checks but before the other formulas.
  That means: If a communication error occurs, the same value change cannot trigger a command because the checking has been canceled beforehand.

The combination of the above-defined behavior pattern means: If a value change leads to a command being triggered and the command is accepted and executed, and the condition to be checked (input lock, phase done condition or a condition for a status change for example) would also be set to *TRUE*, the following condition is no longer checked. The checking is then continued in the new status.

CONDITIONS

A respective formula (on page 303) can be defined for this condition for phase commands and recipe commands:

Command	Description
Pause phase	Condition for the pausing of the phase.
Resume phase	Condition for the continuing of the phase.
Hold phase	Condition for the holding of the phase.
Restart phase	Condition for the restarting of the phase.
Pause recipe	Condition for the pausing of the recipe.
Resume recipe	Condition for the continuation of the recipe.
Hold recipe	Condition for the holding of the recipe.
Restart recipe	Condition for the restarting of the recipe.
Stop recipe	Condition for the stopping of the recipe.

Command	Description
Abort recipe	Condition for the canceling of the recipe.

6.2.6.4 Waiting periods

The recipe creator can define waiting periods. The configuration of waiting periods (time outs) prevents that time-critical processes take too long because of unforeseen events. If the condition is not fulfilled within the defined waiting period, a corresponding event (on page 50) is triggered. With the reactions (on page 58) you can react on the event and influence the recipe process.

For all waiting periods the following is true:

- If *0d 00:00:00* is defined as waiting period, the event is not triggered.
- The waiting periods are independent of the recipe status (e.g. *Recipe paused*) and continue to run even when the Runtime is closed and restarted.
- If a phase is held and restarted, the waiting periods are also restarted.
- If a phase is passed through several times, the waiting periods are started again for every pass.
- Waiting periods themselves do not influence the process. They are simply used to generate an event. The reaction must be defined in the event. After the event is triggered, it is still waited for the fulfillment of the condition.

6.2.7 Value change during execution

Values for initial parameters and value parameters can each be processed between the start of writing and a successful check of the input locking or phase done condition and written to the PLC. You can write amended values in the Runtime (on page 276) immediately or in bundled form.

CONFIGURATION

To change values during the execution of a phase:

1. Activate the **Tag values changeable during phase execution** property in the Editor.
2. Decide whether values are written immediately or in bundled form after confirmation.

WRITE VALUES IMMEDIATELY

To write values immediately:

1. Go to the **Value change during execution** properties group for the phase.

2. Activate the checkbox for the **Immediately write value changes of parameters** property. The **Timeout for value change acknowledgement** property is thus automatically deactivated.
3. Configure the required events
   - *Initial TAG modified during process*
   - *Value TAG modified during process*
   - *Amended initial TAG fully written*
   - *Amended value TAG fully written*

Value changes do not need to be confirmed in the Runtime. If a phase is in the corresponding stage (check of the input lock and/or check of the phase done condition), amended values of initial parameters or value parameters are written immediately after the change. A corresponding symbol and information are displayed in the tooltip.

WRITE VALUES IN BUNDLED FORM

Amended values can also be collated and written to the PLC manually by clicking on a button.
To write values in bundled form:

1. Go to the **Value change during execution** properties group for the phase.
2. Deactivate the **Immediately write value changes of parameters** property.
3. Configure the **Timeout for value change acknowledgement** property.
   You thus stipulate how much time can pass between the first change of a parameter and the writing of the values by clicking on a button in the Runtime. The corresponding result is triggered after expiry of the waiting time. This informs the operator that amended values have not yet been sent.
   The value *0* means: no waiting time.
4. Configure the required events
   - *Initial TAG modified during process*
   - *Value TAG modified during process*
   - *Amended initial TAG fully written*
   - *Amended value TAG fully written*
   - *Waiting time for confirmation of the value change of the initial TAG exceeded*
   - *Waiting time for confirmation of the value change of the value TAG exceeded*
5. Configure the **Confirm Changes** button in the screen.

Note: If parameter values are amended during recipe execution, this is indicated by a symbol to the phase. As long as the amended values have not been confirmed with the **Confirm Changes** button, they are not written to the variables. This can led to execution of the function being blocked.

BEHAVIOR IN THE RUNTIME

The setting for the **Tag values changeable during phase execution** property is noted when writing the parameter.
If a phase is still in the validation phase during execution, parameter values can be amended regardless of the property's setting.

Tag values changeable during phase execution Checkbox setting:

- *Active*:
  - Parameter list in the screen: Keyboard is called up. Changes are possible in the dialog to edit the parameters.
  - Edit phase dialog: Changes are possible in the dialog to edit the parameters.
- *Inactive*:
  - Parameter list in the screen: Keyboard is not called up. The dialog to edit the parameters is write protected.
  - Edit phase dialog: The dialog to edit the parameters is write protected.

6.2.8 Reactions

Reactions are important objects to influence the recipe process and to communicate with the controller. Reactions are always based on events. These can be REE events (e.g.: *Phase started*), but also general events (e.g. *Exit Runtime initiated*).

With the help of reactions:

- You can, for example, tell the controller when a phase has been started or ended in the REE and when all command tags have been written.
- Send the status of the phase to the controller. Otherwise the control has no information about the process of the recipe.

Example: If you stop the REE or the phase, the event *Status change: Stop* will be triggered. With a reaction, you can transfer this status change to the control as set value input. Only then can the control react and stop the **process action**. You can find an example in chapter Example for status parameter (on page 37).

Note: The counterpart to the reactions are the conditions (on page 40). With a reaction, you can transfer this status change to the control as set value input.

6.2.8.1 Context menu reactions unit tree

Command	Description
New reaction	Creates a new reaction in the detail view.
Replace links	Opens the dialog to replace links (on page 67).
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Help	Opens online help.

6.2.8.2 Toolbar and context menu detail view reactions

Symbol/Command	Description
New reaction	Creates a new reaction in the detail view.
Execution order: Earlier	For reactions of the same type: Moves the reaction forward in the execution order.
Execution order: Later	For reactions of the same type: Moves the reaction backward in the execution order.
Execution order: Change places	Only active if exactly two reactions are chosen. The two selected reactions change their places in the execution order.
Replace links	Opens the dialog to replace links (on page 67).
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be

Symbol/Command	Description
	selected.
Replace text in selected column	Opens the dialog to search and replace texts for the selected column.
Properties	Opens the **Properties** window.
Help	Opens online help.

6.2.8.3 Engineering

With each phase the node **Reactions** is created automatically. In this node you can created any reactions. To create a new reaction:

1. Click on **Reactions**.
2. Select the **New Reaction** command in the context menu.

   A new reaction is shown in the detail view.

   A **Reaction name** is created automatically. Amend this if required. The name must be unique for each phase.
3. In the detail view, click on the entry in the **Event** column.
4. Select the desired event from the drop-down list and define the desired reactions (on page 58) in the property window.

Several reactions can be configured for each event. They are executed according to the configured sequence. Reactions of the same type can only be sorted using the toolbar or context menu (on page 48).

PROJECT CONFIGURATION RULES

- Several reactions are possible for each event. They are executed according to the configured sequence.
- The variables of all parameters are signed in to the driver for reading. If a value is needed for a reaction but is not yet available or invalid, the alternate value is written. The writing of the value is done without write confirmation.
- Some of the reactions are triggered only once in the process - e.g. time outs. If the phase is restarted, these reactions are also retriggered if necessary.

BEHAVIOR IN THE RUNTIME

For each reaction, a range of actions can be configured in the **Reactions** properties group, such as **Tag write set value**, **Function**, **Allocate tag** or **Recipe influence**. For example, in order to change the REE mode after restarting the Runtime, or to execute an REE command when writing command parameters.

In doing so, please note:

- The execution of these actions in the Runtime can be prevented by other events.
  For example, the status of the phase can prevent the execution of an action.
- Several different reactions are processed in the same execution cycle. However, only one REE command per cycle is permitted. In general, the last REE command is executed.

Note: Actions of reactions are logged in the CEL if the **Create CEL entry** property has been activated for the reaction. *System* is entered as a user.

6.2.8.4Events

Each **reaction** is based on an event. This is stipulated for the reaction using the **Event** property. As many **Reactions** as desired can be assigned the same event. The sequence of the execution of the reactions is stipulated in the detail view.

In the event of an incorrect configuration of a reaction, the name of the event is displayed when validating recipes.
Syntax: **(EventName.x)** whereby **event name** corresponds to the **Event**. **x** is a figure that displays the position in the execution sequence.

EVENTS ON RESTART

The reactions *phase activated* and *phase started* are always only executed once. These reactions are not triggered again after the phase has been restarted (after being held). The phase starts to run again, however it was not executed in full beforehand.

Along the same lines, the reaction *phase deactivated* is only triggered once the phase has been ended and not during a restart.

The *phase started* reaction is triggered if the unit allocation and the parallel execution detection has been executed. If the procedure has not exceeded this detection on restart, the reaction is triggered for the *Restart* command. If the process is already in an advanced state, the reaction is not carried out again.

ALLOW EXECUTION BEFORE "PHASE STARTED"

For reactions, it is possible to configure whether these are also to be executed if the *phase started* event has not yet been triggered. This can be set with the **Allow execution before start event** property. The property is only available for reactions whose result can occur both before and after the start event. The property is deactivated for events that can only occur before or after the start event. In this case, the value is automatically set according to the event type. For details, see the tables with the events.

Note that no commands can be sent to the PLC before the *phase started* event. Otherwise it may be the case that the execution of the PLC program is influenced unintentionally, such as if an instance of the phase is started whilst another instance of the same phase is currently being executed.

EVENTS FOR REACTIONS

From the drop-down list you can select the following events:

UNIT CLASSES

These events are only available in phases of unit classes.

Event	Description	Allow execution before start event
Placeholder not linked to instance	Is triggered if, at the time of execution, a placeholder is not linked to a unit instance.	*always*
Maximum waiting time for instance assignment exceeded	Is triggered if the waiting time for the linking of a unit instance has been exceeded.	*always*

PROCEDURE

Events in the procedure.

Event	Description	Allow execution before start event
Phase activated	Is the first event which is triggered.	*always*
Unit allocation not possible	Is triggered if the unit was not allocated successfully at first try.	*always*

Event	Description	Allow execution before start event
Finished writing initial TAG	Is triggered if all initial parameters have been successfully written to the PLC.	*always*
Initial TAG modified during process	Is triggered if the value of an initial parameter is amended whilst the input locking is checked.	*adjustable*
Amended initial TAG fully written	Is triggered if amended initial parameters have been successfully written to the PLC.	*adjustable*
Initial TAG modified during process	Is triggered if the value of an initial parameter is amended whilst the input locking is checked.	*adjustable*
Phase started	Is triggered after writing the initial parameters. With this event you inform the PLC that the phase has been started in the REE and that it is expected that the command tags will be written soon. Other events can be executed before the event if the **Allow execution before start event** has been activated for the corresponding event. For details, see the **Allow execution started before phase started** section.	-
Input lock blocked	Is triggered if the input lock is blocked (FALSE) after the initial checking (after writing the initial parameters). Makes only sense if property **Input interlocking** was configured.	*never*
Input interlock checked successfully	Is triggered as soon as the input lock happens (formula for the input lock = TRUE). Makes only sense if property **Input interlocking** was configured.	*never*
Finished writing value TAG	Is triggered if all command tags have been written.	*never*

Event	Description	Allow execution before start event
	Recommendation: Use this event to inform the PLC that the phase has written all command tags and that the PLC can start processing the technological function.	
Value TAG modified during process	Is triggered if the value of a value parameter is amended whilst the phase done condition is checked.	*never*
Amended value TAG fully written	Is triggered if amended value parameters have been successfully written to the PLC.	*never*
Phase done condition completed	Is triggered if the phase has been executed in full (formula for the phase done condition = TRUE).	*never*
Phase deactivated	Is triggered if the phase is deactivated regardless of whether it has been executed successfully, stopped or canceled.	*adjustable*

EXECUTION ERROR

Events in relation to execution errors.

Event	Description	Allow execution before start event
Checking error	After a phase is activated, its parameters are checked. With an invalid configuration (such as value outside limits), the checking error event is triggered and the execution of the phase is held until the error has been rectified.	*always*
Phase started multiple times	A phase can only be active once. If an instance of a phase is activated whilst another instance of the same phase is active, this event is triggered and the execution of the phase is held until the other phase has been deactivated.	*always*

Event	Description	Allow execution before start event
Waiting period unit allocation exceeded	Is triggered if the waiting duration for the unit allocation runs. Can also occur during *Paused* and *Held*.	*always*
Waiting time for confirmation of the value change of the initial TAG exceeded	Is triggered if amended initial parameters have not yet been confirmed after expiry of the waiting time.	*adjustable*
Waiting period input lock exceeded	Is triggered if the waiting duration (time out) for the input interlocking expired. Makes only sense if a **Input interlocking** was defined.	*never*
Value TAG without value	Is triggered if a Boolean command tag is to be toggled and the variable assigned to the tag does not have a valid value.	*never*
Waiting time for confirmation of the value change of the value TAG exceeded	Is triggered if amended value parameters have not yet been confirmed after expiry of the waiting time.	*never*
Maximum execution duration exceeded	Is triggered if the waiting duration for waiting for the phase done condition (maximum execution duration) has been exceeded.	*never*
Waiting period following condition exceeded	Is triggered if the phase was not deactivated within the envisaged waiting time (time out) although the phase done condition has been met.	*adjustable*
Linked variable interrupted	▸ If the value of a variable with state INVALID should be used, this event is created once per invalid variable and phase. ▸ If the variable status changed from INVALID to not INVALID and back to INVALID, the reaction is again triggered when the variable is used. ▸ If the phase is restarted, the event is triggered again when an invalid	*adjustable*

Event	Description	Allow execution before start event
	variable is used. At the following activities it is checked for invalid variable: - Source variable in reaction - Variable for phase done condition - Variable for input interlocking - Write command tag inversely - Variables for conditions with transient status - **Note:** In the event of an *INVALID*, the events are not necessarily processed in the order in which they are received. If an *INVALID* occurs whilst another event is being processed, this event can overtake the one that is currently being executed.	

STATUS CHANGE

Status change events. If the phase changes its status, the corresponding event is triggered.

Event	Description	Allow execution before start event
Status change: In execution	The phase is executed.	*adjustable*
Status change: Pausing	The phase is currently being paused.	*adjustable*
Status change: Paused	The phase has been paused.	*adjustable*
Status change: Resume	The phase is resuming after a break. A status change in the object from *Paused* to *Running* triggers the events *Resuming* and *Running*.	*adjustable*
Status change: Holding	The phase is held at the moment.	*adjustable*

Event	Description	Allow execution before start event
Status change: Held	The phase was stopped.	*adjustable*
Status change: Restarting	The phase is restarting at the moment.	*adjustable*
Status change: Stopping	The phase is stopping at the moment.	*adjustable*
Status change: Stopped	The phase was stopped.	*adjustable*
Status change:Aborting	The phase is aborted at the moment.	*adjustable*
Status change: Aborted	The phase was aborted.	*adjustable*
Status change: Executed	The phase is finished.	*adjustable*
Escape condition started	Is triggered if the **Escape condition** for exiting from a phase is started.	*adjustable*
Escape condition met	Is triggered if the **Escape condition** for exiting from a phase is met.	*adjustable*

MODE CHANGE

Events in relation to mode switch in the REE

Event	Description	Allow execution before start event
Mode change: Automatic	The REE switched to *automatic* mode.	*adjustable*
Mode change: Semi-automatic	The REE was switched to *semi-automatic* mode.	*adjustable*
Mode change: Manually	The REE switched to mode *Manual*.	*adjustable*

CLOSE AND RESTART RUNTIME

Events in relation to closing and restarting Runtime.

Event	Description	Allow execution before start event
Exit Runtime initiated	Is triggered if the Runtime is exited. This is an especially critical state for the Batch Control module as the recipe process does not stop in the control immediately. Therefor exiting the Runtime is prevented as long as module Batch Control saved all data. A process image is created which can later be used as starting point. Likewise it is made sure that the tags of action **Write set value** safely arrive at the control. Internally the phase is paused only when the writing confirmation from the driver ensued. For more details about existing the Runtime see chapter: Exit and restart Runtime (on page 280). When this event occurs, no reaction types of group **Influencing the recipe** are possible.	*adjustable*
Runtime restart	The Runtime was restarted.	*adjustable*

INTERRUPTIONS AND ERRORS

Events in relation to interruptions and errors in communication and on the PLC.

Event	Description	Allow execution before start event
Loss of communication	This event reports that communication to the PLC has been interrupted. This is triggered by a failed write attempt of command tags, or if the formula for the communication fault is TRUE.	Only from the point at which the values for the **Loss of communication** property are waited for.
Loss of communication fixed	This event reports that the communication failure has been rectified.	*adjustable*

Event	Description	Allow execution before start event
Loss of communication acknowledged	This event reports that a displayed communication failure has been acknowledged.	*adjustable*
PLC error	Is triggered if there is a PLC error.	*adjustable*
PLC error fixed	Is triggered if a PLC error has been rectified.	*adjustable*
PLC error fixed by deactivation of the phase	Is triggered if there was a PLC error when a phase was deactivated. This was changed to *rectified* when deactivated. Does not apply for a restart of the phase.	*adjustable*

6.2.8.5 Reaction types

In the properties of the reactions the reaction types are more precisely defined and engineered. The following properties are available in the **Reactions** group:

Group	Description
Tag write set value	Influences both command tags as well as return / PLC sync tag. All tag data types can be used. **Attention:** The value must be within of the set value limits of the variables which are linked at the tag. If this is not the case, an error message is created during the validation. Reactions can only use tags of their own phase. If reactions are copied from other phases, they try to use tags with the same name of the name phase.
CEL entry	Creates an entry in the CEL. With this the reaction can be documented and the recipe process can be tracked later. To do this, the **Create CEL entry** property must be activated. The text for the CEL is defined in the **CEL message text** property. As a user, the *system* CEL message is entered.
Function	Makes it possible to link any zenon function. With this you can e.g. call up a pop-up in order to inform the user about a certain status or start a data backup.

Group	Description
	Note: In the network the function is always executed at the server.
Allocate tag	Makes it possible to perform a value assignment from **Source tag** to another **Target tag**. Both command tags and return / PLC tags can be used. The data type of source and target tag must be identical otherwise an error is displayed at the validation of the recipe. Reactions can only use tags of their own phase. If reactions are copied from other phases, they try to use tags with the same name of the name phase.
Recipe influence	Make it possible to: - Change the execution mode - Execute recipe commands - execute phase commands It is thus possible to react to serious events such as *invalid linked variable*. **Note:** Use this reaction type carefully because this reaction type influences the entire recipe process For each event you can only once: - set the **Mode** and - set a **Command**. Because e.g. it does not make sense to pause and hold the recipe at the same time with the same event.

6.2.8.6 Reactions in unit classes and unit instances

The following is applicable for reactions in phases of a unit instance:

- If a reaction is created, deleted or renamed in a phase of the unit class, this change is applied to the linked unit instances.
  Changes to the reaction type and the execution sequence are also applied.
- If a reaction is newly created in a phase of a unit instance, this is a local reaction.
  It is highlighted in the detail view and selection lists with its own background color. It has no linking to a higher-level class and is configured completely locally. It is displayed with an

activated checkbox for the **Local** property. This property is only present for linked phases. It is not also displayed for local phases.

- If a reaction in a unit class is renamed and this name does not correspond to a local reaction, the local reaction is automatically renamed. The uniqueness of the reaction names is thus retained.
  Schematic diagram for renaming: **local <n> <OriginalName>**
  Example: **Reaction** becomes **local 0 reaction**.
- The properties of linked reactions can be amended locally in the unit instance.
  Exception: The **Event** property cannot be changed.
- Linked reactions cannot be deleted in the unit instance.
- If a reaction is copied and pasted from an instance, it loses any linkings to a class that may be present. It is always local.
- Only the locally-created reactions can be renamed or deleted in a unit instance.
- When compiling for Runtime, each derived reaction for all properties that are linked to a class takes on the values configured in the unit class. In the Runtime, the reaction has precisely the values that were displayed in the Editor.

INHERITANCE OF THE EXECUTION ORDER

If several reactions have been defined for the same event, the order of their execution must be defined. This happens using the **Execution order** or in the list via the toolbar or the context menu of the event. Valid values depend on the existing reactions.

The following is applicable for the inheritance of the execution sequence:

- Inheritance is per type, not per reaction.
  The execution sequence of the reactions always depends on the other reactions of the same type. The inheritance and restoration is thus always carried out for all reactions of a type.
- The following is applicable for an active inheritance of the order:
  - All local reactions of a type are always sequenced behind the inherited.
  - Changes to the order from the unit class are applied directly
  - If new reactions are added in the unit class, their instances are then inserted at the same place in the execution sequence. All local reactions of the type are moved to the back.
- Breaking the inheritance of the sequence.
  The inheritance of the execution sequence can be interrupted by:
  - Moving a reaction in the sequence.
  - Swapping the sequence of two reactions.
  - Setting the sequence of a reaction.
  - Explicit separation of the inheritance by means of the property's context menu.

  - If, in the event of an interrupted inheritance, new reactions are created in the unit class, the instances that arise as a result are added to the unit instance at the last place.

  Note: A change to the sequence only causes a a breaking of the inheritance if a linked reaction is affected as a result. If only local reactions are affected, the inheritances are retained.

- Restoration of the inheritance of the sequence:
  If the inheritance is restored, the sequence for all inherited reactions is set to the value in the unit class. All local reactions are placed in sequence behind it.

  Note: The relative sequence of the local reactions to one another is not changed in the process.

INHERITANCE OF PROPERTIES FOR STATUS OR MODE COMMANDS

The properties for *status* or *mode* commands can only be edited individually. A simultaneous change to reactions of the same type is not possible.

The following is applicable for their inheritance from a unit class:

- Inheritance per type only, not per object.

  The inheritance of the command is only possible for each reaction type and not per object. Only one reaction of a type can have a status command or mode command. Dependencies between the reactions thus arise.

- If the inheritance is active, changes to the commands are applied from the unit class directly. If changes in the unit class are in conflict with the commands of local reactions, the commands are removed from the local reactions
- The following is applicable if the inheritance is interrupted:
  - Changes to properties are no longer applied.
  - If a new reaction is inserted into the unit class whose commands are in conflict with settings in the unit instance, the command is removed from the new reaction. This is also applicable for changes to the type of a reaction with commands.
- Restoring the inheritance:

  When restoring the inheritance of one of the command properties, all values of the inherited reactions are applied from the unit class again. If there is a conflict with a local reaction in the process, the command is removed from the local reaction.

6.2.8.7 Behavior in the Runtime

For each reaction, a range of actions can be configured in the **Reactions** properties group, such as **Tag write set value**, **Function**, **Allocate tag** or **Recipe influence**. For example, in order to change the REE mode after restarting the Runtime, or to execute an REE command when writing command parameters.

In doing so, please note:

- The execution of these actions in the Runtime can be prevented by other events.
  For example, the status of the phase can prevent the execution of an action.
- Several different reactions are processed in the same execution cycle. However, only one REE command per cycle is permitted. In general, the last REE command is executed.

Note: Actions of reactions are logged in the CEL if the **Create CEL entry** property has been activated for the reaction. *System* is entered as a user.

6.2.9 Control strategies

Control strategies make it possible to set parameters for different versions of a phase. Only the command parameters allocated to the control strategy are sent for each control strategy.

CONFIGURING CONTROL STRATEGIES

To use control strategies, these must be activated in the phase. To do this:

1. Highlight the desired phase
2. Go to property group **Control strategies**\
3. Activate the checkbox in front of the **Active control strategies** property.
4. Select a parameter in the **control strategy tag** property. This parameter defines which control strategy is active in the Runtime.
5. The phase is thus displayed in the Editor with the **Control strategies** node.
6. Right click on the node and select the entry **New control strategy** in the context menu.
7. A new control strategy is created.
8. Configure the properties of the control strategy.
   In doing so, please note:
   - **Name** and **control strategy number** of the control strategy must be unique within the phase.
   - The **Name** must not be empty, contain a dot, consist of only spaces and must be within a maximum of 256 characters long.
9. Add the desired command parameters.

   Note: Clicking on a parameter add its properties.
   Only the following properties in the **Write set value** group can be edited:

   - **Tag value**

- **Min. value**
- **Max. value**
- **Parameter value can be changed in master recipe**

All other properties cannot be edited. To edit these, switch to the parameter list of the phase.

COPYING CONTROL STRATEGIES

Control strategies can be copied using commands in the context menu and the toolbar and inserted into the same or other phases.

If control strategies are copied throughout phases, units or a project, only the parameter linkings that are also to be triggered in the new phase are inserted. In doing so, the conditions are the same as for inserting parameter linking (on page 66).

6.2.9.1 Control strategies node context menu

Right-clicking on the **Control strategies** node opens a context menu with the following entries:

Parameters	Description
New Control strategy	Creates a new control strategy
Paste control strategy	Pastes a control strategy from the clipboard. Copied control strategies are adapted when pasted into a phase so that **Name** and **control strategy number** are made unique, if this is not already the case in the phase.
Help	Opens online help.

6.2.9.2 Context menu selected control strategy

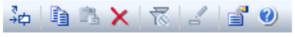
Right-clicking on the control strategy opens a context menu with the following entries:

Parameter	Description
Add command TAG	Opens the dialog to select command parameters (on page 308).
Paste	Adds parameters that have been copied from a different control strategy to the selected control strategy. **Note:** Copy the parameters to the control strategy list.

Parameter	Description
Rename	Highlights the name to be renamed.
Delete	Deletes the selected control strategy after a confirmation message.
Copy	Copies the selected control strategy. This can be pasted using **Paste** in the context menu of the **Control strategies** node. Copied control strategies are adapted when pasted into a phase so that **Name** and **control strategy number** are made unique, if this is not already the case in the phase.
Help	Opens online help.

6.2.9.3 Toolbar and control strategy list context menu

Entries in the control strategy list can be edited using symbols or entries in the context menu.



The following are available in the context menu and the toolbar:

Parameter	Description
Add command parameters	Opens the dialog to select command parameters (on page 308).
Copy	Copies the selected entries to the clipboard.
Paste	Pastes the content from the clipboard. If an entry with the same name already exists, the content is pasted as "**Copy of...**".
Delete	Deletes selected entries after a confirmation from list.
Remove all filters	Removes all filter settings. **Note:** Only active if the current view is filtered.
Edit selected cell	Opens the selected cell for editing. The binocular symbol in the header shows which cell has been selected in a highlighted line. Only cells that can be edited can be selected.

Parameter	Description
Properties	Opens the **Properties** window.
Help	Opens online help.

6.2.9.4Control strategies in unit classes and unit instances

The following is applicable for control strategies in phases of a unit instance:

- If a control strategy is created, deleted or renamed in a phase of the unit class, this change is applied to the linked unit instances.
- If a control strategy is newly created in a phase of a unit instance, this is a local control strategy.
  It is highlighted in the detail view and selection lists with its own background color and it gets its own symbol. It has no linking to a higher-level class and is configured completely locally. It is displayed with an activated checkbox for the **Local** property. This property is only present for linked phases. It is not also displayed for local phases.
- If a control strategy in a unit class is renamed and this name does not correspond to a local control strategy, the local control strategy is automatically renamed. The uniqueness of the reaction names is thus retained.
  Schematic diagram for renaming: **local <n> <OriginalName>**
  Example: **Control strategy** becomes **local 0 control strategy**.
- If there are the same control strategy numbers present in in the unit class and in the unit instance, the number of the local control strategy is changed to a free number. This change is documented in a log entry.
- The properties of linked control strategies cannot be edited locally in a unit instance.
  Exception: The **Description** property can be amended.
- Linked control strategies cannot be deleted in the unit instance.
- If a control strategy is copied and pasted from an instance, it loses any linkings to a class that may be present. It is always local.
- Only the locally-created control strategies can be renamed or deleted in a unit instance.
- When compiling for Runtime, each derived control strategy for all properties that are linked to a class takes on the values configured in the unit class. In the Runtime, the control strategy has precisely the values that were displayed in the Editor.

PARAMETERS IN CONTROL STRATEGIES OF UNIT INSTANCES

The following is applicable for parameters in control strategies of a unit instance:

- With linked control strategies, only local command tags can be linked or removed.
- With local control strategies, local command tags and the command tags of the class can be linked.
- If a parameter is created, deleted or renamed in a control strategy of the unit class, this change is applied to the linked unit instances.
- If a parameter is newly created in a control strategy of a unit instance, this is a local tag. It is highlighted in the detail view and in selection lists with its own background color. It has no linking to a higher-level class and is configured locally. It is then shown by the **Local** property with an activated checkbox.

INHERITANCE OF THE PARAMETER VALUES

Parameter values in control strategies are applied as:

- Phase or unit instance
  or
- Control strategy of the phase of the unit class

Derivations result from the unit class.
Exception: The value of the parameter in the phase of the unit instance has been changed locally. The derivation is then local.

6.2.9.5 Parameters in control strategies

LINKING PARAMETERS

There are different methods available to link parameters to a control strategy:

- To do this, select the **Add command parameter** command in the context menu of the control strategy.
- Select, in the detail view of the control strategies in the toolbar or in the context menu of a parameter, the **Add command parameter** command.
- Drag the parameter from the parameter list of the phase by dragging & dropping it onto the control strategy. Only command parameters (on page 35) are linked.

Note: If a parameter is deleted for a phase, the attendant parameter linking is also deleted for all control strategies.

DELETING OR RESTORING LINKED VALUES

Parameters of the control strategies take on the values of the parameters with which they are linked. This link can be removed by:

- Overwriting the value
- Separating the linking via the context menu

The context menu can also be used to restore the link to the source parameter again.

You can read details on linked values in the **Linked properties** chapter in the **Editor** manual.

COPYING PARAMETERS

Existing parameter linking can be copied between control strategies. Copying is possible throughout via phases, units and projects. When inserting parameter linkings, an attempt is made to link the names accordingly. No new parameters are created.

Insertion is not possible if there is no parameter with the copied name in the target phase or the corresponding parameter is already part of the control strategy.

PARAMETERS IN CONTROL STRATEGIES OF UNIT INSTANCES

For the behavior of parameters in control strategies of a unit instance, see the corresponding section in the **Control strategies in unit classes and unit instances** (on page 65) chapter.

6.3 Replace links

Linking of variables and functions can be replaced automatically in units, phases and reactions. This process corresponds to the process for **replacing linking for screen switching** and **replacing linking in the Editor screen.**

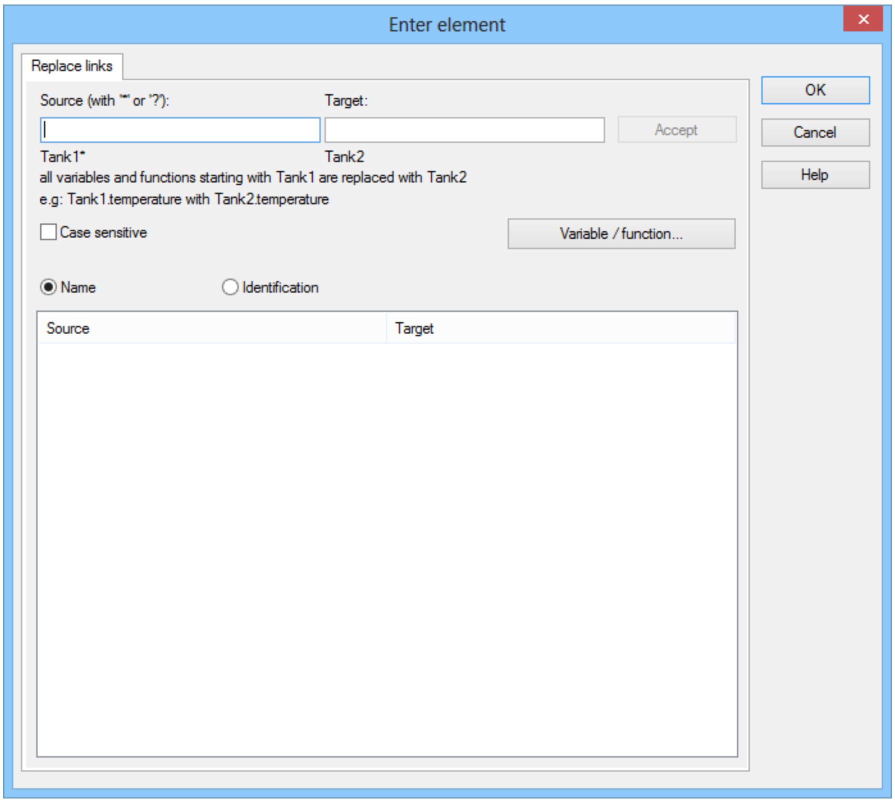
The following can be replaced:

- Units and unit classes: linked variables for Runtime information
- Reactions: linked functions
- Parameter: linked variables

To replace elements:

1. Select the **Replace linking** command in the context menu or the toolbar

2. The dialog for the replacement of links opens.



Tank1* Tank2
all variables and functions starting with Tank1 are replaced with Tank2
e.g: Tank1.temperature with Tank2.temperature

□ Case sensitive

● Name ○ Identification

Property	Description
Source	Enter the partial string to be searched for. The placeholders ***** and **?** can be used: - *****: desired number of characters ***aste** finds **Taste**, **Stopptaste**, ... ***aste*** finds **Tasten**, **Tastendruck**, **Pasteurisierung**, ... - **?**: precisely one character. For example: **T?ste** finds **Teste** and **Taste**. The placeholder ***** is permitted as a prefix and/or suffix. When using placeholders, note when a character appears more than once. The first thing found is always replaced when several things are found: - Example character sequence: **01{SU(00,Test1)}** **Test1** should be replaced with **Test 2**. - Source entry **1* and target entry *2* finds and replaces the *1* in *01* but not in the subsequent following *Test1*. Result: **02{SU(00,Test1)}**

Property	Description
	- Source entry **test1* and target entry *test2* finds and replaces *Test1*. Result: **01{SU(00,Test2)}**
Target	Entry of the partial string **Note:** Source and target must be in the same project.
Apply	Swaps target strings from the **source** for those defined in the **target**.
Note capitalization	When swapping, be sure that any capitalization is an exact match.
Name	Swaps information in process variable names.
Identification	Exchanges information in the identification
Variable/function	Opens the selection list for variables/functions in relation to the selected line in the list. Clicking on the variable or function in the list defines new target variable or target function. Alternative: Double-click on the corresponding source variable or source function.

REPLACE

REPLACE WITH MANUAL SELECTION

To replace elements manually:

- Select the element from the list that you would like to replace as the source.
- Select a target element via the **Variable/Function** button

  The previous element is replaced by the new one.

AUTOMATED REPLACEMENT WITH RULES

To automatically replace elements on the basis of rules:

- In the **Source** input field, define the parameters for the element that you wish to replace
- Define the parameter for the new variable/function in the **Target** input field
- Specify what is to be replaced via **Name/Identification**.
- Click on **Accept**.

Information

The target variable or target function can also be in a different project as the source variable or source function. In doing so, all projects concerned must be started and available on the same computer in Runtime.

Internal variable IDs are used for replacement. This means that if variables are used or functions are renamed, the replacement remains.

⚠Attention

When replacing variables, be aware of the type and signal resolution. If you replace a variable with one of an incompatible type, this can lead to errors during execution. You will be warned when making the substitution; the substitution will however be carried out.

6.4 Create screen of type Batch Control

ENGINEERING

Two procedures are available to create a screen:

- The use of the screen creation dialog
- The creation of a screen using the properties

Steps to create the screen using the properties if the screen creation dialog has been deactivated in the menu bar under **Tools**, **Settings** and **Use assistant**:

1. Create a new screen.

   To do this, select the **New screen** command in the tool bar or in the context menu of the **Screens** node.

2. Change the properties of the screen:

   a) Name the screen in the **Name** property.

   b) Select *Batch Control* in the **Screen type** property.

   c) Select the desired frame in the **Frame** property.

3. Configure the content of the screen:

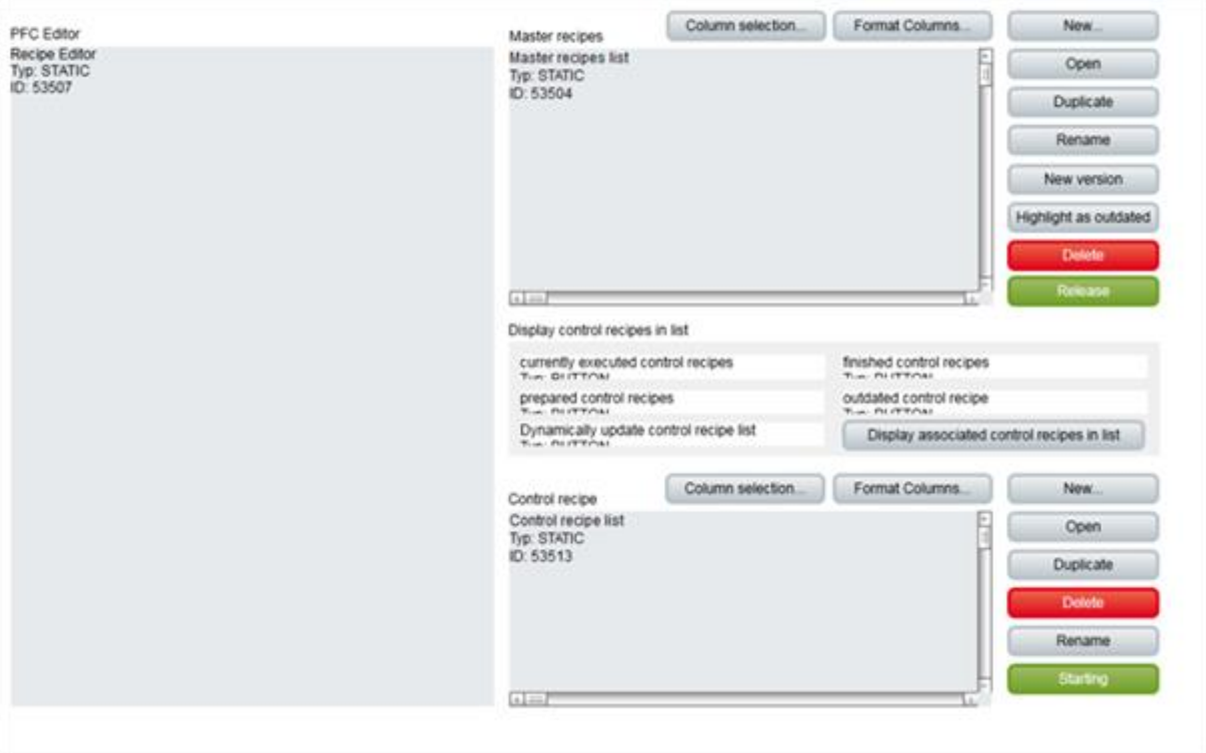
   a) Select the **Elements (screen type)** menu item from the menu bar.

b) Select *Insert template* in the drop-down list.
The dialog to select pre-defined layouts is opened. Certain control elements are inserted into the screen at predefined positions.

c) Remove elements that are not required from the screen.

d) If necessary, select additional elements in the **Elements** drop-down list. Place these at the desired position in the screen.

4. Create a screen switch function.



6.4.1 Elements (Batch Control)

You add control elements to the screen using the **Elements (Batch Control)** menu.

INSERT TEMPLATE

Entry	Description
Insert template	Opens the dialog for selecting a template for the screen type. Templates are shipped together with zenon and can also be created by the user. Templates add pre-defined control elements to pre-defined position in the screen. Elements that

Entry	Description
	are not necessary can also be removed individually once they have been created. Additional elements are selected from the drop-down list and placed in the zenon screen. Elements can be moved on the screen and arranged individually.

ELEMENTS (BATCH CONTROL)

Control element	Description
Default (master recipes)	Inserts control elements for master recipes on predefined locations on the screen. These control elements can be supplemented, reduced and positioned newly.
Default (control recipes)	Inserts control elements for control recipes on predefined locations on the screen. These control elements can be supplemented, reduced and positioned newly.
Recipe Editor	Adds the licenses editor for creating master and control recipes.

MATRIX RECIPE CREATION

Control elements to create matrix recipes.

Control element	Description
Delete column/row	Deletes a line or column. (depending on selection.)
Move columns to the left/steps up	Moves columns to the left or moves steps up (depending on selection.)
Move columns to the right/steps down	Moves columns to the right or moves steps down (depending on selection.)
Activate selected elements	Activates the selected elements. (depending on selection.)
Deactivate selected elements	Deactivates the selected elements. (depending on selection.)

RECIPE CONTROL

Control elements to control recipes.

Control element	Description
General	General control elements: - **Save master recipe** - **Check recipe for errors** - **Switch recipe to test mode** - **Release recipe** - **Switch recipe to edit mode** - **Close recipe** - **Close all other recipes** - **Edit element** - **Graphical design** - **Confirm changes**
Execution commands of the recipe	Control elements for recipe commands: - **Start recipe** - **Pause recipe** - **Resume recipe** - **Hold recipe** - **Restart recipe** - **Stop recipe** - **Abort recipe**
Execution commands of the phases	Control elements for phase commands: - **Pause phase** - **Resume phase** - **Hold phase** - **Restart phase** - **Escape phase**
Switch execution mode	Control elements for execution modes: - **Switch to automatic mode** - **Switch to semi-automatic mode** - **Switch to manual mode**

Control element	Description
	- **Compatible elements**
Execution navigation	Control elements for navigation in recipes: - **Continue recipe at selected execution position only** - **Continue recipe at all execution positions** - **Skip active condition**
Operation navigation	Control elements for navigation in operations: - **Change to operation template** - **Switch to main recipe**

Compatible elements:

Standard Win32 control elements that have been replaced or removed by zenon elements (*dynamic text*, *switch*) and continue to be available due to compatibility reasons. These elements are not taken into account with automatic insertion of templates.

For the description, see current elements.

- **Switch to automatic mode**
- **Switch to semi-automatic mode**
- **Switch to manual mode**

MASTER RECIPES

Control elements for master recipes.

Control element	Description
Master recipes list	In this list all master recipes can be displayed. The display can be limited by filters to an individual selection. The filtering can be preset in the zenon Editor in the screen switch function (on page 83). Filtering in Runtime is also possible. These filter settings are not saved in the Runtime. This means that the filters must be entered again when called up again. A permanent pre-setting of the filter is only possible with configuration in the zenon Editor. All commands are also possible in the context menu of the list. The commands for list management can be called from the header of the

Control element	Description
	list. The commands for recipe management can be called at editing one or more recipes. The recipes in the list cannot be edited directly in the list. Renaming, changing the description or changing the recipe status is only possible with the corresponding commands.
Column selection master recipe...	Opens a dialog in order to determine which columns should be displayed (on page 115). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.
Format columns master recipe...	Opens a dialog to edit the column settings (on page 119). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.
New master recipe...	Opens dialog for creating a new master recipe (on page 162).
Create new version of master recipe	Creates a new version (on page 217) of the selected master recipe. This must be approved or marked as obsolete.
Rename master recipe	Only active if exactly one master recipe was selected. The dialog for the input of a unique name and the description is opened. Recipes can only be renamed if they are in status *Editable*. Also use this function in order to change the description of the control recipe. When renaming a recipe, a CEL entry is created.
Duplicate master recipe	Only active if exactly one recipe is selected. Created a copy of the selected recipe. At the creation of the copy, the version of the recipe saved on the hard disk is used. If the recipe is just edited in another computer and the changes have not yet been saved, the changes are not applied. The dialog for the input of a unique name and the

Control element	Description
	description is opened. The copy of the recipe automatically receives status *Editable* and can be edited further. When duplicating a recipe, a CEL entry is created.
Delete master recipe	Deletes the selected recipes irrevocably. If the recipe is opened on another computer for editing, it is automatically closed there. Deleting is only possible if there are no control recipes which are based on the master recipe. First you must delete all control recipes. Recipes which are currently executed in test mode (master recipe status: *Test in execution*) cannot be deleted. First they must be *finished*, *stopped* or *canceled*. If recipes must not be deleted - e.g. in an FDA-regulated environment - it is recommended that this button is not configured or that it is given an appropriate **Authorization level**. A CEL entry is created when a recipe is deleted.
Open master recipe	Opens the selected master recipe in the recipe editor if screen element *Recipe editor* exists in the screen. Each selected master recipe is opened in a separate tab of the recipe editor.
Switch master recipe to edit mode	Changes the master recipe status of the selected recipes to *Editable*. In this status, recipes can again be edited completely. Only recipes in *Test mode* can be set back to *Editable*.
Switch master recipe to test mode	Changes the master recipe status of the selected recipes to *Test mode*. Only faultless recipes can be switched to test mode. If error occur during the validation (on page 249), you must first fix them. Recipes in the test mode can be executed but no longer reengineered. For details about the states see chapter Recipe types and recipe states (on page 160).

Control element	Description
Release master recipe	Changes the master recipe status of the selected recipes to *Released*. Only recipes without errors can be released. If error occur during the validation (on page 249), you must first fix them. Only recipes in status *Test mode* and *Editable* can be released. Released recipes can no longer be edited. Control recipes can only be created from released recipes. For details about the states see chapter Recipe types and recipe states (on page 160). When releasing a recipe, a CEL entry is created.
Highlight master recipe as outdated	Changes the status of the recipe to *outdated*. The recipe can no longer be edited or approved. No control recipe can be created on the basis of this recipe.
Show related control recipes in list	Displays all control recipes that are based on the selected master recipe and that comply with the set filter criteria.
Dynamically update control recipe list.	Deactivates the **Display associated control recipes in list** button. When selecting a master recipe, all attendant control recipes are displayed automatically.
Filter for displaying the control recipe	Makes it possible to filter control recipes for the following criteria: **Prepared control recipes**: Display only control recipes which are prepared for execution. **Currently executed control recipes**: Displays control recipes which are currently executed. **Completed control recipes**: Displays control recipes which have already been executed. **Outdated control recipes:** Shows control recipes with the status *outdated*.

Control element	Description
	Filtering only takes effect once you click on **Show associated control recipes in list**.
Compatible elements	Standard Win32 control elements that have been replaced or removed by zenon elements (*dynamic text*, *switch*) and continue to be available due to compatibility reasons. These elements are not taken into account with automatic insertion of templates. For the description, see current elements. Compatible elements - **Dynamically update control recipe list** - **Prepared control recipes** - **Currently executed control recipes** - **Completed control recipes** - **Outdated control recipes**

CONTROL RECIPE

Control elements for control recipes.

Control element	Description
Control recipe list...	In this list all control recipes can be displayed. The display can be limited by filters to an individual selection. Per default the list is empty. The following must be the case for the list to be filled: - Master recipes must be selected - The filters: **currently-executed control recipes**, **prepared control recipes** and **completed control recipes** must be set - click button **display associated control recipes in list** In addition to the filters mentioned above, you can filter the list itself.The filtering can be preset in

Control element	Description
	the zenon Editor in the screen switch function (on page 83). Filtering in Runtime is also possible. These filter settings are not saved in the Runtime. This means that the filters must be entered again when called up again. A permanent pre-setting of the filter is only possible with configuration in the zenon Editor. All commands are also possible in the context menu of the list. The commands for list management can be called from the header of the list. The commands for recipe management can be called at editing one or more recipes. The recipes in the list cannot be edited directly in the list. Renaming, changing the description or starting the recipes is only possible with the corresponding commands.
Column selection control recipes...	Opens a dialog in order to determine which columns should be displayed (on page 115). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.
Format columns control recipes...	Opens a dialog to edit the column settings (on page 119). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.
New control recipe...	Opens the dialog (on page 233) for entering a unique name and a description for the control recipe. The uniqueness of the name is also checked in the zenon network. The name must only be unique within the master recipes. Control recipes which are based on other master recipes may have the same name. The uniqueness within module Batch Control is achieved by always referencing the master recipe name and the control recipe name. When creating a control recipe, a CEL entry is created.
Configure control recipe	Only active if exactly one control recipe was

Control element	Description
	selected. The dialog for the input of a unique name and the description is opened. Recipes can only be renamed if they have the status of *prepared*. Also use this function in order to change the description of the control recipe.
Duplicate control recipe	Only active if exactly one recipe is selected. Created a copy of the selected recipe. At the creation of the copy, the version of the recipe saved on the hard disk is used. If the recipe is just edited in another computer and the changes have not yet been saved, the changes are not applied. The dialog for the input of a unique name and the description is opened. The copy of the recipe automatically gets the status *Prepared* and can therefore be edited and started. The execution status (on page 263) of the duplicate is set to *automatic*. When duplicating a recipe, a CEL entry is created.
Delete control recipe	Deletes the selected recipes irrevocably. If the recipe is opened on another computer for editing, it is automatically closed there. Deleting is only possible if all selected recipes are not executed (control recipe status: *In execution*). First they must be *finished*, *stopped* or *canceled*. If recipes must not be deleted - e.g. in an FDA-regulated environment - it is recommended that this button is not configured or that it is given an appropriate **Authorization level**. A CEL entry is created when a recipe is deleted.
Open control recipe	Opens the selected control recipes in the recipe editor if screen element *Recipe editor* exists in the screen. Each selected control recipe is opened in a separate tab of the recipe editor.
Start control recipe	Starts the selected control recipe in the set execution mode. The recipes are executed automatically at the Server. It is not necessary that

Control element	Description
	the recipe is opened in the recipe editor.

PARAMETER LISTS

List fields for the display of parameters.

Two list boxes can be created. These are configured in the screen switching (on page 108).

Control element	Description
TAG List 1	First parameter list.
TAG List 2	Second parameter list

CONTROL STRATEGIES

Control elements for control strategies.

Control element	Description
Control strategies list	Shows the control strategies available for the selected phase.
Apply control strategy	Applies the control strategy selected in the list to the phase.

PLACEHOLDER

Control elements for placeholders.

Control element	Description
Placeholder list	Shows the available placeholders for units. Elements in the list can be edited in the Runtime using a context menu: ▸ **New**: Creates a new placeholder. ▸ **Configure**: Opens the dialog to name the element and to assign it to a group. ▸ **Delete:** Deletes the selected element. Multiple selection is possible. ▸ **New group**: Creates a new group. ▸ **Allocate unit instance**: Opens the dialog to allocate a unit instance.

Control element	Description
	If unit classes have been deleted, they are shown in the list and given a notice. If changes from another client are saved in the zenon network, the list is updated for all clients.
New	Creates a new placeholder. Available for master recipes and operations in edit mode With partial recipes, the entries are displayed in the list of the host recipe.
Configure	Opens the dialog to configure the selected placeholder or the placeholder group. Here you can also assign placeholders to a placeholder group. The assignment of placeholders to a placeholder group is also possible by means of drag&drop.
Delete	Deletes the selected placeholder after a request for confirmation. Multiple selection is possible. Deleting a placeholder makes phases of the placeholder that already exist in the recipe invalid; they must be reconfigured. If a placeholder group is deleted, the placeholders contained therein are assigned to the group of **placeholder without group**. This group cannot be deleted.
New group	Creates a new placeholder group. A placeholder is assigned to a group using the **Configure** entry in the context menu of the placeholder.
Allocate unit instance	Opens a dialog to assign a unit instance to a placeholder. To do this, all selected placeholders must be linked to the same unit class.
Column selection	Opens the dialog (on page 115) to configure the columns that are displayed

Control element	Description
Column format	Opens a dialog (on page 119) to format the columns.

XML IMPORT/EXPORT

Control element for XML import/export.

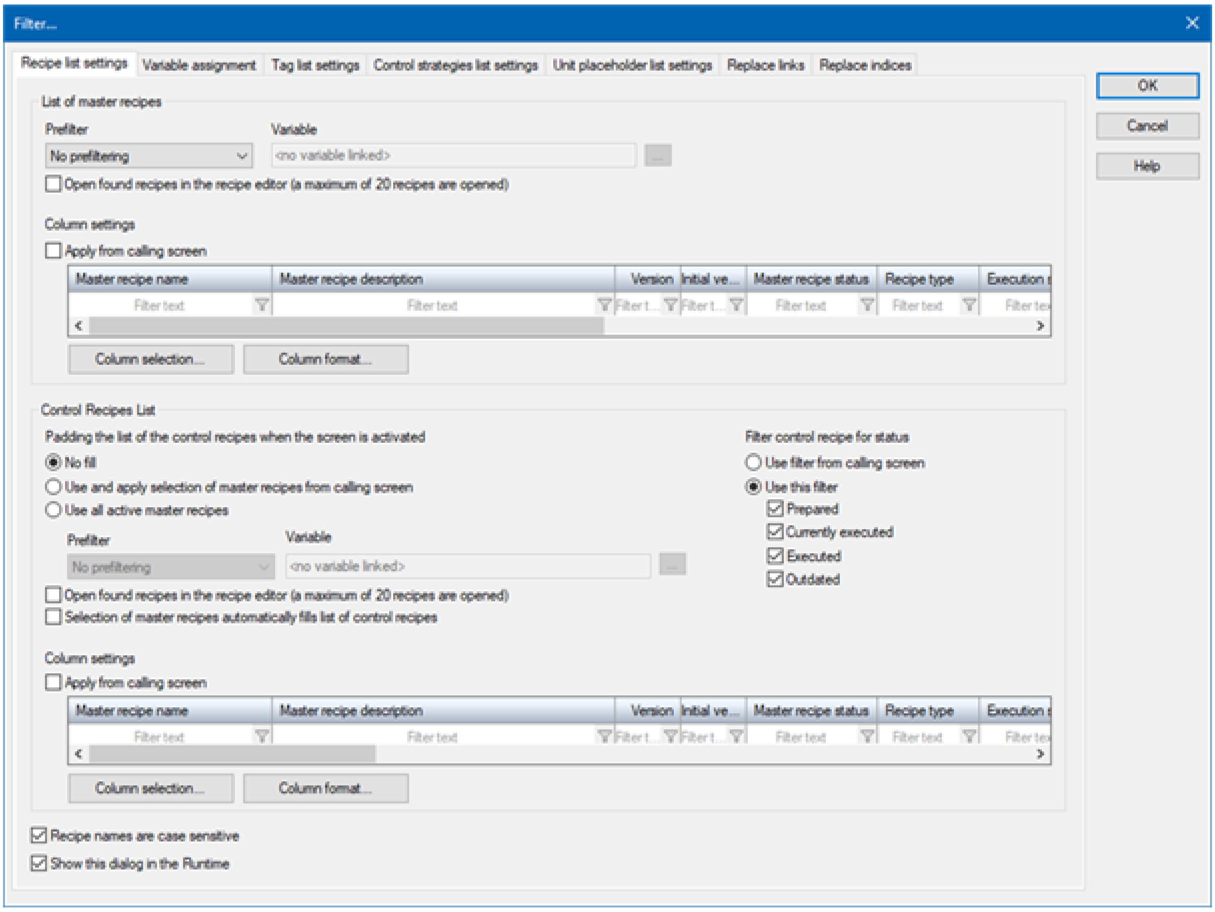
Control element	Description
Export master recipes...	Exports the selected master recipe as an XML file.
Export control recipes...	Exports the selected control recipe as an XML file.
Import recipes...	Imports recipes from the selected XML file.

6.5 Screen switch Batch Control

To use Batch Control in the Runtime, engineer a screen switch function to a screen of type *Batch Control*:

1. Select the **New function** command in the **Functions** node
2. Select the **Screen switching** function
3. Select *Batch Control* screen.

The filter dialog (on page 84) is displayed.



List of master recipes

Prefilter: No prefiltering

Variable: <no variable linked>

□ Open found recipes in the recipe editor (a maximum of 20 recipes are opened)

Column settings

□ Apply from calling screen

Control Recipes List

Padding the list of the control recipes when the screen is activated

● No fill

○ Use and apply selection of master recipes from calling screen

○ Use all active master recipes

Prefilter: No prefiltering

Variable: <no variable linked>

□ Open found recipes in the recipe editor (a maximum of 20 recipes are opened)

□ Selection of master recipes automatically fills list of control recipes

Filter control recipe for status

○ Use filter from calling screen

● Use this filter

☑ Prepared

☑ Currently executed

☑ Executed

☑ Outdated

Column settings

□ Apply from calling screen

☑ Recipe names are case sensitive

☑ Show this dialog in the Runtime

4. Configure the:

   a) Settings (on page 84) for the list of the master recipes/control recipes including prefiltering (on page 88)

   b) Variable allocations (on page 91)

   c) Tag list settings (on page 108)

   d) Settings for the list of control strategies (on page 112)

   e) Settings for the unit placeholder list (on page 114)

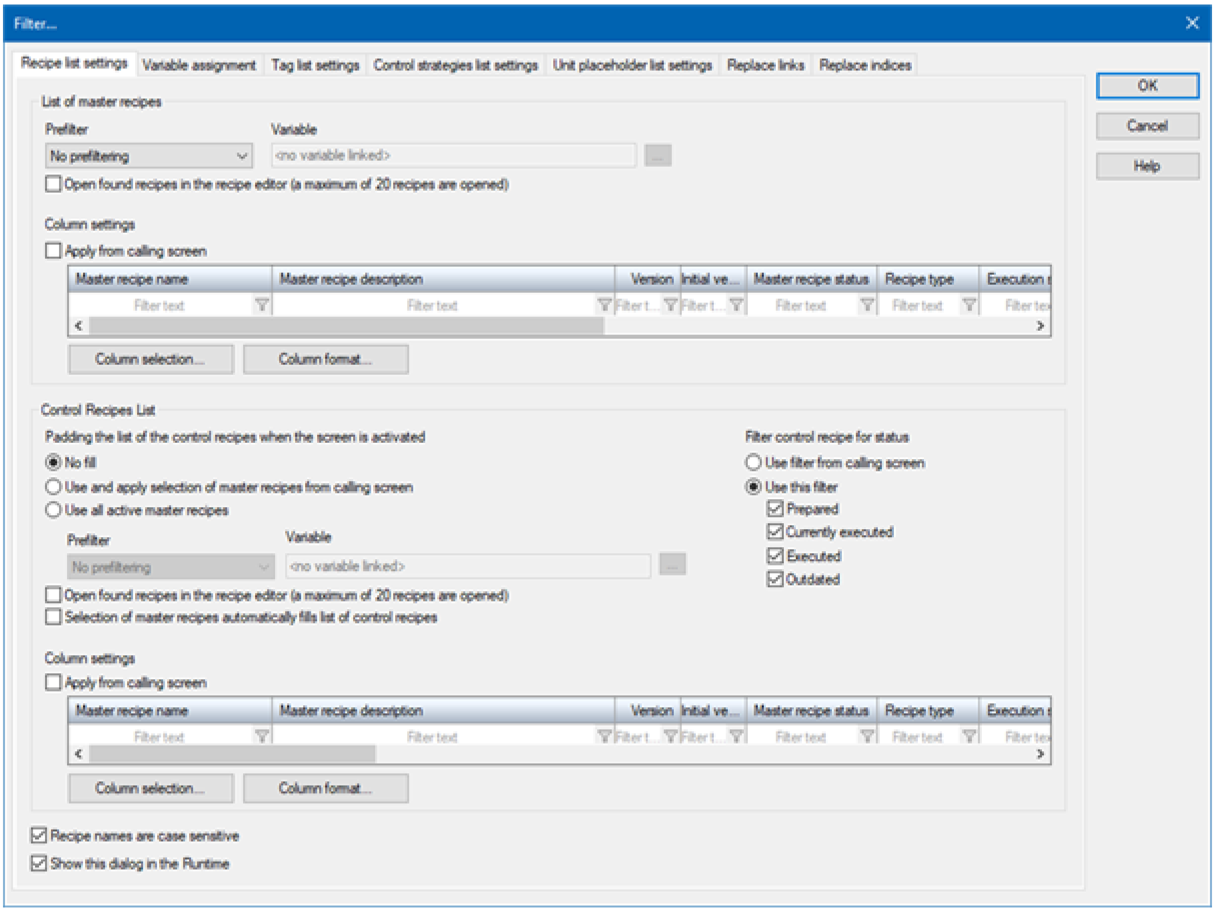
5. Link the function to an element in order to switch in the Runtime.

6.5.1 Recipe list settings

The settings are configured for:

- List of master recipes
- Control Recipes List

- Prefiltering (on page 88)



Filter...

Recipe list settings | Variable assignment | Tag list settings | Control strategies list settings | Unit placeholder list settings | Replace links | Replace indices

List of master recipes

Prefilter: No prefiltering

Variable: <no variable linked>

□ Open found recipes in the recipe editor (a maximum of 20 recipes are opened)

Column settings

□ Apply from calling screen

Master recipe name	Master recipe description	Version	Initial ve...	Master recipe status	Recipe type	Execution s
Filter text	Filter text	Filter t...	Filter t...	Filter text	Filter text	Filter tex

Control Recipes List

Padding the list of the control recipes when the screen is activated

● No fill

○ Use and apply selection of master recipes from calling screen

○ Use all active master recipes

Prefilter: No prefiltering

Variable: <no variable linked>

□ Open found recipes in the recipe editor (a maximum of 20 recipes are opened)

□ Selection of master recipes automatically fills list of control recipes

Filter control recipe for status

○ Use filter from calling screen

● Use this filter

☑ Prepared

☑ Currently executed

☑ Executed

☑ Outdated

Column settings

□ Apply from calling screen

Master recipe name	Master recipe description	Version	Initial ve...	Master recipe status	Recipe type	Execution s
Filter text	Filter text	Filter t...	Filter t...	Filter text	Filter text	Filter tex

☑ Recipe names are case sensitive

☑ Show this dialog in the Runtime

LIST OF MASTER RECIPES

Configuration for master recipes.

Option	Description
Prefiltering	Select from a drop-down list whether master recipes should be pre-filtered when called up. Possible settings: - *No prefiltering*: Recipes are not pre-filtered. All master recipes are shown in the list. - *ID from variable*: Recipes are filtered according to ID. Filter condition is defined in the **Variable** property. - *Name from variable*: Recipes are filtered according to recipe name. Filter condition is defined in the **Variable** property.

Option	Description
Variable	Definition of the variables that provide the values for the prefiltering. Click on button **...** in order to open the dialog for selecting the variable.
Apply column settings from calling screen	*Active*: The column settings are transferred from the calling screen in the Runtime, i.e. the screen in which the button is configured with the screen switching function. The calling screen must also be a *Batch Control* screen. The corresponding properties can no longer be configured in the Editor.
List field	Display of the configured columns.
Column selection	Opens the dialog for selecting the columns (on page 115).
Column format	Opens a dialog (on page 195) to format the columns.

CONTROL RECIPES LIST

Configuration for control recipes.

Option	Description
Filling the list of the control recipes when the screen is activated	Settings for the filling of the list when called up. Select an option. - *No filling* - *Apply and use selection of the master recipes from calling screen* - *Apply all active master recipes*
Prefiltering	Select from a drop-down list whether control recipes should be prefiltered when called up. Possible settings: - *No prefiltering*: Recipes are not pre-filtered. - *ID from variable*: Recipes are filtered according to ID. Filter condition is defined in the **Variable** property. - *Name from variable*: Recipes are filtered according to recipe name. Filter condition is defined in the **Variable** property. - *Job ID from variable*: Recipes are filtered according to job ID. Filter condition is defined in the **Variable** property.
Variable	Definition of the variables that provide the values for the prefiltering. Click on button **...** in order to open the dialog for

Option	Description
	selecting the variable.
Open found recipes in recipe editor	*Active*: The recipes shown in the list when called up are opened in the recipe editor. **Note:** The first 20 recipes found is the maximum that can be opened.
Selection of master recipes automatically fills list of control recipes	*Active*: The list of control recipes always shows the control recipes of the master recipes at the respective point in time. in the Runtime, it is then only necessary to click on the **Dynamically update control recipe list** button.
Apply column settings from calling screen	*Active*: The column settings are applied in the screen that is calling them up in the Runtime. The corresponding properties can no longer be configured in the Editor.
List field	Display of the configured columns.
Column selection	Opens the dialog for selecting the columns.
Column format	Opens a dialog (on page 195) to format the columns.
Filter control recipe for state	Settings for the filtering of the control recipes according to their status.
Apply filter from calling screen	*Active*: The filter is accepted from the calling screen.
Use this filter	Selection of criteria for the status of a recipe that is to be called up. *Active*: Several statuses can be selected by selecting the corresponding checkbox: ▸ *Prepared* ▸ *Currently executed* ▸ *Executed* ▸ *Outdated*

GENERAL OPTIONS

Recipe names are case sensitive	*Active*: General options Default: *inactive*
Show this dialog in the	*Active*: When calling up the function in the Runtime, this dialog

Recipe names are case sensitive	*Active*: General options Default: *inactive*
Runtime	is opened and the user can amend the configuration before execution. The dialog is displayed on the current computer in the Runtime. During network operation when activating the client the dialog is also displayed on the client

CLOSE DIALOG

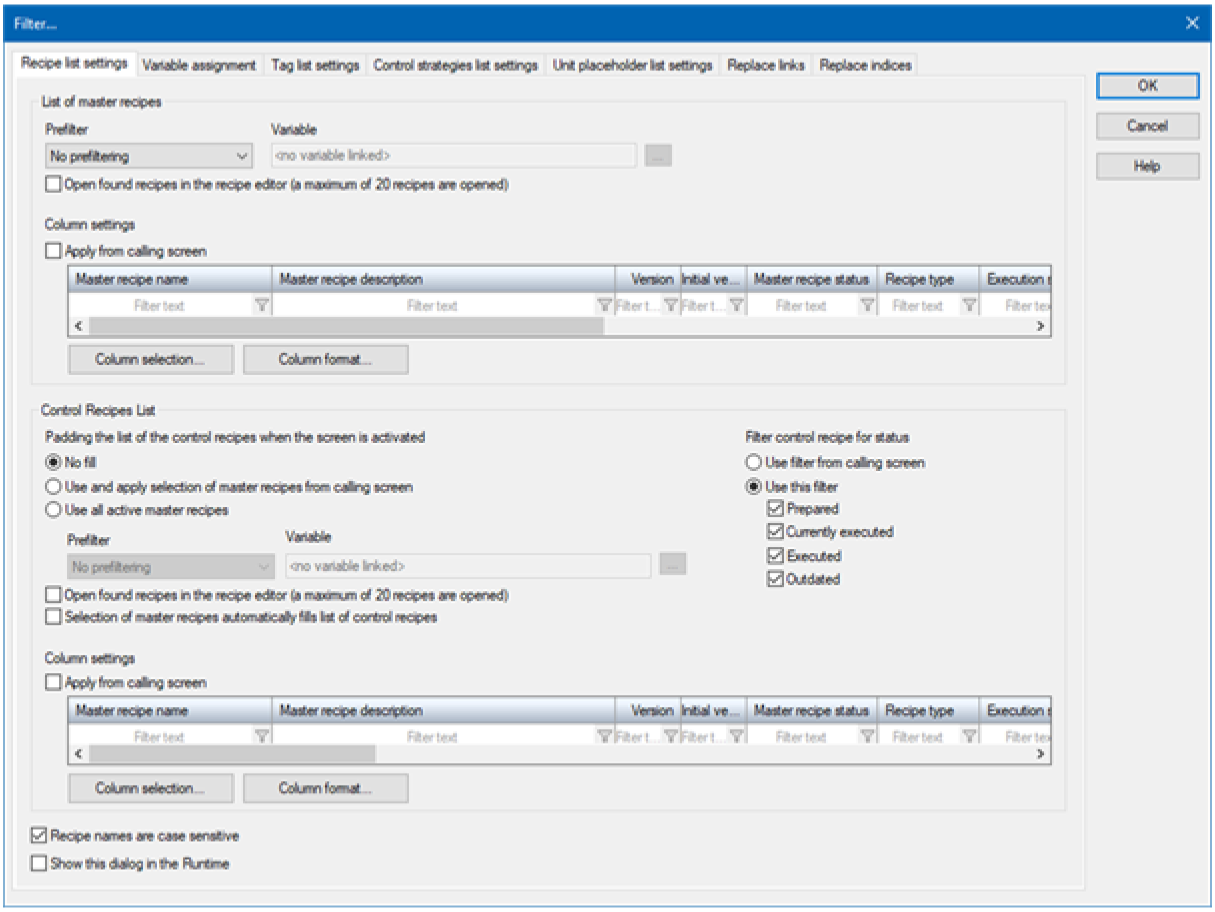
Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

Note for variable selection using name or ID: For the selection of variables according to name or ID, numerical variables and string variables can be selected respectively. The data types are converted to the respective correct form.

6.5.1.1 Prefilter

To eliminate the need for all recipes to always be loaded in the recipe list in Runtime, you can define filters for master recipes and control recipes in the screen switch function. Then, the only recipes that appear in the list of master recipes and the control recipes are those that correspond to the configured filter conditions. If activated, these recipes are also opened in the recipe editor.

Note: Screen switching in the Runtime can take some time due to the large amounts of data required for the recipes to be loaded. The progress is shown in a progress bar.



List of master recipes

Prefilter: No prefiltering

Variable: <no variable linked>

□ Open found recipes in the recipe editor (a maximum of 20 recipes are opened)

Column settings

□ Apply from calling screen

Control Recipes List

Padding the list of the control recipes when the screen is activated

● No fill

○ Use and apply selection of master recipes from calling screen

○ Use all active master recipes

Prefilter: No prefiltering

Variable: <no variable linked>

□ Open found recipes in the recipe editor (a maximum of 20 recipes are opened)

□ Selection of master recipes automatically fills list of control recipes

Filter control recipe for status

○ Use filter from calling screen

● Use this filter

☑ Prepared

☑ Currently executed

☑ Executed

☑ Outdated

Column settings

□ Apply from calling screen

☑ Recipe names are case sensitive

□ Show this dialog in the Runtime

If you want it to be impossible for users to remove the filters in the runtime environment:

1. Deactivate the **Show this dialog in Runtime** option.
2. Block the column filter: Open the **Column format...** (on page 119) dialog and activate the **Lock column filter in the Runtime** option. As a result of this, the user cannot modify the filter in Runtime and therefore they do not get the recipes that they cannot modify displayed.

FILTERING FOR MASTER RECIPES

Configure:

1. **Prefiltering**

   Stipulate if recipes are to be prefiltered. You can filter according to name or ID. The filter condition is queried in the Runtime using a variable.

2. **Column settings**

   a) In the screen switch function, click on a column filter for the **list of the master recipes** or the **list of control recipes**.

b) Enter the desired filter text; wildcards in the form of asterisks (*) are permitted.

c) Confirm the filter text with the **Enter key** in order for it to be applied.

3. **Automatic recipe switching**

   Stipulate if the recipes found when switching are also to be opened in the Recipe Editor straight away.

   Note: The first 20 recipes found is the maximum that can be opened automatically.

> **Information**
>
> When reloading Runtime, the filter settings of the prefiltering for master recipes are applied again. This also applies if the value of the filter variables changes or new recipes are added to the list that do not correspond to the filter. The list is always recreated exactly after reloading.

FILTERING FOR CONTROL RECIPES

Configure:

1. **Recipe status**

   Filter the control recipes according to their status. You can select several states using checkboxes or accept the filter from the calling screen.

2. **Filling the recipe list**

   Stipulate if and how the list of control recipes is to be filled when called up.

3. **Prefiltering**

   Stipulate if recipes are to be prefiltered. You can filter according to ID, name or job ID. The filter condition is queried in the Runtime using a variable

4. **Column settings**

   a) In the screen switch function, click on a column filter for the **list of the master recipes** or the **list of control recipes**.

   b) Enter the desired filter text; wildcards in the form of asterisks (*) are permitted.

   c) Confirm the filter text with the **Enter key** in order for it to be applied.

5. **Automatic recipe switching**

   Stipulate if the recipes found when switching are also to be opened in the Recipe Editor straight away.

   Note: The first 20 recipes found is the maximum that can be opened automatically.

Information

When reloading Runtime, the filter settings of the prefiltering for control recipes are applied again. Instead, all control recipes that correspond to the current filter in the Runtime are displayed (master recipes, status, column filter).

6.5.2 Variable assignment

You link variables to elements in the recipe in this tab. This way you can display the statuses of a phase or an operation in another screen and react to these. The execution status of the recipe and the selected object can be displayed by means of string variables or numerical variables. Numerical variables are suitable, for instance, for linking to a combined element. For details on the status, see the Coding of the execution status (on page 95) section.

To display statuses:

1. Create a new screen with the desired elements.
2. Link the elements to variables
3. Link these variables in the screen switching filter to the corresponding objects

LINK VARIABLES

To link a variable:

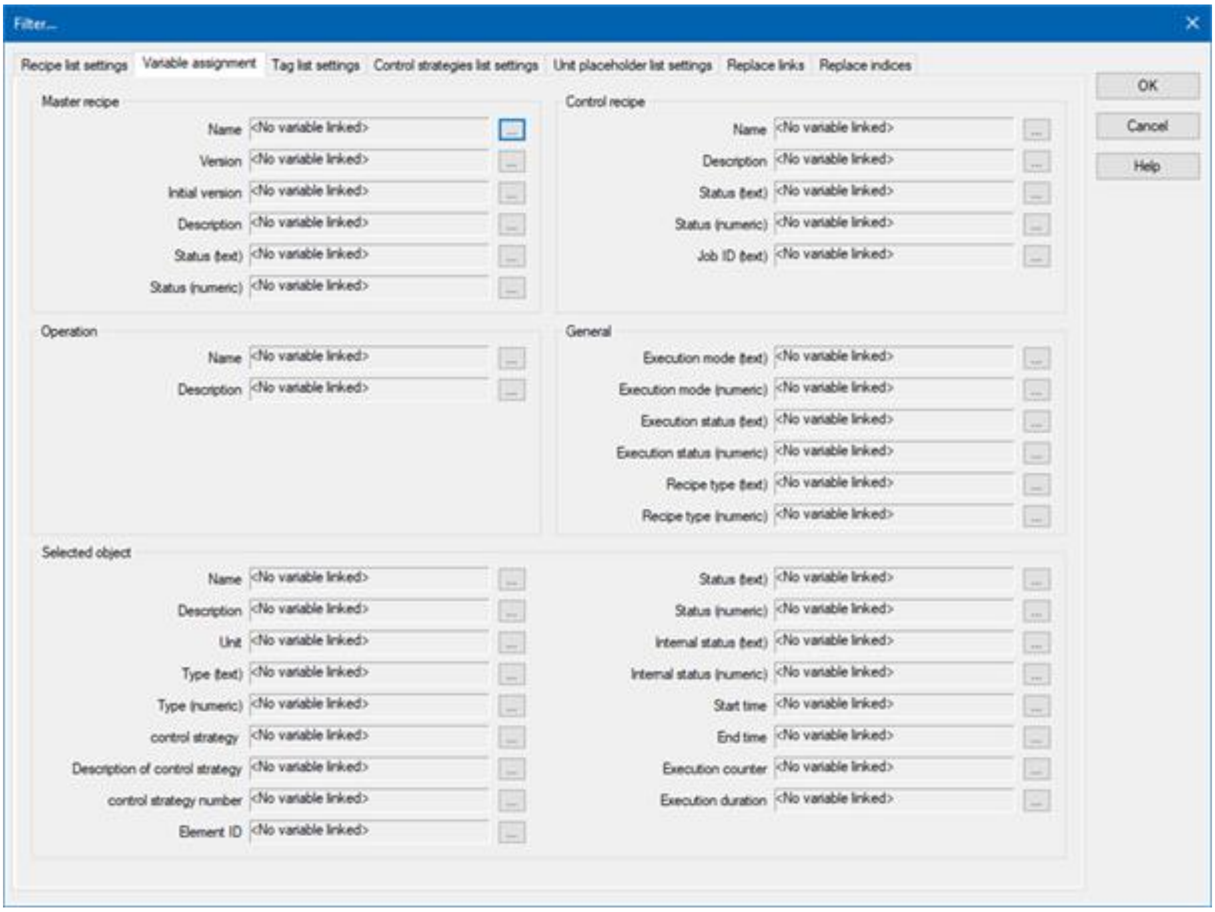
1. Click on the **...** button
2. The dialog for selecting a variable is opened
3. Select the desired variable

DISPLAY

The following are displayed:

- For the recipe that is in focus:
  - Name
  - Description
  - Status
  - Execution mode
  - Execution status
  - Recipe type

- For the object in the recipe that is in focus (phase or operation):
  - Name
  - Description
  - Unit
  - Type
  - Status
  - Internal state
  - Start time
  - Time of the end
  - Execution counter
  - Duration of execution



Master recipe

Name <No variable linked>
Version <No variable linked>
Initial version <No variable linked>
Description <No variable linked>
Status (text) <No variable linked>
Status (numeric) <No variable linked>

Control recipe

Name <No variable linked>
Description <No variable linked>
Status (text) <No variable linked>
Status (numeric) <No variable linked>
Job ID (text) <No variable linked>

Operation

Name <No variable linked>
Description <No variable linked>

General

Execution mode (text) <No variable linked>
Execution mode (numeric) <No variable linked>
Execution status (text) <No variable linked>
Execution status (numeric) <No variable linked>
Recipe type (text) <No variable linked>
Recipe type (numeric) <No variable linked>

Selected object

Name <No variable linked>
Description <No variable linked>
Unit <No variable linked>
Type (text) <No variable linked>
Type (numeric) <No variable linked>
control strategy <No variable linked>
Description of control strategy <No variable linked>
control strategy number <No variable linked>
Element ID <No variable linked>

Status (text) <No variable linked>
Status (numeric) <No variable linked>
Internal status (text) <No variable linked>
Internal status (numeric) <No variable linked>
Start time <No variable linked>
End time <No variable linked>
Execution counter <No variable linked>
Execution duration <No variable linked>

Parameter	Description
Master recipe	Variable linkings for the master recipe (on page 96). Status is displayed if the recipe has the focus or a phase or

Parameter	Description
	operation is highlighted. The following can be linked: - Name - Version - Initial version - Description - Status (text) - Status (numeric)
Control recipe	Variable linkings for the control recipe. (on page 97) Status is displayed if the recipe has the focus or a phase or operation is highlighted. The following can be linked: - Name - Description - Status (text) - Status (numeric) - Job ID (text)
Operation	Variable linkings for the operation. (on page 98) Status is displayed if the recipe has the focus or a phase or operation is highlighted. The following can be linked: - Name - Description
General	Variable linkings for general information. (on page 98) Status is displayed if the recipe has the focus or a phase or operation is highlighted. The following can be linked: - Execution mode (text) - Execution mode (numerical) - Execution status (text)

Parameter	Description
	- Execution status (numerical) - Recipe type (text) - Recipe type (numeric)
Selected Object	Variable linkings for the selected object (phase or operation) (on page 100). Status is displayed if a phase or an operation in the recipe is highlighted. The following can be linked: - Name - Description - Unit - Type (text) - Type (numeric) - Control strategy - Description of control strategy - control strategy number - Status (text) - Status (numeric) - Internal state (text) - Internal state (numeric) - Start time - Time of the end - Execution counter - Duration of execution

CLOSE DIALOG

Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

Note: No information is displayed if several objects are highlighted.

6.5.2.1 Coding of the execution status

EXECUTION STATE

The execution state (both for the recipe and the selected object) is coded with different information:

- Byte *0*: Execution state
- Byte *1*: Status bits for the status
- Byte *2*: Type of object that is part of the status

If no recipe is opened then the string variables are empty and the numerical variables have the value *0*.

BYTE 0

Execution states.
The values that are possible are determined by the object type.

String	Decimal	Binary
Idle	*1*	*1*
In execution	*2*	*10*
Executed	*3*	*11*
Pause (starting from state: running)	*4*	*100*
Paused	*5*	*101*
Hold (starting from state: running)	*6*	*110*
Hold (starting from state: Paused)	*7*	*111*
Hold (starting from state: Restart)	*8*	*1000*
Held	*9*	*1001*
Restarting (starting from state: Held)	*10*	*1010*
Stopping (starting from state: running)	*11*	*1011*
Stopping (starting from state: Paused)	*12*	*1100*
Stopping (starting from state: Held)	*13*	*1101*

String	Decimal	Binary
Stopped	*14*	*1110*
Aborting (starting from state: running)	*15*	*1111*
Aborting (starting from state: Paused)	*16*	*10000*
Aborting (starting from state: Held)	*17*	*10001*
Aborted	*18*	*10010*

BYTE 1

Status bits for the status.
The text is added to the string variable. The corresponding bits are set in the numerical variables

Value for string variable	Value for numeric variable
n elements in different states	*0b00000001*
Waiting for n element(s)	*0b00000010*

BYTE 2

Type of object that is part of the status.
For numerical variables only.

Decimal	Binary	Meaning
1	*1*	Recipe
2	*10*	Phase
3	*11*	Operation object in the recipe

6.5.2.1.1 Master recipe

NAME

Name of the currently-opened master recipe or the master recipe that belongs to the currently-opened control recipe or operation.

DESCRIPTION

Description of the currently-opened master recipe or the master recipe that belongs to the currently-opened control recipe or operation.

STATUS (TEXT AND NUMERIC)

Status of the currently-opened master recipe or the master recipe that belongs to the currently-opened control recipe or operation.

Variable values:

String	Decimal	Binary
No recipe in active view	*0*	*0*
Creation (not visible)	*1*	*1*
Edit mode	*2*	*10*
Released	*3*	*11*
Test mode	*4*	*100*
Test in execution	*5*	*101*
Terminated with error	*6*	*110*
Outdated	*9*	*1001*

6.5.2.1.2Control recipe

NAME

Name or selection of the control recipe. The recipe must be open and have the focus. Is filled in the control recipe and in the operation instance in the control recipe.

DESCRIPTION

Description of the currently-selected control recipe. The recipe must be open and have the focus. Is filled in the control recipe and in the operation instance in the control recipe.

STATUS (TEXT AND NUMERIC)

Status of the currently-selected control recipe. The recipe must be open and have the focus. Is filled in the control recipe and in the operation instance in the control recipe.

Variable values:

String	Decimal	Binary
no control recipe active	*0*	*0*
Creation (not visible)	*1*	*1*
Prepared	*2*	*10*
In execution	*3*	*11*
Executed	*4*	*100*
Terminated with error	*5*	*101*
Outdated	*7*	*111*

6.5.2.1.3Operation

NAME

Name of the operation that is currently open. Is filled in the operation template and in the operation instance.

DESCRIPTION

Description of the operation currently open. Is filled in the operation template and in the operation instance.

6.5.2.1.4General

Attention: The content of these variables is not compatible between zenon 7.10 and 7.11.

EXECUTION MODE (TEXT AND NUMERIC)

Currently-set execution mode for the currently-selected recipe. Is filled for all recipe cycles.

Variable values:

String	Decimal	Binary
Automatic	*1*	*01*
Semi-automatic	*2*	*10*
Manual	*3*	*11*

EXECUTION STATUS (TEXT AND NUMERIC)

Status of the recipe

BYTE 0

Variable values:

String	Decimal	Binary
Idle	*1*	*1*
In execution	*2*	*10*
Executed	*3*	*11*
Pausing (starting from state: *running)*	*4*	*100*
Paused	*5*	*101*
Holding (starting from state: *running)*	*6*	*110*
Held	*9*	*1001*
Restarting (starting from state: *Held*)	*10*	*1010*
Stopping (starting from state: *running)*	*11*	*1011*
Stopped	*14*	*1110*
Aborting (starting from state: *running)*	*15*	*1111*
Aborted	*18*	*10010*

BYTE 1

Value for string variable	Value for numeric variable
n elements in different states	*0b00000001*
Waiting for n element(s)	*0b00000010*

BYTE 2

Object type. For the numerical value only

Decimal	Binary	Meaning
1	1	Recipe

RECIPE TYPE (TEXT AND NUMERIC)

Recipe type of the current selected recipe. Is filled for all recipe cycles.

Variable values:

String	Decimal	Binary
Master recipe	*1*	*1*
Control recipe	*2*	*10*
Operation template	*4*	*100*
Operation instance in the master recipe	*9*	*1001*
Operation instance in the control recipe	*10*	*1010*

6.5.2.1.5Selected Object

The variables for the selected object always contain data if a phase or an operation has been selected in the currently-selected operation. It is filled for all recipe types. If no object or several objects are selected, then the string variables are empty and the numerical values are *0*.

NAME

Is filled with the name of the phase or the operation.

DESCRIPTION

Is filled with the description of the phase of the operation.

UNIT

Is filled with the name of the unit of the selected phase.

TYPE (TEXT AND NUMERIC)

Variable values:

String	Decimal	Binary
Phase	*3*	*11*
Operation	*13*	*1101*

STATUS (TEXT AND NUMERIC)

Is filled with the current execution status of the element.

VALUE FOR A PHASE

BYTE 0

Variable values:

String	Decimal	Binary
Idle	*1*	*1*
In execution	*2*	*10*
Executed	*3*	*11*
Pausing (Starting from state: *running)*	*4*	*100*
Paused	*5*	*101*
Holding (Starting from state: *running)*	*6*	*110*
Holding	*7*	*111*

String	Decimal	Binary
(Starting from state: *Paused*)		
Holding (Starting from state: *Restarting*)	*8*	*1000*
Held	*9*	*1001*
Restarting (Starting from state: *Held*)	*10*	*1010*
Stopping (Starting from state: *running)*	*11*	*1011*
Stopping (Starting from state: *Paused*)	*12*	*1100*
Stopping (Starting from state: *Held*)	*13*	*1101*
Stopped	*14*	*1110*
Aborting (Starting from state: *running)*	*15*	*1111*
Aborting (Starting from state: *Paused*)	*16*	*10000*
Aborting (Starting from state: *Held*)	*17*	*10001*
Aborted	*18*	*10010*

BYTE 1:

Always empty.

BYTE 2:

Object type. For the numerical value only

Variable values:

Meaning	Decimal	Binary
Phase	*2*	*10*

VALUE FOR AN OPERATION INSTANCE

BYTE 0

Variable values:

String	Decimal	Binary
Idle	*1*	*1*
In execution	*2*	*10*
Executed	*3*	*11*
Pausing (Starting from state: *running)*	*4*	*100*
Paused	*5*	*101*
Holding (Starting from state: *running)*	*6*	*110*
Held	*9*	*1001*
Restarting (Starting from state: *Held*)	*10*	*1010*
Stopping (Starting from state: *running)*	*11*	*1011*
Stopping (Starting from state: *Paused*)	*12*	*1100*
Stopping (Starting from state: *Held*)	*13*	*1101*
Stopped	*14*	*1110*
Aborting (Starting from state: *running)*	*15*	*1111*

String	Decimal	Binary
Aborting (Starting from state: *Paused*)	*16*	*10000*
Aborting (Starting from state: *Held*)	*17*	*10001*
Aborted	*18*	*10010*

BYTE 1

Value for string variable	Value for numeric variable
n elements in different states	*0b00000001*

The bit is always set if there are objects in the operation with a different status to that of the operation. *Idle* and *finished* are not included in this.

BYTE 2:

Object type. For the numerical value only.

Variable values:

Meaning	Decimal	Binary
Operation in the recipe.	*3*	*11*

INTERNAL STATUS (TEXT AND NUMERIC)

Is filled with the internal execution status of the selected element.

Composition of the internal status:

- Byte *0*: Status
- Byte *1*: Status Bits
- Byte *2*: Object type
- Byte *3*: PLC error status
  - Bit *0*: PLC error occurred
  - Bit *1*: PLC error rectified

VALUE FOR A PHASE

BYTE 0

Internal status. The text can be different for other objects.

VARIABLE VALUES:

String	Decimal	Binary
Idle	*1*	*1*
Waiting for phase to be ready for starting	*2*	*10*
Waiting for unit allocation	*3*	*11*
Waiting for unit allocation - timeout	*4*	*100*
Waiting for exclusive execution	*5*	*101*
Waiting for input lock	*6*	*110*
Waiting for the input lock - timeout	*7*	*111*
Waiting for phase done condition	*8*	*1000*
Waiting for phase done condition - timeout	*9*	*1001*
Waiting for phase done condition - error writing value	*10*	*1010*
Waiting for minimum execution duration	*11*	*1011*
Wait until recipe has status "running"	*12*	*1100*
Waiting for following conditions	*13*	*1101*
Waiting for following condition - timeout	*14*	*1110*
Writing the value TAGs	*18*	*10010*
Writing the initial TAGs	*19*	*10011*
Placeholder not linked to instance	*23*	*10111*
Validation error	*24*	*11000*

String	Decimal	Binary
Timeout for instance allocation	*25*	*11001*
Renewed writing of the initial TAGs	*26*	*11010*
Renewed writing the value TAGs	*27*	*11011*
Waiting for confirmation of the changes made before writing the initial TAGs	*28*	*11100*
Waiting for confirmation of the changes made before writing the initial TAGs - time out	*29*	*11101*
Waiting for confirmation of the changes made before writing the value TAGs	*30*	*11110*
Waiting for confirmation of the changes made before writing the value parameters - timeout	*31*	*11111*

BYTE 1

Status. For numerical variables only.

Variable values:

Meaning	Decimal	Binary
There is an execution error.	*1*	*1*
There is a communication error.	*8*	*1000*
Communication error due to failed write attempt	*9*	*1001*
Loss of communication fixed.	*16*	*10000*
Communication reestablished.	*64*	*1000000*
Communication reestablished after failed write attempt	*65*	*1000001*

With the phase, only one of the bits can be active for the communication error. With an operation a bit is always set if it is relevant to at least one internal phase.

BYTE 2

Object type. For the numerical value only.

Variable values:

Meaning	Decimal	Binary
Operation in the recipe.	*3*	*11*

POSSIBILITIES FOR PHASE

BYTE 0

All possibilities.

BYTE 1

All possibilities.

BYTE 2

Meaning	Value for numeric variable: Decimal	Value for numeric variable: Binary
Phase	*2*	*10*

POSSIBILITIES FOR OPERATION INSTANCES

BYTE 0

Variable values:

String	Decimal	Binary
Idle	*1*	*1*
Execution of the internal objects	*8*	*1000*
Waiting for following conditions	*13*	*1101*

BYTE 1

Variable values:

Meaning	Decimal	Binary
Execution of the internal objects. Errors are displayed.	1	1

BYTE 2

Variable values:

Meaning	Decimal	Binary
Operation in the recipe.	3	11

START TIME

Time at which the execution of the selected object has started.

END TIME

Time at which the execution of the selected object has finished.

DURATION OF EXECUTION

Time period that has expired during the execution.

EXECUTION COUNTER

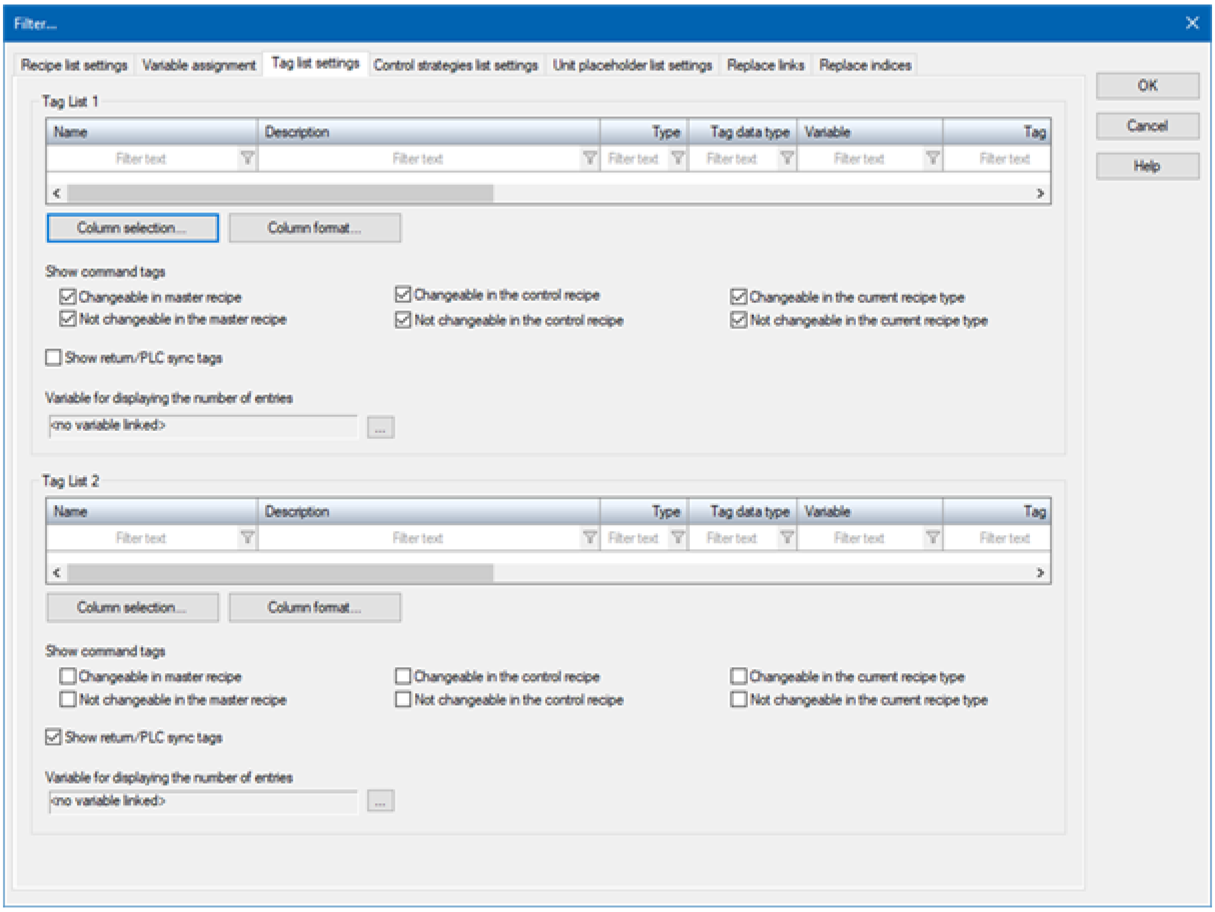
Number denoting how often the element was executed

6.5.3 Tag list settings

Two parameter lists with identical options are available. These can be configured and used individually.

The parameter lists are created in the screen and configured in screen switching.

If you want to edit the list directly using the monitor, activate the Multi-Touch functionality. You can find detailed information in relation to this in the Configure interactions chapter.



Filter...

Recipe list settings | Variable assignment | Tag list settings | Control strategies list settings | Unit placeholder list settings | Replace links | Replace indices

OK | Cancel | Help

Tag List 1

Name	Description	Type	Tag data type	Variable	Tag
Filter text	Filter text	Filter text	Filter text	Filter text	Filter text

Column selection... | Column format...

Show command tags

☑ Changeable in master recipe
☑ Not changeable in the master recipe
☑ Changeable in the control recipe
☑ Not changeable in the control recipe
☑ Changeable in the current recipe type
☑ Not changeable in the current recipe type

□ Show return/PLC sync tags

Variable for displaying the number of entries: <no variable linked>

Tag List 2

Name	Description	Type	Tag data type	Variable	Tag
Filter text	Filter text	Filter text	Filter text	Filter text	Filter text

Column selection... | Column format...

Show command tags

□ Changeable in master recipe
□ Not changeable in the master recipe
□ Changeable in the control recipe
□ Not changeable in the control recipe
□ Changeable in the current recipe type
□ Not changeable in the current recipe type

☑ Show return/PLC sync tags

Variable for displaying the number of entries: <no variable linked>

TAG LIST 1

Parameter	Description
TAG List 1	Display of the configured lists. The columns can: - be filtered - have their width changed by dragging the column title with the mouse - be arranged by means of Drag&Drop with the mouse **Note:** The filtering and configuring in the Runtime can also be prevented for lists by means of the properties of the **Header** group. **Attention:** If a filter is created in the Editor and the display of the filter line is then prevented in the Runtime, the filter is applied but can no longer be configured.

Parameter	Description
Column selection	A dialog for choosing the columns (on page 115) which shall be displayed is opened
Column format	Opens a dialog to format the columns (on page 119).
Display command TAGs	Selection of the command parameters which should be displayed. Possible selection by activating the checkboxes: ▸ **Parameter value can be changed in master recipe**: Command tags for which the **Parameter value can be changed in master recipe** option has been set in the Editor are displayed. ▸ **Changeable in control recipe**: Command tags for which the **Changeable in control recipe** option has been set in the master recipe are displayed. ▸ **Changeable in the current recipe type**: Command tags that can be changed in the current recipe type are displayed. ▸ **Cannot be changed in the master recipe**: Command tags for which the **Parameter value can be changed in master recipe** option has not been set in the Editor are displayed. ▸ **Not changeable in the control recipe** Command tags for which the **Changeable in control recipe** option has not been set in the master recipe are displayed. ▸ **Cannot be changed in the current recipe type**: Command tags that cannot be changed in the current recipe type are displayed. As many connections as desired can be configured. Activating all checkboxes leads to all command parameters being displayed.
Show return/PLC sync TAG	*Active*: Return / PLC tags are displayed.
Variable for displaying the number of entries	Allows the linking of a variable that contains the number of parameters displayed in the list in Runtime. Click on the **...** Button to open the dialog to select a variable.

TAG LIST 2

Parameter	Description
TAG List 2	Parameter list 2 with identical configuration possibilities to **Parameter Tag 1**.

Longer texts can also be displayed in the Runtime over several lines using the **Automatic word wrap** property.

In the Editor, go to **Representation** in the properties of the respective list properties and activate the checkbox of the **Automatic word wrap** property.

The line height must be amended manually.

CLOSE DIALOG

Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

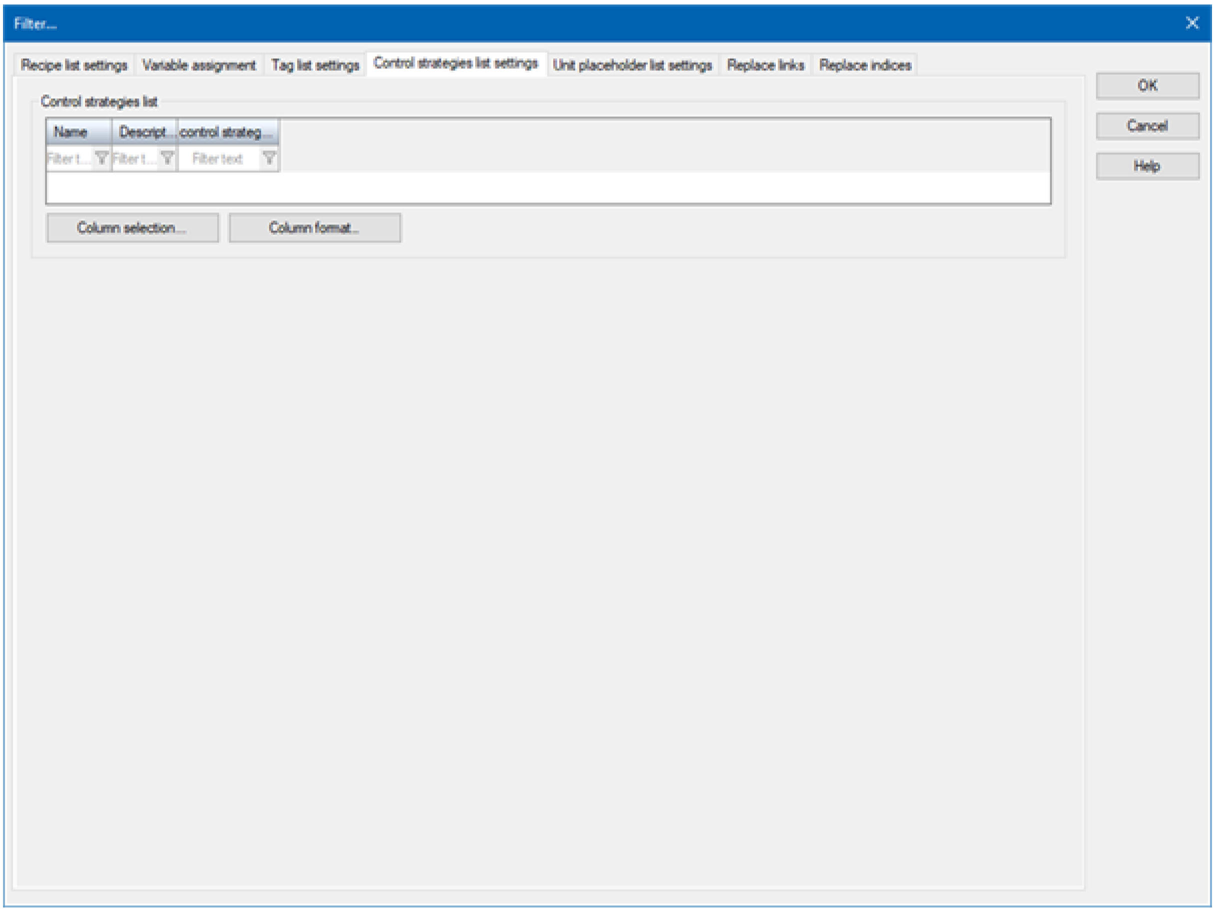
The parameter lists are recreated in the Runtime if:

- A different element is selected in the recipe
- The selected phase was edited
- The recipe was saved (even if the client has saved it)

> ## Information
>
> Backward compatibility for **displaying command parameters**:
>
> The option was expanded with version 7.11 and changed from radio buttons to checkboxes. For backward compatibility, this means:
>
> Backward compatible writing is possible if:
>
> Only one checkbox is set for the changeable parameters
>
> or the combination of all checkboxes lets all parameters through
>
> If the combination of the checkboxes results in a setting that was not previously configurable with zenon 7.10 or earlier, no parameters are displayed in the list.

6.5.4 Control strategies list settings

In this tab, you define how the list of control strategies is displayed in Runtime. You can also pre-define filters.



Option	Description
List of control strategies	Configuration of how the list is displayed in Runtime. Select the desired columns and define a filter if required. The following can be done with the columns in Runtime: ▶ be filtered ▶ have their width changed by dragging the column title with the mouse ▶ be arranged by means of Drag&Drop with the mouse **Note:** The filtering and configuring in the Runtime can also be prevented for lists by means of the properties of the **Header** group. **Attention:** If a filter is created in the Editor and the

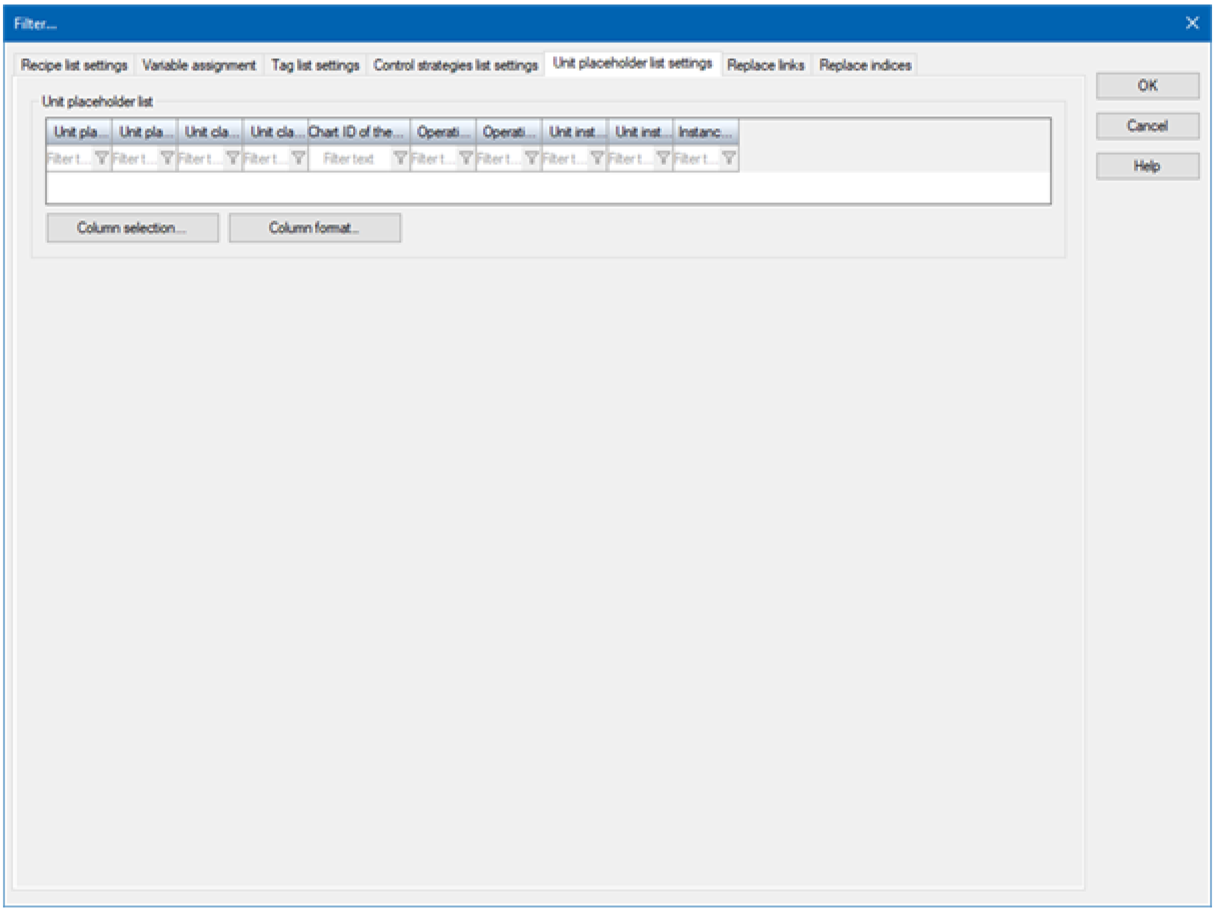
Option	Description
	display of the filter line is then prevented in the Runtime, the filter is applied but can no longer be configured.
Column selection	A dialog for choosing the columns (on page 115) which shall be displayed is opened
Column format	Opens the dialog to format the columns (on page 119).

CLOSE DIALOG

Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

6.5.5 Settings for the unit placeholder list

In this tab, you define how the list of unit placeholders is displayed in the Runtime. You can also pre-define filters.



Option	Description
Unit placeholder list	Configuration of how the list is displayed in the Runtime. Select the desired columns and define a filter if required. The following can be done with the columns in the Runtime: - be filtered - have their width changed by dragging the column title with the mouse - be arranged by means of Drag&Drop with the mouse **Note:** The filtering and configuring in the Runtime can also be prevented for lists by means of the properties of the **Header** group.

Option	Description
	Attention: If a filter is created in the Editor and the display of the filter line is then prevented in the Runtime, the filter is applied but can no longer be configured.
Column selection	A dialog for choosing the columns (on page 115) which shall be displayed is opened
Column format	Opens the dialog to format the columns (on page 119).

CLOSE DIALOG

Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

6.5.6 Column selection

Column selection is available for all configurable lists. Configuration is carried out by means of a dialog that offers the same respective options. The columns available depend on the list to be configured. Examples of concrete lists then follow the documentation of the options.

OPTIONS

Option	Function
Available columns	List of columns that can be displayed in the table.
Selected columns	Columns that are displayed in the table.
Add ->	Moves the selected column from the available ones to the selected items. After you confirm the dialog with OK, they are shown in the detail view.
Add all ->	Moves all available columns to the selected columns.
<- Remove	Removes the marked columns from the selected items and shows them in the list of available columns. After you confirm the dialog with OK, they are removed from the detail view.

Option	Function
<- Remove all	All columns are removed from the list of the selected columns.
Up	Moves the selected entry upward. This function is only available for unique entries, multiple selection is not possible.
Down	Moves the selected entry downward. This function is only available for unique entries, multiple selection is not possible.

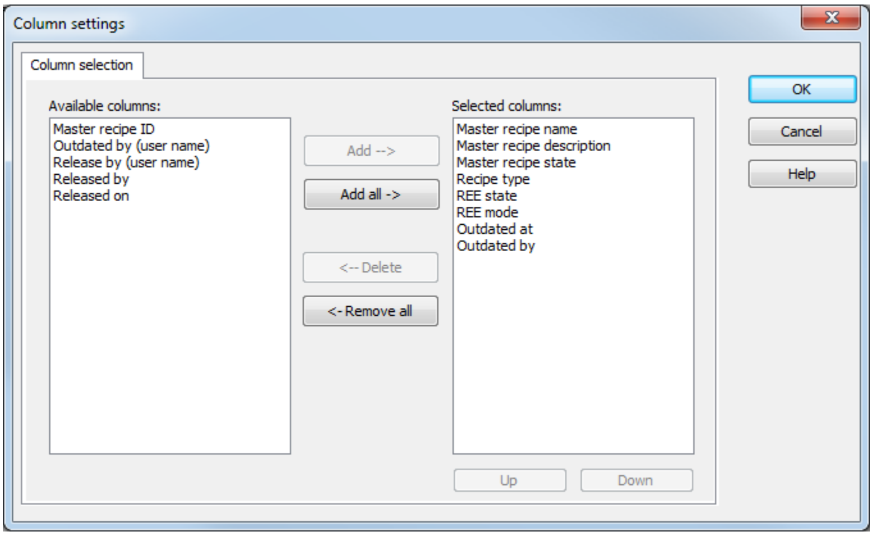
CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

Note: These settings are only used in the Runtime for dockable windows (on page 151) if there is no Runtime profile available for the user who is logged in.

EXAMPLES OF COLUMN SELECTION

Column selection for list of the master recipes (on page 161):



Column settings

Column selection

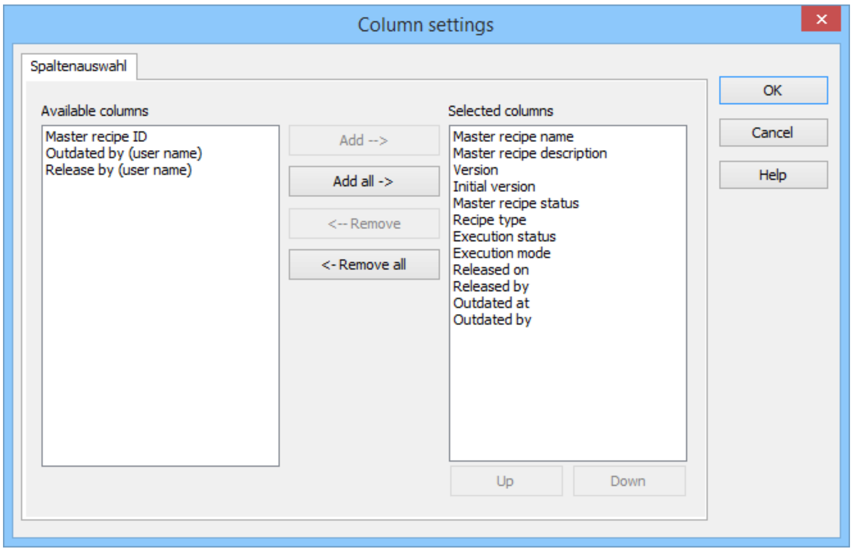
Available columns:

Master recipe ID
Outdated by (user name)
Release by (user name)
Released by
Released on

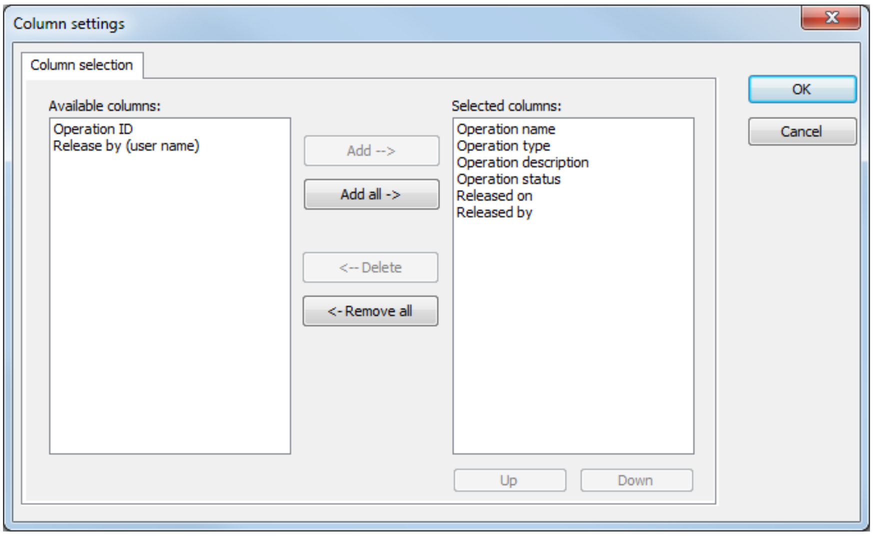
Selected columns:

Master recipe name
Master recipe description
Master recipe state
Recipe type
REE state
REE mode
Outdated at
Outdated by

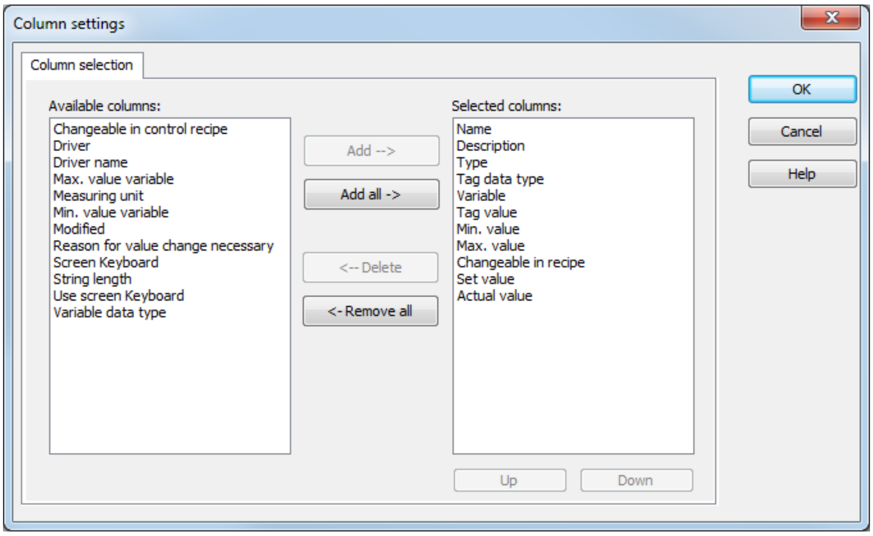
Column selection for list of the control recipes (on page 232):



Column selection for list of the partial recipes (on page 220) (only available in the Runtime):



Column selection for Tag lists (on page 108)



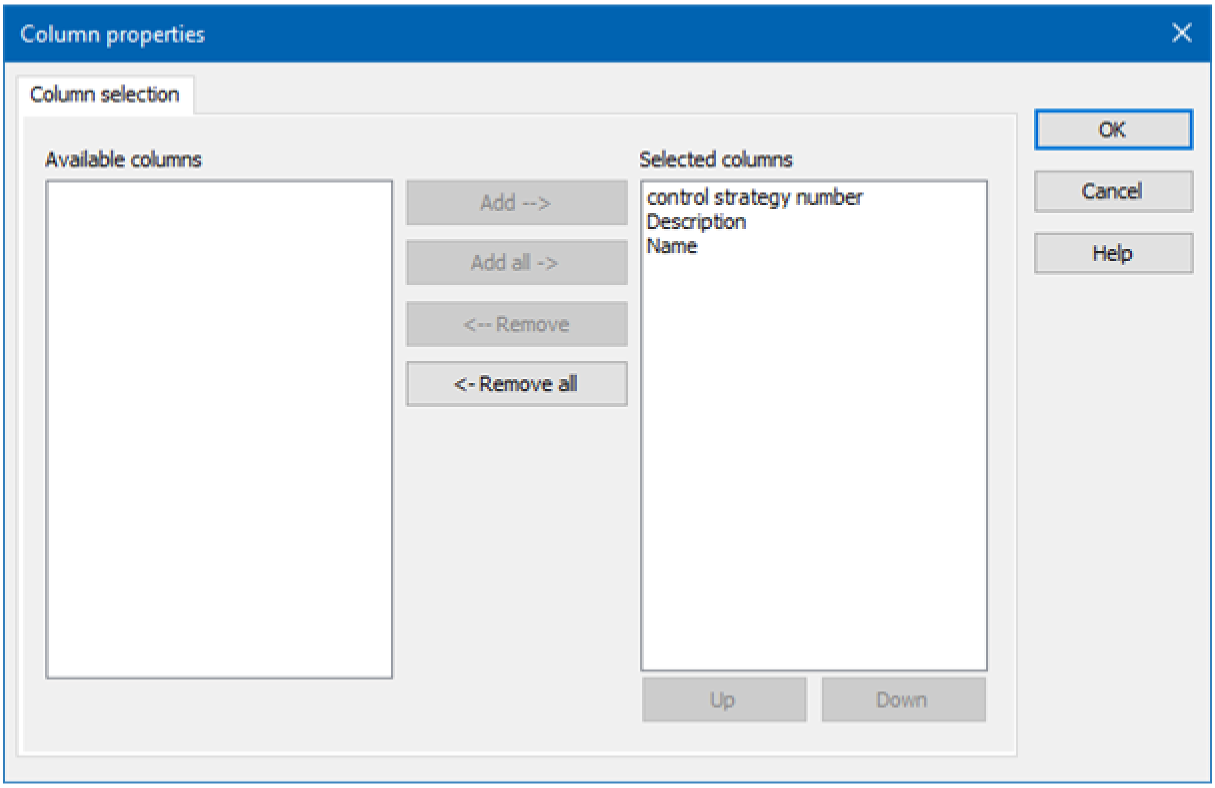
Available columns:

Changeable in control recipe
Driver
Driver name
Max. value variable
Measuring unit
Min. value variable
Modified
Reason for value change necessary
Screen Keyboard
String length
Use screen Keyboard
Variable data type

Selected columns:

Name
Description
Type
Tag data type
Variable
Tag value
Min. value
Max. value
Changeable in recipe
Set value
Actual value

Column selection for control strategies (on page 112):

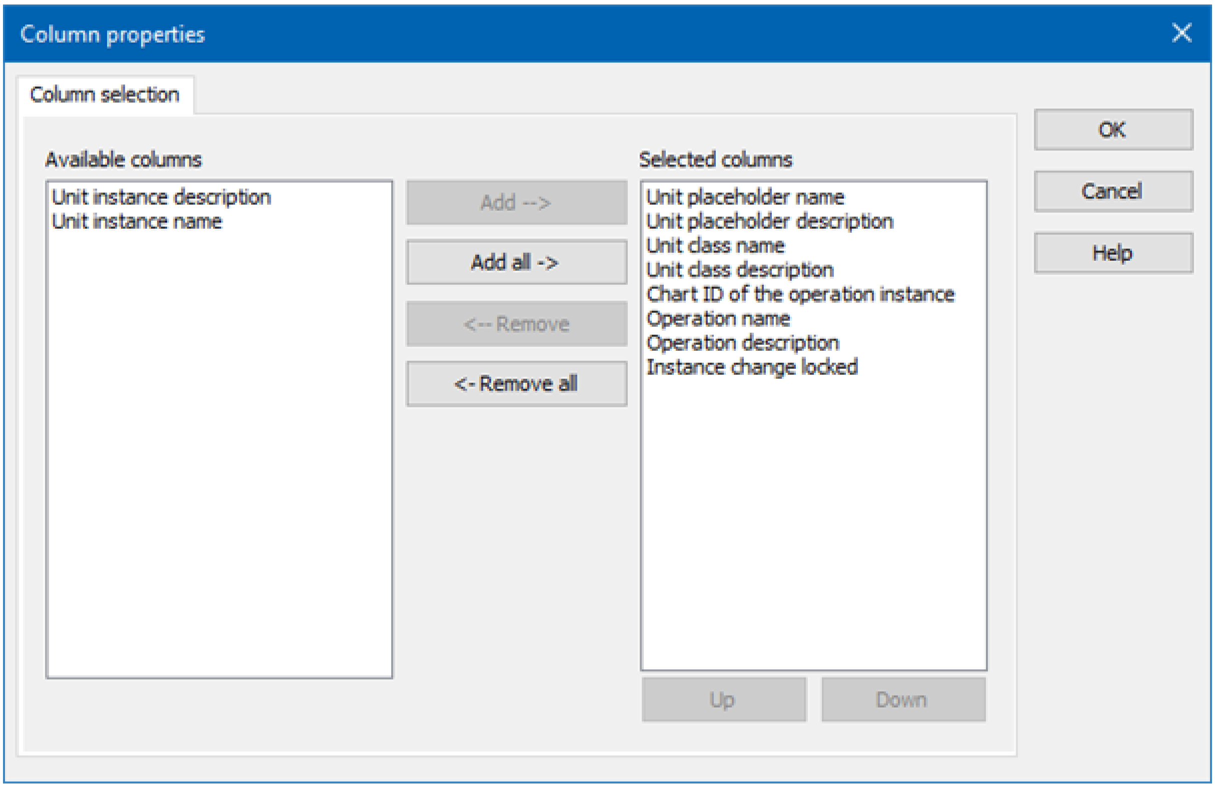


Available columns

Selected columns

control strategy number
Description
Name

Column selection for unit placeholder list (on page 114):



Available columns

Unit instance description
Unit instance name

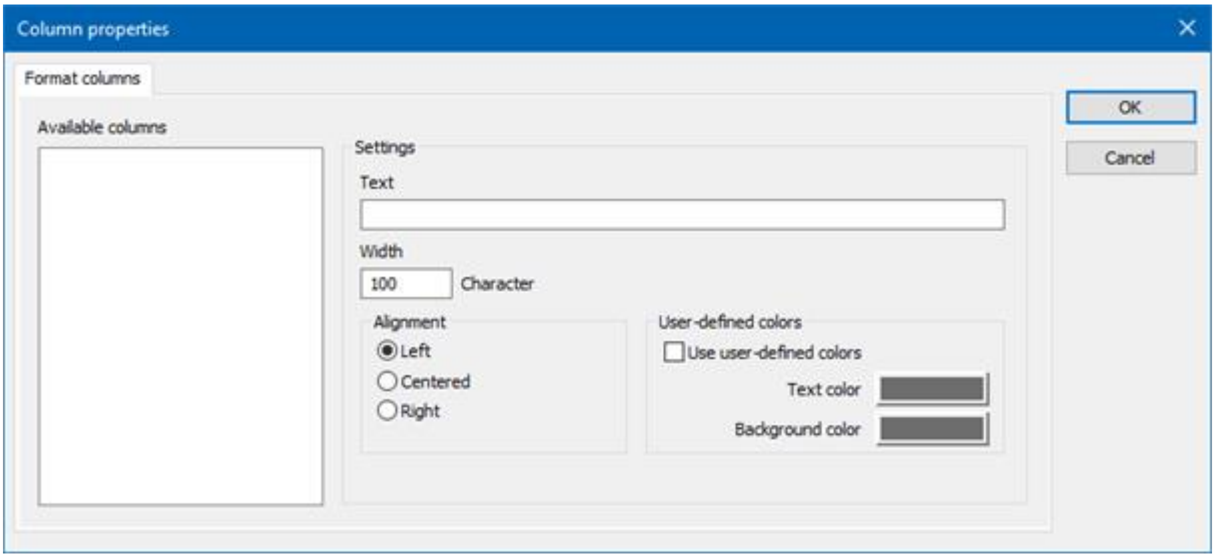
Selected columns

Unit placeholder name
Unit placeholder description
Unit class name
Unit class description
Chart ID of the operation instance
Operation name
Operation description
Instance change locked

6.5.7 Column Format

In this dialog you define the column format:

Configuration of the properties of the columns for configurable lists. The settings have an effect on the respective list in the Editor or - when configuring screen switching - in Runtime.



Text

Width

100 Character

Alignment

● Left
○ Centered
○ Right

User-defined colors

☐ Use user-defined colors

AVAILABLE COLUMNS

Option	Description
Available columns	List of the available columns via **Column selection**. The highlighted column is configured via the options in the **Settings** area.

SETTINGS

Option	Description
Settings	Settings for selected column.
Labeling	Name for column title. The column title is online language switchable. To do this, the **@** character must be entered in front of the name.
Width	Width of the column in characters. Calculation: Number time average character width of the selected font.
Alignment	Alignment. Selection by means of radio buttons. Possible settings: - **Left**: Text is justified on the left edge of the column. - **Centered**: Text is displayed centered in the column. - **Right**: Text is justified on the right edge of the column.
User-defined colors	Properties in order to define user-defined colors for text and background. The settings have an effect on the Editor and Runtime. **Note:** - These settings are only available for configurable lists. - In addition, the respective focus in the list can be signalized in the Runtime by means of different text and background colors. These are configured using the project properties.
User defined colors	*Active*: User-defined colors are used.

Option	Description
Text color	Color for text display. Clicking on the color opens the color palette to select a color.
Background color	Color for the display of the cell background. Clicking on the color opens the color palette to select a color.
Lock column filter in the Runtime	- *Active*: The filter for this column cannot be changed in the Runtime. **Note:** Only available for: - Batch Control - Extended Trend - Filter screens - Message Control - Recipegroup Manager - Shift Management - Context List

CLOSE DIALOG

Option	Description
OK	Applies all changes in all tabs and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

Note: These settings are only used in the Runtime for dockable windows (on page 151) if there is no Runtime profile available for the user who is logged in.

6.6 Job variables

Job variables make it possible to assign job numbers to control recipes. Allocation is, depending on configuration, either when creating the control recipe or when starting it. A distinction is made between global job variable and individual job variables. As many variables as desired can be linked in the **Individual job variable** property. It is then possible, in the control recipe function, to select which of these variables are to be used. It is thus possible to configure a separate order variable for each production line. It is always the global **Job variable** that is used with manual creation of control recipes.

In the Runtime, the allocated job ID can be displayed in the list of control recipes.

Note: Job numbers are not translated in the recipe list or CEL. This is also applicable if you include the **@** character.

GLOBAL JOB VARIABLES

To link a global **Job variable**:

1. Navigate to the **General**/**Job variable** node in the module
2. Link a variable in the **Job variable** property
3. Select, from the drop-down list of the **Apply value from job variable at** property, the type of transfer in the Runtime:
   - When creating the control recipe:

     When creating the control recipe, the current value of the variables is transferred into the control recipe. The value is displayed in the control recipe configuration dialog. The value must not be empty, otherwise the recipe cannot be created.
   - When starting the control recipe:

     In the Runtime, the content of the variables is written to the recipe when the control recipe is started. The value must not be empty.

   Note about value changes: The value transferred by the variables is always changed into a string, regardless of the type of variables. When changing from real to string, 10 decimal points are taken into account. Zeros at the end are cut off.
4. Define a **Individual job variable** too if you wish.

INDIVIDUAL JOB VARIABLES

To link a **Individual job variable**:

1. Navigate to the **General**/**Job variable** node in the module
2. Link one or more variables in the **Individual job variable** property
3. When configuring the zenon control recipe function (on page 124), create the desired variable and the type of allocation.

Note: In order to ensure that the variable value is available immediately, the job variable is inserted into the global connection and registered when Runtime is started. The variable is requested again on reloading.

6.7 Keyboards for parameter settings

Parameters can be amended in the Runtime. In addition to the standard dialog, individually-designed keyboards can also be used for this.

SELECTION OF KEYBOARDS

You define keyboards for use in the Batch Control module in the properties for the module in the **Edit tag**/**Keyboards** group. Stipulate the desired keyboards for all data types here:

- **Binary tags**
- **Numeric tags**
- **String tags**
- **Time period TAGs**

With the **Reason for value change necessary** property, you can also stipulate that each value change must have a reason. If this property is active, a dialog to enter the reason is opened before the change is made.

For each parameter, select whether a keyboard screen is to be used. To do this:

1. Navigate to the **Write set value** group in the parameter properties.
2. Activate, in the **Keyboards** subgroup, the **Use screen Keyboard** property
3. Optionally, define which keyboard screen is to be called up in Runtime in the **Screen Keyboard** property. If you leave this property empty, the keyboard screen suitable for the data type (configured in the module properties) is used.

BEHAVIOR IN THE RUNTIME

In the Runtime, double-clicking on the entry opens a list of the respective linked keyboard. Clicking in different columns can also open different keyboards.

USING A KEYBOARD

Values can be edited in the master recipe and in the control recipe if the recipe status and the parameter settings allow it. The minimum and maximum can only be changed in the master recipe and only for numerical parameters and duration parameters.

The following applies for the use of keyboards Batch Control screens:

- Keyboards are only opened when editing parameter values in the configurable parameter lists. No keyboard screens are used when editing parameter values in the configuration dialog of a phase.

- Only one keyboard can be active at a time. If a new one is called up, the previous one is closed.
- If the keyboard is active and the mouse is double-clicked with the pointer in a column in which nothing can be changed, nothing happens.
- The following is applicable to value, minimum and maximum of a parameter:
  - If a parameter itself is linked to a keyboard, this is used.
  - If no keyboard is linked to the parameter, the keyboard that is generally linked to Batch Control in the **Edit tag** group is used.
  - If a keyboard is linked to the parameter, but this is no longer available, the keyboard that is also generally linked to Batch Control in the **Edit tag** group is used.
  - No keyboard is opened if this also does not exist.
- Keyboards for binary inputs can be provided using the **On**, **Off** and **Toggle** keys.
- When switching the units of measurement, the min/max values and the unit name are sent to the system variables again.
- For the **Changeable in control recipe property**, the keyboard for **Binary tags** defined in the **Edit tag** group is searched for. If none is linked, then a search is carried out for a keyboard with the standard name **SETBOOLKBD**. No keyboard is opened if this also does not exist.
- Keyboards that are directly linked to parameters can no longer be changed after a recipe has been approved. If the keyboard linked to the parameter is deleted, only the keyboard defined in the **Edit tag** group can be used.
- With keyboards, parameter values can also be set outside the defined parameter limits as long as the recipe is in edit mode. However, before the recipe is switched to test mode or approved, valid parameter values must be configured. Validation will fail otherwise.

Note: Use system driver variables to display additional information about the parameter value to be processed in the keyboard screen.

6.8 zenon functions

zenon provides some functions with which actions can be executed automatically in Batch Control:

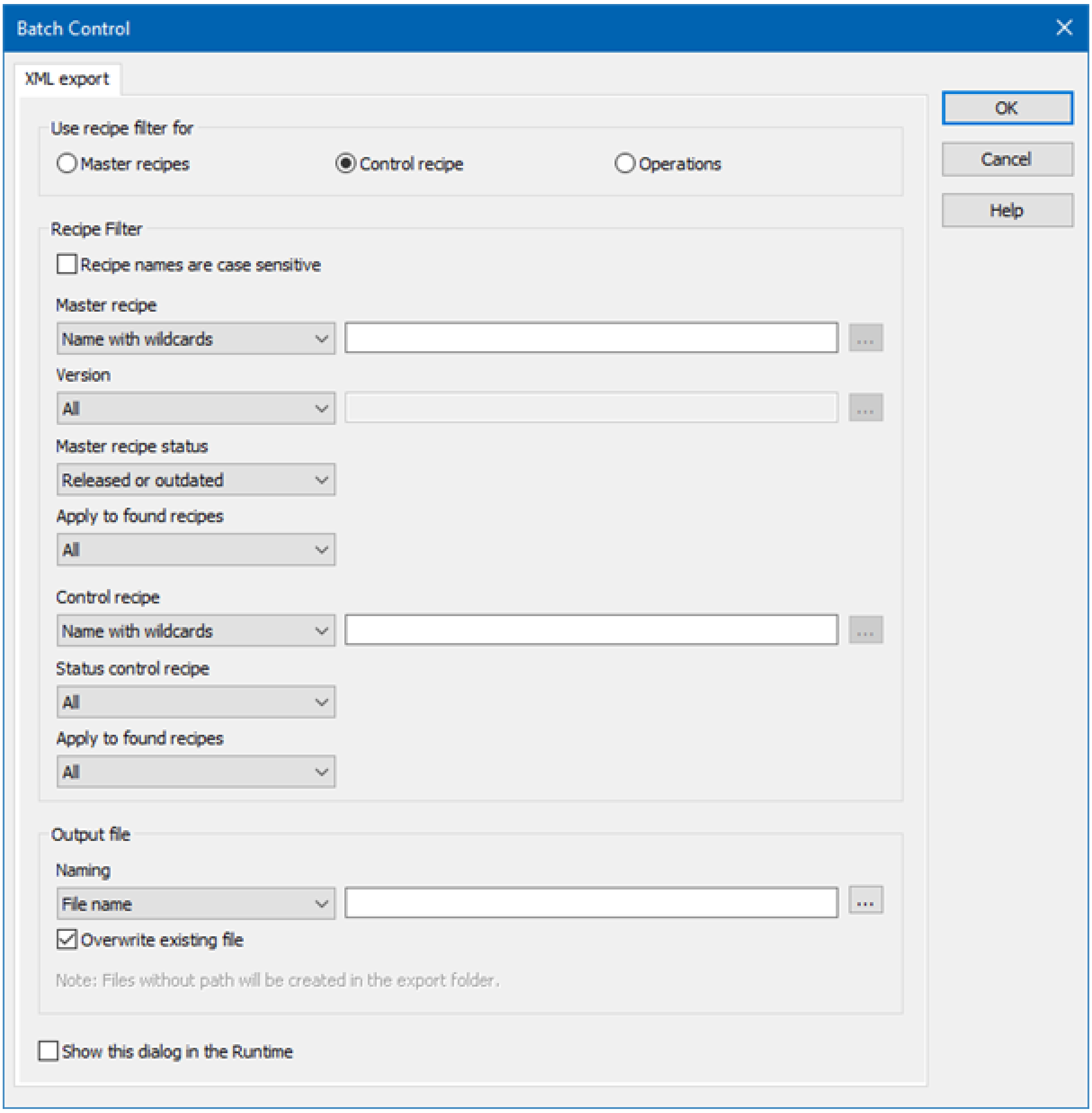
- **Export Batch recipes** (on page 125): Exports recipes to an XML file.
- **Import Batch recipes** (on page 131): Imports recipes from an XML file.
- **Execute recipe command change or mode change** (on page 133): Sends control commands to the batch execution.
- **Create control recipe** (on page 138): Creates control recipes of pre-defined master recipes.
- **Release master recipe** (on page 143): Unlocks a master recipe.

6.8.1 Export Batch recipes

Recipes can be exported to an XML file with the **Export Batch recipes** function. To create the function:

1. In the zenon Editor, navigate to the **Functions** node
2. Select **New function**
3. Go to the **Batch Control** in the function selections
4. Select **Export Batch recipes**

5. the dialog for engineering the function is opened



Use recipe filter for

○ Master recipes ● Control recipe ○ Operations

Recipe Filter

□ Recipe names are case sensitive

Master recipe: Name with wildcards

Version: All

Master recipe status: Released or outdated

Apply to found recipes: All

Control recipe: Name with wildcards

Status control recipe: All

Apply to found recipes: All

Output file

Naming: File name

☑ Overwrite existing file

Note: Files without path will be created in the export folder.

□ Show this dialog in the Runtime

USE RECIPE FILTER FOR

Parameter	Description
Use recipe filter for	Selection of what the recipe filter is applied to: ▸ Master recipe ▸ Control recipe The filter is processed from top to bottom For example, **version** is only applied to the recipes

Parameter	Description
	found in the **master recipe** filter.
Master recipe	*Active*: It is filtered on Master recipes.
Control recipe	*Active*: It is filtered on control recipes. **Note:** The attendant master recipes must also be selected. If no master recipe has been selected for the control recipe, the filter cannot find the recipe being searched for in the Runtime. **Hint:** If the master recipe is not known, filtering of all master recipes with a placeholder is recommended.

RECIPE FILTER

Parameter	Description
Recipe Filter	Configuration of the recipe filter
Master recipe	Parameters for the selection of the master recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*: The name of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables.
Version	Selection of the version (on page 217) from the drop-down list: - *All*: The version stated is ignored and each version found is used. - *Fixed version*: This filters for versions that are entered in

Parameter	Description
	this field. Highest possible version: *4294967295* - *Version from variable*: The version of the master recipe is defined by a variable in the Runtime. Click on button **...** in order to open the dialog for selecting a variable. - *Only oldest version*: Only the recipe with the oldest version number is used. - *Only newest version*: Only the recipe with the newest version number is used. - Default: *All*
State master recipe	Status of the master recipe. Select from drop-down list. **When selecting master recipes for recipe filters:** - *All* - *Edit mode* - *Released* - *Test mode* - *Test running* - *Terminated with error* - *Outdated* **When selecting control recipes for recipe filters:** - *Released or outdated* - *Released* - *Outdated*
Apply to found recipes	Definition of which master recipes the filter is applied to. Select from drop-down list: - *All*

Parameter	Description
	- *Only oldest ID* - *Only newest ID*
Control recipe	Parameters for the selection of the control recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*:__ The name of the control recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. Precisely one recipe can be found if the variable value at the time of execution is a valid ID of a control recipe. *Job ID from variable*: Finds control recipes that have the given job ID number. Any type of variable can be linked. The value is automatically converted into STRING.
State control recipe	Definition of which recipe status the filter is applied to. Select from drop-down list: - *All* - *Prepared* - *In execution* - *Executed* - *Terminated with error* - *Outdated*
Apply to found recipes	Definition of which control recipes the filter is applied to. Select from drop-down list: - *All* - *Only oldest ID*

Parameter	Description
	- *Only newest ID* **Note:** Only the respective IDs are taken into account for master recipes and control recipes. The search for control recipes can find several recipes with this filter. This filter must also be activated for the master recipes for a unique result.
Operation	Parameters for the selection of the operation. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*:_ The name of the partial recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the partial recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. Precisely one recipe can be found if the variable value at the time of execution is a valid ID of an operation.
Status operation	Definition of which recipe status the filter is applied to. Select from drop-down list: - *All* - *Edit mode* - *Released* - *Outdated*
Apply to found recipes	Definition of which operations the filter is applied to. Select from drop-down list: - *All* - *Only oldest ID* - *Only newest ID*
Output file	Selection of the file name and the save location.

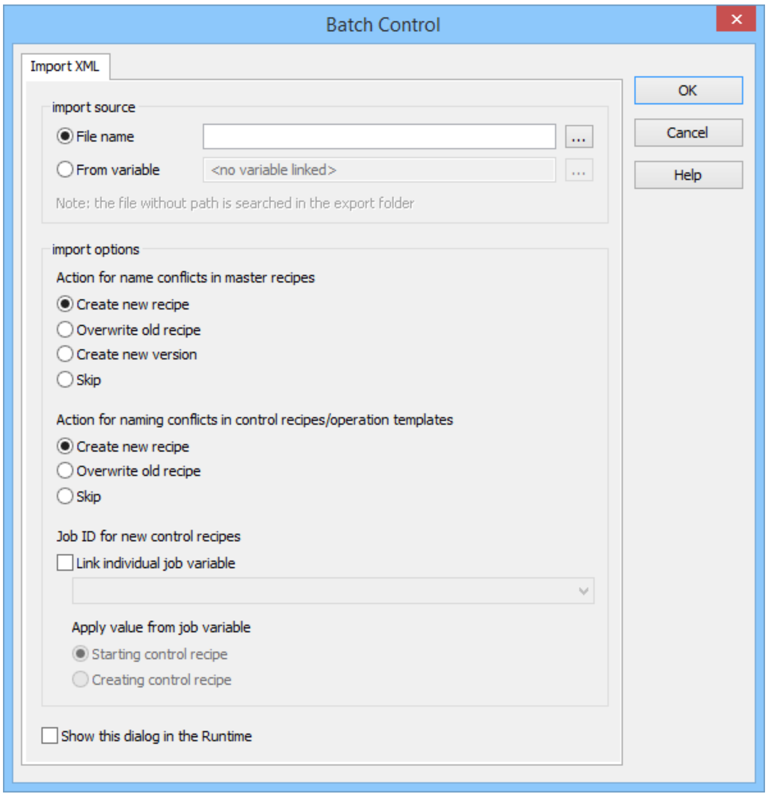
Parameter	Description
Naming	- *File name*: The name of the XML file can be entered manually or when selecting a file from the respective save path. - *File name from variable*: The name of the XML file is taken from the linked variable. - *Create file name from recipe name*: The name of the XML file is made up of the recipe type, recipe name and recipe version. - *Create file name from recipe ID*: The name of the XML file is made up of the recipe type and recipe ID.
Overwrite existing file	*Active*: If there are naming conflicts, the pre-existing XML file is overwritten with the names.
Note: Files without a path will be created in the export folder	Files without the path stated will be automatically created in the export folder.
Show this dialog in the Runtime	Active: The dialog is shown in Runtime and can be operated.
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

6.8.2 Import Batch recipes

Recipes can be imported to an XML file with the **Import Batch recipes** function. To create the function:

1. In the zenon Editor, navigate to the **Functions** node
2. Select **New function**
3. Go to the **Batch Control** in the function selections
4. Select **Import Batch recipes**

5. the dialog for engineering the function is opened



import source

● File name

○ From variable <no variable linked>

Note: the file without path is searched in the export folder

import options

Action for name conflicts in master recipes

● Create new recipe

○ Overwrite old recipe

○ Create new version

○ Skip

Action for naming conflicts in control recipes/operation templates

● Create new recipe

○ Overwrite old recipe

○ Skip

Job ID for new control recipes

□ Link individual job variable

Apply value from job variable

● Starting control recipe

○ Creating control recipe

□ Show this dialog in the Runtime

IMPORT SOURCE

Parameter	Description
File name	Imports the XML file with the selected file name.
From variable	Imports the XML file from the linked variable.

IMPORT OPTIONS

Parameter	Description
Action for name conflicts in master recipes	Selection of a master recipe name that already exists.
Create new recipe	Creates a new recipe with the name.
Overwrite old recipe	Overwrites an existing recipe with the name.
Create new version	Creates a new version of the recipe.
Skip	Skips this step.
Action in the event of naming conflicts in	Selection when issuing a control recipe or

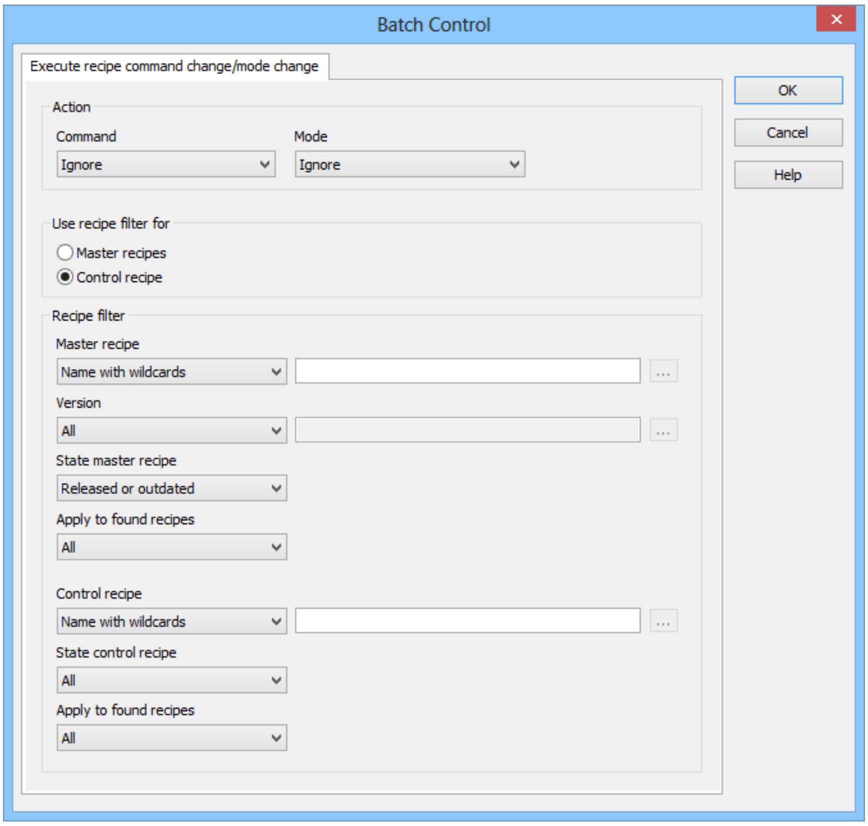
Parameter	Description
control recipes/operations	operation that already exists.
CREATE NEW RECIPE	Creates a new recipe with the name.
Overwrite old recipe	Overwrites an existing recipe with the name.
Create new version	Creates a new version of the recipe.
Skip	Skips this step.
Job ID for new control recipes	Job ID for newly-created control recipe
Link individual job variable	Links an individual order variable for each control recipe
Apply value from job variable in the event of	The value from the job variable is taken with - **Starting control recipe** - **Creating control recipe**
Show this dialog in the Runtime	This dialog can also be operated in Runtime.

6.8.3 Execute recipe command or mode switch

You can send control commands to the batch execution with this function. To create the function:

1. In the zenon Editor, navigate to the **Functions** node.
2. Select **New function**.
3. Go to the **Batch Control** in the function selections.
4. Select **Execute recipe command or change mode**.

The dialog for engineering a function is opened.



Action

Command: Ignore

Mode: Ignore

Use recipe filter for

○ Master recipes

● Control recipe

Recipe filter

Master recipe: Name with wildcards

Version: All

State master recipe: Released or outdated

Apply to found recipes: All

Control recipe: Name with wildcards

State control recipe: All

Apply to found recipes: All

ACTION

Option	Description
Action	Selection of the action to be executed: - Command - Mode
Command	Selection of the command to be executed from drop-down list: - Ignore - Start recipe - Recipe pausing - Resume recipe - Hold recipe - Restart recipe - Recipe stopping

Option	Description
	- Abort recipe
Mode	Selection of the mode to which the recipe is to be switched: - Ignore - Automatic - Semi-automatic - Manual

USE RECIPE FILTER FOR

Parameter	Description
Use recipe filter for	Selection of what the recipe filter is applied to: - Master recipe - Control recipe The filter is processed from top to bottom For example, **version** is only applied to the recipes found in the **master recipe** filter.
Master recipe	*Active*: It is filtered on Master recipes.
Control recipe	*Active*: It is filtered on control recipes. **Note:** The attendant master recipes must also be selected. If no master recipe has been selected for the control recipe, the filter cannot find the recipe being searched for in the Runtime. **Hint:** If the master recipe is not known, filtering of all master recipes with a placeholder is recommended.

RECIPE FILTER

Parameter	Description
Recipe Filter	Configuration of the recipe filter
Master recipe	Parameters for the selection of the master recipe. Select from drop-down list: - *Name with wildcards*:

Parameter	Description
	A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*: The name of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables.
Version	Selection of the version (on page 217) from the drop-down list: - *All*: The version stated is ignored and each version found is used. - *Fixed version*: This filters for versions that are entered in this field. Highest possible version: *4294967295* - *Version from variable*: The version of the master recipe is defined by a variable in the Runtime. Click on button **...** in order to open the dialog for selecting a variable. - *Only oldest version*: Only the recipe with the oldest version number is used. - *Only newest version*: Only the recipe with the newest version number is used. - Default: *All*
State master recipe	Status of the master recipe. Select from drop-down list. **When selecting master recipes for recipe filters:** - *All*

Parameter	Description
	- *Test mode* - *Test running* **When selecting control recipes for recipe filters:** - *Released or outdated* - *Released* - *Outdated*
Apply to found recipes	Definition of which master recipes the filter is applied to. Select from drop-down list: - *All* - *Only oldest ID* - *Only newest ID*
Control recipe	Parameters for the selection of the control recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*: The name of the control recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. Precisely one recipe can be found if the variable value at the time of execution is a valid ID of a control recipe. - *Job ID from variable*: Finds control recipes that have the given job ID number. Any type of variable can be linked. The value is automatically converted into STRING.
State control recipe	Definition of which recipe status the filter is applied

Parameter	Description
	to. Select from drop-down list: - *All* - *Prepared* - *In execution*
Apply to found recipes	Definition of which control recipes the filter is applied to. Select from drop-down list: - *All* - *Only oldest ID* - *Only newest ID*
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

6.8.4 Create control recipe

A control recipe for a defined master recipe can be created with the help of the **Create control recipe** function.

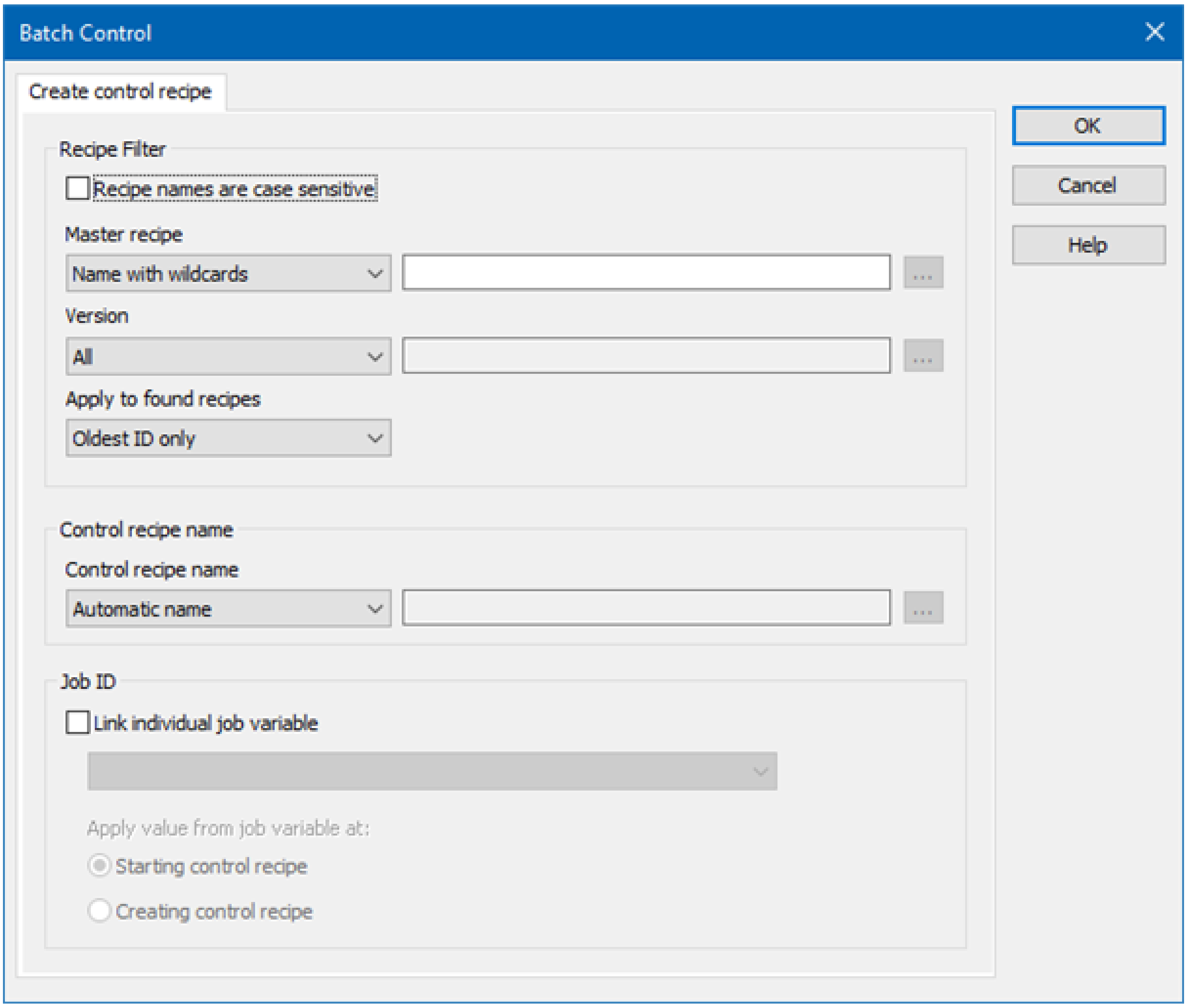
> ⚠**Attention**
>
> Automatic creation of control recipes via functions or API: Ensure that a new control recipe is only created once the variable values for the one previously created are present.
>
> This means the following for the values of the monitoring system driver variables:
>
> - **[Batch Control] Create control recipe result numerically**: must be greater than *-1*.
> - **[Batch Control] Create control recipe result XML**: must not be empty
>
> You can find details about this in the **SYSDRV** manual, in the **Topic - Batch Control** chapter.

To create the function:

1. In the zenon Editor, navigate to the **Functions** node.
2. Select **New function**.
3. Go to the **Batch Control** node in the function selection.
4. Select **Create control recipe**.

   The dialog for engineering a function is opened.

CONFIGURATION DIALOG



Batch Control

Create control recipe

Recipe Filter

□ Recipe names are case sensitive

Master recipe: Name with wildcards

Version: All

Apply to found recipes: Oldest ID only

Control recipe name

Control recipe name: Automatic name

Job ID

□ Link individual job variable

Apply value from job variable at:

● Starting control recipe

○ Creating control recipe

RECIPE FILTER

Parameters	Description
Recipe Filter	Configuration of the recipe filter The filter is processed from top to bottom For example, **version** is only applied to the recipes

Parameters	Description
	found in the **master recipe** filter.
Recipe names are case sensitive	- *Active*: Recipe name are case sensitive. A distinction is made between **Recipe** and **RECIPE**. - *Inactive*: Capitalization is not taken into account. **Recipe** is handled the same as **RECIPE**.
Master recipe	Parameters for the selection of the master recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*: The name of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables.
Version	Selection of the version (on page 217) from the drop-down list: - *All*: The version stated is ignored and each version found is used. - *Fixed version*: This filters for versions that are entered in this field. Highest possible version: *4294967295* - *Version from variable*: The version of the master recipe is defined by a variable in the Runtime. Click on button **...** in order to open the dialog for selecting a variable. - *Only oldest version*: Only the recipe with the oldest version

Parameters	Description
	number is used. - *Only newest version*: Only the recipe with the newest version number is used. - Default: *All* **Note:** The recipe status is checked in the Runtime after this filter is applied. If, for example, filtering for *most recent version only* is carried out and the appropriate master recipe is not unlocked, no recipe is selected.
Apply to found recipes	Definition of which recipes the filter is applied to. Selection of ID from drop-down list: - *All* - *Oldest ID only* - *Latest ID only* **Note:** The recipe status is checked in the Runtime before this filter is applied. Only the IDs of approved master recipes are checked.

CONTROL RECIPE NAME

Parameters	Description
Control recipe name	Configuration of the name of the control recipe.
Control recipe name	Selection of the naming from the drop-down list: - *Automatic name*: Name is automatically issued on creation - *Name from variable*: Name is taken from a variable. Click on button **...** in order to open the dialog for selecting a variable. If there is already a recipe with the name that has been transferred from the variable in the Runtime, no new control recipe is created.

JOB ID

Parameters	Description
Job ID	Configuration of the Job ID.
Link individual job variable	*Active*: A job variable (on page 121) can be linked. The variable must already be configured. Selection of the variable from the drop-down list:
Apply value from job variable in the event of	Definition of the time at which the job ID is transferred. During: - Starting control recipe - Creating control recipe
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog. The function is nevertheless created, however without a defined target.
Help	Opens online help.

All filters always have an effect on a group of recipes with the same name. Depending on the configuration, more than one recipe can remain left over.

For example: Recipes with the **Test*** filter are searched for. The result is 5 versions of **Test_1** and 3 versions of **Test_2**. If filtering for the latest version is continued, then two control recipes are created, one each for the recipe with the highest version number per group.

Note for variable selection using name or ID: For the selection of variables according to name or ID, numerical variables and string variables can be selected respectively. The data types are converted to the respective correct form.

ROLE OF THE RECIPE STATUS IN THE PROCESS

Note the following when configuring the filter: Control recipes can only be created for approved master recipes.

The recipe status is also taken into account when executing the filter in the Runtime. It cannot be configured in the dialog. The status filter is applied after the **version** filter and before the **apply to found recipes** filter.

EXAMPLE

There are **two versions of the master recipe:**

- **Version 1**: approved

- **Version 2**: in progress

The filter acts differently in the Runtime depending on requirements for the **Version** option:

Filter setting	Result
Version: *Only newest version*:	No recipe suits the filter. The most recent version is **version 2**. It is not approved however.
Version: *All* **Apply to found recipes:** *Most recent ID*:	**Version 1** is used. Version 1 is approved and will be selected. The filter for the ID is only applied afterwards. At this point in time, only **version 1** is in the filter and is thus the version with the most recent ID.

6.8.5 Release master recipe

With the help of the **Release master recipe** function, it is possible to approve a master recipe in the Runtime. The function unlocks the master recipes that have been found whose properties for the configuration suit and whose possibly-embedded partial recipes have already been unlocked. It can be instigated by a computer in the zenon network.

Information

A master recipe can only be approved if:

- The recipe is free of errors
- All recipes contained therein have already been approved

When approving, the current configuration is saved for all phases currently used in the recipe. This also includes all phases of placeholders. The configuration is also used in the control recipe. Any possible changes - via the Editor for example - are no longer taken into account.

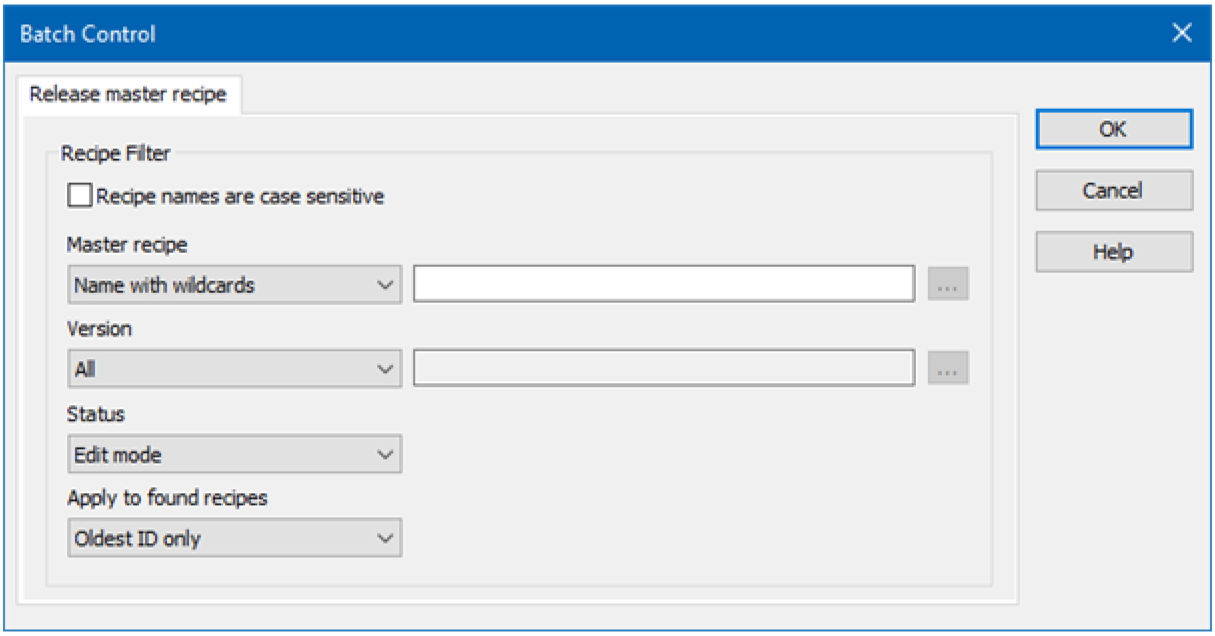
After approval, a master recipe can no longer be edited!
You can however create a copy of the approved master recipe. To do this, click on the **Duplicate Master Recipe** button. This copy can then be edited.

To create the function:

1. In the zenon Editor, navigate to the **Functions** node.
2. Select **New function**.
3. Go to the **Batch Control** node in the function selection.
4. Select **Release master recipe**.

The dialog for engineering a function is opened.

CONFIGURATION DIALOG



Recipe Filter

□ Recipe names are case sensitive

Master recipe: Name with wildcards

Version: All

Status: Edit mode

Apply to found recipes: Oldest ID only

RECIPE FILTER

Parameter	Description
Recipe Filter	Configuration of the recipe filter The filter is processed from top to bottom For example, **version** is only applied to the recipes found in the **master recipe** filter.
Recipe names are case sensitive	- *Active*: Recipe name are case sensitive. A distinction is made between **Recipe** and **RECIPE**. - *Inactive*: Capitalization is not taken into account. **Recipe** is handled the same as **RECIPE**.
Master recipe	Parameters for the selection of the master recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out.

Parameter	Description
	▸ *Name from variable*: The name of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. ▸ *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables.
Version	Selection of the version (on page 217) from the drop-down list: ▸ *All*: The version stated is ignored and each version found is used. ▸ *Fixed version*: This filters for versions that are entered in this field. Highest possible version: *4294967295* ▸ *Version from variable*: The version of the master recipe is defined by a variable in the Runtime. Click on button **...** in order to open the dialog for selecting a variable. ▸ *Only oldest version*: Only the recipe with the oldest version number is used. ▸ *Only newest version*: Only the recipe with the newest version number is used. ▸ Default: *All*
Status	Selection of the status that the recipe must have, from a drop-down list: ▸ *Edit mode* ▸ *Test mode* Default: *Edit mode*
Apply to found recipes	Definition of which recipes the filter is applied to. Selection related to IDs from drop-down list:

Parameter	Description
	- *All* - *Oldest ID only* - *Latest ID only* Default: *Oldest ID only*
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog. The function is nevertheless created, however without a defined target.
Help	Opens online help.

All filters always have an effect on a group of recipes with the same name. Depending on the configuration, more than one recipe can remain left over.
For example: Recipes with the **Test*** filter are searched for. The result is 5 versions of **Test_1** and 3 versions of **Test_2**. If there is further filtering for the most recent version, the most recent version of **Test_1** and the most recent version of **Test_2** is released.

Note for variable selection using name or ID: For the selection of variables according to name or ID, numerical variables and string variables can be selected respectively. The data types are converted to the respective correct form.

6.9 Rules for recipe creation

Rules can be defined for the creation of recipes in Runtime, in the editor in the **General**/**Recipe creation** properties group.

DISALLOW PARALLEL BRANCHING

You stipulate whether parallel branching is permitted using the **Disallow parallel branching**. If the property is activated, validation in Runtime does not accept any parallel branches. Recipes that have already been validated are validated again before starting. Recipes that have already been started are also canceled after this property has been activated.

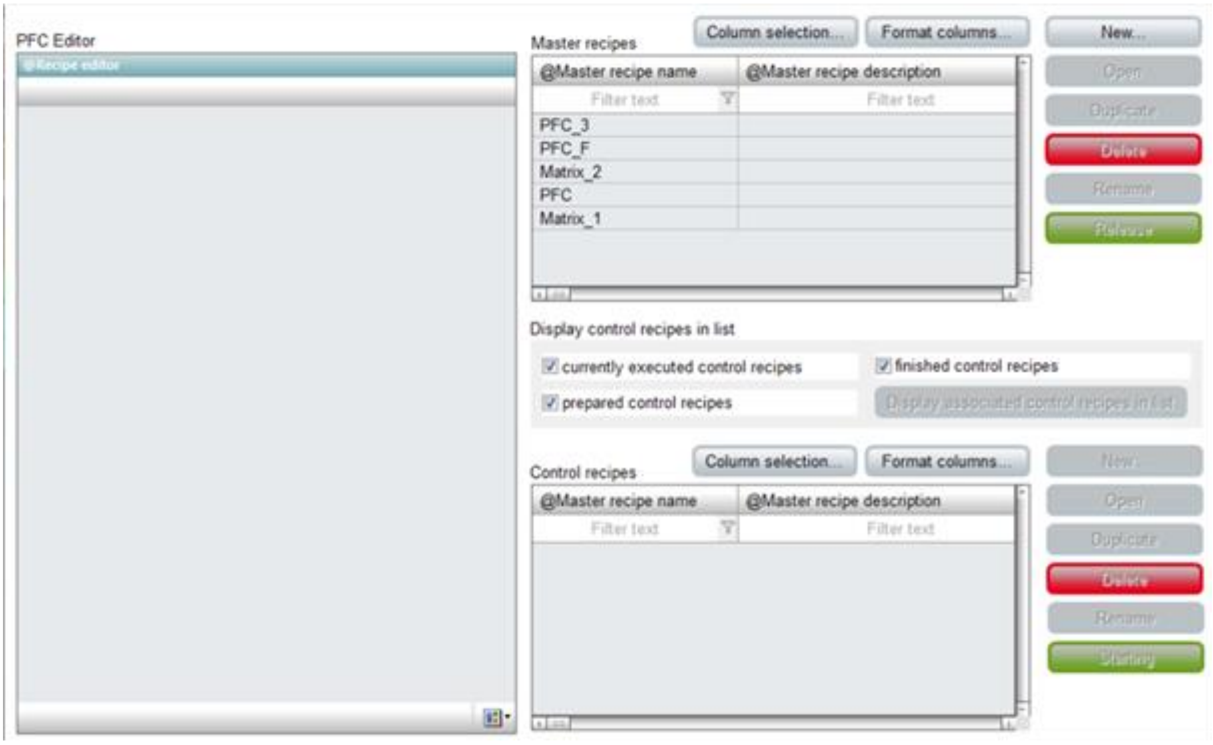
7 Conversion

If the version of zenon is changed, all recipes must be ended before converting the project.

Recipes that are running continue to be executed after a restart. The restart only functions within the same zenon version.

8 Configure and control in the Runtime

The entire management, creation and processing of the recipes is done in the Runtime. Editing in the zenon Editor is not possible.



@Master recipe name	@Master recipe description
Filter text	Filter text
PFC_3	
PFC_F	
Matrix_2	
PFC	
Matrix_1	

@Master recipe name	@Master recipe description
Filter text	Filter text

Note: Runtime files up to and including version 7.00 SP0 are not compatible with subsequent versions. Versions from version 7.10 and later are compatible.

SYNCHRONIZATION

When loading, opening, duplicating and approving a recipe or operation, a check is made to see if the configuration of units, phases etc. has been changed in the superordinate instance, such as the Editor. For details, see the Synchronization (on page 292) chapter.

SELECTION PROCEDURE IN LISTS:

- The keyboard shortcut **Ctrl+A**: selects all elements
- The keyboard shortcut **Ctrl+mouse click**: adds master recipe to the existing selection

- **Shift key + mouse click**: Extends selection from the currently selected master recipe to the clicked master recipe

SAVING OF RECIPES WHEN CLOSING RUNTIME

If Runtime is closed and there are still recipes that have not been saved, you are asked if these recipes are to be saved. In order for this query to not prevent Runtime closing, Runtime is automatically closed after 15 seconds if nothing is entered. Unsaved recipes are then discarded.

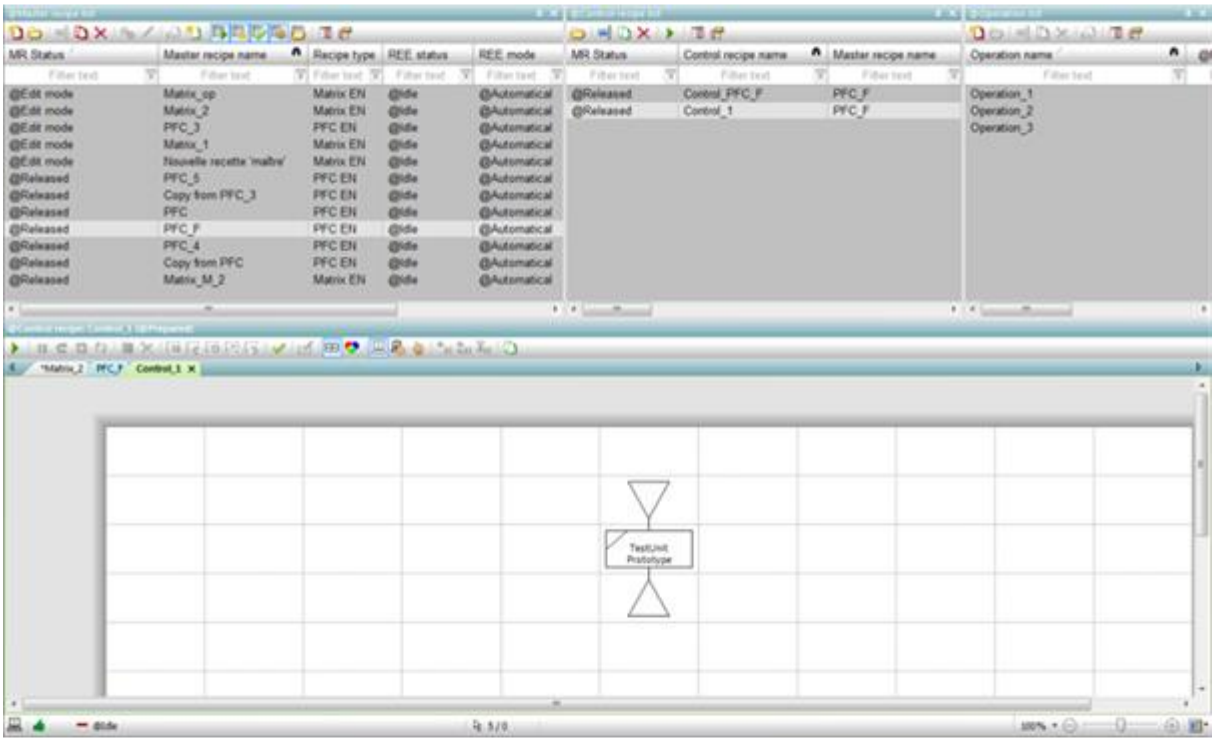
EXPORT AND IMPORT VIA XML

Master recipes and control recipes can be exported and reimported in XML format. When importing, the limit values for parameters are now also re-read and validated.

8.1 User interface

The user interface of the editor in Runtime can be configured with toolbars and dockable windows.

Example: User interface with list of the master recipes, list of the control recipe, list of the operation and PFC editor:



MR Status	Master recipe name	Recipe type	REE status	REE mode
@Edit mode	Matrix_op	Matrix EN	@Idle	@Automatical
@Edit mode	Matrix_2	Matrix EN	@Idle	@Automatical
@Edit mode	PFC_3	PFC EN	@Idle	@Automatical
@Edit mode	Matrix_1	Matrix EN	@Idle	@Automatical
@Edit mode	Nouvelle recette 'maître'	Matrix EN	@Idle	@Automatical
@Released	PFC_5	PFC EN	@Idle	@Automatical
@Released	Copy from PFC_3	PFC EN	@Idle	@Automatical
@Released	PFC	PFC EN	@Idle	@Automatical
@Released	PFC_F	PFC EN	@Idle	@Automatical
@Released	PFC_4	PFC EN	@Idle	@Automatical
@Released	Copy from PFC	PFC EN	@Idle	@Automatical
@Released	Matrix_M_2	Matrix EN	@Idle	@Automatical

MR Status	Control recipe name	Master recipe name
@Released	Control_PFC_F	PFC_F
@Released	Control_1	PFC_F

Operation name
Operation_1
Operation_2
Operation_3

SWITCHING POSITIONS

The position in the recipe is adapted depending on the recipe and status of the recipe:

- When opening a currently-running recipe:

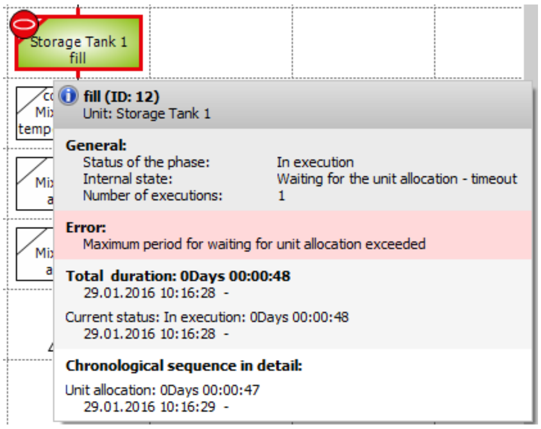
  It is centered on the first active element. The first active element is the one that is at the top. If there are several active elements in the same line, the element that is furthest to the left is selected. Elements with execution positions before or after this are handled the same as the active ones in this case.

- When a recipe is first opened

  Centering is on the start of the recipe. In a PFC recipe, centering is on the start element. A matrix recipe is opened in such a way that the upper left corner is visible.

TOOL TIP

A tool tip informs you about the current and historical events of an element.



fill (ID: 12)
Unit: Storage Tank 1

General:

```
Status of the phase:        In execution
Internal state:             Waiting for the unit allocation - timeout
Number of executions:       1
```

Error:
Maximum period for waiting for unit allocation exceeded

Total duration: 0Days 00:00:48
29.01.2016 10:16:28 -

Current status: In execution: 0Days 00:00:48
29.01.2016 10:16:28 -

Chronological sequence in detail:

Unit allocation: 0Days 00:00:47
29.01.2016 10:16:29 -

The following is displayed:

- Element name
- General information about status and number of executions.
  Note: The counter becomes active if the phase is reactivated, including if the status is *restarting*, but the restart condition has not yet been run through.
- Error
- Total duration including times in status *Paused*.
- Timing

8.1.1 Control elements of the recipe editor

The following are available for the editor:

- Toolbars (can be hidden)

- Tabs (on page 150)
- Dockable windows (on page 151)

TOOLBARS

There are independent toolbars available in the Recipe editor for each type of recipe and the different status. For details, see the Matrix recipe toolbars (on page 209) and Toolbars and PFC recipe context menu (on page 167) chapters. All actions of the individual symbols can also be engineered using their own buttons in the screen. If the toolbars are thus not needed, they can be displayed or hidden using the settings in the zenon Editor.

To show/hide toolbars:

1. In the zenon Editor, highlight the **Recipe editor** control element in the **Batch Control** screen
2. Go to the property group **Representation\Show editor control elements**.
3. Activate or deactivate the **Toolbars** checkbox.

8.1.1.1 Tabs in the recipe editor

If several recipes are open in the recipe editor, these are represented with tabs. Recipes can be displayed and opened in two groups next to each other or underneath each other. To open a recipe in a new group:

1. In the context menu of the recipe, select the **horizontal neighboring group** or **vertical neighboring group** command
2. Select the type of display:
   - Move
   - Open in parallel

The control elements are always only applicable to the active recipe of the active group.

- The active group is emphasized in color.
- The active tab is emphasized with bold font. Information on the active tab is shown in the title bar of the editor.
- Tabs can be moved and arranged by dragging & dropping, including between groups.

TAB CONTEXT MENU

Parameter	Description
Save	Saves changes in the recipe.
Close	Closes the recipe.

Parameter	Description
Close all others	Closes all other open recipes. Only the recipe in which the context menu was activated remains open.
Group horizontally	Opens the recipe in a new group below the other recipes. - Move display: The recipe is moved and the upper group is removed. - Move display parallel. The recipe is displayed in parallel in both groups.
Group vertically	Opens the recipe in a new group to the right of the active group. - Move display: The recipe is moved and removed from the left group. - Move display parallel. The recipe is displayed in parallel in both groups.

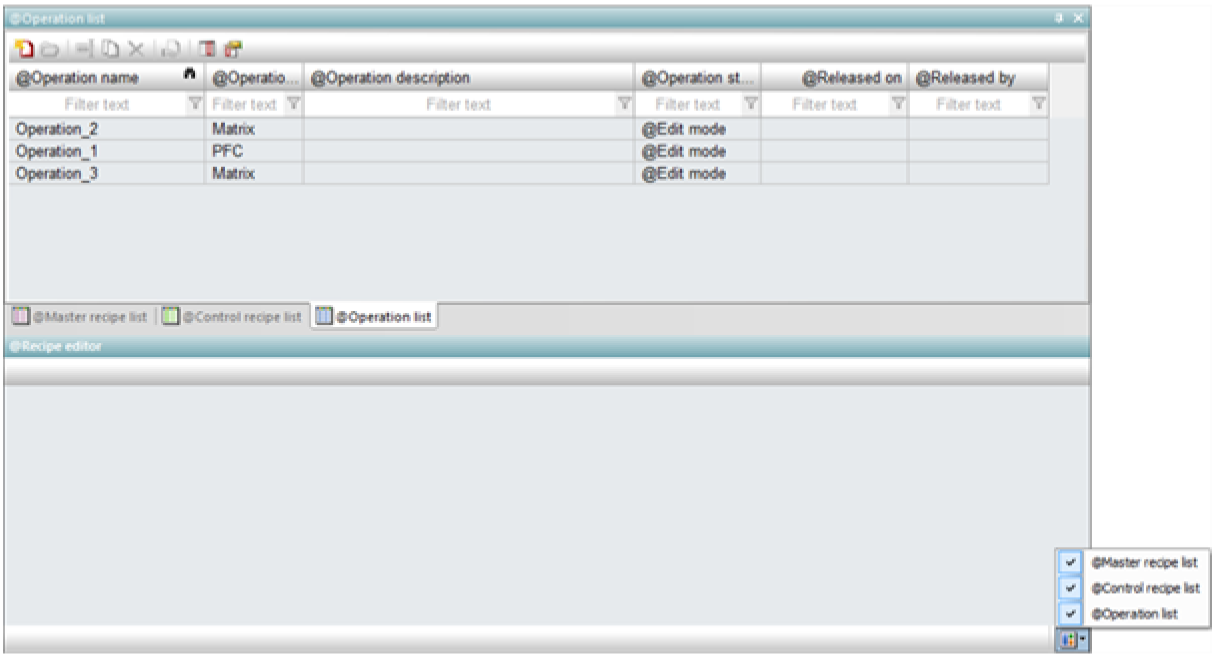
8.1.1.2 Dockable windows

In the recipe editor, three windows can be shown, positioned and docked in the Runtime as desired:

- List of master recipes (on page 161)
- List of control recipes (on page 232)
- List of operations (on page 220)

For details on the selection and positioning, see the Selection and positioning (on page 154) chapter.

The settings are saved individually for each computer and user.



@Operation name	@Operatio...	@Operation description	@Operation st...	@Released on	@Released by
Filter text	Filter text	Filter text	Filter text	Filter text	Filter text
Operation_2	Matrix		@Edit mode		
Operation_1	PFC		@Edit mode		
Operation_3	Matrix		@Edit mode		

SHOW/HIDE LIST

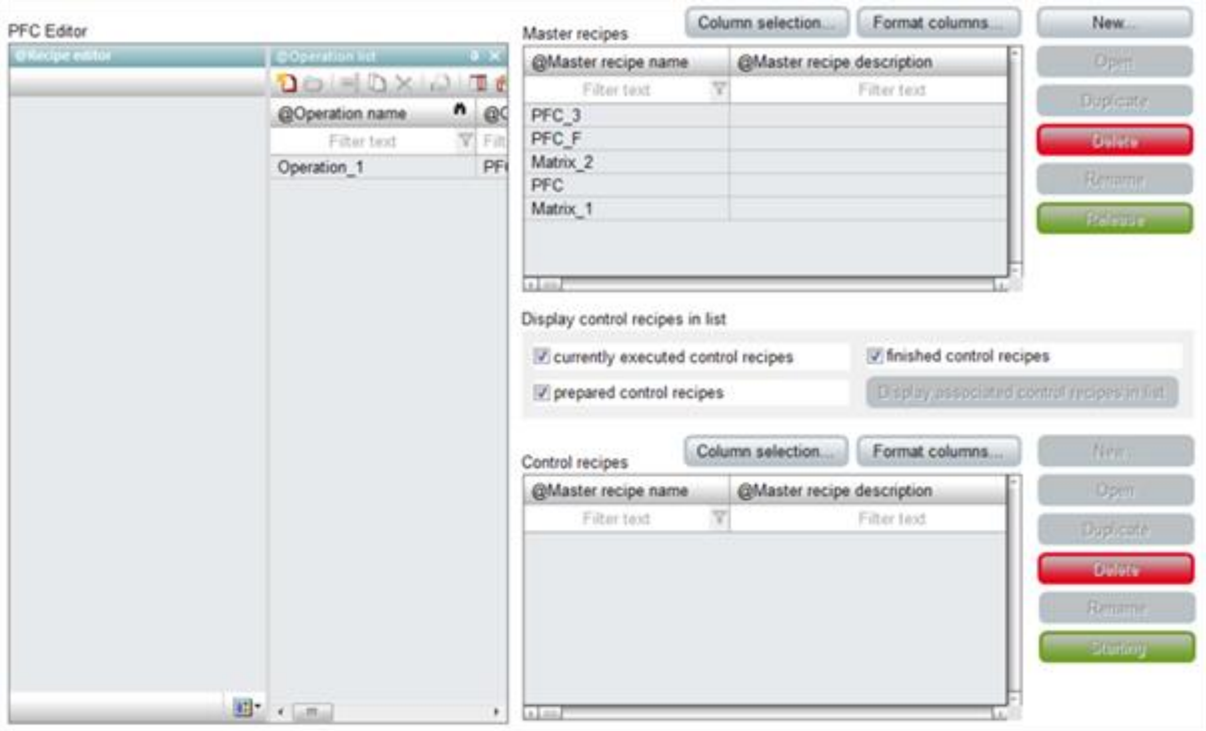
The list of dockable windows can be displayed or hidden by means of settings in the zenon Editor.

To show/hide the list:

1. In the zenon Editor, highlight the **Recipe editor** comment area in the **Batch Control** screen
2. go to property group **Representation\Show editor control elements**
3. Activate or deactivate the **Dockable windows** checkbox

EXAMPLES

Additional list of the operations shown in the Recipe Editor:

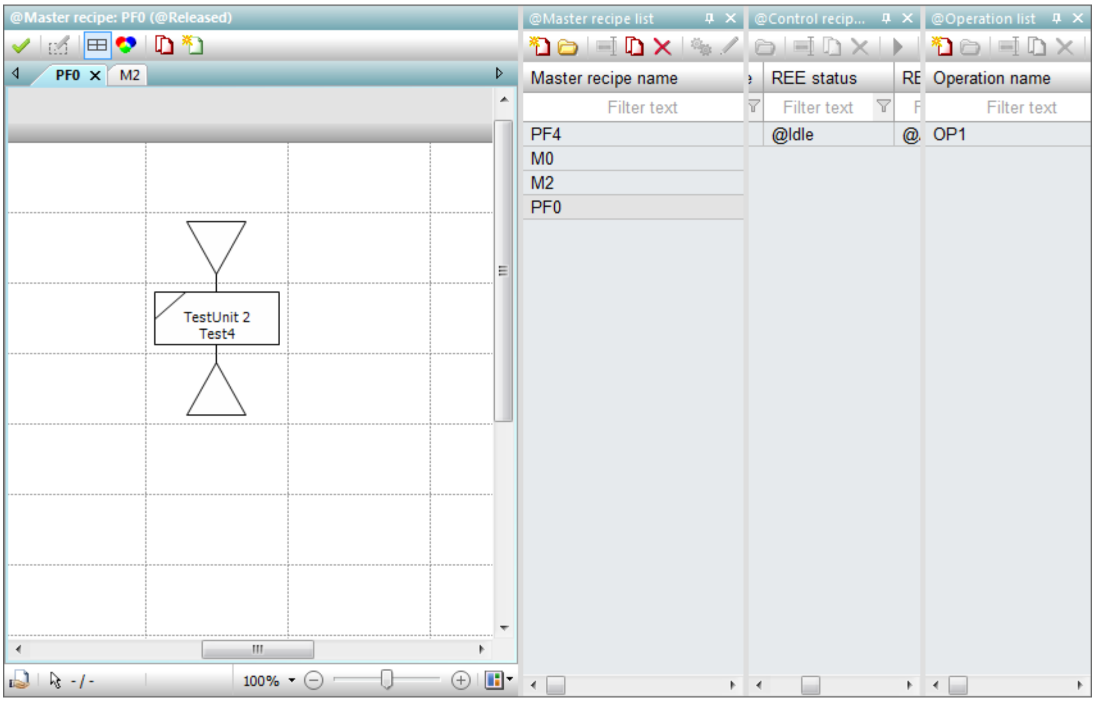


@Operation name	@O
Filter text	Filt
Operation_1	PF(

@Master recipe name	@Master recipe description
Filter text	Filter text
PFC_3	
PFC_F	
Matrix_2	
PFC	
Matrix_1	

@Master recipe name	@Master recipe description
Filter text	Filter text

Additional list of master recipes, control recipes and operations shown in the Recipe Editor:

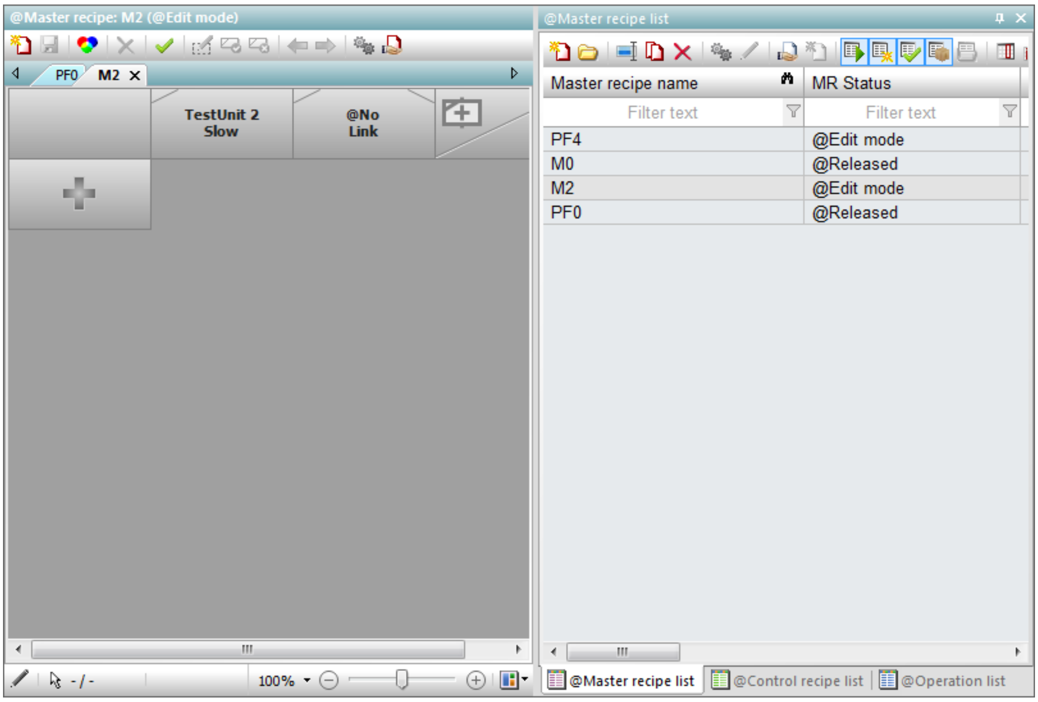


Master recipe name
Filter text
PF4
M0
M2
PF0

REE status	RE
Filter text	F
@Idle	@

Operation name
Filter text
OP1

Lists of master recipes, control recipes and operations shown in the Recipe Editor:



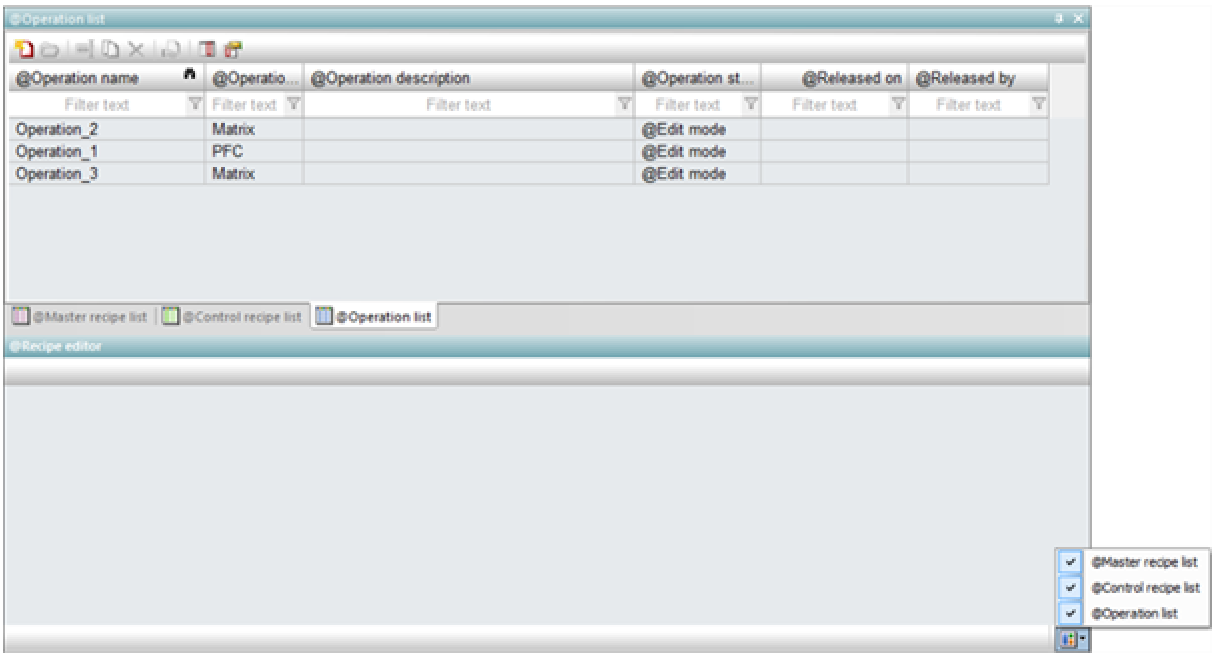
Master recipe name	MR Status
Filter text	Filter text
PF4	@Edit mode
M0	@Released
M2	@Edit mode
PF0	@Released

8.1.1.2.1 Selection and positioning

To show and dock the window in the Editor:

- Select the desired window using the drop-down list of the **Selection of dockable windows**
- move the window to the desired position
- Locate this with the positioning aid

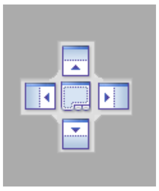
The **Selection of dockable windows** is located at the right edge of the status bar of the Editor. Clicking on the symbol opens the list of dockable windows. Selection is carried out by activating/deactivating the checkboxes.



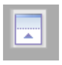
@Operation name	@Operatio...	@Operation description	@Operation st...	@Released on	@Released by
Filter text	Filter text	Filter text	Filter text	Filter text	Filter text
Operation_2	Matrix		@Edit mode		
Operation_1	PFC		@Edit mode		
Operation_3	Matrix		@Edit mode		

POSITIONING AIDS

When moving windows from the Editor interface, positioning aids are displayed. These represent windows or their borders.



This element represents a window area in the Editor.



This element represents the border area of the Editor.

POSITION WINDOW

To position an element as docked:

1. Move the element with the mouse into the desired area
2. The positioning aid is displayed
3. This represents a window and its areas:

a) Center: whole window

b) Top: upper half

c) Bottom: lower half

d) Right: right half

e) Left: left half

or the border of the Editor

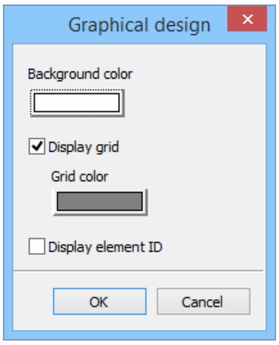
4. Move the mouse to the central positioning aid or to a positioning aid on the border of the editor and from there to the desired area
5. The area in the Editor where the element was placed when the mouse button was released is colored in blue
6. Move the mouse within the positioning aid to the desired area that is displayed in blue
7. Let the mouse button go and the element is placed

If a window is placed on a pre-existing window, both windows are displayed at the same location using tabs.

Note: You can read more about positioning in the chapter on the zenon Editor in the User interface/Positioning windows section.

8.1.2 Graphical design

Clicking on the symbol for the Graphical design in the toolbar opens the dialog for configuring the colors, grid settings and display of the element ID in the recipe that is currently open.



Parameter	Description
Background color	Defines the background color of the diagram. Click on the color in order to open the palette for selecting a color.
Display grid	▸ *Active*: Display the grid ▸ *Inactive*: Grid is hidden. Can only be configured for PFC recipes.

Parameter	Description
Grid color	Defines the line color of the grid. Click on the color in order to open the palette for selecting a color.
Show element IDs	*Inactive*: No element ID is shown in the recipe. **Note:** This setting is recommended for normal operation. *Active*: The ID of the elements is displayed in the recipe. The exception is lines. The display is in the upper left-hand corner of the element. **Note:** This setting is recommended for troubleshooting.
OK	Applies all settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

8.1.3 Status line

The status line is automatically adapted to each recipe that contains the focus and initialized with its data. The status line the following is displayed:

- Mode of the recipe
  - Edit mode
  - Manual mode
  - Semi-automatic mode
  - Automatic mode
- Status of the recipe:
  - Ready for start
  - Error: Currently there is an error in the recipe. The number of errors is also displayed.
  - Historical error: During the execution at least one error occurred. Currently not error exists.
  - No error: Until now the execution runs faultless.
- Current execution status (on page 263).
- Mouse cursor position and changes such as moving the connection line
- Approval of a recipe
- Finished without errors
- Zoom level of the current view; this can also be controlled here

The global statuses of the main recipe are also displayed when executing operations.

8.1.4 Touch operation

You can do the following to simplify touch operation in Runtime:

- Optimize the operability of the active elements
- Display some symbols in matrix recipes as control elements (on page 209)
- Assign control strategies (on page 229) using control elements of a phase

OPTIMIZE ACTIVE ELEMENT FOR TOUCH

In order to optimize the operability of the active elements in a recipe that is running, you can activate the **Touch execution positions** property in the editor. You can find these in the properties of the recipe in the **Representation**/**Multi-Touch** nodes.

If the option is active, semi-transparent circles are placed behind the active elements in the recipe editor; the active contact areas are enlarged as a result.

8.2 Engineering rules for recipes

At the engineering the rules defined in standard ANSI/ISA-S88 are generally true.

Important principles:

GENERAL

- For all elements all connection points must be connected.
  Exception: Jump targets. Only two of the three input connection points need to be linked there.
- The **begin element** is always present only once with PFC recipes and marks the beginning of the process. It is not visible with matrix recipes.
- The **end element** is always present only once with PFC recipes and marks the end of the process. It is not visible with matrix recipes.
- Phases can be inserted anywhere. You can also place several phases in succession.
- At least one active phase should be present in the recipe.

TRANSITIONS

- Transitions only exist in PFC recipes.
- Two transitions may not lie one after the other.

BRANCHES

- Branches only exist in PFC recipes.
- The first element after a **Begin branch** must be a transition.
- The individual branches which start at **Begin branch** must all end in an **End branch** never in an **End parallel branch**. Any element can be placed between begin and end of a branch even parallel branches as long as they are closed before the **End branch** element. An end branch can be replaced with jump targets at any point, including within a parallel branch.
- It is not necessary to have an **End branch** for each **Begin branch**. You can, for example, have two **Begin branch** elements ending in one **End branch**, or the other way round.
- It is not necessary to have an **end branch** for a **begin branch**. If for example you have a **Begin branch** element with two paths and one of the paths ends in a jump target, it does not make sense to have an **End branch**.

PARALLEL BRANCHES

- Each parallel branch must contain at least one phase.
- The first element after a **Begin parallel branch** must not be a transition.
- The individual branches that start at a **Begin parallel branch** must all end in one **End parallel branch**, but must never end in an **End branch**. You may use any elements between **Begin parallel branch** and **End parallel branch** even branches as long as they are closed before the **End parallel branch**.
- Not all branches which were started in a **Begin parallel branch** must end in an **End parallel branch**. It is enough when all branches converge over an **End parallel objects**. Equally branches from different **Begin parallel branch** objects may converge in a single **End parallel branch**.
- parallel branches allows embedding of additional parallel branches.
  In doing so: each embedded parallel branch must recombine with the superordinate parallel branch

LINES IN THE PFC EDITOR

- Lines may be used as connections between any objects. It is allowed to add any number of lines after another.
- Lines must not be used to connect two equal connection points.
  For example: The both inputs of two phases must not be connected directly with a line. In the engineering this connection is allowed. It is however displayed in red (error) and in the validation (on page 249) an error message is displayed.

JUMP TARGETS

- Jump targets only exist in PFC recipes.
- Jump targets correspond to an **end branch**. They are intended to
  - jump between branches,
  - jump out of branches,
  - engineer loops

  In doing so, the following applies: A path which ends in a jump target must have started with a **Begin branch**. Otherwise the end is not reached.
- Jump targets consist of tree inputs and one output. At least two inputs and the output must always be connected. At this it makes no difference which input connection point is connected.
- Jump targets can be switched consecutively if at least two input connection points are allocated.
- Jumps are prohibited:
  - between parallel branches
  - to jump out of a parallel branch
  - to jump in a parallel branch.

CONFIGURATION OF OBJECTS

REACTIONS

- Reactions can appear in each object state.
- For each reaction type several reactions are possible. They are sorted at the triggering and are executed in accordance to their priority. At this *1* is the highest priority.
- All variables of all parameters are signed in to the driver for reading. If a value is needed for a reaction but is not yet available or invalid, the alternate value is written. The writing of the value is done without write confirmation.
- Some of the reactions are triggered only once in the process - e.g. time outs. If the phase is restarted, these reactions are also retriggered if necessary.

8.3 Recipe types

Batch Control distinguishes between the following recipe types:

- PFC recipes: For free, complex processes. These are created with the PFC editor (on page 166).

- Matrix recipes: For simple, sequential, parallel processes. They are created with the matrix editor (on page 208).

Recipes are created and edited in the Runtime.

Note: If a screen that contains recipes that have not been saved is closed, the user is asked if the recipes are to be saved. If there is no input from the user within 15 seconds, the unsaved changes are discarded and the screen is closed.

VERSIONS

The following type of recipes are available:

- **Master recipes** (on page 161):
  Form the basis for the control recipes. They are created in state *Editable*. They can no longer be modified if they are in *released* state.
- **Operations** (on page 220):
  Form a substructure that can be embedded in recipes. This can provide a better overview in complex recipes. Operation templates are created in a similar way to master recipes. Operation instances can be inserted into master recipes.
- **Host**:
  Main recipe that contains operation instances.
- **Control recipes** (on page 232):
  Are recipes which can be executed once. Each control recipe is based on a master recipe. The arrangement of the elements cannot be changed in control recipes. It is defined by the master recipe. Changeable are only command tags for which option **Changeable in control recipe** was activated.

PLACEHOLDER

Recipes can also be created with placeholders. These are only placed with actual units during operation. Placeholders are always administered as a separate list and replaced with unit instances (on page 16).

8.4 Master recipes

Master recipe are the basis of control recipes. The recipe process is defined and tested with the help of master recipes. After a master recipe is released, its content and structure can no longer be changed.

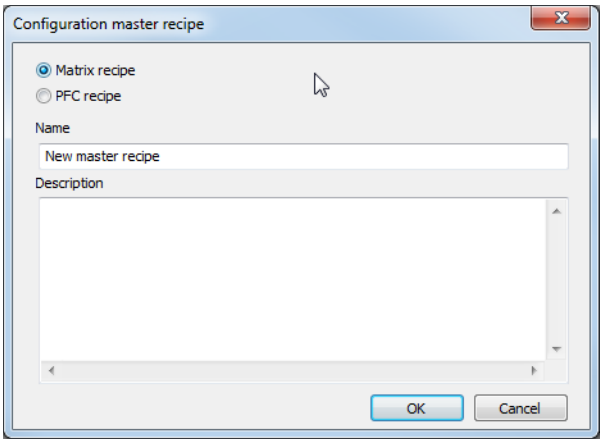
For the creation of master recipes two editors are available:

- Matrix editor
- PFC editor

Depending on the license, either just the matrix editor or both editors are available.

8.4.1 Create master recipe

A click on the **Create Master Recipe** button opens the **master recipe configuration** dialog.

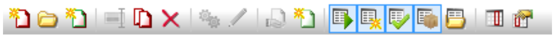


Parameters	Description
Matrix recipe (on page 207)	Activate this radio button if you want to create a matrix recipe (on page 207). **Note:** Only possible if the corresponding license is available.
PFC recipe (on page 165)	Activate this radio button if you want to create a PFC recipe (on page 165). **Note:** Only possible if the corresponding license is available.
Name	Unique name for the recipe. The name must not contain a dot (**.**), a question mark (**?**), a @ or an asterisk (*****). Maximum length: *256* characters. **Note:** When you copy a recipe the existing name is complemented with the prefix "**Copy of**". If the maximum length is exceeded by this, the name is shortened to the allowed length starting from the last character. The uniqueness is checked in the entire network. Therefore it can happen that you cannot take over the name as another user on another computer in the zenon network already has used the same name and you do not see the recipe in the list of the master recipes yet. The recipe names can be changed later but only as long as the recipe is in status *Editable*.
Description	Optional description for the recipe that is to be created.

Parameters	Description
	You can change the description later, but only as long as the recipe is in *editable* status. To change the description, select the **Configure Master Recipe** symbol.
OK	Applies all settings and created a new recipe.
Cancel	Closes the dialog without creating a recipe.

8.4.2 Toolbar and context menu for master recipe list view

TOOLBAR



Symbol	Description
New master recipe	Opens the dialog for creating a new master recipe.
Open master recipe in Editor	Opens the selected recipe in the recipe editor.
Create new version	Creates a new version (on page 217) of the selected master recipe. This must be approved or marked as obsolete.
Configure master recipe	Opens dialog to configure the selected recipe.
Duplicate master recipe	Creates a copy of the selected recipe and opens the dialog to rename the duplicate.
Delete master recipe	Deletes selected recipes.
Export selected as XML...	Exports the selected master recipe as an XML file.
Import XML	Imports the selected XML file as master recipe(s).
Switch master recipe to test mode	Switches selected recipe to test mode after requesting confirmation.
Switch master recipe to edit mode	Switches selected recipe to edit mode after requesting confirmation.
Release master recipe	Approves selected recipe after requesting confirmation.
New control recipe	Opens the dialog for creating a new control recipe.

Symbol	Description
Include running control recipes in the display	Includes running control recipes in the display.
Include prepared control recipes in the display	Includes prepared control recipes in the display.
Include finished control recipes in the display	Includes finished control recipes in the display.
Include outdated control recipes in the display	Includes outdated control recipes in the display.
Display list of attendant control recipes in control recipe list	Shows all control recipes for the selected approved control recipes.
Column selection	Opens the dialog to select the columns to be displayed.
Column format	Opens the dialog for configuring the column formats.

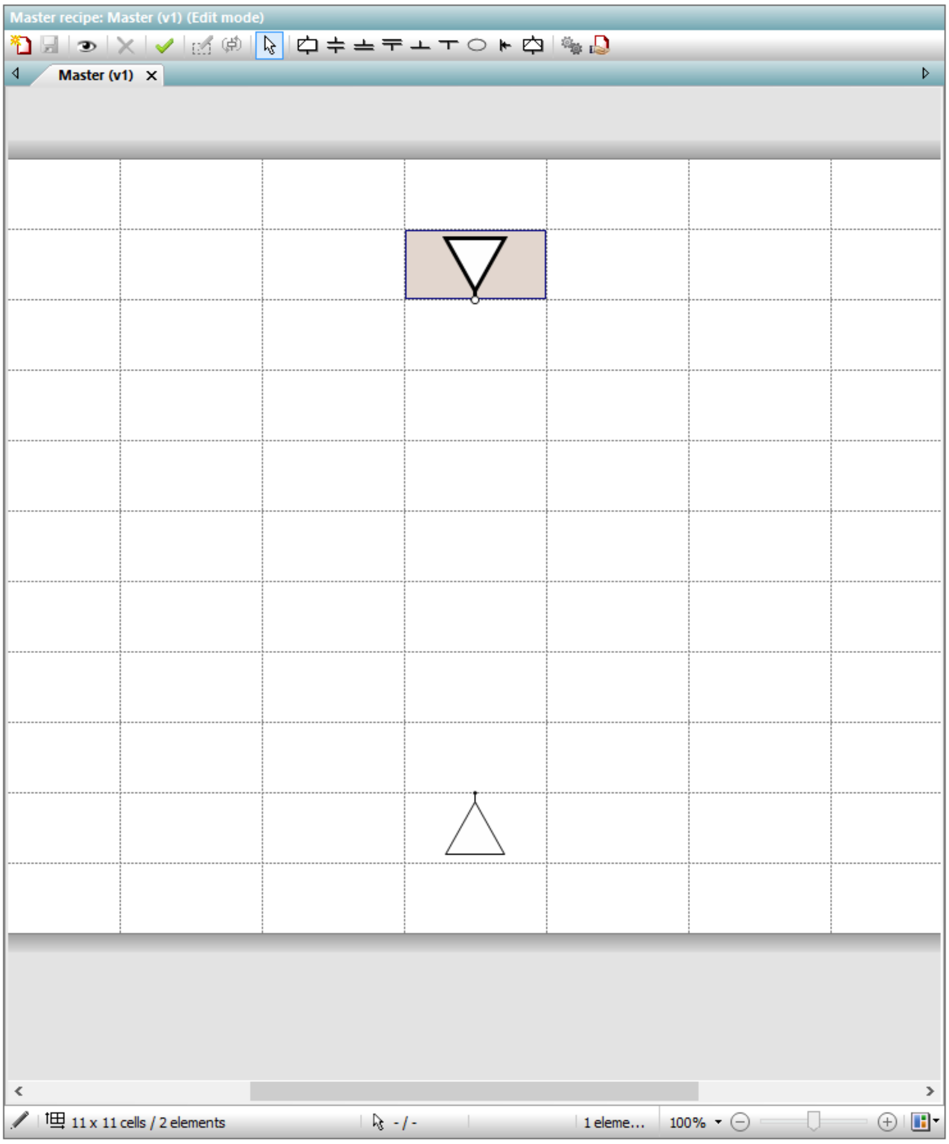
CONTEXT MENU

Command	Description
New master recipe	Opens the dialog for creating a new master recipe.
Open in Recipe Editor	Opens the selected recipe in the recipe editor.
Rename...	Opens dialog to rename the selected recipe.
Duplicate...	Creates a copy of the selected recipe and opens the dialog to rename the duplicate.
Delete	Deletes selected recipes.
Export selected as XML...	Exports the selected master recipe as an XML file.
Import XML...	Imports the selected XML file as master recipe(s).
Switch to edit mode	Switches selected recipe to edit mode after requesting confirmation.
Switch to test mode	Switches selected recipe to test mode after requesting confirmation.
Release	Approves selected recipe after requesting confirmation.
Highlight as outdated	Marks the selected recipe as outdated.

Command	Description
New control recipe	Opens the dialog for creating a new control recipe.
Show related control recipes in list	Shows all control recipes that belong to selected approved master recipes.

8.4.3 PFC recipe

If you selected PFC recipe in the **master recipe configuration** dialog and exited the dialog with OK, the newly-created recipe opens on a new tab in the PFC editor (on page 166).

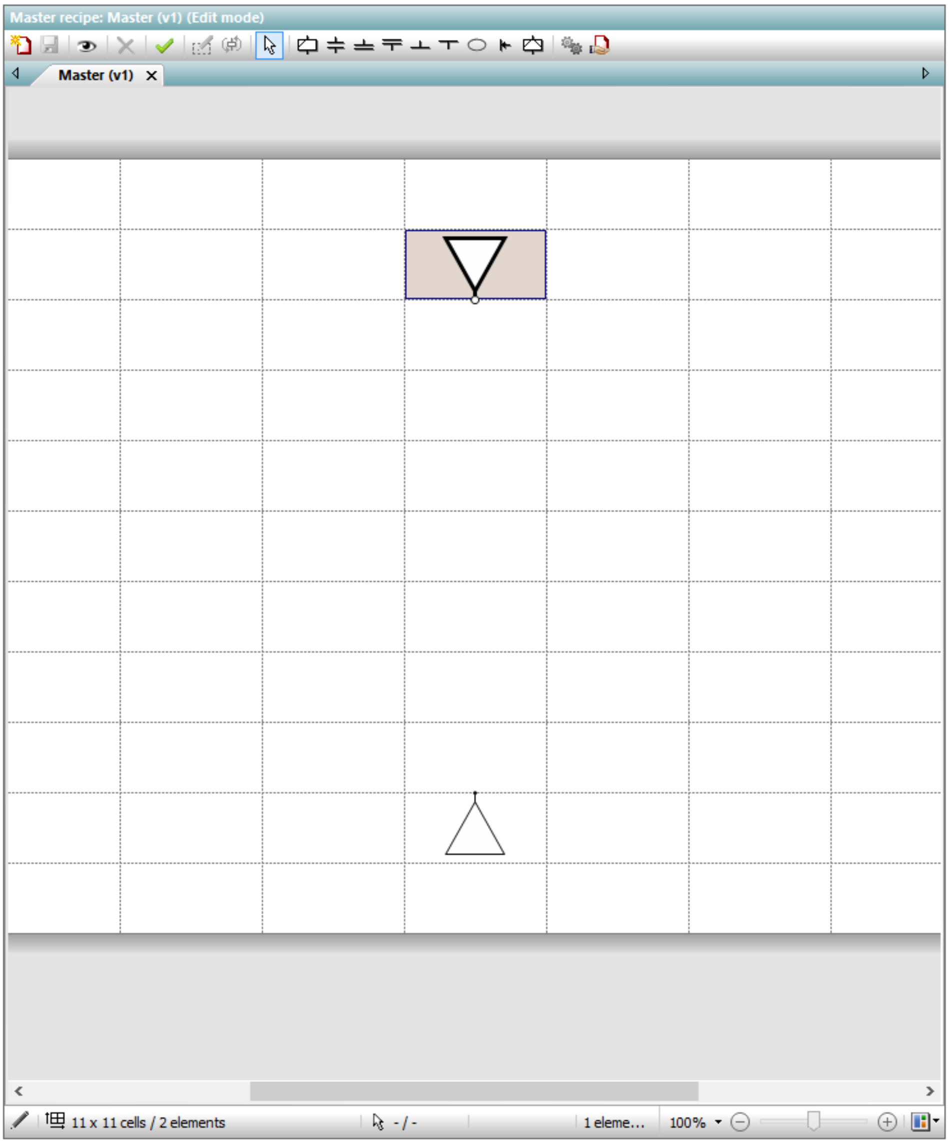

Master recipe: Master (v1) (Edit mode)
Master (v1)
11 x 11 cells / 2 elements
- / -
1 eleme...
100%

As each recipe needs a begin and an end element, these two elements already exist in the recipe and cannot be deleted from it.

Add the desired phases, transitions, branches, parallel branches and unit allocations to your recipe. Fields which are unsuitable for adding an element turn red when you move the element above it.

8.4.3.1 PFC editor

In the PFC editor you can create your recipes graphically.


Master recipe: Master (v1) (Edit mode)
Master (v1)
11 x 11 cells / 2 elements
- / -
1 eleme...
100%

TECHNICAL DETAILS

- Sheet size:
  - Default: 11 x 11 cells

  - Minimum: 5 x 5 cells
  - Maximum: 500 x 1000 cells
- Cell size
  - Default: 155 x 111 pixels
- Outside edge: 100 pixel
- Grid: is displayed by default; can also be hidden
- Scroll bar: Is displayed if the document is larger than the frame.
- Scrolling with a mouse wheel: up and down or, if you press and hold the **Shift key**, left and right.
- Zooming: **CTRL key + scroll wheel**
- Selection of elements: **left mouse click**
- Multiple selection: **CTRL key + mouse click**
- Move symbol: Click element and move it over the diagram while holding the left mouse button pressed. Content can be dropped to cells with green background. If a cell turns red when you move over it, you cannot drop the content.

8.4.3.2 Toolbar and context menu PFC recipe

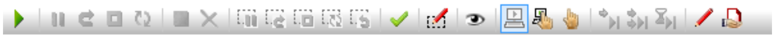
TOOLBAR PFC EDITOR: EDIT MODE

Parameter	Description
New master recipe...	Opens the dialog for creating a new master recipe.
Save master recipe	Saves the master recipe which is open for editing.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Delete	Deletes the selected elements from the diagram. Multiple selection via **Ctrl + mouse click keyboard shortcut**.
Check recipe for errors	Checks recipe for errors and displays found errors in an information window. If there are several errors, the

Parameter	Description
	first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element.
Replace phase/operation	Opens dialog to select a phase or an operation and replaces the existing phase with the newly-selected one or the existing operation with a newly-selected one. Several phases or operations can be selected and replaced together. Shortcut: **Shift+double click**
Edit mode	Toggles between insert mode and edit mode.
Insert phase	Adds a phase.
Insert transition	Adds a transition.
Insert Begin parallel branch	Adds the start of a parallel branch.
Insert End parallel branch	Adds an end of a parallel branch.
Insert Begin branch	Adds a begin branch.
Insert End branch	Adds an end branch.
Insert unit allocation	Adds a unit allocation.
Insert jump target	Adds a jump target.
Insert operation	Adds an operation (on page 220).
Switch recipe to test mode	Switches recipe to the test mode. For this the recipe must be without errors.
Release recipe	Releases the recipe. With this a control recipe can be created.

Note: Insertions remain active until you change to the edit mode using the **Edit mode** symbol, the **Esc key** or you change to another insert option via a symbol for adding a new element.

TOOLBAR PFC RECIPE TEST MODE



Parameter	Description
Start recipe	Starts the recipe process.
Recipe pausing	Pauses the recipe.
Resume recipe	Resumes paused recipe.
Hold recipe	Holds recipe.
Restart recipe	Restarts held recipe.
Recipe stopping	Stops the recipe.
Abort recipe	Aborts the recipe process.
Phase pausing	Pauses the selected phase.
Resume phase	Resumes the process of a selected paused phase.
Hold phase	Holds the selected phase.
Restart phase	Restarts the selected held phase.
Escape phase	Starts process to exit from the phase.
Check recipe for errors	Starts recipe validation (on page 249).
Edit element	Opens the corresponding dialog for editing the selected element.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Switch to automatic mode	Switches process to *automatic mode*.
Switch to semi-automatic mode	Switches process to *semi-automatic mode*.
Switch to manual mode	Switches process to *manual mode*.
Continue recipe only at selected execution position	Continues a recipe at the selected position.
Continue recipe at all execution positions	Continues a recipe on every available position.
Skip active condition	Skips a pending condition of the selected phase. Only possible in the *manual mode*.

Parameter	Description
Edit mode	Switches from test mode to edit mode.
Release recipe	Releases the recipe. With this a control recipe can be created.

PFC RECIPE TOOLBAR: APPROVED

Parameter	Description
Check recipe for errors	Checks recipe for errors and displays found errors in an information window. If there are several errors, the first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element. The dialog is opened in write-protected mode, because it is no longer possible to edit approved recipes.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Duplicate recipe	Created a copy of the recipe. The dialog for the input of a unique name and the description is opened.
Create control recipes	Creates a control recipe on the basis of the approved master recipe.

Information

The functions of the individual symbols can also be configured using buttons and thus be made touch-operable. Toolbars can therefore also be hidden (on page 149).

8.4.3.3 Begin element

First element in the process. It is automatically created in the editor when a recipe is created and it cannot be deleted.

8.4.3.4 Add and configure phase

To add a phase:

1. In the toolbar select the symbol for **Insert phase**
2. Move the mouse to the desired position.
3. Place the phase.

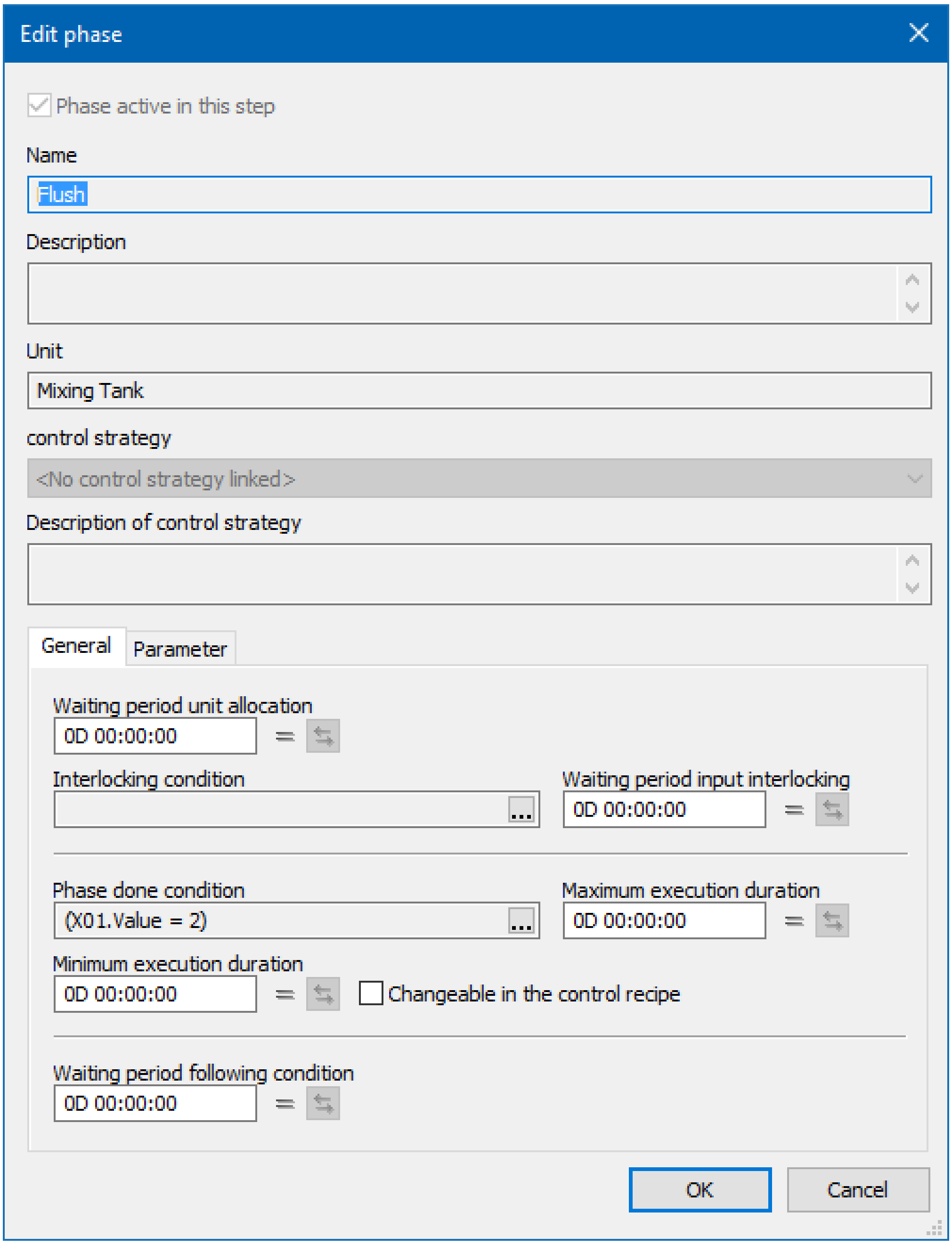
   the dialog for selecting a phase (on page 174) is opened
4. Select the desired phase.
   Phases from placeholders are always displayed in color. If a phase of a placeholder is selected, a reference to the unit placeholder is shown in the recipe.

To configure a phase:

1. Double click the phase
2. The dialog for configuration is opened
   If an element has not yet been assigned a phase, double-clicking opens the dialog to select a phase.

Note: The **shift key** plus a **double click** always opens the dialog to select a phase.

CONFIGURATION



☑ Phase active in this step

Name: Flush

Description:

Unit: Mixing Tank

control strategy: <No control strategy linked>

Description of control strategy:

Waiting period unit allocation: 0D 00:00:00

Interlocking condition:

Waiting period input interlocking: 0D 00:00:00

Phase done condition: (X01.Value = 2)

Maximum execution duration: 0D 00:00:00

Minimum execution duration: 0D 00:00:00

□ Changeable in the control recipe

Waiting period following condition: 0D 00:00:00

Parameter	Description
Phase active in this step	*Active*: Phase is active in this step
Name	Name of the phase. Display only.
Description	Comment about the phase. Display only.

Parameter	Description
Unit	Unit on which the phase is carried out. Only display.
Control strategy	Selection of a control strategy (on page 229) from a drop-down list. Only available if control strategies have been configured (on page 62) for this phase. Default: *no control strategy linked*
Control strategy description	Description of the selected control strategy. Display of the description entered in the Editor only.
General	Tab for configuration of general properties.
Parameter	Tab for configuration of parameters.
OK	Applies all changes on all tabs and closes the dialog.
Cancel	Discards all changes on all tabs and closes the dialog.

Configuration of the tabs see chapter:

- General: (on page 175) Display and configuration of the settings for the phase
- Parameter (on page 179): Configuration of the tags

The entry of a reason can be requested in order to make changes. To do this, either the **Reason for value change necessary** property must be activated in the **Edit tag** node for the module in general, or the **Reason for value change necessary** property in the **General** group for individual functions.

RULES FOR EDITING A PHASE

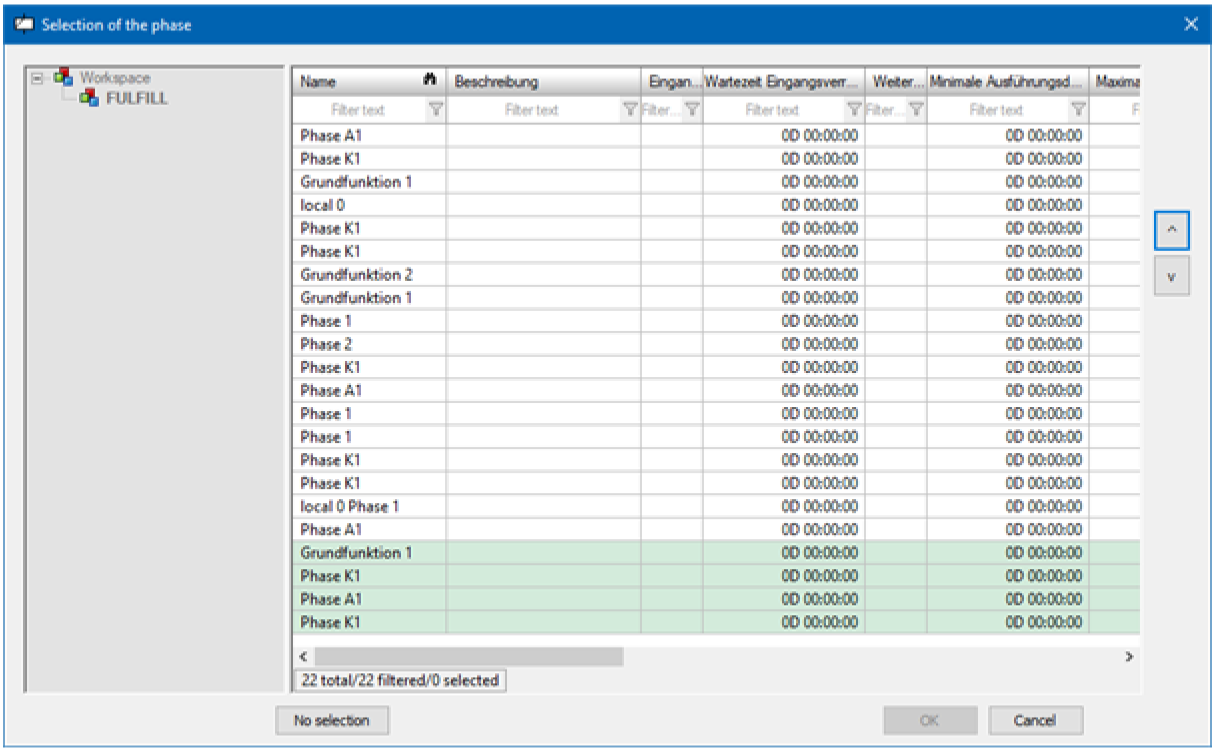
A phase can be edited:

- In a master recipe in edit mode: If the user has sufficient rights.
- In a master recipe in test mode: If the user has sufficient rights and the phase is not active.
- In a control recipe with *prepared* status: If the user has sufficient rights and the **Changeable in the control recipe** option has been activated.
- In a control recipe with *running* status: If the user has sufficient rights, the **Changeable in the control recipe** option has been activated and the phase is not active.

The phase can no longer be edited in pre-configured control recipes and in approved master recipes.

8.4.3.4.1Selection phase

If a phase is added, the dialog for selecting a phase is opened.



Name	Beschreibung	Eingan...	Wartezeit Eingangsverr...	Weiter...	Minimale Ausführungsd...	Maxima
Filter text	Filter text	Filter...	Filter text	Filter...	Filter text	F
Phase A1			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
Grundfunktion 1			0D 00:00:00		0D 00:00:00	
local 0			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
Grundfunktion 2			0D 00:00:00		0D 00:00:00	
Grundfunktion 1			0D 00:00:00		0D 00:00:00	
Phase 1			0D 00:00:00		0D 00:00:00	
Phase 2			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
Phase A1			0D 00:00:00		0D 00:00:00	
Phase 1			0D 00:00:00		0D 00:00:00	
Phase 1			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
local 0 Phase 1			0D 00:00:00		0D 00:00:00	
Phase A1			0D 00:00:00		0D 00:00:00	
Grundfunktion 1			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	
Phase A1			0D 00:00:00		0D 00:00:00	
Phase K1			0D 00:00:00		0D 00:00:00	

22 total/22 filtered/0 selected

Option	Description
Project tree	Displays the current project from which the phases can be selected.
List of phases	In the list all phases engineered in the Editor are displayed. Phases from placeholders have a colored background. The list can be filtered. The filtering is case-sensitive. Placeholders ***** and **?** can be used. The display (on page 194) and labeling (on page 195) of the columns can also be configured by right-clicking on the title line above the corresponding dialog.
Arrow keys	Move the selection up or down.
No selection	Deletes already selected phases from the element.
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

All settings of the dialog are saved user-specifically when the dialog is closed.

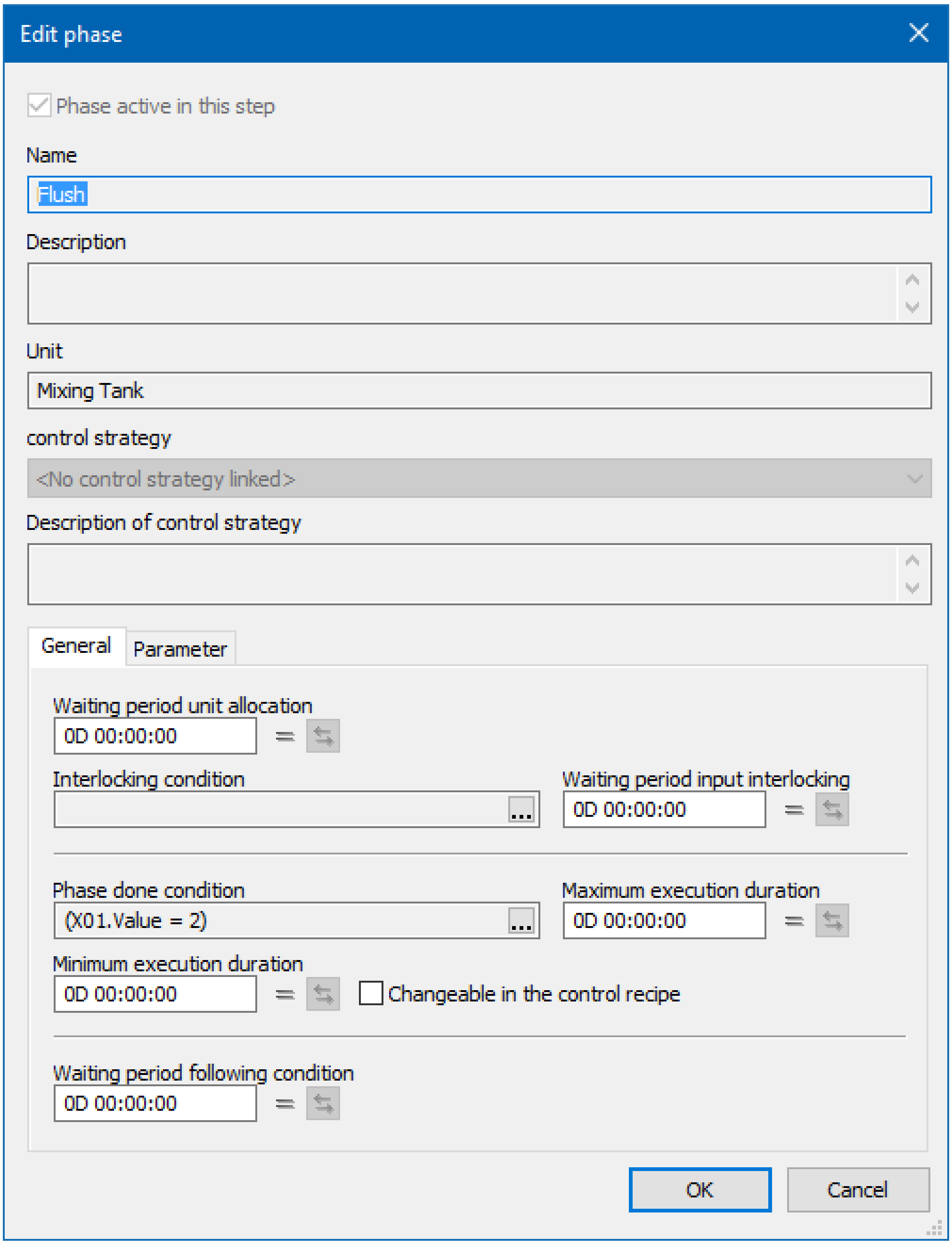
8.4.3.4.2 General

The properties of tab **General** are set in the Editor with the exception of **Changeable in the control recipe**. The values can changed in the master recipe. A symbol indicates whether the value in the dialog matches the value in the Editor. For different values you can again apply the value defined in the Editor.

Meaning of the symbols next to the values:

Symbol	Description
= Equals sign	Value in the dialog and the value in the Editor match.
<> Angle brackets	Value in the dialog and the value in the Editor do not match.
⇆ Arrows in opposite direction	Only active if the values in the recipe and Editor do not match. Click on button to apply the value from the Editor. It overrides the value in the master recipe.

EDIT PHASE



Edit phase

☑ Phase active in this step

Name: Flush

Description:

Unit: Mixing Tank

control strategy: <No control strategy linked>

Description of control strategy:

General | Parameter

Waiting period unit allocation: 0D 00:00:00

Interlocking condition:

Waiting period input interlocking: 0D 00:00:00

Phase done condition: (X01.Value = 2)

Maximum execution duration: 0D 00:00:00

Minimum execution duration: 0D 00:00:00

☐ Changeable in the control recipe

Waiting period following condition: 0D 00:00:00

For information on configuration of the basic data on these tabs, see the Add and configure phases (on page 171) chapter.

Option	Description
Waiting period unit allocation	Time in days, hours, minutes and seconds which is waited for the allocation of the unit. The waiting period can be changed in the Runtime as long as the recipe has the status *editable*.

Option	Description
	After the defined period has exceeded, the event *Waiting period unit allocation exceeded* is triggered and the element is highlighted. Additional actions must be defined by the engineer. If no further actions take place, the waiting is continued. Maximum: *9999d 11:59:59 PM* Default: *0T 00:00:00* **Note:** A unit can only be allocated by a single recipe with status *In execution* at a time. Value is predefined in the Editor and can be changed here as long as the recipe is not released.
Input lock	Defines entry interlocking. Click on button **...** to open the formula editor (on page 303) for defining the condition. If an input lock is configured, the phase is only executed in the Runtime if the condition for the input lock has been met. The formula can consist of one or more command tags and return / PLC sync tags of the phase. Value and status of the variables can be used. The formula returns *TRUE* or *FALSE* as result. The condition can be displayed in the Runtime but cannot be changed. The waiting period for the input interlocking is configured with the help of property **Input interlocking timeout**.
Waiting period input interlocking	Time period in days, hours, minutes and seconds in which the condition defined in the **Input interlocking** property must return the value *TRUE*. The waiting period begins with the checking of the entry interlocking. If the condition is not fulfilled with the waiting period, the event *Waiting period entry interlocking exceeded* is triggered and the waiting continues. If no reaction was defined on the event which forces another behavior, it is waited until the condition is fulfilled. If *0d 00:00:00* is defined as waiting period, the event is not triggered. - Minimum: *0d 00:00:00* - Maximum: *9999d 11:59:59 PM* - Default: *0T 00:00:00* Value is predefined in the Editor and can be changed here as

Option	Description
	long as the recipe is not released.
Phase done condition	Defines phase done condition. Click on the **...** Button to open the formula editor (on page 303) to define a condition. The condition can be displayed in the Runtime but not changed. The period which is waited for the fulfillment of the condition is defined via property **Maximum execution duration**.
Maximum execution duration	Time period in days, hours, minutes and seconds in which the condition defined in property **Phase done condition** must return value TRUE. The waiting period begins with the writing of the command tag. If the condition is not fulfilled within the waiting period, the event *Maximum execution period exceeded* is triggered and the waiting is continued. If no reaction was defined on the event which forces another behavior, it is waited until the condition is fulfilled. **Note:** Time keeps running when the recipe/the phase is paused. If *0d 00:00:00* is defined as waiting period, the event is not triggered. - Minimum value: *0d 00:00:00* - Maximum value: *9999d 11:59:59 PM* - Default: *0T 00:00:00* Value is predefined in the Editor and can be changed here as long as the recipe is not released. Makes only sense if property **Phase done condition** was configured. Read more about execution duration in chapter Execution duration (on page 32).
Minimum execution duration	Minimum execution duration of the phase. - *>0*: Period which is at least waited after the writing of the command tag regardless of whether the phase done condition is fulfilled. - *0*: No minimum execution duration defined. **Note:** Minimum execution duration can exceed **Maximum execution duration**.Value is predefined in the Editor and

Option	Description
	can be changed here as long as the recipe is not released. Changeable in the control recipe if option **Changeable in the control recipe** was activated in the master recipe. Read more about execution duration in chapter Execution duration (on page 32).
Changeable in the control recipe	*Active:* Property can be changed in the control recipe.
Waiting period following condition	Time period in days, hours, minutes and seconds in which the phase must be deactivated. The waiting period begins when the phase done condition is reached. If the condition is not fulfilled within the waiting period, the event *Waiting period following condition exceeded* is triggered and the waiting is continued. If no reaction was defined on the event which forces another behavior, it is waited until the condition is fulfilled. If *0d 00:00:00* is defined as waiting period, the event is not triggered. - Minimum value: *0d 00:00:00* - Maximum value: *9999d 11:59:59 PM* - Default: *0T 00:00:00* **Note:** If the following condition exists depends on the recipe structure. A waiting period should thus not be configured in the Editor but in the Runtime.Value is predefined in the Editor and can be changed here as long as the recipe is not released. For more information see chapters **Times** (on page 45) and **Following conditions** (on page 273).
OK	Applies all changes on all tabs and closes the dialog.
Cancel	Discards all changes on all tabs and closes the dialog.

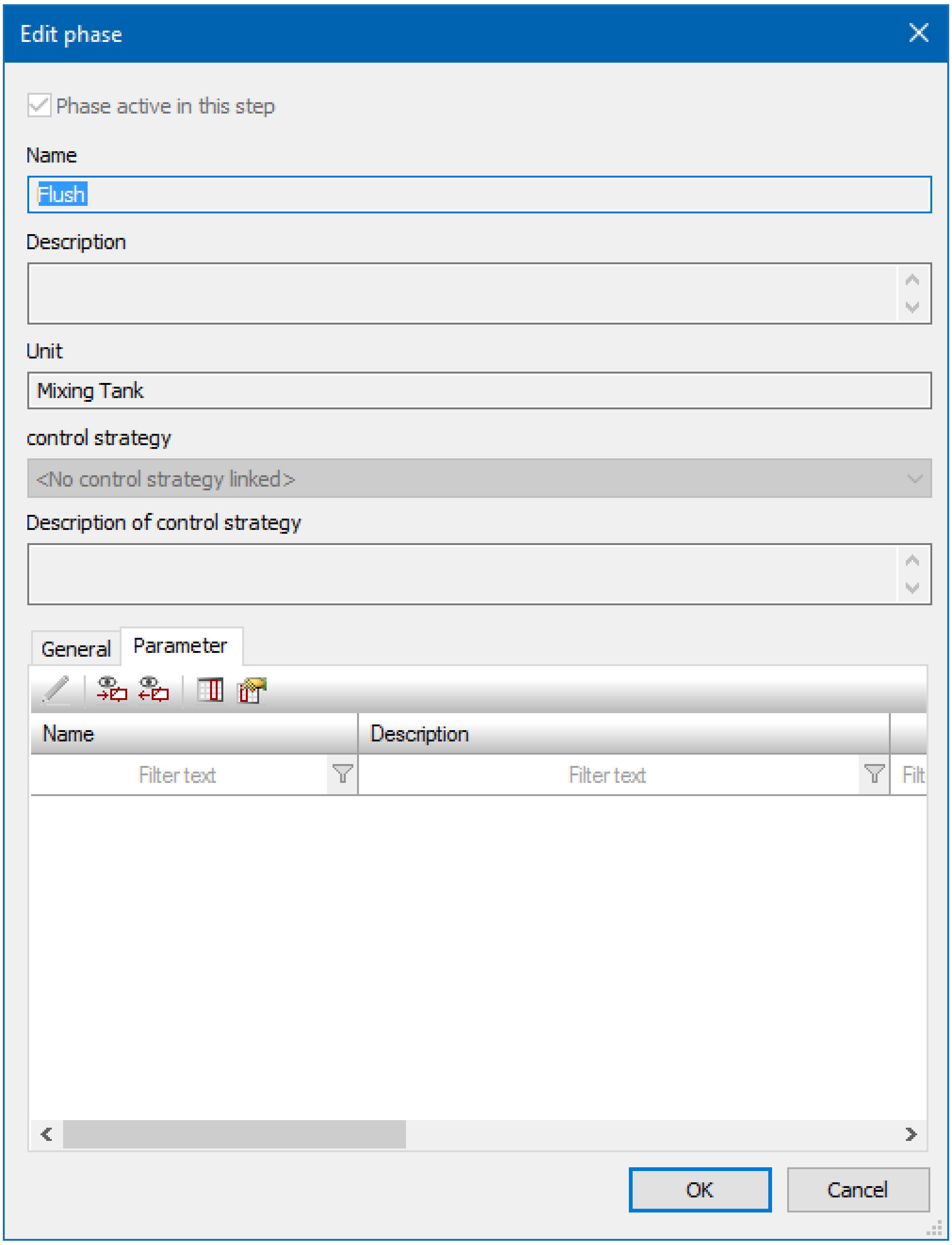
8.4.3.4.3 Parameter

This tab allows the editing of the parameters of a phase that have been approved for this.

Information

Tags can also be changed during the execution of a phase.

PARAMETER DIALOG



Edit phase

☑ Phase active in this step

Name: Flush

Description:

Unit: Mixing Tank

control strategy: <No control strategy linked>

Description of control strategy:

Parameter	Description
TAG list	Displays the tag configured in the Editor. Tags can be

Parameter	Description
	filtered and sorted according to columns. Click on the **Edit tag** symbol, double-click the tag, menu item in the context menu or press the **Enter** key to open the dialog (on page 181) to edit a tag.
OK	Applies all changes on all tabs and closes the dialog.
Cancel	Discards all changes on all tabs and closes the dialog.

TOOLBAR

Symbol	Meaning
Edit TAG	Opens the dialog (on page 181) for editing the tag.
Display all command TAGs	Toggles between the display of the changeable tags and all tags.
Show return/PLC sync TAG	In addition to the command tags, also displays the return / PLC sync parameter tags or hides them.
Column selection	Opens the dialog (on page 194) for selecting the columns which should be displayed.
Column format	Opens dialog (on page 195) to format the columns.

8.4.3.4.4 Edit tag

To edit a tag in the Runtime:

1. In the **edit phase** dialog, select the **Tag** tab.
2. Highlight the desired tag
3. Open the dialog to edit the tag by clicking on the **Edit tag** symbol, the context menu, double-clicking on the tag or pressing the **Enter** key.

   The dialog for editing is opened.

For each data type an own dialog is opened:

- Numerical
- Binary

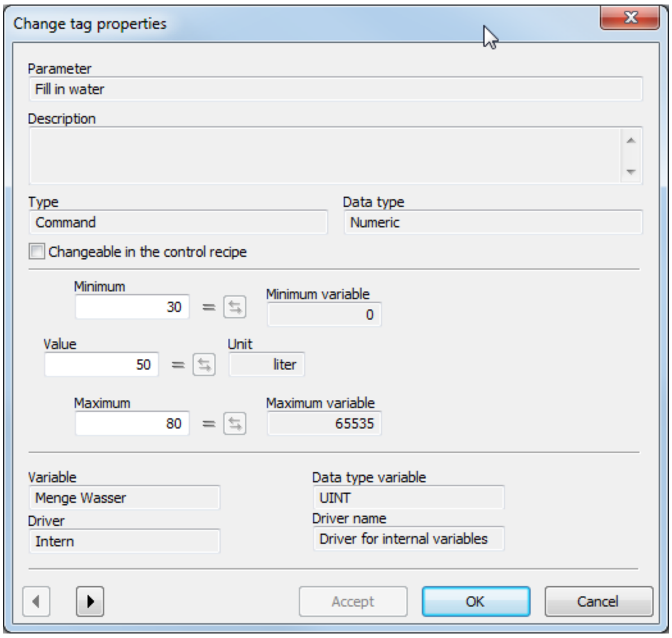
- String
- Time period

Depending on the configuration in the Editor, the values displayed in the dialog can be edited or only read. In order to be able to amend tag values the **Parameter value can be changed in master recipe** property must be activated in the Editor (on page 32).

Notes:

- In order to define tag values using variables (on page 191), the **Placeholder for tag value variable** property must be defined in the Editor (on page 32). In this case, no values are displayed in the Runtime in recipes in edit mode. Instead, an option to select a variable is offered. This variable determines the parameter value.
- In order for parameters to be able to edited in an active phase, the **Tag values changeable during phase execution** property must be activated in the Editor.
- Amended values are sent to the Runtime in bundled form or immediately (on page 276).

NUMERIC TAGS



Change tag properties

Parameter: Fill in water

Description:

Type: Command

Data type: Numeric

□ Changeable in the control recipe

Minimum: 30

Minimum variable: 0

Value: 50

Unit: liter

Maximum: 80

Maximum variable: 65535

Variable: Menge Wasser

Data type variable: UINT

Driver: Intern

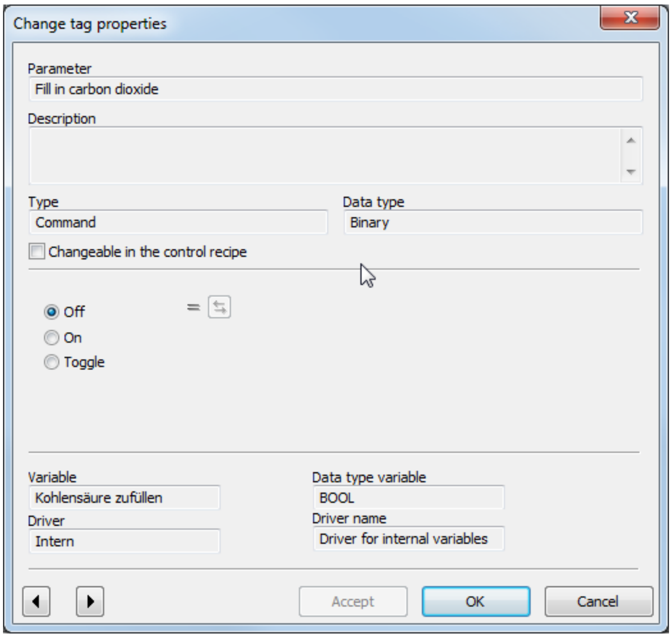
Driver name: Driver for internal variables

Parameter	Meaning
Parameter	TAG name Display only.
Description	Free description of the tag. Display only.
Type	Type of the tag: *Initial, value* or *return*. Display only.
Data type	Data type of the tag. Display only.

Parameter	Meaning
Changeable in the control recipe	*Active:* Value of the property can be changed in the control recipe. Only available if: - The parameter is an *initial* or *value* parameter - it was configured in the Editor that the tag can be changed in the Batch recipe (property: **Parameter value can be changed in master recipe**).
Minimum	Minimum value for the tag. Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters.
Minimum variable	Allowed minimum value of the variable.
Value	Value of the tag. Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters. With configuration via a variable, an option to select a variable (on page 191) is displayed instead of a variable.
Measuring unit	Unit of the value.
Maximum	Maximum value for the tag. Display only.

Parameter	Meaning
	Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters.
Maximum variable	Allowed maximum value of the variable. Display only.
Variable	Variable which is linked to the tag. Display only.
Variable Data Type	Data type of the variable. Display only.
Driver	Driver of the variable. Display only.
Driver name	Description of the driver of the variable. Display only.
Arrow keys	Navigating through the tags. They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping. If changes were done, there is a prompt before you can change to another tag whether the changes should be applied or discarded. If changes should be applied, the input is checked before advancing.
Apply	Applies all changes if the check of the changes was successful. The dialog remains open for further editing.
OK	Applies all changes and closes the dialog if the check of the changes was successful.
Cancel	Discards all changes which have not been taken over yet and closes the dialog.

BINARY TAG



Parameter: Fill in carbon dioxide

Description:

Type: Command

Data type: Binary

□ Changeable in the control recipe

● Off

○ On

○ Toggle

Variable: Kohlensäure zufüllen

Data type variable: BOOL

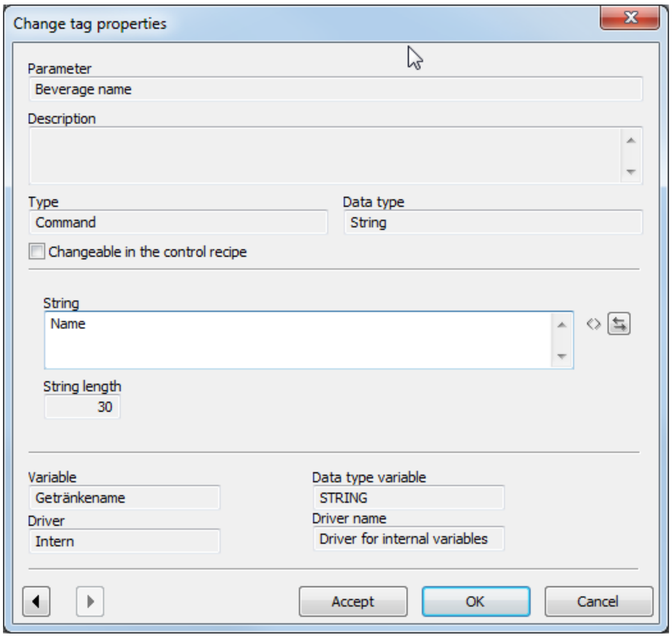
Driver: Intern

Driver name: Driver for internal variables

Parameter	Meaning
Parameter	TAG name Display only.
Description	Free description of the tag. Display only.
Type	Type of the tag: *Initial, value* or *return*. Display only.
Data type	Data type of the tag. Display only.
Changeable in the control recipe	*Active:* Value of the property can be changed in the control recipe. Only available if: - The parameter is an *initial* or *value* parameter - it was configured in the Editor that the tag can be changed in the Batch recipe (property: **Parameter value can be changed in master recipe**).
Off	Condition: Off. Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with

Parameter	Meaning
	the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters. With configuration via a variable, an option to select a variable (on page 191) is displayed instead of a variable.
On	Condition: On.
Toggle	Toggles between the states.
Variable	Variable which is linked to the tag. Display only.
Variable Data Type	Data type of the variable. Display only.
Driver	Driver of the variable. Display only.
Driver name	Description of the driver of the variable. Display only.
Arrow keys	Navigating through the tags. They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping. If changes were done, there is a prompt before you can change to another tag whether the changes should be applied or discarded. If changes should be applied, the input is checked before advancing.
Apply	Applies all changes if the check of the changes was successful. The dialog remains open for further editing.
OK	Applies all changes and closes the dialog if the check of the changes was successful.
Cancel	Discards all changes which have not been taken over yet and closes the dialog.

STRING TAG



Change tag properties

Parameter: Beverage name

Description:

Type: Command

Data type: String

☐ Changeable in the control recipe

String: Name

String length: 30

Variable: Getränkename

Data type variable: STRING

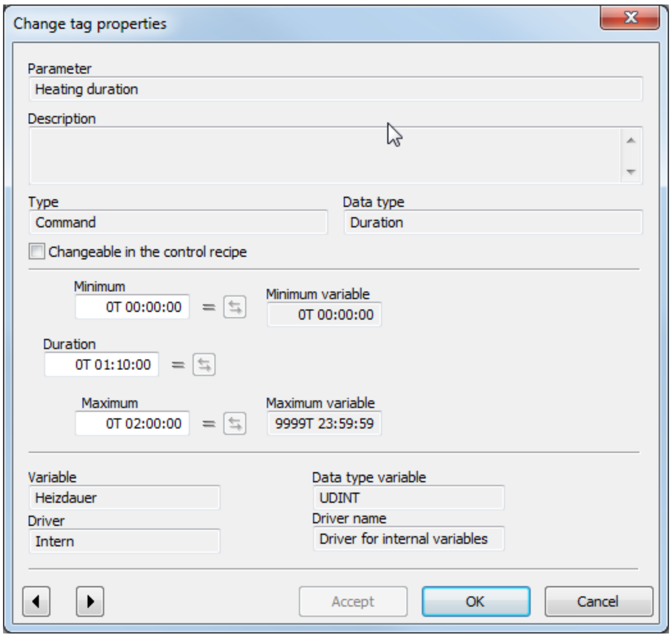
Driver: Intern

Driver name: Driver for internal variables

Parameter	Meaning
Parameter	TAG name Display only.
Description	Free description of the tag. Display only.
Type	Type of the tag: *Initial, value* or *return*. Display only.
Data type	Data type of the tag. Display only.
Changeable in the control recipe	*Active:* Value of the property can be changed in the control recipe. Only available if: - The parameter is an *initial* or *value* parameter - it was configured in the Editor that the tag can be changed in the Batch recipe (property: **Parameter value can be changed in master recipe**).
String	Alphanumeric character string. Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with

Parameter	Meaning
	the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters. With configuration via a variable, an option to select a variable (on page 191) is displayed instead of a variable. Possible length is limited by the **String length** engineered in the variable.
String length	Defines possible length of the string. Display only.
Variable	Variable which is linked to the tag. Display only.
Variable Data Type	Data type of the variable. Display only.
Driver	Driver of the variable. Display only.
Driver name	Description of the driver of the variable. Display only.
Arrow keys	Navigating through the tags. They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping. If changes were done, there is a prompt before you can change to another tag whether the changes should be applied or discarded. If changes should be applied, the input is checked before advancing.
Apply	Applies all changes if the check of the changes was successful. The dialog remains open for further editing.
OK	Applies all changes and closes the dialog if the check of the changes was successful.
Cancel	Discards all changes which have not been taken over yet and closes the dialog.

TIME PERIOD



Change tag properties

Parameter
Heating duration

Description

Type
Command

Data type
Duration

☐ Changeable in the control recipe

Minimum
0T 00:00:00

Minimum variable
0T 00:00:00

Duration
0T 01:10:00

Maximum
0T 02:00:00

Maximum variable
9999T 23:59:59

Variable
Heizdauer

Data type variable
UDINT

Driver
Intern

Driver name
Driver for internal variables

Parameter	Meaning
Parameter	TAG name Display only.
Description	Free description of the tag. Display only.
Type	Type of the tag: *Initial, value* or *return*. Display only.
Data type	Data type of the tag. Display only.
Changeable in the control recipe	*Active:* Value of the property can be changed in the control recipe. Only available if: - The parameter is an *initial* or *value* parameter - it was configured in the Editor that the tag can be changed in the Batch recipe (property: **Parameter value can be changed in master recipe**).
Minimum	Minimum value for the tag in format: **D hh:mm:ss**. Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values

Parameter	Meaning
	can be overwritten with the default from the Editor with the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters.
Minimum variable	Allowed minimum value of the variable.
Duration	Value of the parameter in the format: **D hh:mm:ss.** Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters. With configuration via a variable, an option to select a variable (on page 191) is displayed instead of a variable.
Measuring unit	Unit of the value.
Maximum	Maximum value for the tag in format: **D hh:mm:ss**. Default from the Editor can be changed when the **Parameter value can be changed in master recipe** property was activated in the Editor (on page 32). If the value is changed, it is marked with the symbol (on page 175) on the right side of the value. Changed values can be overwritten with the default from the Editor with the help of the button right next to it. For control strategies, synchronization takes place with the values that were set in the control strategy when linking parameters.
Maximum variable	Allowed minimum value of the variable. Display only.
Variable	Variable which is linked to the tag. Display only.
Variable Data Type	Data type of the variable. Display only.

Parameter	Meaning
Driver	Driver of the variable. Display only.
Driver name	Description of the driver of the variable. Display only.
Arrow keys	Navigating through the tags. They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping. If changes were done, there is a prompt before you can change to another tag whether the changes should be applied or discarded. If changes should be applied, the input is checked before advancing.
Apply	Applies all changes if the check of the changes was successful. The dialog remains open for further editing.
OK	Applies all changes and closes the dialog if the check of the changes was successful.
Cancel	Discards all changes which have not been taken over yet and closes the dialog.

8.4.3.4.5 Define parameter value via variable

Parameter values can also be read in using variables. The variables that are available in the Runtime are configured in the Editor (on page 32) using the **Placeholder for tag value variable** property.

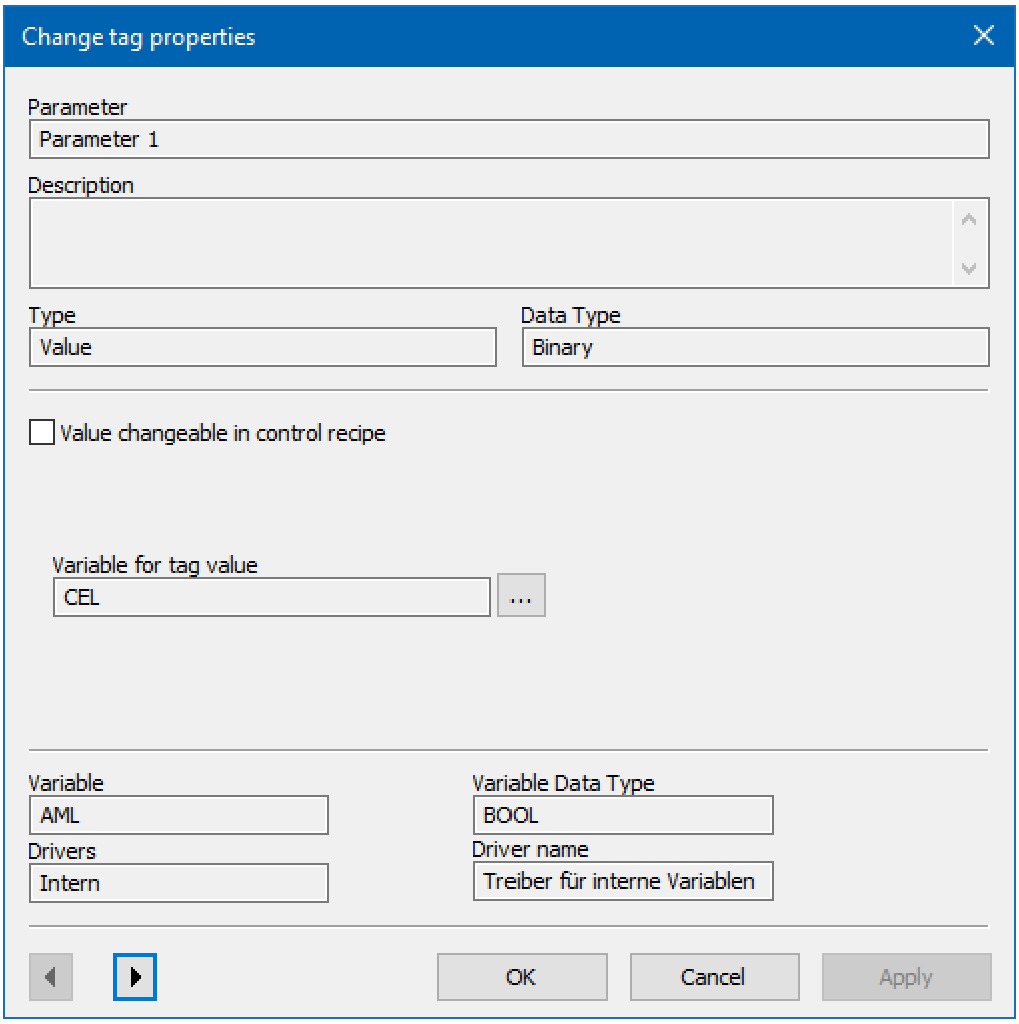
To configure parameter values using placeholders:

1. Go to the **Placeholder for tag value variable** property in the parameter properties in the Editor.
2. Enter a filter text into the field.
   In doing so, you can use the die wildcards ***** and **?**.
3. Assign a variable to the parameter when configuring the master recipe.
   This determines the value of the parameter.

To select a variable for the parameter value:

1. Open the phase to be edited in the Runtime.
2. Open the desired parameters to edit there.

Instead of the parameter values, an option to select the variable for the parameter value is displayed.



Change tag properties

Parameter: Parameter 1

Description:

Type: Value

Data Type: Binary

□ Value changeable in control recipe

Variable for tag value: CEL

Variable: AML

Variable Data Type: BOOL

Drivers: Intern

Driver name: Treiber für interne Variablen

3. Please select a variable.
   To do this, click on the **...** button.
   The variable selection dialog is opened
   Only variables that correspond to the filter defined in the **Placeholder for tag value variable** property are offered.
4. Close all open dialogs.

ACTION OF VALUES FROM VARIABLES

The parameter values are read from the configured variables and written as values in the parameter when:

- Switching a master recipe to test mode
- Creating a master recipe

If values are written, they can be amended in the control recipe like manually-set values. The variable is no longer shown in the dialogs.

> ⚠Attention
>
> - The value supplied by the variable must be within the defined limit value.
> - The values of the linked variables must not change when the control recipe is created.
> - The data type of the variables must not be changed as long as the recipe is in test mode.

8.4.3.4.6 Change parameter values using the keyboard

Values for **numerical parameters** and **string parameters** can only be amended in the parameter list directly using a keyboard screen.

Configuration in the Editor:

1. Link the parameter variable to a reaction matrix.
   Numerical variables and **string variables** are permitted.

   The reaction matrix must be the same type as the variable.
   For each value, you need a status with:

   a) **Value** *same*

   a) Limit value text

2. Create a keyboard for the statuses configured in the reaction matrix with control elements for:
   - **Display value as text**
   - **Set value input**
   - **OK**
3. Link the keyboard to the parameter.

Application in Runtime:

1. Click on the cell for the value in the parameter list.

   The keyboard is opened.

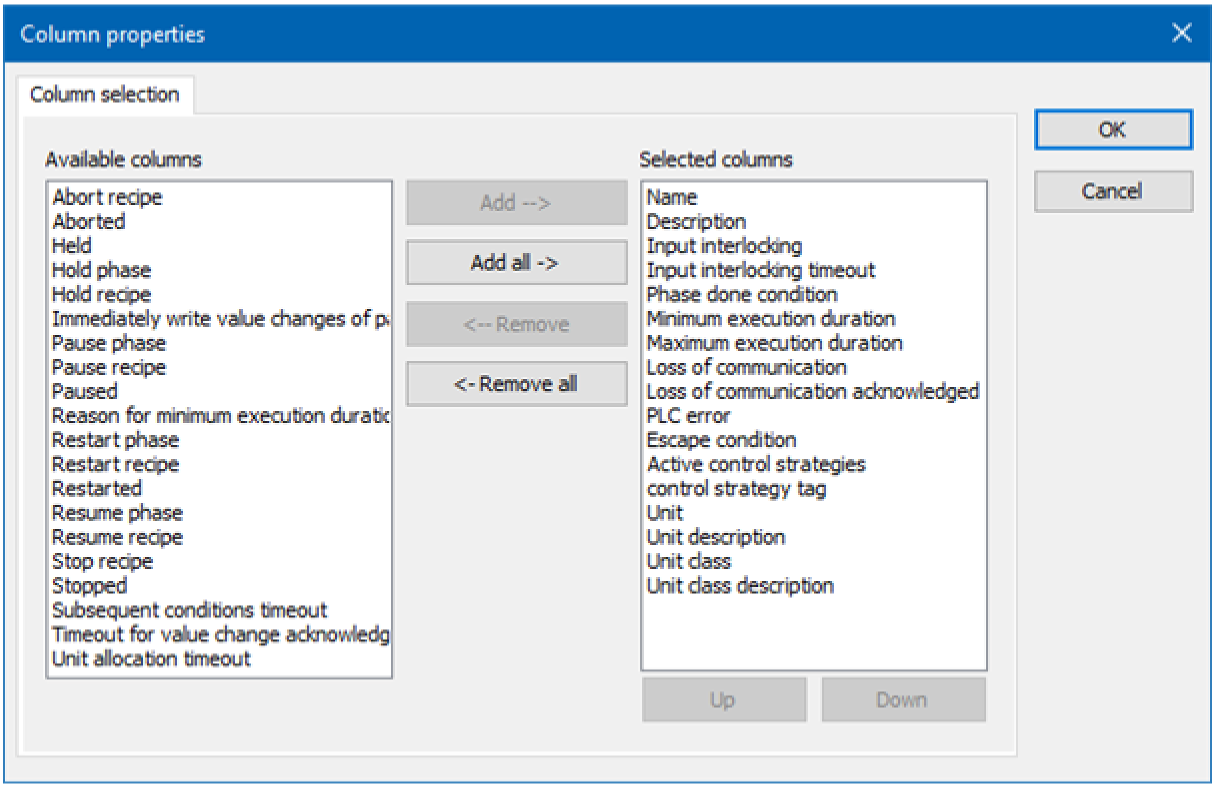
2. Click on the desired condition.

   The value is displayed in the **set value input**.

3. Click on **OK**.

   The value is written to the parameters.

8.4.3.4.7 Column selection



Available columns

- Abort recipe
- Aborted
- Held
- Hold phase
- Hold recipe
- Immediately write value changes of p
- Pause phase
- Pause recipe
- Paused
- Reason for minimum execution duratic
- Restart phase
- Restart recipe
- Restarted
- Resume phase
- Resume recipe
- Stop recipe
- Stopped
- Subsequent conditions timeout
- Timeout for value change acknowledg
- Unit allocation timeout

Selected columns

- Name
- Description
- Input interlocking
- Input interlocking timeout
- Phase done condition
- Minimum execution duration
- Maximum execution duration
- Loss of communication
- Loss of communication acknowledged
- PLC error
- Escape condition
- Active control strategies
- control strategy tag
- Unit
- Unit description
- Unit class
- Unit class description

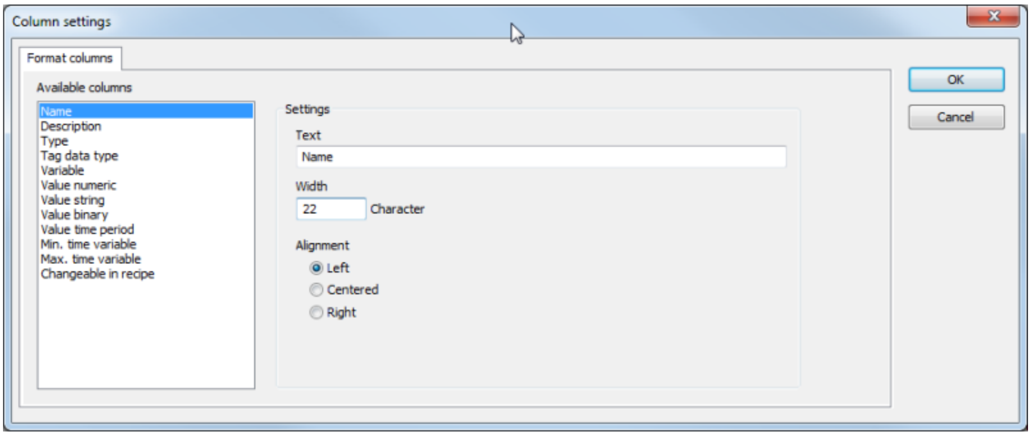
Option	Function
Available columns	List of columns that can be displayed in the table.
Selected columns	Columns that are displayed in the table.
Add ->	Moves the selected column from the available ones to the selected items. After you confirm the dialog with OK, they are shown in the detail view.
Add all ->	Moves all available columns to the selected columns.
<- Remove	Removes the marked columns from the selected items and shows them in the list of available columns. After you confirm the dialog with OK, they are removed from the detail view.
<- Remove all	All columns are removed from the list of the selected columns.
Up	Moves the selected entry upward. This function is only available for unique entries, multiple selection is not possible.

Option	Function
Down	Moves the selected entry downward. This function is only available for unique entries, multiple selection is not possible.

CLOSE DIALOG

Options	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.
Help	Opens online help.

8.4.3.4.8 Column format



Available columns

- Name
- Description
- Type
- Tag data type
- Variable
- Value numeric
- Value string
- Value binary
- Value time period
- Min. time variable
- Max. time variable
- Changeable in recipe

Settings

Text: Name

Width: 22 Character

Alignment: ● Left ○ Centered ○ Right

Parameter	Description
Available columns	List of the available columns via **Column selection**. The column selected here is configured using the settings in the **Parameters** section.
Parameter	Settings for selected column.
Label	Name for column title. The column title is online language switchable. To do this, you must enter the **@** character in front of the name.
Width	Width of the column in characters.
Alignment	Alignment.

Parameter	Description
	Possible settings: - Left: Text is justified on the left edge of the column. - Centered: Text is displayed centered in the column. - Right: Text is justified on the right edge of the column.
OK	Applies settings and closes the dialog.
Cancel	Discards settings and closes the dialog.

8.4.3.5 Insert operations

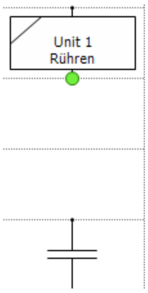
To insert an operation:

1. Select the symbol for **Insert Operation** in the toolbar.
2. Move the mouse to the desired position
3. Place the operation.
4. The dialog to select a template (on page 226) for the operation is opened.
5. Select the desired operation.
6. The operation is inserted.

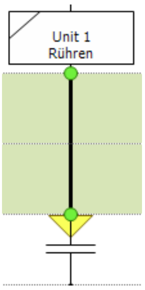
8.4.3.6 Lines

Lines connect elements via free connection points. To connect connection points with each other:

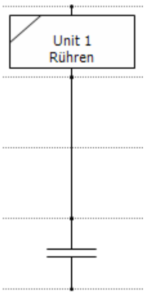
1. Activate a point with the help of the mouse:
   The connection point turns green. Red means that the connection point is already taken.



2. Drag a line to another connection point:
   A yellow arrow shows the direction of the line.. Green fields can be crossed. Red fields may not be crossed by the line.



3. As soon as the yellow tip of the line touches the next connection point, the line is created.



LINE HANDLING

Lines:

- are dragged with the mouse
- Can be moved (press and hold **Ctrl** key)
  In doing so, all existing connections are separated and an attempt is made to reconnect the line if there are objects with connection points in the right direction at the target.
  If several lines are highlighted, the line that has a cell with mouse cursor in it is moved.
- Can be deleted by highlighting them and pressing the **Del** key.
- are deleted when re-dragging them from beginning to end
- have a tool tip displaying its ID.

If a line reaches a connection point of an object, the connection point becomes active. If a connection is possible, it turns green otherwise red. Connections connecting two connections points of the same type - two inputs, two outputs, etc. - are not allowed. The line can be added in any case. Not allowed connections are displayed in red and trigger a corresponding error message at testing.

The connection points of the elements are always displayed in the edit mode even if the connection point in question is connected. In status "*Release*" no connection points are displayed.

Properties connection point:

- connected: highlighted red; connection is separated when the line is dragged and a new connection point can be chosen
- open: highlighted green; at dragging a new line is created

8.4.3.7Transitions

Transitions are used after phases in order to ensure a defined transition from one phase to another. Transitions display their internal state during the process and inform via a tool tip about status and process duration. For details on configuring the condition of transitions, see also the Engineering in the Editor (on page 15) section in the conditions (on page 40) chapter.

To add a transition:

1. In the toolbar, select the symbol for **Insert Phase**.
2. Move the mouse to the desired position.
3. Place the transition.

   The formula editor is opened.
4. Select one or more parameters (on page 308) and configure the formula (on page 303).

To configure a transition:

1. Double click on the transition.

   The dialog for configuration is opened

PARAMETERS IN TRANSITIONS

All parameters of all phases are available for the condition of a transition. The origin of the parameters is visible in the parameter selection dialog. The name of the phase and the unit type (*unit*, *placeholder* or *placeholder group*) and the unit name is displayed in the list for this.

The following is applicable for parameters:

- The corresponding phase must be used at least once in the recipe.
- The following is displayed if a configured parameter is no longer available: **<Invalid TAG>**.
- Phases from partial recipes are not taken into account.
- Each parameter can only be selected once.

PARAMETERS OF PLACEHOLDER PHASES

The following is also applicable for parameters from placeholder phases:

- Only parameters of the unit classes are available. Local parameters of the unit instances are not offered.
- With placeholders and placeholder groups, only the group is displayed, individual placeholders are not available.
- In the selection list, the placeholder name or group name is displayed instead of the unit name.
- Linked parameters from placeholders are displayed in the formula editor as: **Placeholder.[placeholder name].[phase name].[TAG name]**"
  If the TAG is part of a placeholder group, the following is shown: **Placeholder group.[group name].[phase name].[parameter name]**.

COPY CONDITIONS

Transition conditions can be copied between different transitions. To do this, use the **Copy** and **Paste** buttons in the formula editor.

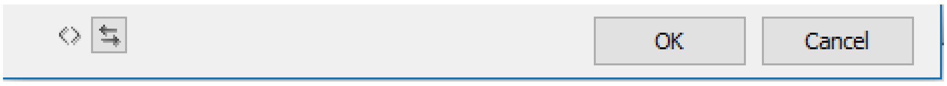
Note: The relationship to placeholders and placeholder groups is established using placeholder names. If there is a placeholder with the same name as in the source recipe in the target recipe, the parameter from the target recipe is used.

CHANGE FORMULA IN OPERATION INSTANCE

Formulas for transitions can be edited in the operation instance. This is possible as long as the master recipe of the instance is in editing mode.

A change to the operation instance disconnects the connection between template and instance. Different conditions can thus be used in the template and the instances. The local condition can be reset to the condition of the template for each instance. This resetting also links instance and template together again.

In order to reset the condition of the instance to the condition of the template, click on the reset button in the formula editor in the footer of the dialog.



VALIDATION

The following applies for validation:

- Connection exists:
  - Errors in the template are also displayed in the instance.
- Connection disconnected:
  - Errors in the instance are only displayed in the instance.

- Errors in the template are only shown in the template.

8.4.3.8 Branches and parallel branches

Recipes can branch (on page 201) and run in simultaneous sequences (on page 202).

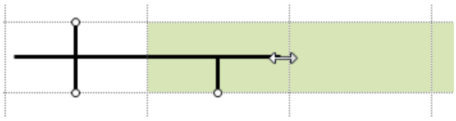
Branches and simultaneous sequences consist of:

- Single or double horizontal lines
- Connection pieces (consisting of connection line and connection point)

CREATE A BRANCH

To create a branch:

1. Select the symbol **Insert begin branch**.
2. Put the branch on the desired location.
3. Connect the input connection point to an output connection point of the preceding object.
4. Connect both output connection points with the desired following objects.
5. Close a branch with the object **Insert end branch**

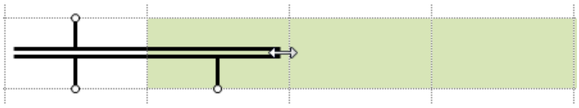


INSERT PARALLEL BRANCHES

To create a parallel branch:

1. Select the symbol **Insert begin parallel branch**
2. Put the branch on the desired location.
3. Connect the input connection point to an output connection point of the preceding object.

4. Connect both output connection points with the desired following objects.
5. Close a parallel branch with the object **Insert end parallel branch**



MODIFY AND MOVE

Branches and parallel branches can be moved and changed in size.

MOVE

To move an object:

1. Click on the object.
2. Hold down the mouse button.
3. Move the object to the desired position.

CHANGE SIZE

Begin/End branch/parallel branch objects can be extended and shortened. To change their size:

1. Move the mouse pointer to the end of the object until it turns into a double arrow.
2. Hold down the left mouse button and move it in the desired direction:
   - away from the object to extend it
   - into the object to shorten it
3. When extending a new connection piece is added:
   - all fields which are concerned by the extension are marked green
   - to add several need connection pieces the process must be repeated
4. All corresponding connection pieces are deleted during shortening.

8.4.3.8.1Branches

A branch offers the possibility to execute one of several possible ways. For this it is necessary that the first element at the beginning of a branch is a transition. This means that **Begin branch** can only be followed by a transition (on page 40).

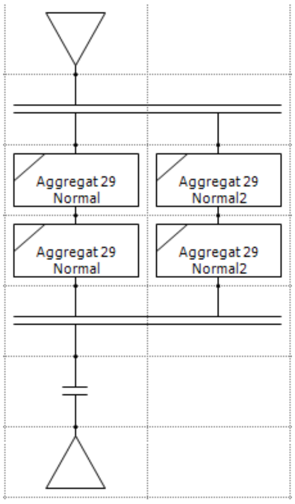
Procedure:

- The path is chosen for which the transition is *TRUE* first.

- If several transitions are *TRUE* at the same time, the leftmost path for which the transition is *TRUE* is always selected.

For begin and end the following is true: If there is a phase in front of the element and a transition behind, the phase remains active until the transition was completed.

In a branch the objects are processed sequentially. Each branch processes its objects independent of other branches.



8.4.3.8.2 Parallel branches

At the parallel branch an execution path parts into several execution paths which are executed in parallel during the process. In the process the intermediate area of the **end parallel branch** is also colored. The color matches the coloring (on page 255) of the phase.

End parallel branch is displayed as active as soon as the first of several previous elements has been completed. There is a wait until all previous elements have been completed. **End parallel branch** is then shown as inactive and phase done.

The **End parallel branch** element is marked as active for:

- Transitions: Immediately after passing.
- Phases: If the phase done condition has been met and the minimum execution duration has expired.

During the execution the status is color-coded.

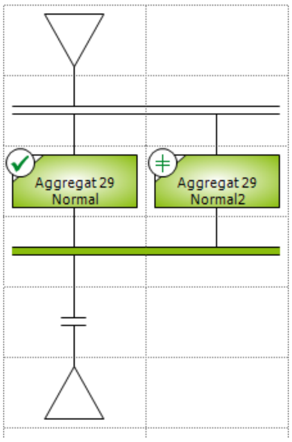
8.4.3.8.3 Split up and combine branches

For parallel branches, the execution splits up a **Begin parallel branch** and combines the single branches again at **End parallel branch**. The paths of the parallel objects are independent branches. Only at **End simultaneous sequence** all branches are synchronized.

The possible branches are defined when the recipe is created. It is evident from the recipe execution in Runtime if the separate branches are allocated or unlocked. A branch is active as long as an object on it is active.

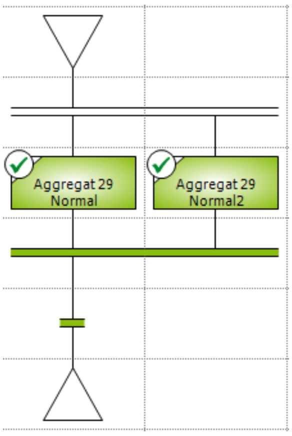
END PARALLEL BRANCH

An **End parallel branch** combines the branches of the connected paths. The object after **End simultaneous sequence** is activated when all paths reaches **End simultaneous sequence** with their process.



The left path is ready. Therefore **End simultaneous sequence** is already active. The transition after **End simultaneous sequence** is not yet active as the right branch is not yet completed.

When the right branch is also completed:



The transition after **End simultaneous sequence** is activated. All objects which were active before are still active.

8.4.3.9Jump targets

Jump targets make it possible to

- jump between branches
- To jump out of branches
- engineer loops

Jump targets consist of tree inputs and one output. At this the output is always at the bottom and the inputs are located at the top and the sides. You can connect any input connection points. A path which ends in a jump target must have started with a **Begin branch**. Otherwise the end is not reached. Jump targets are not allowed for parallel branches.

During the editing all connection points are visible. In the checking mode only the connection points which are connected are displayed.

8.4.3.10 Unit allocations

To allocate a unit:

1. In the toolbar select the symbol for **Insert unit allocation**
2. Move the mouse to the desired position
3. Place the **Unit allocation** element.
4. The element is inserted.

To configure the unit allocation:

1. Double-click on the **Unit allocation** element.

   The dialog for configuration is opened

CONFIGURATION

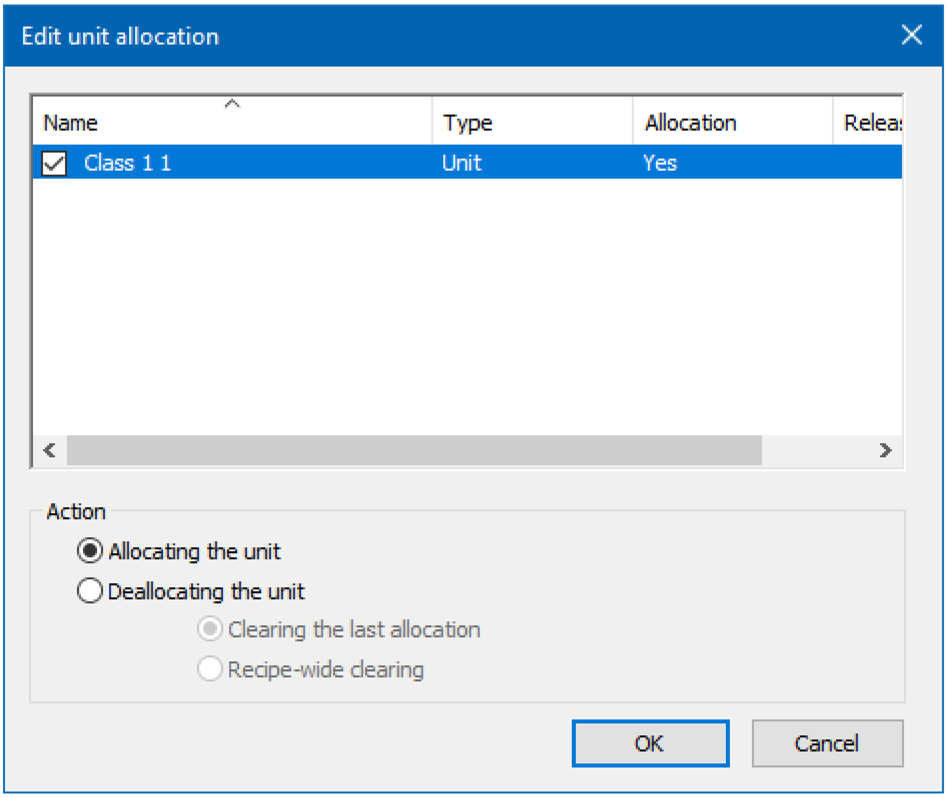
In a configuration dialog you can select the units which exist in the batch recipe. Placeholders and placeholder groups are offered. These are also highlighted in color.

It is distinguished between allocation and release. Releases can be set selectively and globally.

> **⚠Attention**
>
> With grouped placeholders, only the complete placeholder group can be selected. If an ungrouped placeholder that has already been used is added to a group, this leads to an invalid unit allocation.
>
> Placeholder groups are not available in operation templates. Placeholder groups are used in operation instances however. A placeholder can thus become a placeholder group. Assignments with a placeholder group in a recipe can thus also be approved in an operation instance.

Units which were selected before and no longer exist in the recipe, are displayed with an appropriate note. If the dialog is opened in an approved recipe, then access is read only. No entries can then be edited.

UNIT ASSIGNMENT DIALOG



Parameter	Description
List of units	Displays existing units and their assigned actions. The type of the unit is also displayed: Unit, unit placeholder or unit placeholder group.
Action	Assigns an action to the unit selected in the list.
Allocation of the unit	Allocates the unit.
Release of the unit	Releases the unit: ▸ Release of the last allocation: The last allocation is released. ▸ Recipe-spanning release: All allocations in the recipe are released.
OK	Applies settings and closes the dialog.

Parameter	Description
Cancel	Discards changes and closes dialog.

ALLOCATE AND RELEASE UNITS

The allocation of a unit by the unit allocation element is only possible if the unit has not yet been allocated or has only been allocated in the same recipe. All units that are linked in the element are assigned and approved at the same time. It is always waited until all units which should be allocated are released. Then all allocations and releases are executed. Phases that are located in front of an element unit allocation remain active until the allocation was successful. When the recipe is finished and there are still allocations of **unit allocation** elements active, they are released implicitly.

ACTIONS

For each unit which is used in the recipe, you can define an action:

- no action
- allocate
- release

TOOL TIP

In the tool tip of the **unit allocation** element, all units that are marked for allocation or release are displayed. During the execution the Execution duration (on page 32) is stated and all units for which you must wait are color-coded. You must wait for units if they are allocated in another recipe.

UNIT ALLOCATION ELEMENT WITH PLACEHOLDERS

When assigning the unit allocation element, a check is made to see whether all placeholders are linked to unit instances. Placeholders without linking are not permitted.

Procedure:

- Each placeholder is allocated to the assigned instance. If it is a group, the allocation is carried out for all placeholders contained in the group.
- If all placeholders are correctly linked, all placeholders affected are blocked. Execution can also be paused or held at this point.
- If there is at least one non-linked placeholder, there is a wait until all placeholders are linked. This is shown with the result *waiting for instance allocation*. In this status, the element allocation is shown with a red border. The error is shown in the tooltip. The border remains red after the error has been rectified.
- The unit allocation element displays its status with different symbols:

- Active: Allocation is carried out.
- End of phase: Allocation has been ended.
- Waiting for instance allocation: There is a wait for a unit instance.

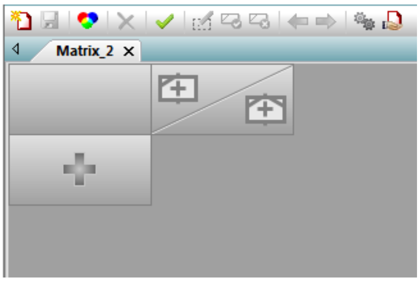
- Skip: The symbol to skip is activated in the tool bar, but acts differently for:
  - Waiting for the allocation of a unit instance: When the symbol is clicked on, a notification is displayed stating that this step cannot be skipped.
  - Waiting for allocation: It is still possible to skip. Affected placeholders are blocked for the instance allocation if there is a wait for the allocation.

8.4.3.11 End element

Last element in the process. It is automatically created in the editor when a recipe is created and it cannot be deleted.

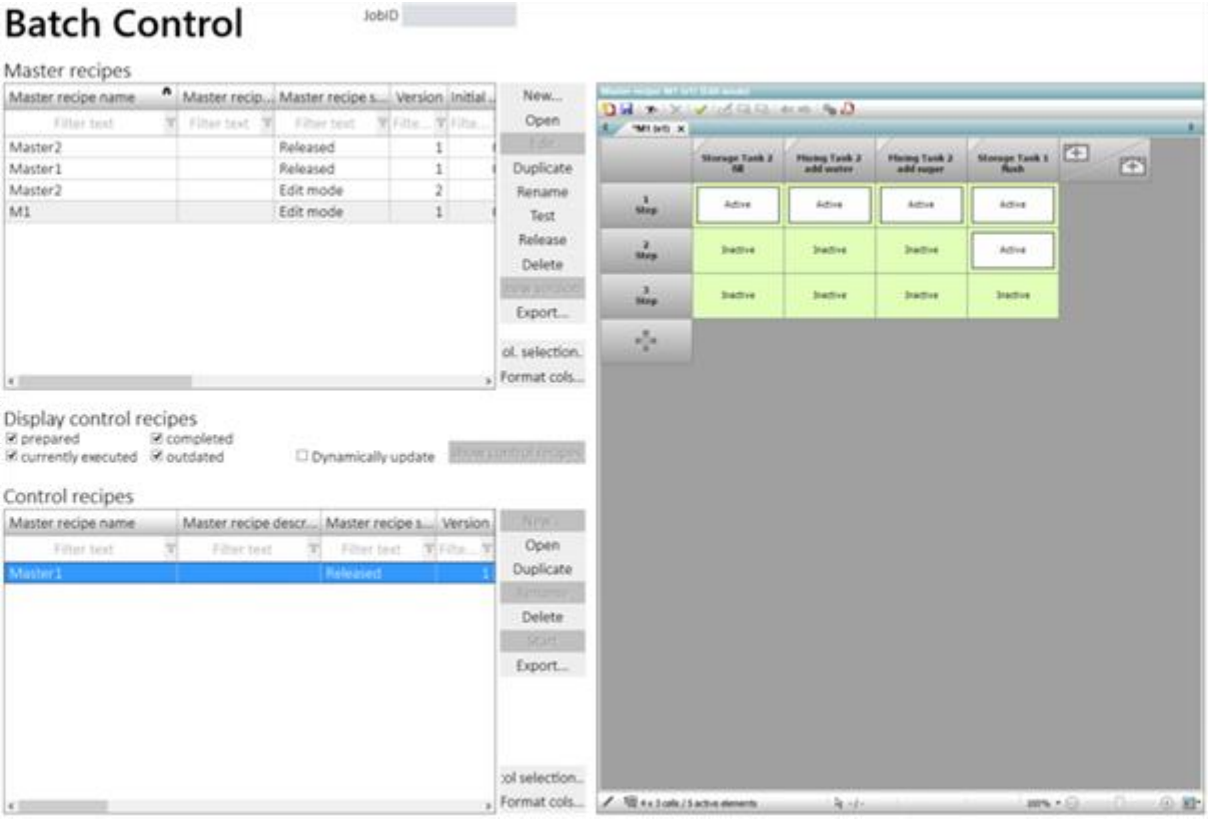
8.4.4 Matrix recipe

If you selected matrix recipe in dialog **Configuration master recipe** and exited the dialog with OK, the newly created recipe opens on a new tab in the matrix editor (on page 208).



8.4.4.1 Matrix editor

In the matrix editor you can create your recipes and operations in a matrix.



Batch Control

JobID

Master recipes

Master recipe name	Master recip...	Master recipe s...	Version	Initial...
Filter text	Filter text	Filter text	Filte...	Filte...
Master2		Released	1	
Master1		Released	1	
Master2		Edit mode	2	
M1		Edit mode	1	

Display control recipes

☑ prepared ☑ completed
☑ currently executed ☑ outdated
□ Dynamically update

Control recipes

Master recipe name	Master recipe descr...	Master recipe s...	Version
Filter text	Filter text	Filter text	Filte...
Master1		Released	1

	Storage Tank 2 fill	Mixing Tank 2 add water	Mixing Tank 2 add sugar	Storage Tank 1 flush
1 Step	Active	Active	Active	Active
2 Step	Inactive	Inactive	Inactive	Active
3 Step	Inactive	Inactive	Inactive	Inactive

Technical details:

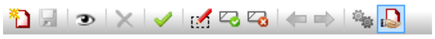
- Columns contain phases (on page 212) and operations (on page 215):
  Phases are inserted by clicking on the symbol above the diagonal
  Phases are inserted by clicking on the symbol below the diagonal
- Lines contain steps with active/inactive phases.
  Clicking on the plus sign inserts a new line or column with a phase or operation into the matrix.
- Delete column/row Press the **Del key** to delete the highlighted row or column.
- Move lines/columns: Lines and columns can be moved via drag & drop with the help of the mouse. Individual labels of steps remain; the step number is adapted automatically.
- Border cells:
  - *Double click* on phase: opens the dialog (on page 212) for configuring the phase.
  - **Double click** on step: opens the dialog for labeling (on page 212) the step.
- Selection of lines/columns:
  - **left mouse click** in border cell: selects a line/column.

    - **Ctrl key + mouse click** in border cell: selects additional line/column.
- Cells:
    - **left mouse click**: selects empty cell.
    - **Shift key + click**: activates/deactivates phase.
    - **Ctrl key + click**: selects several cells.
    - **Double click** on cell: opens the dialog for configuring (on page 212) the phase.
- Scroll bar: Is displayed if the document is larger than the frame.
- Scrolling with a mouse wheel: up and down or, if you press and hold **Shift**, left and right.

8.4.4.2Toolbars matrix recipe

Some symbols can also be configured as control elements (on page 71) in the screen. Touch operation is thus simplified.

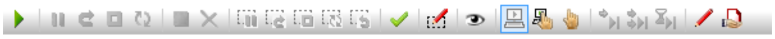
TOOLBARS MATRIX RECIPE: EDIT MODE



Symbol	Description
New master recipe...	Opens the dialog for creating a new master recipe.
Save master recipe	Saves the master recipe which is open for editing.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Delete	Deletes the selected elements from the diagram. Multiple selection via **Ctrl + mouse click keyboard shortcut**.
Check recipe for errors	Checks recipe for errors and displays found errors in an information window. If there are several errors, the first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element.

Symbol	Description
Activate selected elements	Activates the phase of the operation of the selected cells.
Deactivate selected elements	Deactivates the phase or the operation of the selected cells.
Move selected columns to the right or move selected steps up	Moves the selected column to the left by one position or moves selected step up by one position.
Move selected columns to the right or move selected steps down	Moves the selected column to the right by one position or moves selected step down by one position.
Switch recipe to test mode	Switches recipe to the test mode. For this the recipe must be without errors.
Release recipe	Releases the recipe. With this a control recipe can be created.

TOOLBARS MATRIX RECIPE: TEST MODE



Parameter	Description
Start recipe	Starts the recipe process.
Recipe pausing	Pauses the recipe.
Recipe resuming	Resumes paused recipe.
Recipe holding	Holds recipe.
Restart recipe	Restarts held recipe.
Recipe stopping	Stops the recipe.
Recipe aborting	Aborts the recipe process.
Phase pausing	Pauses the phase.
Phase resuming	Resumes the process of a paused phase.
Phase holding	Holds phase.
Restart phase	Restarts held phase.

Parameter	Description
Escape phase	Starts process to exit from the phase.
Check recipe for errors	Starts recipe validation (on page 249).
Edit element	Opens the corresponding dialog for editing the selected element.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Switch to automatic mode	Switches process to *automatic mode*.
Switch to semi-automatic mode	Switches process to *semi-automatic mode*.
Switch to manual mode	Switches process to *manual mode*.
Continue recipe only at selected execution position	Continues a recipe at the selected position.
Continue recipe at all execution positions	Continues a recipe on every available position.
Skip active condition	Skips an active condition. Only possible in the *manual mode*.
Edit mode	Switches from test mode to edit mode.
Release recipe	Releases the recipe. With this a control recipe can be created.

MATRIX RECIPE TOOLBAR: APPROVED

Parameter	Description
Check recipe for errors	Checks recipe for errors and displays found errors in an information window. If there are several errors, the first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element.
Graphical design	Opens the dialog (on page 156) to configure

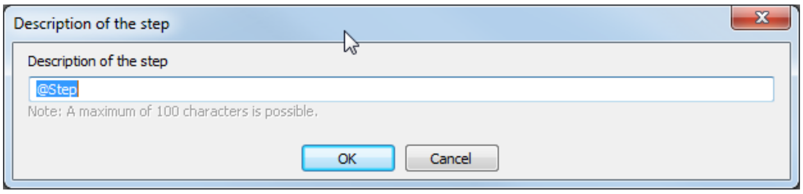
Parameter	Description
	background colors, grid, and display of element ID.
Duplicate recipe	Created a copy of the recipe. The dialog for the input of a unique name and the description is opened.
Create control recipe	Creates a control recipe on the basis of the approved master recipe.

Information

The functions of the individual symbols can also be configured using buttons and thus be made touch-operable. Toolbars can therefore also be hidden (on page 149).

8.4.4.3Name steps

Steps in the matrix editor can be named individually. Double click on the cell to open the dialog for entering an individual name.



@Step

Note: A maximum of 100 characters is possible.

The name can have up to 100 characters and is language switchable is preceded by a @.

8.4.4.4Add and configure phase

To add a phase:

1. Click on the plus sign above the diagonal in the last column of the matrix.

   The dialog to select a phase (on page 174) is opened.
   Phases from placeholders are displayed in color.
2. Select the desired phase.
   If a phase of a placeholder is selected, a notice of the unit placeholder is displayed in the recipe.

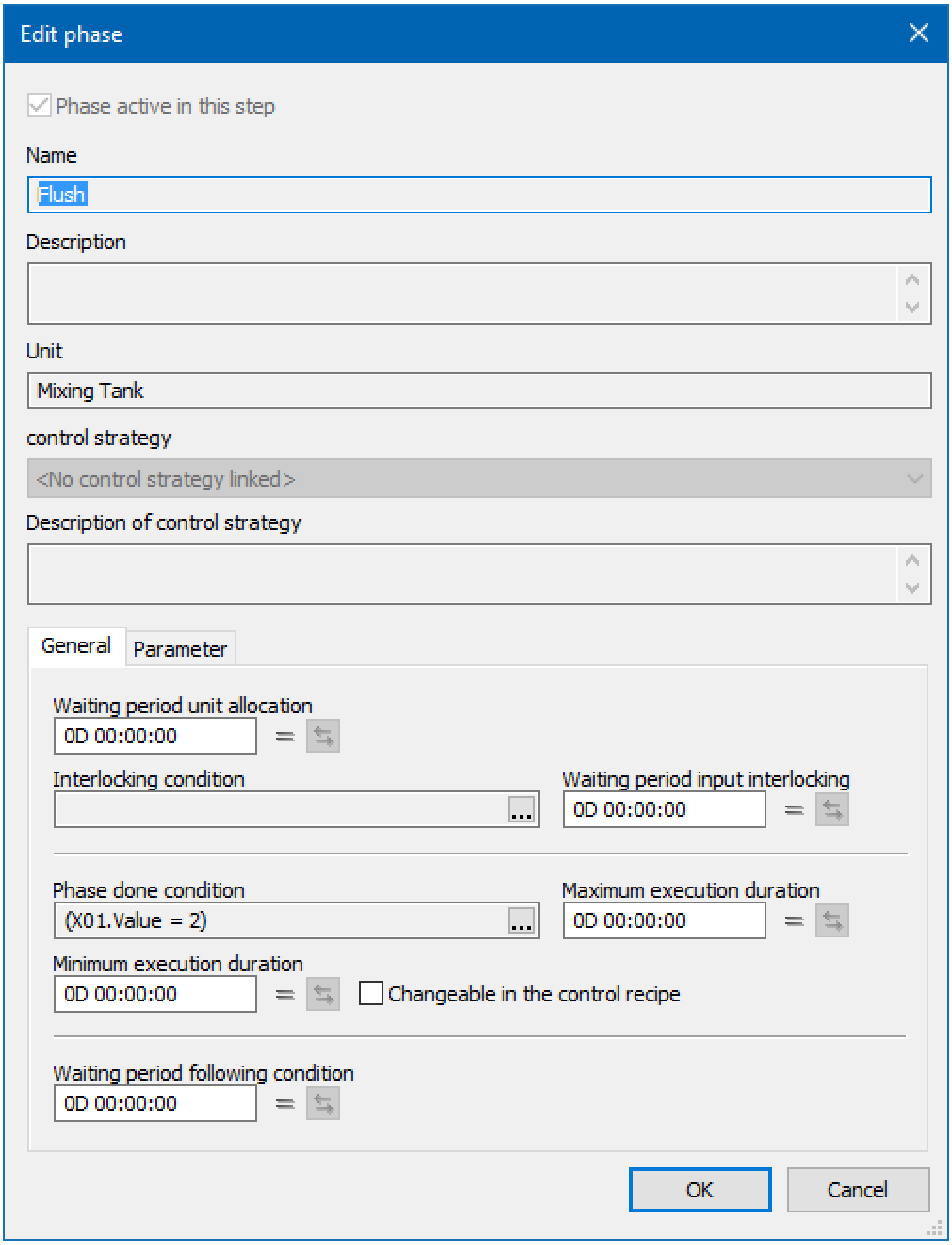
To configure a phase:

1. Double click the matrix filed with the desired phase.

   The dialog for configuration is opened

Note: If a phase is set to inactive, the configuration remains saved as long as the recipe is in edit mode.

CONFIGURATION



Edit phase

☑ Phase active in this step

Name: Flush

Description:

Unit: Mixing Tank

control strategy: <No control strategy linked>

Description of control strategy:

General | Parameter

Waiting period unit allocation: 0D 00:00:00

Interlocking condition:

Waiting period input interlocking: 0D 00:00:00

Phase done condition: (X01.Value = 2)

Maximum execution duration: 0D 00:00:00

Minimum execution duration: 0D 00:00:00

□ Changeable in the control recipe

Waiting period following condition: 0D 00:00:00

Parameter	Description
Phase active in this step	*Active*: Phase is active in this step
Name	Name of the phase. Display only.
Description	Comment about the phase. Display only.
Unit	Unit on which the phase is carried out. Only display.
Control strategy	Selection of a control strategy (on page 229) from a drop-down list. Only available if control strategies have been configured (on page 62) for this phase. Default: *no control strategy linked*
Control strategy description	Description of the selected control strategy. Display of the description entered in the Editor only.
General	Tab for configuration of general properties.
Parameter	Tab for configuration of parameters.
OK	Applies all changes on all tabs and closes the dialog.
Cancel	Discards all changes on all tabs and closes the dialog.

Configuration of the tabs see section PFC editor (on page 166)/chapter:

- General: (on page 175) Display and configuration of the settings for the phase
- Parameter (on page 179): Configuration of the tags

RULES FOR EDITING A PHASE

A phase can be edited:

- In a master recipe in edit mode: If the user has sufficient rights.
- In a master recipe in test mode: If the user has sufficient rights and the phase is not active.
- In a control recipe with *prepared* status: If the user has sufficient rights and the **Changeable in the control recipe** option has been activated.
- In a control recipe with *running* status: If the user has sufficient rights, the **Changeable in the control recipe** option has been activated and the phase is not active.

The phase can no longer be edited in pre-configured control recipes and in approved master recipes.

8.4.4.5 Insert operation

To insert an operation (on page 220):

1. Click on the plus sign below the diagonal in the last column of the matrix.

   The dialog for selecting an operation template (on page 226) is opened.

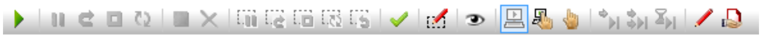
2. Select the desired operation template.

   A new operation is inserted.

8.4.5 Master recipe - test mode

The test mode is used to test master recipes without releasing it and creating control recipes. In addition in the test mode changes in the Editor can be applied directly via reloading the Runtime. **Exception:** During the execution of a recipe, the reloading of a recipe is delayed. Not until the recipe is finished, stopped or aborted, the reloading process is executed.

In test mode you cannot can the principle recipe process. You can only change values of the command tags. Changes are directly saved in the master recipe. It is not necessary to save explicitly.



Parameter	Description
Start recipe	Starts the recipe process.
Recipe pausing	Pauses the recipe.
Recipe resuming	Resumes paused recipe.
Recipe holding	Holds recipe.
Restart recipe	Restarts held recipe.
Recipe stopping	Stops the recipe.
Recipe aborting	Aborts the recipe process.
Phase pausing	Pauses the selected phase.
Phase resuming	Resumes the process of the selected paused phase.
Phase holding	Holds the selected phase.
Restart phase	Restarts the selected held phase.
Check recipe for errors	Checks recipe for errors and displays found errors in

Parameter	Description
	an information window. If there are several errors, the first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Switch to automatic mode	Switches process to *automatic mode*.
Switch to semi-automatic mode	Switches process to *semi-automatic mode*.
Switch to manual mode	Switches process to *manual mode*.
Continue recipe only at selected execution position	Continues a recipe at the selected position.
Continue recipe at all execution positions	Continues a recipe on every available position.
Skip active condition	Skips an active condition. Only possible in the *manual mode*.
Switch recipe to edit mode	Switches from test mode to edit mode.
Release recipe	Releases the recipe. With this a control recipe can be created.

8.4.6 Release master recipe

You can approve a master recipe by means of:

- Selection in the list and clicking on the **Release Master Recipe** button.
  Several recipes can also be selected and approved together. Approval must be confirmed by means of a dialog.
- The zenon function (on page 143) **Release master recipe**.
  Approves all master recipes suitable for configuration.

> **Information**
>
> A master recipe can only be approved if:
>
> - The recipe is free of errors
> - All recipes contained therein have already been approved
>
> When approving, the current configuration is saved for all phases currently used in the recipe. This also includes all phases of placeholders. The configuration is also used in the control recipe. Any possible changes - via the Editor for example - are no longer taken into account.
>
> After approval, a master recipe can no longer be edited!
> You can however create a copy of the approved master recipe. To do this, click on the **Duplicate Master Recipe** button. This copy can then be edited.

8.4.7 Highlight recipe as outdated

Recipes that are no longer valid but have not been deleted should be marked as outdated. If a recipe is set to this status, it can no longer be edited or approved. No control recipe can be created on the basis of this recipe either. The recipe can however be duplicated and thus be used as the basis for new master recipes.

Only recipes that have the status *approved* can be marked as outdated. The following applies for attendant control recipes:

- Control recipes that are currently being executed continue to be executed
- Control recipes with the *prepared* status can no longer be executed

8.4.8 Versioning for master recipes

Master recipes can also be versioned. In doing so, a copy of an approved or obsolete report is created. This copy is in edit mode and contains a unique version number. The new recipe can be edited, but not renamed. Individual versions, including the source recipe itself, can be deleted.

CREATING A VERSION

To use versioning in the Runtime:

1. In the Editor, go to the **General**/**Versioning** properties group in the Batch Control node.
2. Activate the **Versioning active** property.
3. Versioning is switched on and used in the Runtime.

To create a new version of a recipe in the Runtime:

1. Select the desired master recipe.
   Note: The recipe must be approved or obsolete.
2. Select, in the context menu or on the toolbar, **Create new version** or click on the corresponding button in the screen.
3. A new recipe is created.

RULES FOR VERSIONING

The following applies to versioning:

- A new version of a recipe contains the same name as the source recipe.
- New versions of a recipe cannot be renamed. Not even if the version is in editing mode and it is the only remaining version.
- The description can be changed for each version.
- Each version contains a unique version number that is issued on a serial basis. Version numbers of deleted recipes remain blocked and are not reissued.
- The version number of the new recipe and the version number of the source recipe are displayed in their own columns in the list of master recipes.
- Version numbers are also displayed in the title bar of the recipe editor and in the tab of the recipe as well as in the list of the control recipes and in tooltips.

ONLY APPROVE ONE VERSION

It is possible to only allow one version of each master recipe to be approved. To do this, activate the **Release one version only** property in the Editor. Only one version of each master recipe can be approved in the Runtime. If a different version is approved, the one that was approved before must first be deleted or marked as obsolete.

8.4.9 Actions

Action	Description
Check recipe for errors	Checks the recipe for errors and displays error messages.
Edit element	Opens the corresponding dialog for editing the selected element.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.

Action	Description
Duplicate recipe	Copies the selected recipe and adds it as copy to the list.
Create control recipe	Creates a control recipe on the basis of the approved master recipe.
New master recipe	Opens the dialog (on page 162) for creating a new recipe.
Save master recipe	Saves all changes which were done since the last saving.
Delete	Deletes the selected element.
Exchange phase	Opens the dialog (on page 174) for selecting a phase. The present phase is replaced by the newly selected phase.
Edit mode	Switches the mouse cursor from adding an element to edit mode. The switch back to the edit mode can also be achieved by pressing the **Esc key**.
Insert phase	Occupies the mouse cursor with a phase (on page 171). It can be added to any allowed, free location via click.
Insert transition	Occupies the mouse cursor with a transition (on page 198). It can be added to any allowed, free location via click.
Insert Begin parallel branch	Occupies the mouse cursor with a begin parallel branch (on page 202). It can be added to any allowed, free location via click.
Insert End parallel branch	Occupies the mouse cursor with an end parallel branch (on page 202). It can be added to any allowed, free location via click.
Insert Begin branch	Occupies the mouse cursor with a begin branch (on page 200). It can be added to any allowed, free location via click.
Insert End branch	Occupies the mouse cursor with an end branch (on page 200). It can be added to any allowed, free location via click.

Action	Description
Insert unit allocation	Occupies the mouse cursor with a unit allocation (on page 204). It can be added to any allowed, free location via click.
Insert jump target	Occupies the mouse cursor with a jump target (on page 203). It can be added to any allowed, free location via click.
Switch recipe to test mode	Switches recipe to the test mode (on page 215).
Release recipe	Releases (on page 216) the recipe. With this a control recipe can be created.

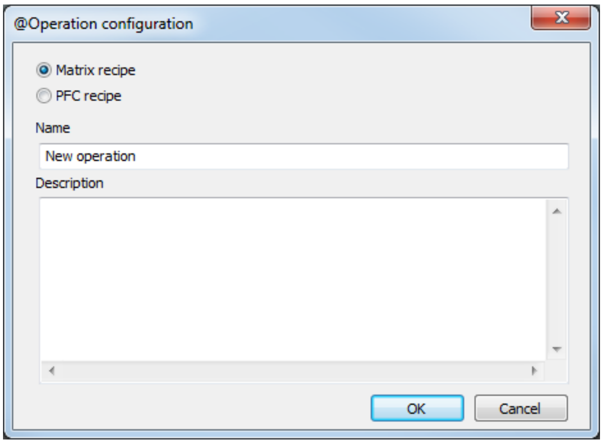
8.5 Operations

Operations form a substructure that can be embedded in recipes. This can provide a better overview in complex recipes. Operations are created in a similar manner to matrix recipes or PFC recipes. The operations are created on the basis of templates (on page 226) and as an instance in existing matrix recipes or PFC recipes.

CREATING AN OPERATION

To create a new operation:

1. In the Recipe editor (on page 148) switch the list of operations to visible (activate checkbox)
2. Select, in the toolbar or context menu (on page 223) of the list, the **New operation** command.
3. The dialog for configuring an operation is opened.



@Operation configuration

● Matrix recipe
○ PFC recipe

Name
New operation

Description

Parameters	Description
Matrix recipe	Activate this radio button if you want to create an operation in the form of a matrix recipe (on page 207). **Note:** Only possible if the corresponding license is available.
PFC recipe	Activate this radio button if you want to create an operation in the form of a PFC recipe (on page 165). **Note:** Only possible if the corresponding license is available.
Name	Unique name for the operation. The name must not contain a dot (**.**), a question mark (**?**), a @ or an asterisk (*****). Maximum length: *256* characters. **Note:** When you copy an operation the existing name is complemented with the prefix "**Copy of**". If the maximum length is exceeded by this, the name is shortened to the allowed length starting from the last character. The uniqueness is checked in the entire network. Therefore it can happen that you cannot take over the name as another user on another computer in the zenon network already has used the same name and you do not see the recipe in the list of the operations yet. The recipe names can be changed later but only as long as the recipe has the state *editable*.
Description	Optional description for the operation which should be created. The description can be changed later but only as long as the operation has the state *editable*. To change the description select the **Configure Operation** symbol.
OK	Applies all settings and created a new operation.
Cancel	Closes the dialog without creating an operation.

Configuration in the Recipe Editor is similar to the creation and configuration of PFC recipes (on page 165) and Matrix recipes (on page 207).

Changes to operations are only visible for the user in the operation instance if the operation instance has been saved.

OPENING OPERATIONS

Existing operations can be opened by:

- Double-clicking on an operation in the list of the operations

- The **Open in Recipe Editor** command in the context menu of an operation
- Clicking on the **Open** symbol in the toolbar

USING OPERATIONS

Operations can be inserted in recipes in the matrix editor or in the PFC editor and used as part of the recipe there.

You insert operations:

- In the PFC recipe using the **Insert operation** (on page 196) symbol
- In the matrix recipe using the **Add operation** (on page 215) symbol

Phases and commands can be executed within operations. The position of the object in the recipe is also given in the CEL when a command is executed.

When inserting an operation into a recipe, the currently-saved version is always inserted. If the operation is open for editing, all unsaved changes are thus also not part of the inserted instance.

SAVING OPERATIONS

Operations are, as instances, always part of the master recipe in which they are integrated. If an operation instance is saved, the attendant master recipe is also saved automatically.

TOOL TIP

Operations have a tooltip. The following are displayed as soon as they have been executed once:

- Execution status:
  Contains current status and original status (status from which the operation comes). The original status is evident from the visual coloring and does not always correspond to the actual last status.
- Execution counter
- Error:
  Shows the number of objects that currently have an error status and the number of objects that have had an error status.
- Exit from phase:
  Number of objects that are being exited from or have been exited from
- Total duration with time points of activation and deactivation
- Information about the status of the internal objects: Number of objects with the respective status (except **idle**)

8.5.1 Toolbar and context menu operations

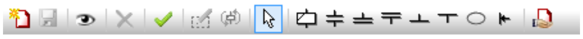
TOOLBAR LIST

Parameter	Description
New operation	Creates a new operation.
Open operation in Editor	Opens the selected operation in the recipe editor.
Configure operation	Opens the dialog to configure an operation.
Duplicate operation	Duplicates the selected operation.
Delete operation	Deletes the selected operation.
Export selected as XML...	Exports the selected operation as an XML file.
Import XML	Imports the selected XML file as operation(s).
Release operation	Checks the selected operation and approves it if no errors were found.
Column selection	Opens the dialog to select a column (on page 115).
Column format	Opens the dialog to format a column (on page 119).

CONTEXT MENU

Parameter	Description
New operation	Creates a new operation.
Configure	Opens the dialog to configure an operation.
Duplicate	Duplicates the selected operation.
Delete	Deletes the selected operation.
Export selected as XML...	Exports the selected operation as an XML file.
Import XML...	Imports the selected XML file as operation(s).
Open in Editor	Opens the selected operation in the recipe editor.
Release	Checks the selected operation and approves it if no errors were found.

TOOLBAR PFC EDITOR: EDIT MODE



Parameter	Description
New operation...	Opens the dialog for creating a new operation.
Save operation	Saves the master recipe which is open for editing.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Delete	Deletes the selected elements from the diagram. Multiple selection via **Ctrl key + mouse click**.
Check operation template for errors	Checks recipe for errors and displays found errors in an information window. If there are several errors, the first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element.
Exchange phase	Opens dialog for selecting a phase and replaces the existing phase by the newly selected phase. Several phases can be selected and replaced together Shortcut: **Shift key + double-click**
Edit mode	Toggles between insert mode and edit mode.
Insert phase	Adds a phase.
Insert transition	Adds a transition.
Insert Begin parallel branch	Adds the start of a parallel branch.
Insert End parallel branch	Adds an end of a parallel branch.
Insert Begin branch	Adds a begin branch.
Insert End branch	Adds an end branch.
Insert unit allocation	Adds a unit allocation.

Parameter	Description
Insert jump target	Adds a jump target.
Release operation	Releases the recipe.

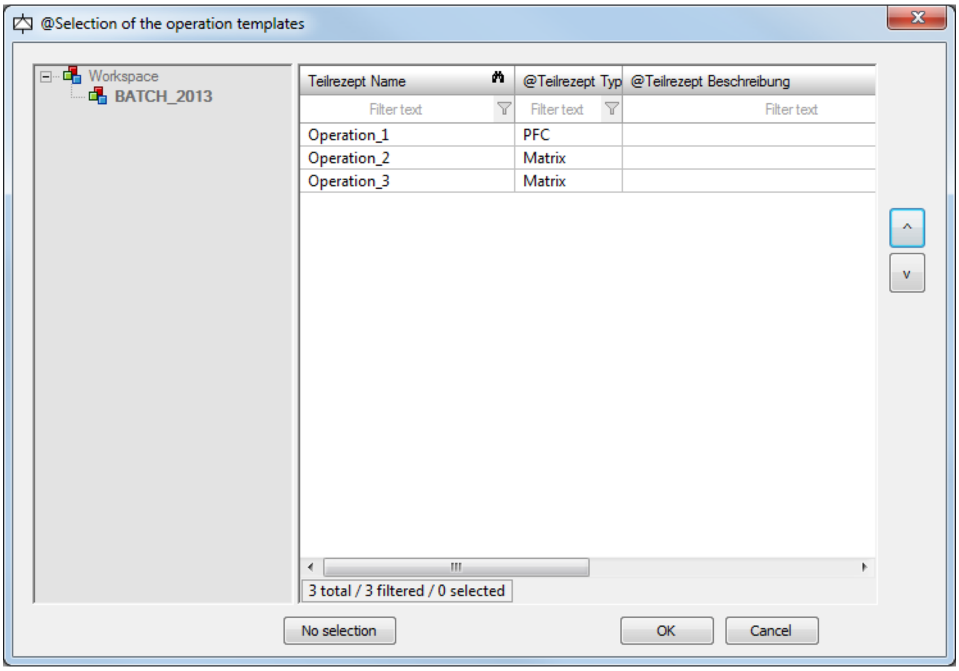
Note: Insertions remain active until there is a switch to edit mode using the **Edit Mode** symbol, the **Esc key** or via a symbol to add an element to another insert option.

PFC RECIPE TOOLBAR: APPROVED

Parameter	Description
Check operation template for errors	Checks recipe for errors and displays found errors in an information window. If there are several errors, the first errors are shown. An error message contains the error number, the ID of the element, its location and a message in plain text.
Edit element	Opens the corresponding dialog for editing the selected element. The dialog is opened in write-protected mode, because it is no longer possible to edit approved recipes.
Graphical design	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Duplicate operation	Created a copy of the recipe. The dialog for the input of a unique name and the description is opened.

8.5.2 Selection of the template for an operation

Operations can be inserted into recipes as an instance. When inserting a partial recipes into a matrix recipe (on page 215) or a PFC recipe (on page 196), the dialog to select a template for an operation is opened. This contains all previously-configured (on page 220) operations:



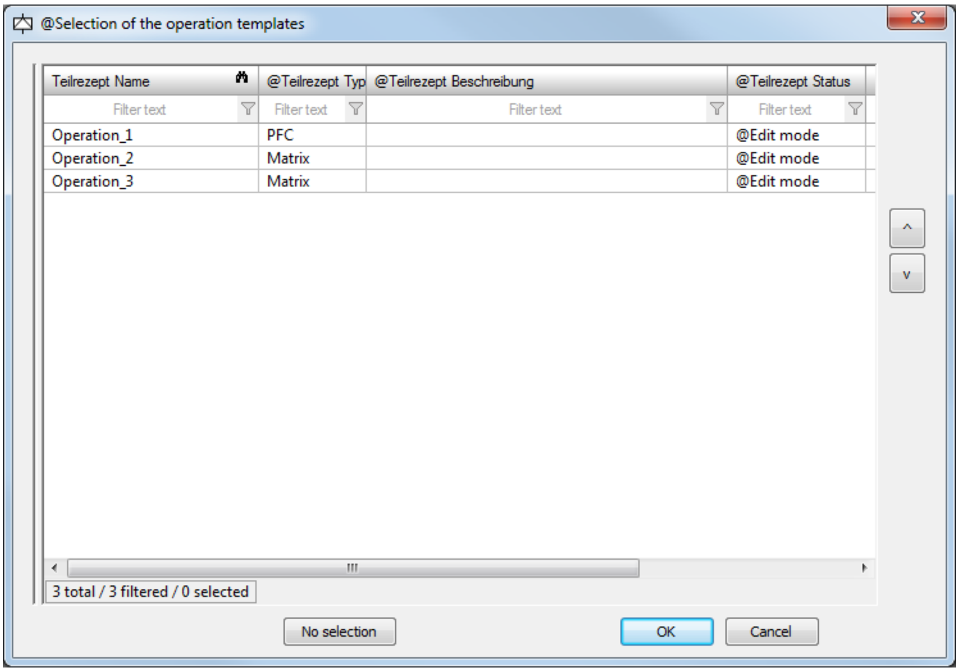
Teilrezept Name	@Teilrezept Typ	@Teilrezept Beschreibung
Filter text	Filter text	Filter text
Operation_1	PFC	
Operation_2	Matrix	
Operation_3	Matrix	

3 total / 3 filtered / 0 selected

Parameter	Description
List workspace	In the standard dialog to display the projects present in the workspace. No function in Batch Control, because operations can always be selected from the respective active project. **Hint:** Drag this area down to the minimum size. It is then only displayed as minimized* in the future.
Operation list	Contains all operations that have been created. Any desired operations (matrix or PFC) can be selected for both editors (matrix and PFC). Entries can: - be sorted by clicking on the column title; another click inverts the sorting - be filtered into filter rows by alphanumeric entries (wildcards can be used)
Arrow keys	Move the selection: - Up

Parameter	Description
	- Down
Status line	Display: - Number of entries - Number of filtered entries displayed - Number of selected entries
No selection	Cancels existing selection for existing operation and closes the dialog.
OK	Accepts selection, closes dialog and inserts operation.
Cancel	Discards selection and closes dialog without selecting or amending a template.

*workspace minimized:



@Selection of the operation templates

Teilrezept Name	@Teilrezept Typ	@Teilrezept Beschreibung	@Teilrezept Status
Filter text	Filter text	Filter text	Filter text
Operation_1	PFC		@Edit mode
Operation_2	Matrix		@Edit mode
Operation_3	Matrix		@Edit mode

3 total / 3 filtered / 0 selected

8.5.3 Status operation

The status shown for an operation always represents the status of the object with the highest priority in the operation.

The following applies to the colored identification:

- Non-transient status: the whole operation is colored
- Transient status: Original status and target status are displayed
- *HOLDING* as original status: *RUNNING* (green) is always displayed

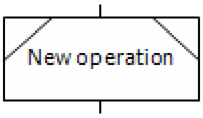
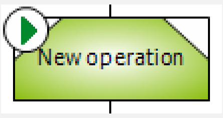
PRIORITY

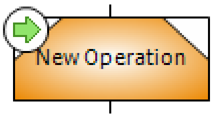
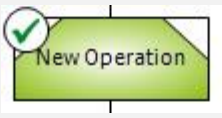
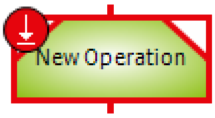
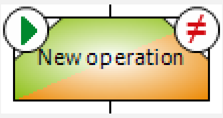
Priority of the objects in an operation, starting with the highest priority:

1. *ABORTING*
2. *ABORTED*
3. *STOPPING*
4. *STOPPED*
5. *RESTARTING*
6. *HOLDING*
7. *HELD*
8. *PAUSING*
9. *PAUSED*
10. *RUNNING*
11. *COMPLETE*
12. *IDLE*

8.5.4 Symbol for execution

The symbols correspond to the symbols (on page 255) generally used the REE. Operations are symbolized in the REE by triangles in the left and right corner.

Symbol	Description	Tool tip
	*Idle*: No object is active in the operation.	Idle
	*In execution*: A symbol is always displayed whilst an operation is running. If objects are still running in the operation, the progress symbol is shown.	Execution of the internal objects.

Symbol	Description	Tool tip
	*Active element active*	
	*End of operation*: If objects are already active after the operation, the symbol for **following condition** is displayed.	Waiting for **Following conditions**.
	*In the operation at least one execution position is waiting*: Is shown if, in semi-automatic or manual mode, execution after a phase is in the operation.	
	If an object has a different status within the operation, then a symbol is also displayed on the top right. There is a different status if an object has an empty status within the operation that does not correspond to the status of the operation and is not *idle* or *complete*.	

8.6 Control strategies

Control strategies can be selected in the Runtime when configuring the phase in the PFC recipe (on page 171) or matrix recipe (on page 212). They must have already been configured in the zenon Editor (on page 62).

Control strategies can only be changed:

- In master recipes
  - In editing mode
  - Test mode
- In operation templates:
  - In editing mode

The control strategy cannot be changed in operation instances.

COMMAND PARAMETERS IN THE PHASE

If control strategies are activated for a phase (**Active control strategies** property in the Editor), all command parameters are initially removed when this phase is inserted into a recipe.

If the control strategy is changed in the dialog to edit a phase, then:

- All parameters currently in this phase are removed
- The linked parameters in the newly-selected control strategy are inserted

  These parameters are only initialized with the values that the parameter linkings currently have. The existing Runtime configurations of the parameters is lost in the process.

If a control strategy is selected for a phase, then the parameters to write to **control strategy number** are inserted in addition to the linked parameters. This parameter gets the parameter number as a numerical value. It is not a recipe parameter. If the same parameter is already linked to the control strategy, the configuration of the linking is ignored!

SYNCHRONIZATION OF CHANGES TO CONTROL STRATEGIES AND CONTROL STRATEGY ACTIVATION

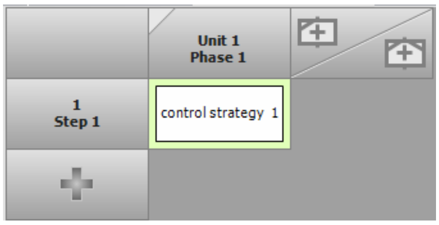
During synchronization (on page 292), changes made in the Editor for control strategies are carried over for phases. For details, see the **synchronization** (on page 292) chapter.

DISPLAY OF THE CONTROL STRATEGY

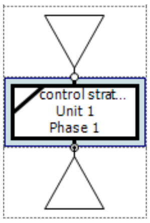
Selected control strategies are displayed:

- In the tooltip of the phase (no display if no control strategy has been selected)
- In the phase in the recipe.

  Matrix:

  PFC:

- In the unit information for the active phases as an appendix

- In the report (on page 298) for a phase

CONTROL STRATEGIES AND VARIABLES

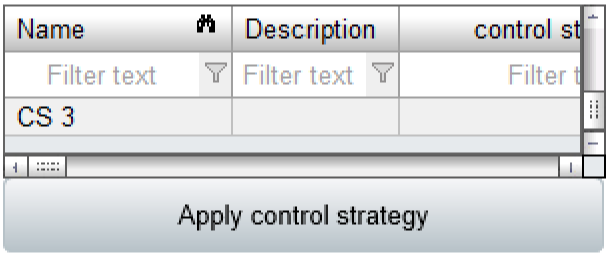
If a phase is selected in the recipe, the variables for the control strategy are filled.

Parameter	Data type	Description
Control strategy name	*STRING*	Displays the **Name** of the control strategy that is linked to the selected phase. Is empty if: - No phase is selected - The phase does not use any control strategies - There is no control strategy currently linked to the phase - With multiple selection
Description of control strategy	*STRING*	Displays the **Description** of the control strategy that is linked to the selected phase. Is empty if: - No phase is selected - The phase does not use any control strategies - There is no control strategy currently linked to the phase - With multiple selection
control strategy number	*LINT*	Displays the **control strategy number** of the control strategy that is linked to the selected phase. Is *-1* if: - No phase is selected - The phase does not use any control strategies - There is no control strategy currently linked to the phase - With multiple selection

TOUCH OPERATION

Control strategies can also easily be assigned a phase in the Runtime by means of touch operation instead of the dialog.
To do this:

1. Configure the control elements for the list and the button for assignment in the Editor.
2. Stipulate the columns to be displayed in the screen switching.
   You can also pre-define filters if required.
3. Highlight the desired phase in the Runtime.
4. Select the desired control strategy in the filter list.
5. Tap the **Apply control strategy** button.



Name	Description	control st
Filter text	Filter text	Filter t
CS 3		

The control strategy is applied to the phase.

8.7 Control recipe

Control recipes are copies of approved master recipes that are used for production.

Attention

Automatic creation of control recipes via functions or API: Ensure that a new control recipe is only created once the variable values for the one previously created are present.

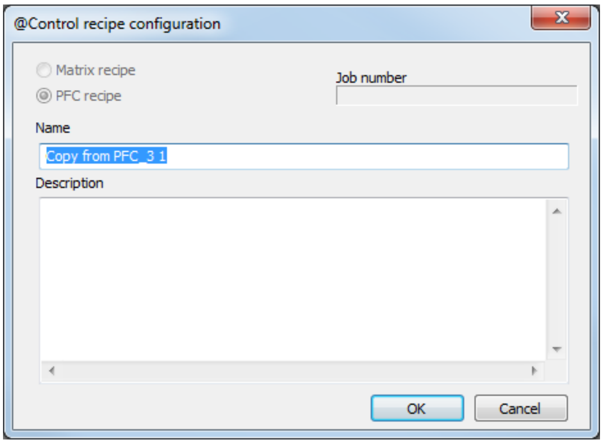
This means the following for the values of the monitoring system driver variables:

- **[Batch Control] Create control recipe result numerically**: must be greater than -1.
- **[Batch Control] Create control recipe result XML**: must not be empty

You can find details about this in the **SYSDRV** manual, in the **Topic - Batch Control** chapter.

8.7.1 Create control recipe

You can create control recipes only based on released master recipes. Select the released master recipe in the list of the master recipes, which should serve as basis for your control recipe and click on button **New control recipe...**.



@Control recipe configuration

○ Matrix recipe
● PFC recipe

Job number

Name: Copy from PFC_3 1

Description

Parameter	Description
Matrix recipe	*Active*: A matrix control recipe is created. Display only.
PFC recipe	*Active*: A PFC control recipe is created. Display only.
Job ID	The job ID provided by the job variables (on page 121).
Name	Unique name for the control recipe. Non-permitted characters: **.? @ *** The uniqueness is checked in the entire network. Therefore it can happen that a name is not accepted as another user already used the same name on another computer in the zenon network at the same time. You can change the name afterwards as long as the recipe has status *Prepared*.
Description	Optional description of the recipe.
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

If the control recipe was created using the symbol in the toolbar, then it is automatically opened in a new tab in the recipe editor provided this has been configured in the screen. If the control recipe has been created using the context menu, it is not opened in the recipe editor.

The newly created recipe is also displayed in the list of control recipes even if it does not match the set filter criteria.

8.7.2 Toolbar and context menu for control recipe list view

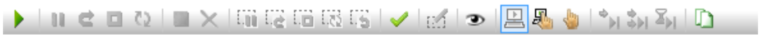
TOOLBAR

Symbol	Description
Open control recipe in Editor	Opens the selected recipe in the recipe editor.
Configure control recipe	Opens dialog to configure the selected recipe.
Duplicate control recipe	Creates a copy of the selected recipe and opens the dialog to rename the duplicate.
Delete control recipe	Deletes selected recipes.
Export selected as XML...	Exports the selected control recipe as an XML file.
Import XML	Imports the selected XML file as operation(s).
Start control recipe	Starts selected control recipe.
Column selection	Opens the dialog to select the columns to be displayed.
Column format	Opens the dialog for configuring the column formats.

CONTEXT MENU

Command	Description
Open in Recipe Editor	Opens the selected recipe in the Editor.
Rename	Opens dialog to rename the selected recipe.
Duplicate	Created a copy of the recipe. The dialog for the input of a unique name and the description is opened.
Delete	Deletes selected recipes.
Export selected as XML...	Exports the selected control recipe as an XML file.
Import XML...	Imports the selected XML file as operation(s).
Start	Starts selected control recipe.

8.7.3 Control recipe edit mode toolbar



Parameter	Type	Description
Start recipe	Command	Starts the recipe process.
Pause recipe	Command	Pauses the recipe.
Resume recipe	Command	Resumes paused recipe.
Hold recipe	Command	Holds recipe.
Restart recipe	Command	Restarts held recipe.
Stop recipe	Command	Stops the recipe.
Abort recipe	Command	Aborts the recipe process.
Pause phase	Command	Pauses the phase.
Resume phase	Command	Resumes the process of a paused phase.
Hold phase	Command	Holds phase.
Restart phase	Command	Restarts held phase.
Escape phase	Command	Starts process to exit (on page 274) from the phase.
Check recipe for errors	Action	Starts recipe validation (on page 249).
Edit element	Action	Opens the corresponding dialog for editing the selected element.
Graphical design	Action	Opens the dialog (on page 156) to configure background colors, grid, and display of element ID.
Switch to automatic mode	Command	Switches process to *automatic mode*.
Switch to semi-automatic mode	Command	Switches process to *semi-automatic mode*.
Switch to manual mode	Command	Switches process to *manual mode*.
Continue recipe at selected execution position only	Command	Continues a recipe at the selected position.

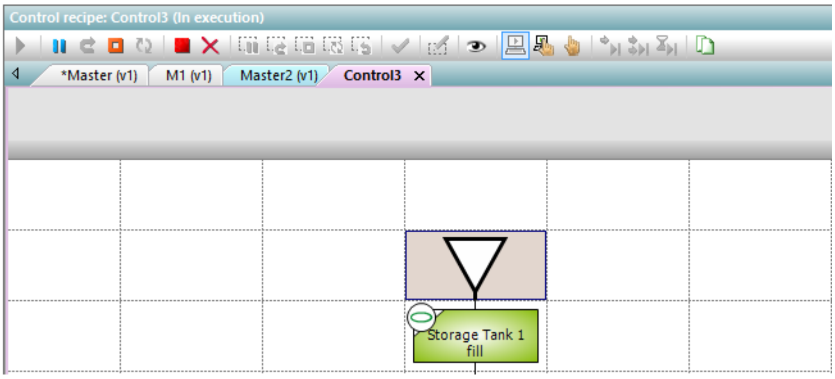
Parameter	Type	Description
Continue recipe at all execution positions	Command	Continues a recipe on every available position.
Skip active condition	Command	Skips an active condition. Only possible in the *manual mode*.
Copy recipe	Action	Created a copy of the recipe. The dialog for the input of a unique name and the description is opened. The copy of the recipe automatically gets the status *Prepared* and can therefore be edited and started. The execution status (on page 263) of the duplicate is set to *automatic*. When duplicating a recipe, a CEL entry is created.

8.7.4 Execute control recipe

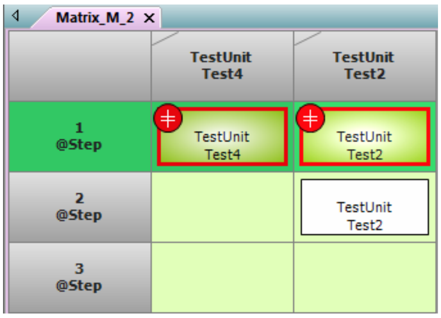
Control recipes can be started:

- after selecting a control recipe in the list of the control recipes:
  - via click on button **Start control recipe**
  - via click on menu item **Start** in the context menu
- via click on symbol **Start control recipe** in the toolbar if the control recipe is opened

PFC control recipe:



Matrix control recipe:



Matrix_M_2

	TestUnit Test4	TestUnit Test2
1 @Step	TestUnit Test4	TestUnit Test2
2 @Step		TestUnit Test2
3 @Step		

CHANGE VALUES

If, for parameters in the master recipe, the **Changeable in the control recipe** option was activated, these values can be amended in the control recipe as long as the phase is not active yet. In this case a button for synchronization is displayed next to the value. A click on this button take over the defined value from the master recipe.

No more values can be changed in control recipes that have already been executed.

MEANING OF THE SYMBOLS NEXT TO THE VALUES:

Symbol	Description
= Equals sign	Value in the control recipe and in the master recipe match.
<> Angle brackets.	Value in the control recipe and in the master recipe are different.
⇆ Arrows in opposite direction.	Click on button to apply the value from the master recipe. It overrides the value in the control recipe. Only active if the values in the control recipe and the master recipe do not match.

VALUE CHANGE DURING EXECUTION

To change values during the execution of a phase:

1. Activate the **Tag values changeable during phase execution** property in the Editor.
2. Decide whether values are written immediately or in bundled form after confirmation.

WRITE VALUES IMMEDIATELY

To write values immediately:

1. Go to the **Value change during execution** properties group for the phase.
2. Activate the checkbox for the **Immediately write value changes of parameters** property. The **Timeout for value change acknowledgement** property is thus automatically deactivated.
3. Configure the required events
   - *Initial TAG modified during process*
   - *Value TAG modified during process*
   - *Amended initial TAG fully written*
   - *Amended value TAG fully written*

Value changes do not need to be confirmed in the Runtime. If a phase is in the corresponding stage (check of the input lock and/or check of the phase done condition), amended values of initial parameters or value parameters are written immediately after the change. A corresponding symbol and information are displayed in the tooltip.

WRITE VALUES IN BUNDLED FORM

Amended values can also be collated and written to the PLC manually by clicking on a button.
To write values in bundled form:

1. Go to the **Value change during execution** properties group for the phase.
2. Deactivate the **Immediately write value changes of parameters** property.
3. Configure the **Timeout for value change acknowledgement** property.
   You thus stipulate how much time can pass between the first change of a parameter and the writing of the values by clicking on a button in the Runtime. The corresponding result is triggered after expiry of the waiting time. This informs the operator that amended values have not yet been sent.
   The value *0* means: no waiting time.
4. Configure the required events
   - *Initial TAG modified during process*
   - *Value TAG modified during process*
   - *Amended initial TAG fully written*
   - *Amended value TAG fully written*
   - *Waiting time for confirmation of the value change of the initial TAG exceeded*
   - *Waiting time for confirmation of the value change of the value TAG exceeded*
5. Configure the **Confirm Changes** button in the screen.

Note: If parameter values are amended during recipe execution, this is indicated by a symbol to the phase. As long as the amended values have not been confirmed with the **Confirm Changes** button, they are not written to the variables. This can led to execution of the function being blocked.

BEHAVIOR IN THE RUNTIME

The setting for the **Tag values changeable during phase execution** property is noted when writing the parameter.
If a phase is still in the validation phase during execution, parameter values can be amended regardless of the property's setting.

Tag values changeable during phase execution Checkbox setting:

- *Active*:
  - Parameter list in the screen: Keyboard is called up. Changes are possible in the dialog to edit the parameters.
  - Edit phase dialog: Changes are possible in the dialog to edit the parameters.
- *Inactive*:
  - Parameter list in the screen: Keyboard is not called up. The dialog to edit the parameters is write protected.
  - Edit phase dialog: The dialog to edit the parameters is write protected.

8.8 Unit placeholders

Placeholders for unit classes can be created for recipes in edit mode. These lists can be used to administer placeholders and replace unit instances (on page 16).

Unit placeholders are available for:

- Master recipes
- Operation templates
- Operation instances (placeholders are displayed in the list of the higher-level recipe)

The unit placeholder list must be configured (on page 71) in the Editor. The initial display is configured (on page 114) with the screen switching.
If a recipe is opened, the list is filled with the pre-existing entries. Several placeholders with the same unit class can be grouped. The grouping ensures that the same unit instance is assigned to all placeholders. This primarily makes sense when using operations.

OPERATIONS

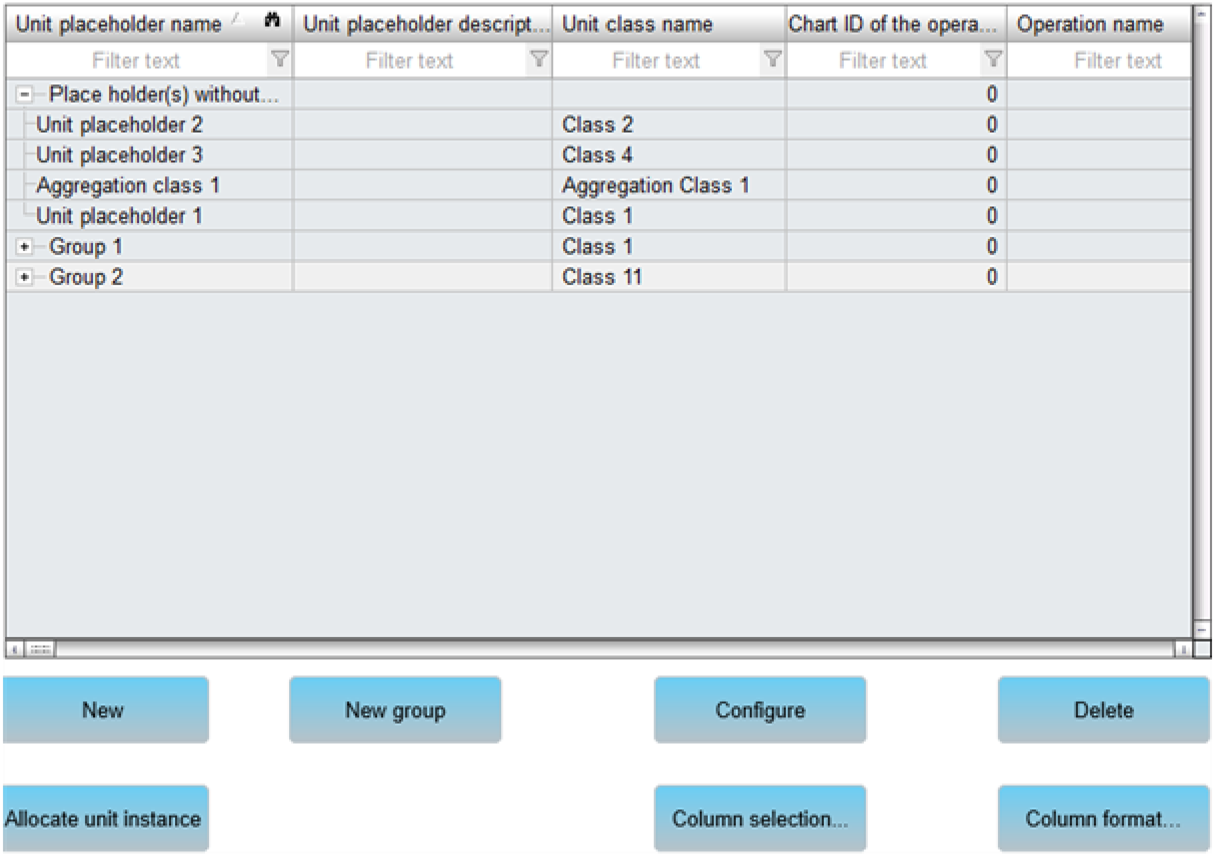
If a master recipe or a partial recipe is opened in the recipe editor, the placeholder list also shows, in addition to the unit placeholders of the master recipe, those of the partial recipe instances. For placeholders in operation instances, all unit instances whereby approval of the master recipe was available.
Placeholders in operations have additional properties that are displayed in the placeholder list:

- Chart ID partial recipe
- Operation name
- Description of partial recipe

All properties can be displayed but not edited.

8.8.1 Unit placeholder list

You administer the placeholders and assign them to these unit instances in the unit placeholder list.



Unit placeholder name	Unit placeholder descript...	Unit class name	Chart ID of the opera...	Operation name
Filter text	Filter text	Filter text	Filter text	Filter text
⊟ Place holder(s) without...			0	
Unit placeholder 2		Class 2	0	
Unit placeholder 3		Class 4	0	
Aggregation class 1		Aggregation Class 1	0	
Unit placeholder 1		Class 1	0	
⊞ Group 1		Class 1	0	
⊞ Group 2		Class 11	0	

Control element	Description
Placeholder list	Shows the available placeholders for units. Elements in the list can be edited in the Runtime

Control element	Description
	using a context menu: - **New**: Creates a new placeholder. - **Configure**: Opens the dialog to name the element and to assign it to a group. - **Delete:** Deletes the selected element. Multiple selection is possible. - **New group**: Creates a new group. - **Allocate unit instance**: Opens the dialog to allocate a unit instance. If unit classes have been deleted, they are shown in the list and given a notice. If changes from another client are saved in the zenon network, the list is updated for all clients.
New	Creates a new placeholder. Available for master recipes and operations in edit mode With partial recipes, the entries are displayed in the list of the host recipe.
Configure	Opens the dialog to configure the selected placeholder or the placeholder group. Here you can also assign placeholders to a placeholder group. The assignment of placeholders to a placeholder group is also possible by means of drag&drop.
Delete	Deletes the selected placeholder after a request for confirmation. Multiple selection is possible. Deleting a placeholder makes phases of the placeholder that already exist in the recipe invalid; they must be reconfigured. If a placeholder group is deleted, the placeholders contained therein are assigned to the group of **placeholder without group**. This group cannot be deleted.

Control element	Description
New group	Creates a new placeholder group. A placeholder is assigned to a group using the **Configure** entry in the context menu of the placeholder.
Allocate unit instance	Opens a dialog to assign a unit instance to a placeholder. To do this, all selected placeholders must be linked to the same unit class.
Column selection	Opens the dialog (on page 115) to configure the columns that are displayed
Column format	Opens a dialog (on page 119) to format the columns.

8.8.2 Placeholders and placeholder groups

You can create, edit and configure placeholders and placeholder groups:

- Create placeholder
- Create placeholder group
- Delete placeholder and/or placeholder group
- Allocate placeholder to a group by means of drag&drop
- Configure placeholders and placeholder groups

CREATE NEW PLACEHOLDER

To create a new placeholder:

1. Click on the **New** control element (if configured) or select the **New** entry in the context menu of the list.

   The dialog to select the unit is opened.
2. Select the desired unit class.
3. Click on **OK**.

   The entry is created.
4. If necessary, configure the entry by clicking on **Configure**.
   You can thus amend the name and description and allocate the entry of a placeholder group of the same unit class.

EDIT PLACEHOLDER

To edit a placeholder:

1. Click on the **Configure** control element (if configured) or select the **Configure** entry in the context menu of the list.

   The dialog to configure the element is opened.

2. Amend the name and description.
3. Amend the name and description.
   Only groups of the same unit class are available.
4. Click on **OK**.

CREATE NEW PLACEHOLDER GROUP

To create a new placeholder group:

1. Click on the **New Group** control element (if configured) or select the **New Group** entry in the context menu.

   The dialog to select the unit is opened.

2. Select the desired unit class.
3. Click on **OK**.

   The placeholder group is created.

EDIT PLACEHOLDER GROUP

To edit a placeholder group:

1. Click on the **Configure** control element (if configured) or select the **Configure** entry in the context menu of the list.

   The dialog to configure the group is opened.

2. Amend the name and description.
3. Click on **OK**.

DELETE PLACEHOLDER AND/OR PLACEHOLDER GROUP

To delete elements from the list:

1. Highlight the desired elements.
2. Click on the **Delete** control element (if configured) or select the **Delete** entry in the context menu of the list or press the **Delete key**.

   A request for confirmation is displayed.

3. Click on **OK**.

Note: If a placeholder group is deleted, all placeholders contained therein are assigned to the group of **placeholders without group** again.

ALLOCATE PLACEHOLDER TO A GROUP BY MEANS OF DRAG&DROP

Placeholders can also be added to a group or removed from a group by means of drag&drop. To do this, drag the placeholder to the desired group or from a group of **placeholders without group**.

Drag&drop only works:

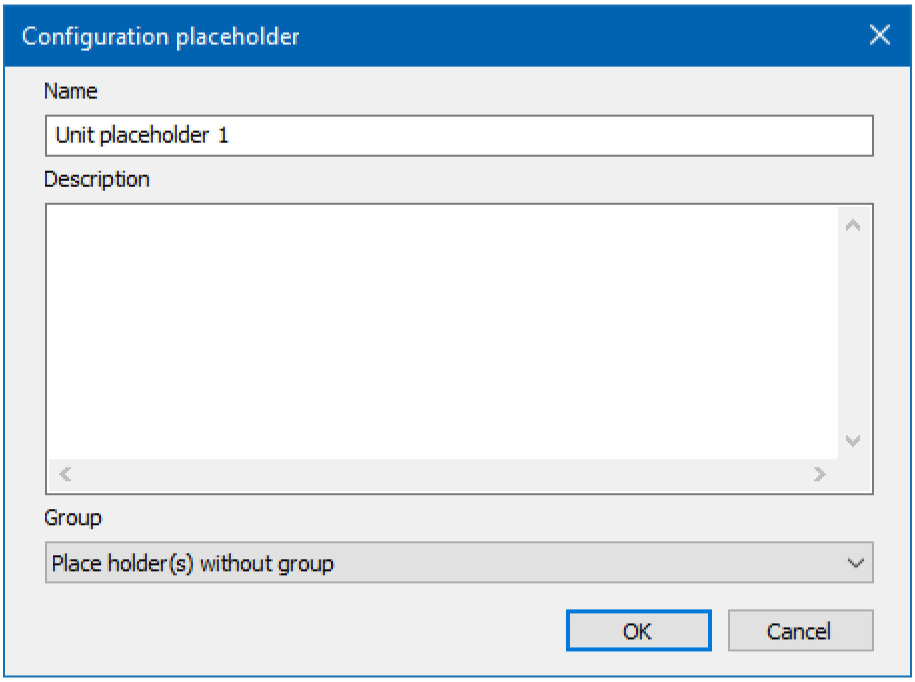
- In master recipes in edit mode.
- If there are appropriate user rights and network tokens.
- If sufficient placeholders have been selected.
  No group can be selected because groups must not contain any subgroups.
- If all selected placeholders are linked to the same unit class.
- If the target group is linked to the same unit class as the selected placeholder or the group of **placeholders without group**.
- The target group is in the same recipe.
  Objects can be moved throughout screens, but only if the lists represent the same recipe.

CONFIGURE PLACEHOLDERS AND PLACEHOLDER GROUPS

Available actions:

Action	Placeholder	Placeholder group
Rename	X	X
Edit description	X	X
Assign a placeholder group	X	--

CONFIGURE PLACEHOLDER



Name: Unit placeholder 1

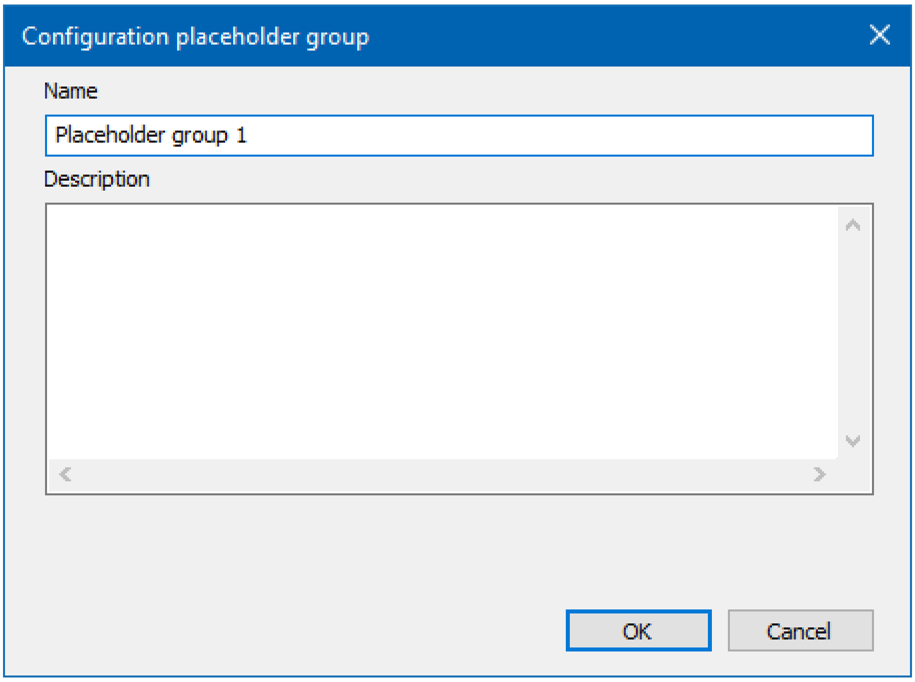
Group: Place holder(s) without group

Option	Description
Name	Name of the placeholder
Description	Description for the placeholder.
Group	Selection of a placeholder group from a drop-down list. Default: *Place holder(s) without group*

CLOSE DIALOG

Option	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

CONFIGURE PLACEHOLDER



Configuration placeholder group

Name

Placeholder group 1

Description

Option	Description
Name	Name of the placeholder group.
Description	Description for the placeholder group.

CLOSE DIALOG

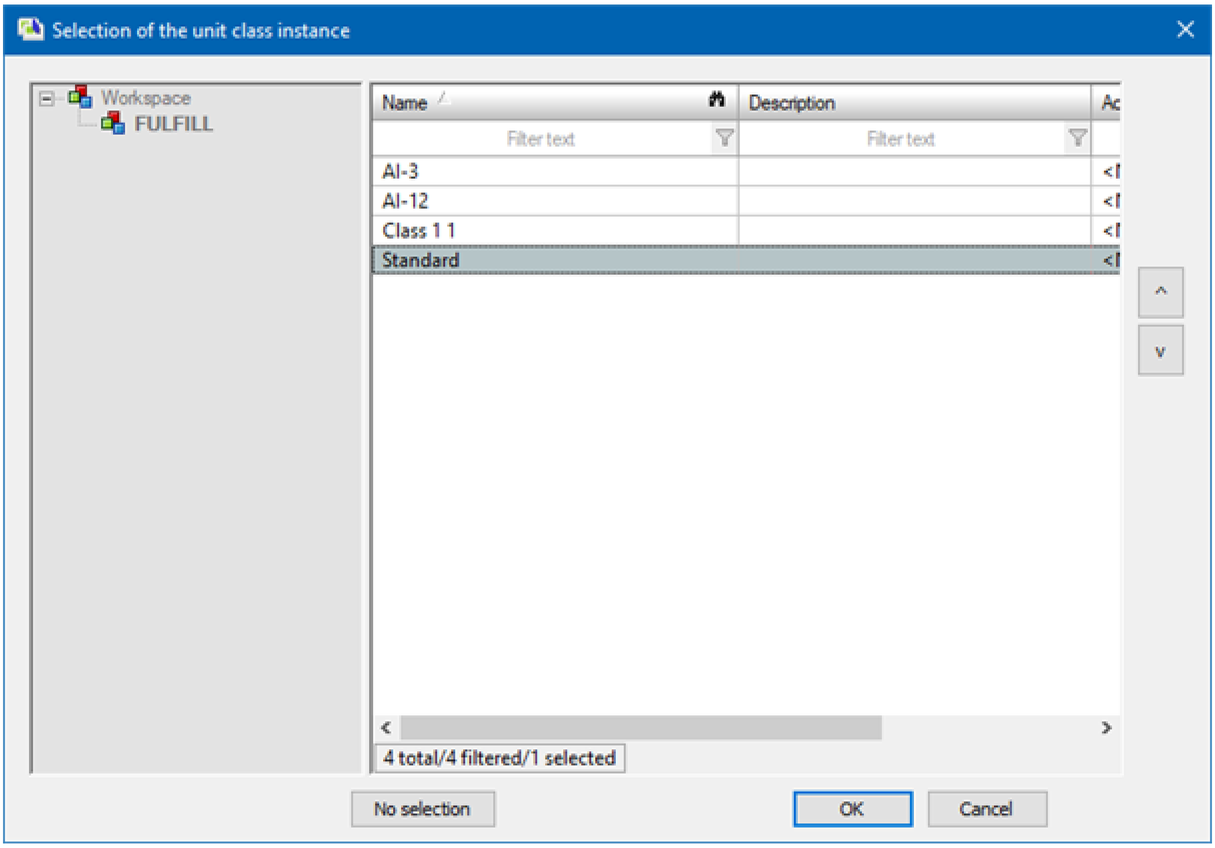
Option	Description
OK	Applies settings and closes the dialog.
Cancel	Discards all changes and closes the dialog.

8.8.3 Allocate unit instance

To allocate a unit instance to a placeholder or a placeholder group:

1. Select a placeholder or a placeholder group in the placeholder list.
2. Click on the **Assign Unit Instance** button or use the corresponding command in the context menu. You can also double-click on the element in the list to allocate it.

The dialog to select a unit instance is opened. All suitable instances for the class of the placeholder are displayed.
If no suitable unit instance is available, a corresponding notice is displayed.



Name	Description	Ac
AI-3		<I
AI-12		<I
Class 1 1		<I
Standard		<I

4 total/4 filtered/1 selected

3. Select the desired unit instance and click on **OK**.

The unit instance is assigned to the placeholder or placeholder group. In doing so, the phases in the recipe are updated accordingly.
Note: To remove a linking, click on **None** and then on **OK**.

> **Attention**
>
> The corresponding function authorizations (on page 297) are necessary for allocation to a placeholder:
>
> - Master recipe:
>   *Batch Control: Master recipe - create, edit, rename, duplicate, save*
> - Control recipe:
>   *Batch Control: Control recipe - edit control recipe*

CHANGE OR BLOCK INSTANCES

Assigned instances can also be changed again later.
Note: If unit instances are removed, the phases are converted to class phases again.

However, changes are no longer possible if:

- A phase of the placeholder or a placeholder of the same placeholder group has successfully carried out the unit allocation
- The unit allocation has been forced manually

As soon as change is possible any longer, the placeholder is blocked for changes. This block is displayed in the list of the placeholders in a separate column. It can no longer be removed by the user.

This block is automatically removed with:

- A switch to test mode
- A switch to edit mode
- Duplication of a recipe
- XML import
- Creation of a new version

RULES OF ALLOCATION

The following is applicable when assigning unit instances:

- Assignments are only carried out if one meets this status:
  - Edit mode (master recipe)
  - Test mode (master recipe)
  - Test in execution (master recipe)
  - Prepared (control recipe)
  - In execution (control recipe)
- Assignments can be amended as long as no unit allocation has taken place.
- If an instance is assigned to a group, all members of the group receive this unit instance.
- If a placeholder is moved into a group, it takes on the unit instance of the group. Exception: If it is moved to the group of **placeholders without group**, it retains the linked instance.
- If an instance that has already been assigned is deleted, the linking is removed and a note of the deleted instance is shown in the placeholder list.
- If an instance is removed from the placeholder list, the data of the class is shown at the phases inserted in the recipe. Local changes are retained.
- If no instance is linked, the data of the class is displayed.
- If an instance is linked, the data of the instance and the local changes are displayed.

- Local control strategies in instances are not displayed.
- If the allocation takes place in the current recipe, a recipe check is carried out. If an error is found, it is shown and the allocation is not carried out.

8.9 Validate recipe

Recipes can be checked automatically or manually for errors during configuration. Recipes are automatically validated for certain actions.

MANUAL VALIDATION

To validate a recipe manually, click on the corresponding symbol in the toolbar of the recipe editor in the Runtime (green tick - **Check recipe for errors**) or on the corresponding button if one has been configured.

The result of the check is displayed as pop-up in plain text. Found errors are also saved in the log file which can be analyzed with the Diagnosis Viewer.

NOTES ON VALIDATION

When validating, the recipe is checked for functionality according to internal rules. The following in particular is checked:

- Syntax (all lines connected, processable from begin to end, etc.)
- Variables
- Data Types
- Parameters (limits)
- Control Strategy: Linking of control strategies and value of the linked control strategy parameter to limits of the variable

Rules which must be adhered to during the engineering can be found in chapter Engineering rules for recipes (on page 158).

> ⚠**Attention**
>
> Operations that are not connected at the time of validation are ignored during validation. Their content and processes are not checked.

VALIDATION OF PARAMETER VALUES

The following is applicable when validating parameter values:

- If the **Tag value** property is active, the parameter values are configured.

- The following is applicable if the **Placeholder for tag value variable** property is active:
  - Real variables must have been used.
  - Configured parameter values are not checked.
  - Parameters for control strategies cannot be linked to **Placeholder for tag value variable** in recipes or operations.
  - In operations: No validation of the parameter value or the variable linking.

RECIPES WITH PLACEHOLDERS

The following is applicable when validating recipes with placeholders:

- All substitutable phases are validated.
  In doing so, the following is validated:
  - When editing: Editor data with the correct control strategy.
  - In the approved recipe: Configuration of the phase saved in the recipe.
  - In approved operation templates: Configuration of the phase saved in the operation template.
- Error message for recipes with placeholder phase: Errors and log entries are displayed with the label **Unit class** instead of with **Unit**.

As long as at least one possible instance exhibits an error, the recipe cannot be approved or started.

VALIDATION OF PHASES IN UNIT CLASSES

The following is applicable when validating phases in unit classes:

- The complete project configuration is not validated. Missing linkings, for example, are permitted.
  Complete validation is only carried out in the phase of the unit instance.
- The **placeholder not linked to instance** reaction must always be correctly configured, including parameters or functions.

VALIDATION OF PHASES IN UNIT INSTANCES

The following is applicable when validating phases in unit instances:

- Only phases that can actually be selected for a unit instance are taken into account:
  The instance must exist in the editor as well as in the saved configuration data of the recipe.
- A check for the existence of at least one available instance is carried out.
- The values that are used after the instance allocation are always used:

  - If a parameter has been edited in the recipe, the edited value is validated.
    Otherwise the default value from the instance is validated.
  - If a limit value is edited, the edited limit value is validated.
    Otherwise the default limit value from the instances is validated.
- The validation is applied to all possible instances of phase that are configured in the recipe.

Syntax error messages: **[Host topology], [operation topology], [potential instance], [error message]**.

8.10 Recipe execution

8.10.1 Commands

The following commands and actions are available in the Runtime:

- Commands effect the recipe process.
- Actions make it possible to edit recipes.

For a command to be accepted by the phase, the following requirements are necessary:

- The REE must run.
- The phase must be active.
- The phase must be in a state in which the command is allowed.

Via multi-selection the command can be sent to several phases in the same execution cycle.

Note: A distinction is made between recipe commands and phase commands:

- Recipe commands affect the execution of all phases active in a recipe, as well as the status of the recipe itself.
- Phase commands are only applied to the selected phase functions (multiple selection is possible).

Command	Description
Start recipe	Starts the recipe process.
Recipe pausing	Pauses the recipe process.
Resume recipe	Resumes a paused recipe.
Hold recipe	Holds the recipe process.
Restart recipe	Restarts all active elements in the held recipe.
Stop recipe	Stops the recipe.

Command	Description
Abort recipe	Aborts the recipe.
Phase pausing	Pauses the phase.
Resume phase	Resumes the paused phase.
Hold phase	Holds phase.
Restart phase	Restarts the held phase.
Escape phase	Abort phase. Only available in manual mode.
Switch to automatic mode	Switches the REE to *automatic mode*.
Switch to semi-automatic mode	Switches the REE to *semi-automatic mode*.
Switch to manual mode	Switches the REE to *manual mode*.
Continue recipe only at selected execution position	Continues a recipe at the selected position.
Continue recipe at all execution positions	Continues a recipe on every available position.
Skip active condition	Skips an active condition. Only possible in the *manual mode*.

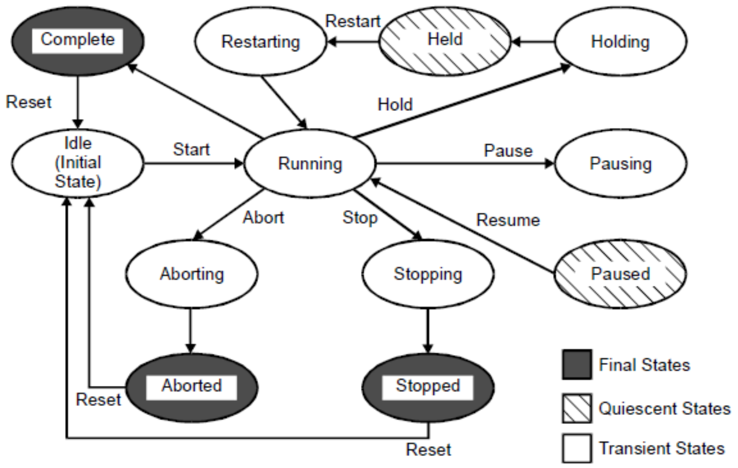
PERMITTED COMMANDS

Execution conditions apply to recipe commands and phase commands. For example, the control recipe can no longer be stopped if it has the status of *Cancel*.

Command		**Start**	**Stop**	**Hold**	**Restart**	**Abort**	**Reset**	**Pause**	**Resume**
Initial State	**No Command End State**	**State Transition Matrix**							
Idle		*Running*							
Running	*Complete*		*Stopping*	*Holding*		*Aborting*		*Pausing*	
Complete							*Idle*		
Pausing	*Paused*		*Stopping*	*Holding*		*Aborting*			
Paused			*Stopping*	*Holding*		*Aborting*			*Running*
Holding	*Holding*		*Stopping*			*Aborting*			
Held			*Stopping*			*Aborting*			
Restarting	*Running*		*Stopping*	*Holding*		*Aborting*			
Stopping	*Stopped*					*Aborting*			
Stopped						*Aborting*	*Idle*		
Aborting	*Aborted*								
Aborted							*Idle*		

Note: The *Reset* command is not implemented in zenon Batch Control.

OVERVIEW OF COMMANDS IN BATCH CONTROL



Note: This overview has been taken from the ANSI/ISA-S88 standard (illustration 18).

Behavior for the Stop command

After a stop command, the phases, transitions and end parallel branch immediately go to stopped status, even if other elements are still waiting for a condition for stopping. Further subsequent commands, such as Cancel are ignored. The Stopped status remains displayed.

BEHAVIOR FOR THE "STOP" COMMAND

After a *stop* command, **transitions** and **end parallel branch** immediately go to *stopped* status, even if other elements are still waiting for a condition for stopping. Further subsequent commands such as *Cancel* are ignored. The *Stopped* status remains displayed.

8.10.2 Recipe status

STATUS FOR MASTER RECIPES

Master recipes can have the following states:

- *Editable*: In this state everything can be changed. Each new master recipe is created in this status.
- *Test mode* (on page 215): In this status the recipe behaves similar as a control recipe. The arrangement of the elements cannot be modified. It can be executed and all commands, actions, modes, etc. are available. You can also change all command tags for which option **Changeable in the master recipe** was activated.
  Exception: If a phase is active at the moment, changes are not possible. Decisive for this is the status at clicking button OK. If the phase is active, the value changes are not taken over and an error message is displayed.
  In test mode recipes can be executed consecutively several times. This is not the case for control recipes.
  Changes from the Editor are only applied after reloading or after Runtime has been restarted, however only once the recipe has been ended.
- *Released* (on page 216): In this status no changes to the recipe are possible.
  Note: Changes to the phases or to the reactions in the Editor are also not transferred to a master recipe. The master recipe always contains the status as at the time of approval. Exception: Changes to variable and function are not considered by these protection mechanisms. This can lead to a master recipe and all control recipes based on it becoming invalid. If e.g. the data type of a variable is changed from Bool to String, the validation function is no longer run through and the recipe can no longer be used as long as the error is not fixed in the Editor. Likewise the units themselves are not protected as they are used recipe-spanning. If a unit name is changed in the Editor, it immediately takes effect after reloading or a Runtime restart on all phases which are based on it.

  To be able to release a master recipe, the recipe must be validated without errors. The validation (on page 249) is done automatically during the release and cannot be avoided. Control recipes can only be created from released recipes.

  If a master recipe with status *Release* should be modified, you must create a copy of the master recipe via command duplicate. The copy gets status *Editable*.

STATUS FOR CONTROL RECIPES

Control recipes can have the following states:

- *Prepared*: A newly created control recipe has this status. In this status, it can be started and command tags, for which option **Changeable in control recipe** was activated, can be changed.
- *In execution*: The control recipe was started and is processed. It remains in this status until it is *Completed*, *Stopped*or *Canceled*. You can learn the exact state of the recipe from the execution status (on page 263).
  In this status, command parameters for which the **Changeable in control recipe** option was activated can be changed.
  Exception: If a phase is active at the moment, changes are not possible. Decisive for this is the status at clicking button OK. If the phase is active, the value changes are not taken over and an error message is displayed.
- *Finished*: The recipe reached its final state. It can no longer be restarted and it also cannot be changed. Changes to command tags are not possible. Finished control recipes can be duplicated and deleted.

8.10.3 Recipe Execution Engine (REE)

The REE (Recipe Execution Engine) executes recipes in the Runtime. You can start any number of recipes.

8.10.3.1 Symbols and Color

The states during the process of a phase are displayed with the help of different symbols. Some symbols are also used for transitions and end parallel branch.

⧗ SYMBOLS FOR THE INNER STATUS:

Symbol	Meaning
⧗	Phase starts.
⟷	Waiting for communication with the controller
⬭	Waiting for unit allocation. The unit of the phase is already being used in another recipe.
🔴	Waiting for unit allocation expired.
🔴	Waiting for exclusive execution. Another instance of the phase is active in the same

Symbol	Meaning
	recipe.
⬇	Writing of the initial tags.
⬇	Writing of the initial tags failed.
⊕	Checking the input interlocking.
⊕	Waiting for input lock expired.
⬇	Writing the value tags.
⬇	Writing the value tags has expired.
🕒	Waiting for minimum execution period.
▶	*Phase:* Waiting for phase done condition. *Transition:* Waiting for transition condition. *End parallel branch:* Wait until execution is finished in all branches.
▶	Maximum waiting period expired.
✓	Waiting for following condition.
⤓	Waiting time for following condition expired.
↻	After the **Restart** command when waiting for the restart condition.
↺	After the **Escape phase** command when waiting for the escape condition.

SYMBOLS FOR ERRORS:

The symbols in the bottom left corner of the phase provide information on the error status of the phase. They can occur ion any desired combination with the symbols for the inner status.

Symb ol	Meaning
⚡	There is a communication error.
⚡	Communication error rectified, wait for acknowledgment of the communication error.

Symbol	Meaning
⚡	Communication error rectified (and acknowledged if required)
🔴	PLC error.
🟢	PLC error rectified.

SYMBOLS FOR "ESCAPE PHASE":

The symbols in the bottom right corner of the phase provide information on the status when escaping from the phase.

Symbol	Meaning
🔴	Waiting for escape condition.
🟢	Escape condition met.

If an error occurs during a phase, the phase is marked as faulty until it is restarted.

STATUS

The execution status (on page 263) of **phases**, **transitions** and **end simultaneous sequence** is signaled in color:

Status	Color
Idle	*white*
In execution	*green*
Executed	*blue*
Stopping	Two colors: - *yellow* - *Original color*
Stopped	*yellow*
Pausing	Two colors: - *orange* - *Original color*

Status	Color
Paused	*orange*
Holding	Two colors: - *Gray* - *Original color*
Held	*gray*
Aborting	Two colors: - *red* - *Original color*
Aborted	*red*
Restarting	Two colors: - *green* - *Original color*
Force *(phases or transitions only)*	*violet border*
Timeout	*red border*

8.10.3.2 Create recipe image

During batch operation, all recipes that are currently being executed can be saved with all recipe data in a recipe image (Image) . Saving is possible as follows:

- *When Runtime is closed*: an image of all running recipes is created automatically.
- *Cyclical*: in a freely-definable time period between *30* and 86400 seconds, a recipe image is written (optional)
- *When activating a phase*: if a phase is activated, arecipe image is written (optional).

These methods can also be combined as desired. It can thus be ensured that Batch operation can also be continued correctly at any time in the event of errors.

RECIPE IMAGE WHEN CLOSING RUNTIME

If the *Exit Runtime initiated* event is triggered, the ending of Runtime is prevented until the Batch Control module has backed up all data. A process image is created that represents the initial status for the restart. Likewise, it is ensured that the parameters of the **Write set value** action arrive at the control unit securely. Internally the phase is paused only when the writing confirmation from the driver ensued. This recipe image contains the images of the REEs, the order of the allocations and all reactions that are needed when restarting. Find out more information in the chapter Exit and restart Runtime.

CYCLIC WRITING OF A RECIPE IMAGE

A recipe image can also be written cyclically during Runtime. As soon as **Activate cyclic writing** is activated, a recipe image is created in the given cycle.

To activate cyclical writing:

1. Navigate to the **Recipe image** group in the properties of the Batch Control module.
2. Activate the **Activate cyclic writing** property.
3. In the **Cycle time [s]** property, create the time period for the writing of the recipe image; the minimum time period is 30 seconds.

With cyclical writing, the last two image files are always retained. Older ones are deleted. Writing is a two-stage process:

1. The recipe image is written to the **TemporaryImg.REE** file.
2. If this was successful, the next version number is issued and the temporary file is renamed to the new version.

With cyclically-created images, all recipes that are currently running are saved in the recipe image. This ensures that the recipe is appropriate to the execution status on restarting. To do this, the recipe from the recipe image overwrites the recipe in Runtime when restarting. It is only possible to restart if the recipe in question still exists and is still in an execution status. The recipe is not restarted if the recipe has already ended after the recipe image has been created.

RECIPE IMAGE WHEN ACTIVATING A PHASE

A recipe image can also be created when activating a phase. To create a recipe image each time a phase is activated:

1. Go, in the properties of the Batch Control module, to the **Recipe image** group.
2. Activate the property **Write recipe image when activating a phase**

A recipe image is written in the Runtime each time a phase is activated.

SAVING AND RESTORING

REE images are stored in the project folder with the following naming convention:
Batch[Version-Hex].REE

With the file extension .**REE**, image files are read in when Runtime is started and the most recent version is identified. The most recent version is the recipe image to be loaded and remains as a file after restarting. All other files with the extension .**REE** are deleted.

8.10.3.3 Behavior of elements in Runtime

The basic principle is: Phases and all elements that follow them (transition, end simultaneous sequence, start simultaneous sequence, allocation) remain active until the next phase becomes active. Exception: A manual skip is carried out.

PHASES

ALL PHASES *PAUSED*

In *manual* mode, it is possible to assign all phases the status *paused*.
For example: The branch continues to be gone through before the *end simultaneous sequence* after the end of this. The active element is thus after the *end simultaneous sequence* and before the next phase. All phases before the *end simultaneous sequence* have the status *paused*. These phases are now only set to *hold* and then to *restart*. The other phases remain paused. If the restarted phase is ended, there is only one active element. If this is activated with the **Next step**, then no active element is present any more. The *end simultaneous sequence* element remains paused however and does not switch through.

Solution: Continuation of the paused phases.

PAUSE AND RESUME

The following applies to pausing and resuming:

- Switching from manual mode to automatic: All paused phases are continued.
- Pausing and resuming with active element: With a *Continue* command, active elements are not continued. Active elements are only continued with the *Continue* phase command, as well as the *Continue recipe only at selected execution position* and *Continue recipe at all execution positions* commands.
- If a phase is *paused* in manual mode and the execution mode is switched to *automatic*, the phase is continued. Switching to automatic execution mode implicitly executes a *Continue phase* command.

  - A *Pause phase* command, followed by a *Pause recipe* and *Continue recipe* command puts the paused phase back into the status of *running*.
- The recipe status changes to *running* after global pausing and resumption. However phases with active elements remain paused and the active element remains unchanged.
- The recipe status remains as *running* after local pausing and resumption. The recipe is resumed at the active elements. This also applies if the active element has been moved.

BRANCHES

The following applies in branchings:

- As long as the left transition does not have a value, the right transition is ignored.
- If the transition condition is met for both transitions, the left branch is selected.

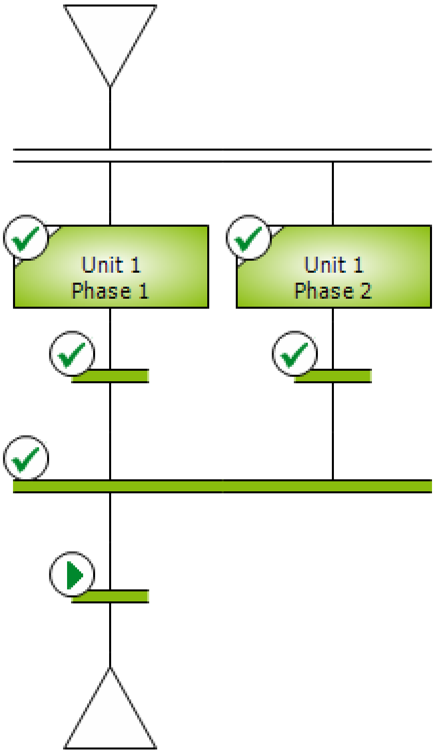
STOPPING ELEMENTS

If an **end simultaneous sequence** is stopped, it becomes inactive immediately and does not react to any more REE commands such as *cancel*. Therefore is also does not change to the status *aborted* after a *cancel* command. The same applies to transitions. In contrast, phases may wait after stopping for further conditions to be met.

TRANSITIONS

- Transitions remain active until a subsequent phase or the end of the recipe has been reached. This also applies if a transition is followed by an End simultaneous sequence and a further transition.
- In branches, impulses for a transition are ignored for as long as the transition to the left of the transition concerned does not have a valid value.

- Transitions are deactivated with *hold* and *restart*. The active element is activated again in manual or semi-automatic mode.



ACTIVE ELEMENT AND JUMP TARGETS

Command	Description
Continue	If an object is paused and an active element is located after it, *continue* has the same effect as *Next step*. This also includes jumps. At a phase command the command only effects a jump in the same branch.
Stop	Depending on the object: - Stop phase: removes execution positions in the branches - Stop recipe: removes all execution positions
Pause	Has now effects for jump targets. Already defined targets remain.
Others	Always causes the deletion of the jumps. For a phase command only the jump in the area of the phase is deleted.

8.10.3.4 Mode and mode change

The REE can run in three modes:

- *Automatic*: The recipe runs entirely automatically.
- *Semi-automatic*: The recipe is executed manually. Conditions cannot be jumped.
- *Manual*: Each step in the recipe or operation is executed manually; conditions that are being waited for can be skipped.

  Note: Not all steps can be skipped, for example if placeholders are no longer linked to a unit instance. If a step cannot be skipped, a corresponding message is shown.

To execute a recipe manually or semi-automatically, the operation types (on page 265) **Step-by-step execution of the recipe** and **Jump** can be used.

When switching to *automatic* mode, all execution positions are removed. Global commands are only executed in branches that have no execution positions.

> **Information**
>
> To react on serious events, you can change the mode during the running process via reaction type (on page 58) *influencing the recipe*.

8.10.3.5 The execution status

The following statuses are possible for recipes:

Status	Description
Idle	The REE is in idle state.
In execution	When starting a control recipe, it changes to the status *running*. However, individual elements in the recipe can have a different status.
Executed	A recipe changes to *Completed* status as soon as: - As soon as the end element has been reached in a PFC recipe or - in a matrix recipe, all phases have been fully processed In this status execution is not possible.
Pausing	After a **Pause** recipe command, the recipe is in the *pausing* status. It remains in this status until all phases in execution are in the *paused* status. Only then does the recipe get the status *paused*.

Status	Description
Paused	All active phases of the recipe are paused or held. Within the phase the process stops at: - Waiting for unit allocation. - Check for parallel execution. - Waiting for input interlocking. - Waiting for phase done condition. - Waiting for following condition.
Holding	After a **hold** recipe, the recipe is in *held* status. Only once all active phases are in the *held* status does the recipe also switch to the *held* status.
Held	All active phases of the recipe are held. Within the phase the process stops at: - Waiting for unit allocation. - Check for parallel execution. - Waiting for input interlocking. - Waiting for phase done condition. - Waiting for following condition.
Restarting	After the **Restart recipe** command, the recipe is in the **Restarting** status. Only once all previously-held phases in the recipe are started does the recipe status also switch to *in execution*.
Stopping	After a **Stop recipe** command, the recipe is in *Stopping* status. Only once all active phases are in *stopped* status does the recipe status switch to *stopped*.
Stopped	All active phases in the recipe have been stopped. A stopped control recipe cannot be continued again or restarted.
Aborting	After an **Abort recipe** command, the recipe is in the *aborting* status. Only once all active phases are in *aborted* status does the recipe status switch to *aborted*.
Aborted	All active phases in the recipe have been aborted. An aborted control recipe cannot be continued again or restarted.
Prepared	Prepared for execution.

8.10.3.6 Step-by-step execution of a recipe and jumps in the recipe

STEP-BY-STEP EXECUTION OF A RECIPE

A recipe can be executed step-by-step if:

- The recipe is in either semi-automatic or manual mode
  and
- The status of the recipe is *running*.

For the step-by-step execution the execution is paused as soon as an element is finished with its execution. The holding is done via the command *Pause* to the concerned execution path. As soon as all active elements in this path have reached the status of *paused*, the active element is marked by a red arrow. Operations are treated like all other objects of a recipe.

The execution is resumed with:

- a selective step: selection of the corresponding arrow (green)
- A global step: all positions with arrows for possible resuming are started

COMMUNICATION ERROR WITH STEP-BY-STEP EXECUTION

If an communication error occurs with a phase whilst this is waiting for an active element, the active element is no longer displayed until the problem is rectified. However, if in the meantime, the execution is switched to *automatic* mode, then the phase is no longer paused and must be continued. This also applies if a switch to manual or semi-automatic mode is made again.

COMMANDS

GLOBAL COMMANDS

For global commands all execution positions are deleted as the execution cannot be resumed from there.

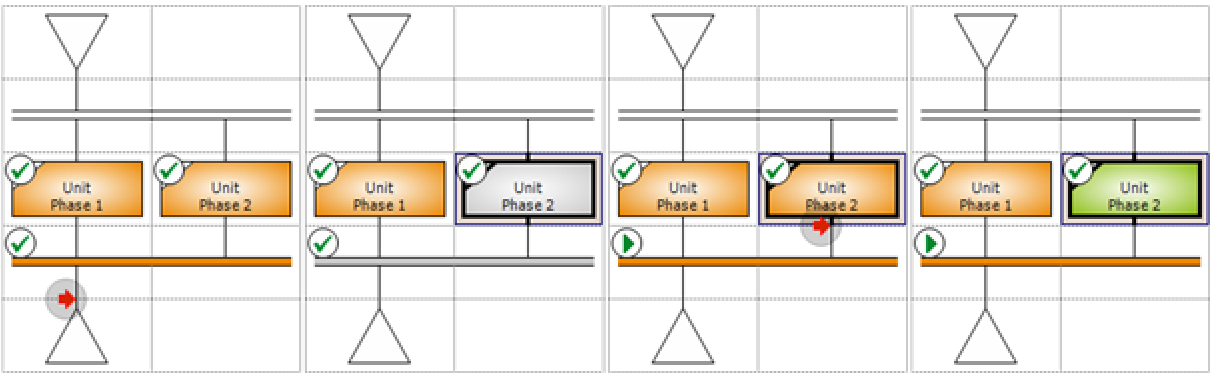
Exceptions: *Pause* and *Resume*. The execution positions remain as they are.

PHASE COMMANDS

- *Hold* for a phase: The active element is deleted from the execution path of the phase.
- *Resume*: If there is an active element, a selective next step is executed in this execution path.

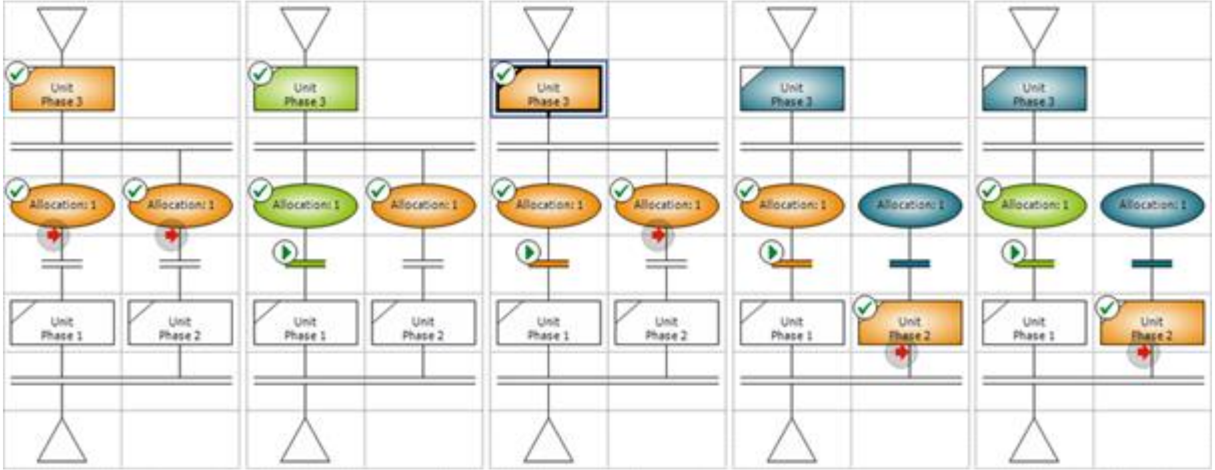
SPECIAL CASES:

PHASE COMMAND HOLD AND RESTART IN A PARALLEL BRANCH FOR ACTIVE END PARALLEL BRANCH



If you hold in this example (images from left to right) a parallel branch and restart it, then you will reach after a step-by-step execution the already paused **end parallel branch**. To resume the execution from here, the left phase must receive command *Resume*.

ONLY ONE PATH IN A PARALLEL BRANCH WITH AN ACTIVE PHASE BEFORE STEP-BY-STEP EXECUTION



If, in a parallel branch with a phase before, only one path is executed completely and waiting is taking place in one of the other parallel branchEs (phase before is *running*), no active element is displayed in the parallel branch. To get them in the other path, the phase must be paused with a phase command. After that it is possible to execute the path completely.

However, if the parallel branch with the active element continues to be executed, the phase before the parallel branch is deactivated. The left path thus remains *paused* and without an active element. As there is not active phase, the execution can only be resumed with a global *Pause* and *Resume*.

JUMP

Jump means to move from one position to another, distant position during execution in order to continue the execution there.

To jump:

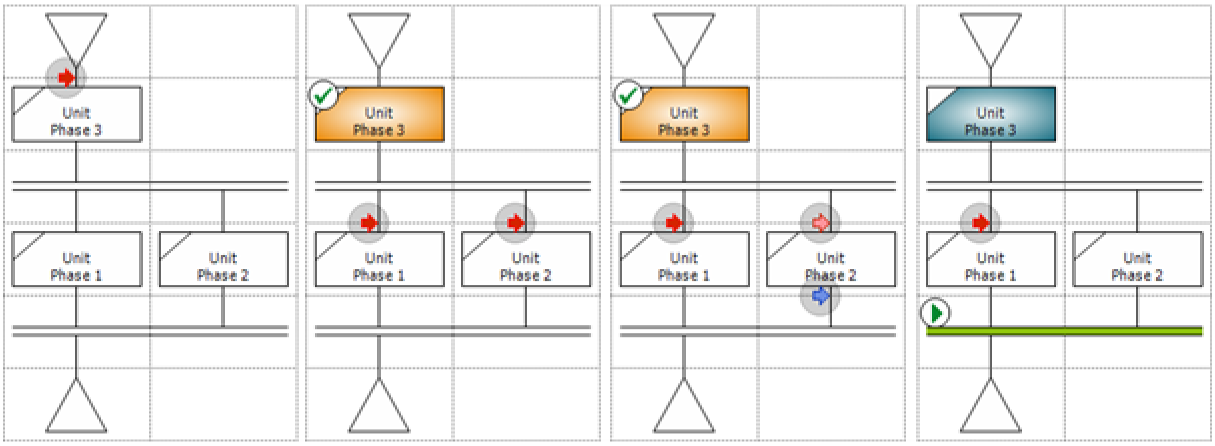
1. Select an active element with the mouse cursor.
2. Move it to one of the offered targets.
3. Execute **Continue recipe only at selected execution position**.

All active elements in front of the source pointer are deactivated and the object after the target is activated.

If a jump is registered for which source and target are analogously the same (jumps over lines, jump targets or end branch objects), this jump is ignored and a simple step is executed.

JUMPING IN SIMULTANEOUS SEQUENCES

If, in parallel branches, a jump over the first phase in the parallel branch is made, the phase before the parallel branch is deactivated. Therefore no phase is active.



The phase before the parallel branch is executed and then a step is executed so that active elements are displayed in both parallel branches. In the right branch, the active element is moved after the phase. If the step is now executed, the phase is deactivated before the parallel branching.

Rule when jumping:

- Objects before a start parallel branch become inactive as soon as objects become active afterwards.
- The first phase thus becomes inactive when skipping.
- The skipped phase in the parallel branch has never become active, likewise the following phases.
- Thus no phase is now active in the recipe.

8.10.3.7 Process of a phase in detail

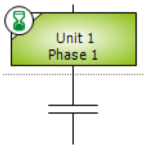
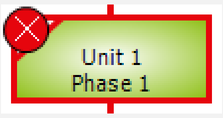
A phase is always processed sequentially after the same pattern. To break down the exact process, you also need a following condition. For this display we use a transition as following condition. You can find additional special process behavior with following condition in chapter Following condition (on page 273).

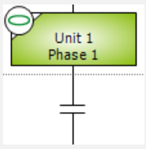
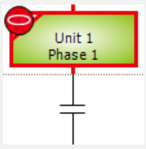
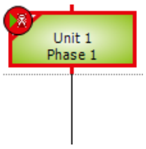
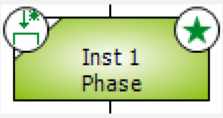
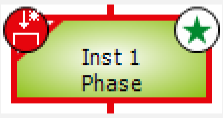
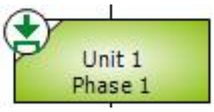
PHASE PROCESS

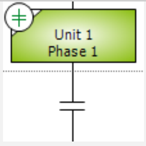
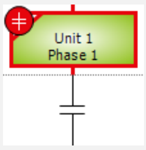
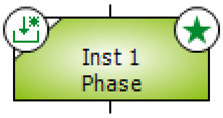
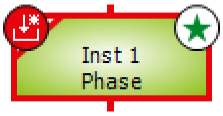
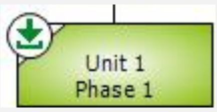
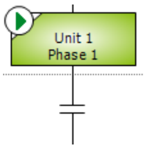
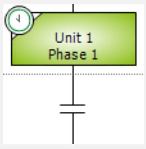
Border colors of phases in the process:

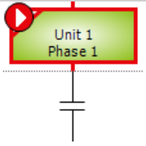
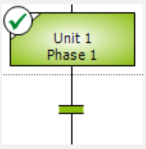
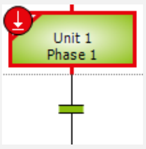
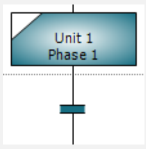
- *Blue*: only for placeholder functions: no unit instance is assigned to the placeholder. The phase cannot be executed.
- *Green*: only for placeholder functions: a unit instance was assigned to the placeholder. This can still be changed.
- *Black*: Allocation can no longer be used.
- *Red*: Error.
- *Purple*: a condition was skipped.

Note: Red and purple borders have priority.

Phase/Transition	Event
Phase is activated  **Note:** With placeholders, a unit instance must be linked for this.	*Phase activated*
Checking of limit values for parameters: A check is carried out to see whether all values for parameters are within the limits. In the event of an error, there is a wait until all values are within the limits.  	In the event of errors: *Validation error:Values outside the limits*
All variables of the phase are registered at the drivers.	

Phase/Transition	Event
Unit allocation is started and *waiting period unit allocation* is started.  	If the unit allocation was not successful in the first try: *Unit allocation not possible* Unit allocation not possible within the waiting period: *Waiting period unit allocation exceeded*
Validation: Is another instance of the phase already being executed? This can happen if the phase is already active in a parallel branch or if the unit allocation was skipped manually and the phase is already being executed in another recipe. In the event of errors: Phase waits. 	*Phase started multiple times*
Waiting for confirmation of the changes before writing the initial parameters.  Timeout: 	Only if the **Immediately write value changes of parameters** property is not active: There is a wait until all changes made in the parameter have been confirmed.
Start sending of the initial parameters and phases. 	*Phase started*
Check of the input interlocking is started and *Waiting period input interlocking* is started.	If the checking of the input interlocking was not successful in the first try: *Input lock blocked* Condition of the input

Phase/Transition	Event
 Timeout: 	interlocking not *TRUE* within the waiting period: *Waiting period input lock exceeded*
Waiting for confirmation of the changes before writing the value tags.  Timeout: 	Only if the **Immediately write value changes of parameters** property is not active: There is a wait until all changes made in the parameter have been confirmed.
Writing the value tags. 	When all value tags have been written: *Finished writing value TAG*
Checking of the phase-done condition is started and time for **Minimum execution duration** and **Maximum execution duration** is started. 	
Optional: Phase-done condition fulfilled but **minimum execution duration** not reached. 	
Optional: Condition of the **Phase-done condition** not *TRUE* within the **Maximum execution duration**.	*Maximum execution duration exceeded*

Phase/Transition	Event
	
Phase-done condition is *TRUE* and **minimum execution duration** has been reached or exceeded.  **Waiting period following** condition is started. Transition is activated	*Phase finished*
All variables of the transition are registered at the drivers.	
Optional: **Following condition** not within waiting period *TRUE*  The transition condition is checked.	*Waiting period following condition exceeded*
The next phase is activated. The following condition can be composed from several objects (e.g. transition + unit allocation). No till the next phase is reached (or the end of the recipe), the following condition counts as fulfilled.	
Phase is informed that the following condition has been met. Transition condition is *TRUE*. - All events of the phase are deactivated. - All variables of the phase are signed off from the drivers. - All variables of the transitions are signed off from the drivers. - The phase is deactivated. - The transition is deactivated. 	*Phase deactivated*

Attention

If a placeholder phase is not linked to a unit instance at the time of execution, the result *placeholder not linked to instance* is set.

GLOBAL PAUSING AND CONTINUATION IN THE EVENT OF COMMUNICATION PROBLEMS

If a phase is paused and there is a communication error, this cannot be simply continued. Phase commands are prevented, recipe commands are circumvented. If the recipe is paused, a *Continue recipe* recipe command can be sent. As a result of this, the recipe changes to the status *running*, but the execution path of the phase concerned remains unaffected.

8.10.3.7.1 Transitions process

When starting a transition, a check is carried out to see whether all parameters are linked to phases of units or unit instances. Placeholders without linking are not permitted.

Procedure:

- If all placeholders are linked correctly, all placeholders concerned are blocked for changes to the unit instance and the transition is executed.
  Caution: This block also takes effect if the placeholder has not yet been allocated.
- If there is at least one non-linked placeholder, there is a wait until all placeholders are linked. This is displayed with the status *at least one used placeholder has not yet been allocated a unit instance*. The transition is shown with a red border in this status. The error is shown in the tooltip. The border remains red after the error has been rectified.
- Skip: The symbol to skip is activated in the tool bar, but acts differently for:
  - Waiting for the allocation of a unit instance: When the symbol is clicked on, a notification is displayed stating that this step cannot be skipped.
  - Waiting for the condition: It is still possible to skip. Affected placeholders are blocked for the instance allocation if there is a wait for the condition.

RULE FOR VALUES OF TRANSITIONS

If the condition of a transition is *TRUE* whilst the transition is active, the transition is marked as ended. If its value should change to *FALSE* later, the execution of the recipe is not influenced.

8.10.3.7.2 Following condition

Once a phase has been executed, it remains active whilst the recipe process is continued. Only once the next phase is activated or the end of the recipe has been reached is the phase deactivated. This is marked as a **following condition**. This is also applicable if, between the phase and the following phase (or the end of the recipe), there are other elements such as a **transition** or **end simultaneous sequence**. This behavior ensures that the unit allocation remains after execution of a phase up to the start of the next phase.

8.10.3.8 Allocate unit

Phases allocate a unit during execution.

The allocation of a unit by a phase is only possible if the unit has not yet been allocated or has only been allocated in the same recipe. The unit allocation remains in place as long as the phase is active.

> **Information**
>
> A phase with the *paused* or *held* status does not attempt to allocate the unit.
> This also applies if the phase is switched to *paused* or *held* whilst waiting.
>
> An attempt to allocate the unit is only made after a restart.
>
> But: In manual mode you can force the allocation of a unit by another recipe. The recipe with the first allocation keeps the control and takes priority at the execution. If this recipe withdraws its main allocation, the recipe with the longest active forced allocation takes over the main allocation.

STARTING A PLACEHOLDER PHASE

Action when starting a placeholder phase

- The unit is allocated when a phase is started.
  With the exception of when one of these conditions is met:
  - Placeholder phase without assigned unit instance.
  - Execution status: *Paused*, *pausing*, *holding* or *held*.

  No allocation is carried out in this case.
- In all other cases, an attempt is made to allocate the unit immediately.
  This can lead to the following statuses:
  - The phase is started. However, a unit instance is not yet linked for the placeholder.
  - When waiting for the assignment of a unit instance, the phase is paused or held by the user.

  - A unit instance is allocated to the placeholder of the phase. The phase thus restarts.
  - The phase is immediately paused or held again.
    The unit is then not yet allocated. The placeholder is not yet blocked for changes.
  - The assigned unit instance can still be changed.
- If a placeholder phase is activated, the previously-active phase is deactivated.This can lead to no unit being allocated during the waiting period

WAITING PERIOD FOR UNIT INSTANCES

For the allocation of a unit instance to a placeholder phase function, a waiting period can be set and linked to a reaction. The operator is thus notified if a recipe cannot run because a placeholder phase has not been assigned a unit instance.
Note: The waiting period is only started if a placeholder phase is activated.

To be notified in the Runtime:

1. Configure the waiting time in the Editor in the **Unit class**/**Instance allocation timeout** property of the phase of the unit class.
   Note: The property is only available for phases of unit classes, not for phases of unit instances.
2. Configure the *maximum waiting time for instance allocation exceeded* (on page 50) reaction.

During recipe execution, an error is triggered if the placeholder phase has not been assigned a unit instance after the set waiting time.

8.10.3.9 Escape phase

It is possible to exit a phase during execution. Execution is then continued after the phase without having to run through the outstanding steps of the phase. It is only possible to exit a phase in manual mode. In the Editor, it is also possible to configure a condition in the **Condition transient status**/**Escape condition** property.

It is also possible to exit if the phase is in this status:

- *In execution*
- *Pausing*
- *Paused*
- *Holding*
- *Held*
- *Restarting*

If the exit command has been reached, the current execution step is canceled and checking of the **Escape condition** starts immediately. If the condition exiting is met, all necessary steps are instigated so that the object changes back to the status of *running*:

- *Restarting*: The normal procedure is continued from the **Wait until the recipe has the status "***running***"**. That means: None of the steps between instigating the exit and **Wait until the recipe has the status "***running***"** are executed.
- *In execution*: The normal procedure continues to be executed after the phase or the phase waits for the *pause* condition to be met.

No more transient conditions are checked from the start of the check for the exit of a phase. Only after a switch to *stopping* or *aborting* is the checking of **Escape condition** interrupted.

Behavior when exiting from the phase depending on the execution status:

- Exit from the phase from *holding* or from *held*:
  A restart is only carried out internally when switching back to *running*. The execution data is retained with the object. This also applies if a restart is carried out during the checking **Escape condition**.
- Exit from the phase if the recipe status is not *running* or *restarting*:
  If there is a different recipe status, once the condition has been met, waiting continues until the recipe status changes to *running* or *restarting*. This also makes it possible to exit with a different recipe status.
  For example: The recipe has the status *held*. The *exit* action is carried out for a phase. Then the end of the exit is only reached if the user executes a global *restart* command. This also makes it possible to exit if the recipe has the status *held*, without using the normal mechanism for a restart.

> **Information**
>
> If an error occurs with communication to the PLC, the check of the exit conditions is postponed until the error has been rectified.

SUPPRESSION OF REACTIONS WHEN EXITING

Most events for the status change and mode change are suppressed whilst a phase is exited. The checking of the **Escape condition** has priority and the status change is also carried out without the condition being checked. Exceptions to this are *stopping* and *aborting*, as well as *End Runtime triggered* and *Runtime restart*, because these have a higher priority than *Exit without phase*. These events are triggered, as well as the event for *Linked variable invalid*.

8.10.3.10 Restart phase

Phases can be restarted. In doing so, all connected active objects are deactivated, including connected simultaneous sequences. A phase always carries out a series of event for *Activation*, *Start* and *Finished*, regardless of the number of restarts. Events that have already been carried out are skipped on restarting. Transient conditions are only checked after the *start* event.

DEACTIVATION

At the restart of a phase, all active objects are deactivated in principle, however only if they are linked. Isolated active objects are not deactivated.

SEQUENCE OF DEACTIVATION

The deactivation starts from the object which has been active the longest. After the restart this object is also going to be activated. If this object is deactivated, it also deactivates all branches to which it can establish a connection via an active object.

ISOLATED BRANCH

A branch is isolated if it is not connected to another branch via an active object. The deactivation only takes place between connected branches. As long as an object does not have status *Completed*, there is not active connection to the following object.

RESTART OF SELECTED PHASES

One or more phases can be selected and restarted. For the selection of several phases, they must be in separate branches.

The oldest object is restarted in the selected branch. With this all active objects in the connected branches are deactivated.

GLOBAL RESTART

The global restart carries out a restart for all phases. The restart is done for the oldest active object and with this all connected, active objects are deactivated. All remaining active objects are in an isolated branch. Here also the oldest object is restarted until all active objects were dealt with.

8.10.3.11Secure writing of the command parameters

The command parameters (initial parameters and value parameters) are sent to the PLC securely. The waiting time can be configured in the Editor.

PROCEDURE

The following applies when writing command parameters:

1. Waiting occurs until all parameters to be written as inverted have a value

- If this is not possible within the configured waiting period, 3 attempts are made.
- If there are still parameters with no value, no parameter is written, not even those with a value.

2. If all attempts have been unsuccessful, the *command parameter without value* event is triggered.
3. Command tags are written.
4. The actual values are compared to the written values.

   These are read in again after the parameter values have been written. If the values read after the set waiting time do not correspond to the written values, all parameters are written again. There is a maximum of 3 repetitions.
5. If secure writing is unsuccessful, a *communication failure* event is triggered. That means:
   - The error must (if configured accordingly) be acknowledged.
   - The phase must be put back to the status "running".
   - Writing is restarted once continued.
   - The execution of the function is restarted in the event of a restart.
6. The procedure can be paused, held etc. using commands whilst secure writing is being carried out. If the phase has the status *Paused* and it is then continued, the writing is also restarted. In doing so, the values to be inverted are recalculated for a command parameter to be toggled.
7. Writing of the command parameters can also be skipped.
8. If the Runtime is ended whilst writing the command parameters, these are rewritten when Runtime is started again. Parameters to be toggled are recalculated.

Duration, start time and end time of the writing are displayed in the tooltip of the phase.

CONFIGURING THE WAITING PERIOD

To configure the waiting period for secure writing:

1. Navigate to the **General**/**Protected writing** group in the properties of the Batch Control module.
2. Enter the desired waiting period in the **Time out for protected writing** property.

Information

The waiting period includes all waiting processes in the whole write process:

- Waiting for values for all parameters to be written.
- Waiting for confirmation of the written parameters.
- Waiting for reading of the variables.

If the default value of 20 seconds is used, all wait processes within 20 seconds must be concluded positively. If the waiting period has expired without a positive result, writing is started over.

VALUE CHANGE DURING EXECUTION

Command tags can each be processed between the start of writing and a successful check of the input locking or phase done condition and written to the PLC. Amended values can be written immediately or in bundled form.

To change values during the execution of a phase:

1. Activate the **Tag values changeable during phase execution** property in the Editor.
2. Decide whether values are written immediately or in bundled form after confirmation.

WRITE VALUES IMMEDIATELY

To write values immediately:

1. Go to the **Value change during execution** properties group for the phase.
2. Activate the checkbox for the **Immediately write value changes of parameters** property. The **Timeout for value change acknowledgement** property is thus automatically deactivated.
3. Configure the required events
   - *Initial TAG modified during process*
   - *Value TAG modified during process*
   - *Amended initial TAG fully written*
   - *Amended value TAG fully written*

Value changes do not need to be confirmed in the Runtime. If a phase is in the corresponding stage (check of the input lock and/or check of the phase done condition), amended values of initial parameters or value parameters are written immediately after the change. A corresponding symbol and information are displayed in the tooltip.

WRITE VALUES IN BUNDLED FORM

Amended values can also be collated and written to the PLC manually by clicking on a button.
To write values in bundled form:

1. Go to the **Value change during execution** properties group for the phase.
2. Deactivate the **Immediately write value changes of parameters** property.
3. Configure the **Timeout for value change acknowledgement** property.
   You thus stipulate how much time can pass between the first change of a parameter and the writing of the values by clicking on a button in the Runtime. The corresponding result is triggered after expiry of the waiting time. This informs the operator that amended values have not yet been sent.
   The value *0* means: no waiting time.
4. Configure the required events
   - *Initial TAG modified during process*
   - *Value TAG modified during process*
   - *Amended initial TAG fully written*
   - *Amended value TAG fully written*
   - *Waiting time for confirmation of the value change of the initial TAG exceeded*
   - *Waiting time for confirmation of the value change of the value TAG exceeded*
5. Configure the **Confirm Changes** button in the screen.

Note: If parameter values are amended during recipe execution, this is indicated by a symbol to the phase. As long as the amended values have not been confirmed with the **Confirm Changes** button, they are not written to the variables. This can led to execution of the function being blocked.

BEHAVIOR IN THE RUNTIME

The setting for the **Tag values changeable during phase execution** property is noted when writing the parameter.
If a phase is still in the validation phase during execution, parameter values can be amended regardless of the property's setting.

Tag values changeable during phase execution Checkbox setting:

- *Active*:
  - Parameter list in the screen: Keyboard is called up. Changes are possible in the dialog to edit the parameters.
  - Edit phase dialog: Changes are possible in the dialog to edit the parameters.
- *Inactive*:

- Parameter list in the screen: Keyboard is not called up. The dialog to edit the parameters is write protected.
- Edit phase dialog: The dialog to edit the parameters is write protected.

8.10.3.12 Exit and restart Runtime

ACTIONS ON RESTARTING

Actions can be predefined for restarting Runtime after closing. These can be defined for:

- **Restart after normal shutdown of the Runtime**
- **Restart on system failure**

One of the following actions can be selected for each of the two properties:

- *Hold recipe*: The complete recipe is held after restarting.
- *Recipe pausing*: The complete recipe is paused after restarting.
- *Retain recipe status*: After restarting, the recipe is set to the same status as before closing.

STATUS CHANGE

After restarting, an attempt is made to execute the configured status change. To do this, the corresponding command must also be executable. The status *Restarting* for recipes and phases is handled in the same way as *in execution*. That means:

- *Paused* is set for: *In execution*, *pausing* and *restarting*.
- *Held* is set for: *In execution*, *pausing*, *paused*, *holding* and *restarting*.

Transient conditions are not checked and events are not set. Therefore the status in the recipe can be brought in line with the status of the equipment, without sending events to the equipment for the status change.

IMAGE FILE

At closing the Runtime an image file (on page 258) of the running recipe is created. It contains the images of the REEs, the order of the allocation and all reactions which are needed for the restart.

ALLOCATIONS

After the restart the allocations match the state before the finishing. It is saved who allocated a unit and who forced an allocation in which order. If a recipe cannot be restarted (e.g. because of failed validation), the allocations for this recipe are removed.

REACTIONS

Reactions which were triggered by the process are also incorporated in the image if they are active. They are then executed after the REE is restarted. This guarantees that the reaction is always executed as a whole either before the image file is created or after the restart.

The *Exit Runtime* reaction is always executed and can never be incorporated in the image.

SYNCHRONOUS WRITING

The REE manages the confirmation for all variables whose write set value should be executed synchronously. The time out for this is defined by the time within which the Runtime must be closed. For each write acknowledgment the time out is restarted. This restart is logged in the LOG file.

Variables that do not access a driver are always written without confirmation, even if write confirmation is requested. Internal drivers do not support acknowledgments.

ALLOCATE TAG

As during the start of the Runtime all drivers are also started, it is possible that they do not provide valid values if they are needed at the restart. During the restart it is not waited for the value update. This does not ensure that the value is written as expected. If no value is available, the alternate value is used.

There is a wait for the values of internal drivers if they are available within 2 cycles.

CHECKING FOR A COMMUNICATION ERROR

A check is also made for communication errors when restarted if this has been configured (on page 343).

8.11 Manage recipes

The entire recipe management is done in the Runtime with the help of one or several screens of type *Batch Control*. Due to suitable filter settings you can achieve already filtered views for master recipes or control recipes.

In the screen different control elements (buttons, lists, editors) are available for different tasks. The screen can also be configured so that it is divided into three main areas which can be also used completely independently from one another:

- Area master recipes (on page 282): Consists of a list and buttons for managing. The area can be used completely independently.

- Area control recipe (on page 286): Consists of a list and buttons for managing. The area can be used only together with the **list of the master recipes** as a master recipe must be selected first before the appertaining control recipes are displayed in the list.
- Recipe editors: Depending on the set recipe type the PFC editor (on page 166) or the matrix editor (on page 208) is used.

8.11.1 Manage master recipes

LIST AND LIST FORMATING

List/action	Description
Master recipes list	In this list all master recipes can be displayed. The display can be limited by filters to an individual selection. The filtering can be preset in the zenon Editor in the screen switch function (on page 83). Filtering in Runtime is also possible. These filter settings are not saved in the Runtime. This means that the filters must be entered again when called up again. A permanent pre-setting of the filter is only possible with configuration in the zenon Editor. All commands are also possible in the context menu of the list. The commands for list management can be called from the header of the list. The commands for recipe management can be called at editing one or more recipes. The recipes in the list cannot be edited directly in the list. Renaming, changing the description or changing the recipe status is only possible with the corresponding commands. **Note for ideal configuration of the list:** The list can be designed diversely concerning content and look: - **Content:** The displayed columns (on page 115) can be selected, the column format (on page 119) (column width, alignment, label) can be changed and you can define filters (on page 88). These settings can be edited in Editor and Runtime. - **Graphics:** At the settings of the list in the Editor you can find diverse setting possibilities in areas **Representation**, **Scroll bars** and **Fill**. With these properties you can even design the list ready for touch operation. Longer texts can also be displayed in lists over several lines using the

List/action	Description
	Automatic word wrap property. The line height must be amended manually in order to do this. - If you want to edit the list directly using the monitor, activate the Multi-Touch functionality. You can find detailed information in relation to this in the Configure interactions chapter.

ACTIONS FOR LIST MANAGEMENT

Action	Description
Column selection master recipe...	Opens a dialog in order to determine which columns should be displayed (on page 115). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.
Format columns master recipe...	Opens a dialog to edit the column settings (on page 119). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.

ACTIONS FOR RECIPE MANAGEMENT

Action	Description
New master recipe...	Creates a new and empty master recipe in status *Editable*. The dialog (on page 162) for entering a unique name and a free description is displayed. The uniqueness of a name is also checked in the zenon network. Depending on the licensing, there may also be the possibility to select the recipe type: Matrix recipe (on page 207) or PFC recipe (on page 165). If only one of these recipe types is licensed, the licensed recipe type is fixedly set. The selected recipe type cannot be changed afterwards. When creating a master recipe, a CEL entry is created.
Create new version	Master recipes can also be versioned. In doing so, a copy of an approved or obsolete report is created. This copy is in edit mode and contains a unique version number. The new recipe can be edited, but not renamed. Individual versions, including the source recipe itself, can be deleted.

Action	Description
Duplicate master recipe	Only active if exactly one recipe is selected. Created a copy of the selected recipe. At the creation of the copy, the version of the recipe saved on the hard disk is used. If the recipe is just edited in another computer and the changes have not yet been saved, the changes are not applied. The dialog for the input of a unique name and the description is opened. The copy of the recipe automatically receives status *Editable* and can be edited further. When duplicating a recipe, a CEL entry is created.
Delete master recipe	Deletes the selected recipes irrevocably. If the recipe is opened on another computer for editing, it is automatically closed there. Deleting is only possible if there are no control recipes which are based on the master recipe. First you must delete all control recipes. Recipes which are currently executed in test mode (master recipe status: *Test in execution*) cannot be deleted. First they must be *finished*, *stopped* or *canceled*. If recipes must not be deleted - e.g. in an FDA-regulated environment - it is recommended that this button is not configured or that it is given an appropriate **Authorization level**. A CEL entry is created when a recipe is deleted.
Export selected as XML...	Exports the selected master recipe as an XML file.
Import XML	Imports the selected XML file as master recipe.
Configure master recipe	Only active if exactly one master recipe was selected. The dialog for the input of a unique name and the description is opened. Recipes can only be renamed if they are in status *Editable*. Also use this function in order to change the description of the control recipe. When renaming a recipe, a CEL entry is created.
Open master recipe	Opens the selected master recipe in the recipe editor if screen element *Recipe editor* exists in the screen. Each selected master recipe is opened in a separate tab of the recipe editor.
Release master recipe	Changes the master recipe status of the selected recipes to *Released*. Only recipes without errors can be released. If error

Action	Description
	occur during the validation (on page 249), you must first fix them. Only recipes in status *Test mode* and *Editable* can be released. Released recipes can no longer be edited. Control recipes can only be created from released recipes. For details about the states see chapter Recipe types and recipe states (on page 160). When releasing a recipe, a CEL entry is created.
Test master recipe	Changes the master recipe status of the selected recipes to *Test mode*. Only faultless recipes can be switched to test mode. If error occur during the validation (on page 249), you must first fix them. Recipes in the test mode can be executed but no longer reengineered. For details about the states see chapter Recipe types and recipe states (on page 160).
Edit master recipe	Changes the master recipe status of the selected recipes to *Editable*. In this status, recipes can again be edited completely. Only recipes in *Test mode* can be set back to *Editable*.
Highlight master recipe as outdated	Changes the status of the recipe to *outdated*. The recipe can no longer be edited or approved. No control recipe can be created on the basis of this recipe.
New control recipe...	Opens the dialog (on page 233) for entering a unique name and a description for the control recipe. The uniqueness of the name is also checked in the zenon network. The name must only be unique within the master recipes. Control recipes which are based on other master recipes may have the same name. The uniqueness within module Batch Control is achieved by always referencing the master recipe name and the control recipe name. When creating a control recipe, a CEL entry is created.

ACTIONS FOR FILLING THE CONTROL RECIPE LIST

As each control recipe can be executed only once, we assume that there are very many control recipes. As during the loading of the list of the control recipes each control recipe is opened on the hard disk, it makes sense to not display all control recipes. Control recipes are therefore not automatically opened by default. They must be called up manually and via filters:

1. Provide the fitting filter options.
2. Select the desired master recipe.
3. Click on button **Display associated control recipes in list**.
4. All control recipes appropriate to the filters and the selection are displayed in the list of the control recipes.

Action/filter	Description
currently executed control recipes	Opens control recipes that are currently being executed. Control recipe status: *In execution*
Prepared control recipes	Opens control recipes which are prepared for execution. Control recipe status: *Prepared*
Completed control recipes	Opens control recipes which have already been executed. Control recipe status: *Executed*
outdated control recipes	Opens control recipes that have been marked as outdated. Control recipe status: *Outdated*
Show related control recipes in list	Displays all control recipes that are based on the selected master recipe and that comply with the set filter criteria.

8.11.2 Manage control recipes

LIST AND LIST FORMATING

List/action	Description
Control recipe list	In this list all control recipes can be displayed. The display can be limited by filters to an individual selection. Per default the list is empty. The following must be the case for the list to be filled: ▸ Master recipes must be selected ▸ The filters: **currently-executed control recipes**, **prepared control recipes** and **completed control recipes** must be set ▸ click button **display associated control recipes in list** In addition to the filters mentioned above, you can filter the list

List/action	Description
	itself.The filtering can be preset in the zenon Editor in the screen switch function (on page 83). Filtering in Runtime is also possible. These filter settings are not saved in the Runtime. This means that the filters must be entered again when called up again. A permanent pre-setting of the filter is only possible with configuration in the zenon Editor. All commands are also possible in the context menu of the list. The commands for list management can be called from the header of the list. The commands for recipe management can be called at editing one or more recipes. The recipes in the list cannot be edited directly in the list. Renaming, changing the description or starting the recipes is only possible with the corresponding commands. **Note for ideal configuration of the list** The list can be designed diversely concerning content and look: - **Content:** The displayed columns (on page 115) can be selected, the column format (on page 119) (column width, alignment, label) can be changed and you can define filters (on page 88). These settings can be edited in Editor and Runtime. - **Graphics:** At the settings of the list in the Editor you can find diverse setting possibilities in areas **Representation**, **Scroll bars** and **Fill**. With these properties you can even design the list ready for touch operation. Longer texts can also be displayed in lists over several lines using the **Automatic word wrap** property. The line height must be amended manually in order to do this. If you want to edit the list directly using the monitor, activate the Multi-Touch functionality. You can find detailed information in relation to this in the Configure interactions chapter.

ACTIONS FOR LIST MANAGEMENT

Action	Description
Column selection control recipes...	Opens a dialog in order to determine which columns should be displayed (on page 115). **Attention:** These changes are discarded when the screen is

Action	Description
	called up again. A permanent definition is only possible in the zenon Editor.
Format columns control recipes...	Opens a dialog to edit the column settings (on page 119). **Attention:** These changes are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.

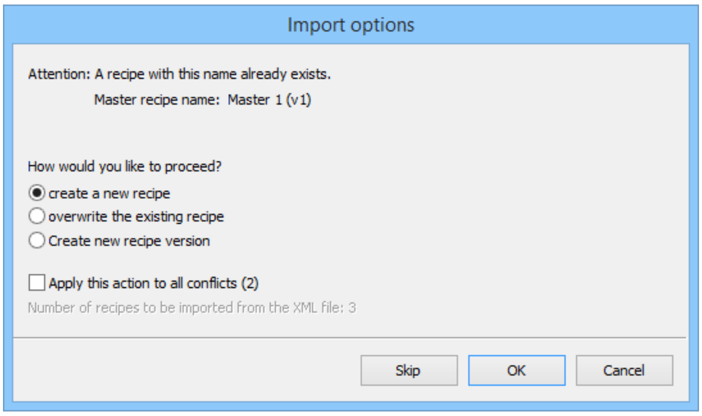
ACTIONS FOR RECIPE MANAGEMENT

Action	Description
Duplicate control recipe	Only active if exactly one recipe is selected. Created a copy of the selected recipe. At the creation of the copy, the version of the recipe saved on the hard disk is used. If the recipe is just edited in another computer and the changes have not yet been saved, the changes are not applied. The dialog for the input of a unique name and the description is opened. The copy of the recipe automatically gets the status *Prepared* and can therefore be edited and started. The execution status (on page 263) of the duplicate is set to *automatic*. When duplicating a recipe, a CEL entry is created.
Delete control recipe	Deletes the selected recipes irrevocably. If the recipe is opened on another computer for editing, it is automatically closed there. Deleting is only possible if all selected recipes are not executed (control recipe status: *In execution*). First they must be *finished*, *stopped* or *canceled*. If recipes must not be deleted - e.g. in an FDA-regulated environment - it is recommended that this button is not configured or that it is given an appropriate **Authorization level**. A CEL entry is created when a recipe is deleted.
Configure control recipe	Only active if exactly one control recipe was selected. The dialog for the input of a unique name and the description is opened. Recipes can only be renamed if they have the status of *prepared*. Also use this function in order to change the description of the control recipe.

Action	Description
Open control recipe	Opens the selected control recipes in the recipe editor if screen element *Recipe editor* exists in the screen. Each selected control recipe is opened in a separate tab of the recipe editor.
Start control recipe	Starts the selected control recipe in the set execution mode. The recipes are executed automatically at the Server. It is not necessary that the recipe is opened in the recipe editor.

8.11.3 Import recipes

When importing a recipe from an XML file, the following dialog is displayed in the event of naming conflicts:



Import options

Attention: A recipe with this name already exists.

Master recipe name: Master 1 (v1)

How would you like to proceed?

● create a new recipe

○ overwrite the existing recipe

○ Create new recipe version

□ Apply this action to all conflicts (2)

Number of recipes to be imported from the XML file: 3

Select how you want to proceed in the dialog:

Parameter	Description
Master recipe name/control recipe/operation	Name of the recipe that is already in use.
Create a new recipe	Creates a new recipe with the name and increments the number at the end of the name.
Overwrite the existing recipe	Overwrites the existing recipe with the name.
Create a new recipe version	Creates a new version of the recipe with the name. **Note**: Only active if **Versioning active** has been selected in the Editor
Apply this action to all conflicts	The selected option is applied to all pending conflicts. The number of conflicts is shown in

Parameter	Description
	brackets.
Number of recipes to be imported from the XML file	Number of recipes that are contained in the selected XML file. Number of recipes that are to be imported in the selected XML file.
Skip	Skips the importation of the currently-displayed recipe.
OK	Accepts the selection and shows the options for the next naming conflict or closes the dialog.
Cancel	Cancels the action and closes the dialog.

If there is a naming conflict there may, under certain circumstances, regardless of the status of the recipe that already exists, not be all important options available:

Recipe type	Status of the existing recipe.	Possible options
Master recipe	*Edit mode*	▶ Create a new recipe ▶ Overwrite the existing recipe ▶ Create a new recipe version
	Released	▶ Create a new recipe ▶ Create a new recipe version
	Outdated	▶ Create a new recipe ▶ Create a new recipe version
Control recipe	*Prepared*	▶ Create a new recipe ▶ Overwrite the existing recipe
	In execution	▶ Create a new recipe
	Executed	▶ Create a new recipe
Operation	*Edit mode*	▶ Create a new recipe ▶ Overwrite the existing recipe
	Released	▶ Create a new recipe

Note: Control recipes cannot be imported to existing master recipes if the version number of the master recipe is different. The attendant master recipe from the control recipe to be imported must have the same version number as the existing master recipe.

8.11.4 Saving on the hard disk and backup scenarios

MASTER RECIPES

Each master recipe has a unique ID under which it is saved on the hard disk with file extension *.MR*; e.g. *9.MR*
Each recipe conforms to one file. The ID of the recipe can be read from the list of the master recipes. For this column **Master recipe ID** must be visible.
The folder for the master recipes is a sub folder of **Runtime folder**:
*\RT\FILES\*zenon*\system\BatchRecipes*

For the recipe management file *Recipe.unique* is responsible which is located in the same folder. It makes sure that the recipe names are unique.

Note: If you delete a recipe manually via the file explorer and therefore outside of the Runtime and the module Batch Control, you must delete file *Recipe.unique* for its content to be correct again. For example if you delete a control recipe manually, you cannot delete the respective master recipe in module Batch Control as the control recipe still exists according to module Batch Control. Only after a reinitialization of file *Recipe.unique* can the master recipe also be deleted.

BACKING UP MASTER RECIPES

The *.MR* files - and with this all master recipes - can be backed up at any time. For example you can use function File operations.

RESTORING MASTER RECIPES

The restoring should only be done if absolutely necessary as more current data is overwritten. Proceed as follows:

1. Exit the Runtime.
2. Save all existing master recipes.
3. Rename file *Recipe.unique* or delete it. It automatically re-created at the Runtime start from the *.MR* files.
4. Restore the *.MR* files from an earlier backup.
5. Restart the Runtime.

CONTROL RECIPE

Each control recipe has a unique ID under which it is saved on the hard disk with file extension *.CR*; e.g. *9.CR*
Each recipe conforms to one file. The ID of the recipe can be read from the list of the control recipes. The **Control recipe ID** column must be visible to do this. Control recipes are always based on a

master recipe and are therefore always assigned to it. The ID number circles are therefore only unique with regard to the underlying master recipe.
Example: The master recipe with ID *9* has the control recipes with IDs *1* and *2*. The master recipe with ID *10* also has the control recipes with IDs *1* and *2*.

Therefore each master recipe has a sub folder in which the control recipes are saved. The name of the folder is always: *<Master recipe ID>.crd*. In our example there is the folder *9.crd* with files *1.CR* and *2.CR* and the folder *10.crd* with files *1.CR* and *2.CR*.

The folders for the command recipes are sub folders of **Runtime folder**: *\RT\FILES\*zenon*\system\BatchRecipes\*. In this folder the individual control recipe folders have been created. In each control recipe folder there is the file *Recipe.unique*. It makes sure that the recipe names are unique.

BACKING UP AND RESTORING CONTROL RECIPES

Proceed in the same was as for the master recipes only that you now need to backup all *.CR* files and the appertaining folder structure. At restoring you must delete all *Recipe.unique* files. They are also restored automatically.

8.12 Synchronization of Runtime and Editor

Project configurations in zenon Editor and in the Runtime are also checked and synchronized at certain times in accordance with certain rules.

CHECKING FOR CHANGES IN THE EDITOR

When loading, opening, duplicating or approving a recipe, a check is made to see if the configuration of units, unit classes, phases, etc. was changed in the Editor. If a change is detected, it is taken over in the object to which the function concerned is assigned. Settings that were overwritten in Runtime are retained. If the recipe has been changed, this is shown by means of an asterisk (*) in the recipe name.

At reloading the recipe is also checked.

Note: Only recipes that are in edit mode or test mode are updated. Master recipes that are currently being executed are only updated after execution has ended. Approved master recipes and control recipes are not updated.

> ⚠**Attention**
>
> Changes to phases in the Editor are applied without warning message when the recipe is approved in the Runtime. For all data which are not overwritten in the Runtime, the Editor is the leading system.

SYNCHRONIZATION OF PHASES

Editing of phases and their parameters is possible at four consecutive levels:

- in the zenon Editor
  - For unit classes (optional)
  - For units or unit instances
  - For phases (optional)
- In zenon Runtime
  - In the operation template (optional)
  - in the master recipe
  - in the control recipe

When instancing, the data from the level above is always used. Synchronization and comparison in the editing dialogs also always relates to the previous level.

When synchronizing, phases and placeholder phases are taken into account.

SYNCHRONIZATION FOR PHASES

Origin of data for comparison or synchronization:

Call context	Position of the object that provides comparative data
Phase of a unit class in zenon Editor	No data to compare.
Phase of a unit in zenon Editor	No data to compare.
Phase of a unit instance in zenon Editor	Phase configured in the unit class.
Control strategy of a phase in zenon Editor	Phase that belongs to the control strategy or control strategy of the phase in the unit class.
Master recipe	Phase or control strategy configured in the zenon Editor.
Control recipe	The phase from the master recipe.
Operation template	Phase or control strategy configured in the zenon Editor.
Operation instance in the master recipe	The phase from the template of the operation.
Operation instance in the control recipe	The phase from the operation instance in the master recipe.

You can find information on the origin of the comparison data in the tooltips of the control elements for synchronization.

SYNCHRONIZATION FOR PLACEHOLDER PHASES

When switching to test mode and when approving master recipes, the current Editor configuration of all phases, as well as all phases used for placeholders, is saved in the recipe. As long as this saved data exists, it is used for synchronization instead of the Editor configuration. This results in the following table for the origin of the data for comparison and/or synchronization:

Call context	Synchronization: 1. Components	Synchronization: 2. Components	Remark
Operation template	Editor: Phase of the unit class.	X	
Master recipe in edit mode	Editor: Phase of the unit class or unit instance.	X	
Master recipe in test mode	Configuration of the phase of the unit class or unit instance saved in the recipe.	X	
Control recipe	Corresponding phase in the master recipe.	Configuration of the phase of the unit class or unit instance saved in the recipe.	Local changes of the phase in the master recipe are noted when combining.
Operation instance in the master recipe host in edit mode	Corresponding phase in the operation template.	Editor: Phase of the unit class or unit instance.	Local changes of the phase in the master recipe are noted when combining.
Operation instance in the master recipe host in test mode	Corresponding phase in the operation template.	Configuration of the phase of the unit class or unit instance saved in the recipe.	Local changes of the phase in the master recipe are noted when combining.
Operation instance in the control recipe host	Corresponding phase in the operation template of the master recipe host	Configuration of the phase of the unit class or unit instance saved in the recipe.	Local changes to the phase in the operation template of the master recipe host are noted when combining.

RECIPES AND OPERATIONS

Synchronization is carried out if:

- A recipe is opened
- A recipe is reloaded
- An operation template is saved and a recipe is opened that has an instance of this template

Values in phases and parameters correspond to the values in the templates as standard and can be overwritten locally.

CHANGES TO CONTROL STRATEGIES AND CONTROL STRATEGY ACTIVATION

The following applies for control strategies (on page 229) during synchronization:

- When synchronizing recipes, phases with active control strategies are provided with the amended information (name change, new tags added, etc.).
- If control strategies were activated for a phase in the Editor, the phase in the Runtime is set to the status where no control strategy is selected. A control strategy must be selected before the recipe can be tested or approved.
- If control strategies are deactivated for a phase in the Editor, the phase contains the complete current configuration of the phase after synchronization. With this switch, all changes that have been made to the phase in the recipe are discarded.

9 Behavior in the network

The module **Batch Control** is fully capable of using a network in terms of Client/Server technology. This means that Batch recipes can be created, duplicated, edited, deleted, etc. on a Client. The whole recipe management remains always on the server. Likewise the whole process control such as **start** recipe, **pause** recipe, **stop** recipe, etc. can be done from the Client. Also mode changes and manual operations such as **jump** are possible.

> ⚠**Attention**
>
> Module Batch Control does not support redundancy. There is no synchronization between Standby Server. When the Server breaks down, the executed Batch recipes are not continued on the Standby! Recipes can also be started whilst the configured Server 2 is the primary server.

For using Batch Control in a network the following is true:

ALLOCATION

The forcing of allocations can be carried out from the Server or Client.

FUNCTIONS

Functions are always carried out at the Server.

PHASES

- Editing phases in the master recipe:
  - Edit mode: Changes a done locally at the Client.
    If during the editing the recipe is saved on another computer in the network, the current configuration is lost. An appropriate message is displayed and the editing dialog is closed. The new data from the server are displayed.
  - Test mode: Changes a done at the Server.
- Control recipe: Changes a done at the Server.
- If a recipe is saved in the network, all Clients using this recipe are updated.
- If a recipe is opened on a client, the current version on the server is always displayed, even if it has not yet been saved there.
- If a recipe is deleted on a computer, a message is displayed on all computer on which the recipe is opened that the recipe has been deleted.

MODE

- The mode (automatic, semi-automatic, manual) can be switched by the server and the client.
- Jumps in the recipe and step-by-step progress of a recipe can be done from Server and Client.

RELOAD

Changes made to the recipes on the client that have not been saved can be overwritten when reloading.

RECIPES

- Recipes can be started and controlled by the zenon server and by zenon clients.
- If parameters in a recipe are changed whilst the recipe is saved on a different zenon client, the change to the parameters is refused and not carried out.

- A master recipe can be changed on the zenon client whilst it switches to test mode on the zenon server and is sent to the zenon client. The changes that were last saved are transferred. This means: If the zenon client saves last, the recipe is switched to editing mode again. If the zenon server saves last, the change to the zenon clients is discarded and the recipe is in test mode.
- If a communication error occurs when deleting a recipe or an operation template, the deletion is refused with an error message.

WEB CLIENT

With a standard web client:

- The settings for grid and color can be changed
- No recipes can be created or edited
- The size of the editing area cannot be changed
- In the toolbar, all symbols that are not permitted are deactivated; it is not possible to select the corresponding objects.

Web client PRO is not affected by these restrictions.

9.1 Redundancy

zenon Batch Control does not support redundancy. In networks that have been set up as redundant, this means:

- If the server fails, the recipe is in an undefined status.
  It is not passed on to the standby server.
- On the standby server, once it has stepped up to become the server:
  - Master recipes can be switched back to edit mode and deleted.
    These changes are carried over to the server once it is working again.
  - Control recipes cannot be edited or executed.
- Starting, pausing or other commands are not possible on the Standby Server.
- If the recipe is not edited in the Standby Server, it runs normally again as soon as the server is online again.

10 Function authorizations

For changes in the Editor and in the Runtime, corresponding function authorizations can be issued to users. A warning is displayed if an operation for which there are no corresponding rights is executed. Exception: No warning is displayed when editing.

If a user does not have the function authorization, then

- In the Runtime: the corresponding functions cannot be executed
- in the Editor: Toolbars and context menus of the corresponding module are grayed out

Users can also log in temporarily for the execution of operations for which they have no special authorizations. For details on this, see the **Permanent and temporary login** section in the User administration manual.

You can find details on the individual authorizations in the User administration chapter, most of all in the **Function authorizations** section.

Note:

- These function authorizations are only available if Batch Control is licensed on the computer.
- Operation instances always use the user rights of the recipe in which they are embedded.

11 Reporting

Reports for configuration of the recipes can be created with the Report Viewer integrated into zenon.

When switching to a Report Viewer screen:

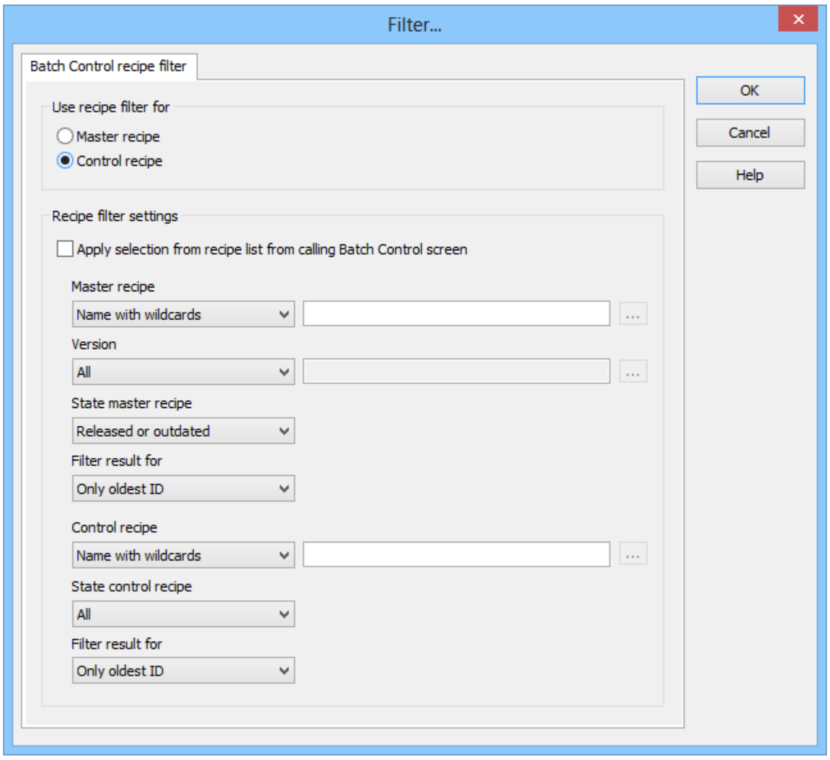
- it is possible to filter for recipes (on page 298)
- Datasets for Batch Control reports can be created:
  - Master recipe
  - Control recipe
  - Recipe screens
  - Matrix cells
  - PFC structure
  - Phases
  - Parameter
  - Transitions
  - Unit allocations
  - Operation instance

11.1 Batch Control recipe filter

When screen switching to a **Report Viewer** screen, a filter can be set for recipes from the Batch Control module. To filter according to recipes:

1. Open the Report definition tab for screen switching.
2. Go to area **Filter**.
3. In the **Module-specific filter** tab, click on **Batch Control:Recipe filter**.

   The dialog for configuring the filter is opened.



Batch Control recipe filter

Use recipe filter for

○ Master recipe

● Control recipe

Recipe filter settings

□ Apply selection from recipe list from calling Batch Control screen

Master recipe: Name with wildcards

Version: All

State master recipe: Released or outdated

Filter result for: Only oldest ID

Control recipe: Name with wildcards

State control recipe: All

Filter result for: Only oldest ID

USE RECIPE FILTER FOR

Option	Description
Use recipe filter for	Selection of the recipe type that is applied to the filter: - Master recipe - Control recipe
Master recipe	*Active*: It is filtered on Master recipes.
Control recipe	*Active*: It is filtered on control recipes. **Note:** The attendant master recipes must also be selected. If no master recipe has been selected for the control recipe, the filter cannot find the recipe being searched for in the Runtime. **Hint:** If the master recipe is not known, filtering of all master recipes with a placeholder is recommended.

RECIPE FILTER SETTINGS

Option	Description
Recipe filter settings	Options for the recipe filter
Apply selection from recipe list from calling Batch Control screen	- *Active*: In the Runtime, the first selected recipe of the batch screen from which the report viewer screen is called up is used. Individual settings in this dialog are then not available. - *Inactive*: The filter settings are changed individually using this dialog.
Master recipe	Parameters for the selection of the master recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*: The name of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables.
Version	Selection of the version (on page 217) from the drop-down list: - *All*: The version stated is ignored and each version found is used. - *Fixed version*: This filters for versions that are entered in this field. Highest possible version: *4294967295* - *Version from variable*: The version of the master recipe is defined by a variable in the Runtime. Click on button **...** in order to open the dialog for selecting a variable. - *Only oldest version*: Only the recipe with the oldest version number

Option	Description
	is used. - *Only newest version*: Only the recipe with the newest version number is used. - Default: *All*
Status of master recipe	State of the recipe Selection from drop-down list, depending on filtering: Filtering for template recipes: - All - Edit mode - Released - Test mode - Test in execution - Ended with an error - Outdated Filtering for control recipes: - Released or outdated - Released - Outdated
Filter result for	Define which ID is to be selected when filtering for names by selecting from the drop-down list: - Only oldest ID - Only newest ID Because a report can only be used for one recipe, it is not possible to filter for "all recipes".
Control recipe	Parameters for the selection of the control recipe. Select from drop-down list: - *Name with wildcards*: A name with placeholder can be entered into the input field. Filtering according to this name is carried out. - *Name from variable*:__

Option	Description
	The name of the control recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. - *ID from variable*: The ID of the master recipe is defined by a variable in the Runtime. Click on button **...** opens the dialog for selecting variables. Precisely one recipe can be found if the variable value at the time of execution is a valid ID of a control recipe. - *Job ID from variable*: Finds control recipes that have the given job ID number. Any type of variable can be linked. The value is automatically converted into STRING. **Note:** If the variable does not have a value, no recipe is sent to the Report Viewer.
State of control recipe	Selection of the recipe status from the drop-down list: - All - Prepared - In execution - Executed - Terminated with error - Outdated
Filter result for	Define which ID is to be selected when filtering for names or job ID by selecting from the drop-down list: - Only oldest ID - Only newest ID
OK	Applies all changes, creates filter and closes the dialog.
Cancel	Discards all changes in all tabs and closes the dialog.
Help	Opens online help.

Note for variable selection using name or ID: For the selection of variables according to name or ID, numerical variables and string variables can be selected respectively. The data types are converted to the respective correct form.

12 Formula editor

The formula editor is automatically opened if you need to enter or edit a formula. Above all:

Editor:

- Properties of the phases
- Interlocking conditions
- Phase done condition
- Condition for transient status
- Conditions for the detection of communication problems and PLC errors

Runtime:

- Phase done condition and interlocking when editing a phase
- Editing transitions

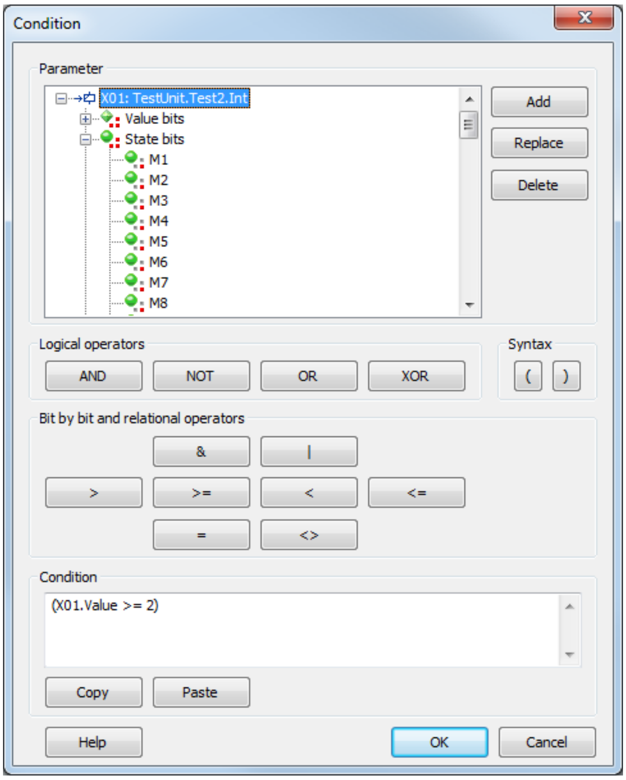
> **Information**
>
> If the phase referenced in the formula is removed and a new phase is added, the operands are reassigned for transitions. To be able to newly assign the operands, the same phase must be added again. Parameters from a different phase are not automatically linked.

ENTER FORMULA

The following input is accepted:

- Constant as decimal number
- Hexadecimal number if it is preceded by an **x**
- Dot as decimal separator; the following is true:
  - Comma is automatically converted into a dot: 23,000 to 23.000

- Decimal places which are only zeros are removed: 23.000 to 23



(X01.Value >= 2)

Parameter	Description
TAG list	List of the tags which can be used for the formula. Each entry contains of: - a basis node for the label - a value - a status - the bits for value and status A symbol at the first node shows whether it is a command tag or return / PLC sync tag. The short indentifier at the beginning of the name is used for the formula.
Add	Opens the dialog for adding a parameter (on page 308). In doing so, the following applies: - The following can be added: numeric and binary tags and tags for time duration. Values for duration are converted to seconds - For conditions of the phase only the tags created for it can

Parameter	Description
	be added. - Tags can be added multiple times.
Replace	Makes it possible to replace a tag. Clicking on the button opens the dialog to add a parameter (on page 308). - Selection of a new parameter replaces the highlighted parameter. - Clicking on the **no selection** button deletes the highlighted parameter from the list. The short identifier remains the same at replacing.
Remove	Removes the highlighted tag. For a tag to be deleted: - the formula must be correct - the selected tag must not be used in the formula
Logical operators	Via the buttons for operators, operators are added to the formula.
AND	logical 'AND'
OR	logical 'OR'
XOR	logical 'EXCLUSIVE OR'
NOT	Negation
Syntax	The operator buttons add the string shown on them to the formula.
(	Open parenthesis
)	Close parenthesis
Bit by bit and relational operators	
&	And
|	Or
>	greater than
>=	greater or equal
<	less than
<=	Less than or equal
=	Equal

Parameter	Description
< >	less or greater
Condition	Configuration and display of the formula.
Copy	Copies the whole formula: - All configured tags from the tag tree - Formula from the field
Paste	Pastes a formula from the clipboard. At this all already configured elements are deleted and replaced by the copied formula. When copying formulas between phases, it is tried to resolve the operands via their names. For tags which are not found invalid entries are created in the operands list. Their point of use in the formula remain the same.
OK	Applies formula and closes the dialog. For this the formula must be correct.
Cancel	Discards all changes and closes the dialog.

Information

You can link up to 99 tags in a formula. X01 to X99. The length of the formula must not exceed 4096 characters.

THE MEANING OF THE BITS:

Parameter	Description
Value bits	32 value bits (from 0 -31) are available. They describe the tag value bit by bit. For binary tags only bit *0* is of importance, for SINT and USINT only the bits from *0 - 7*, etc.
Status Bits	Here you find the most commonly used status bits. You find the exact definition and use of the status bits in the Status Bits List.
value and state	In the formulas, all values (value bits and status bits) are treated as binary values and can be logically linked with AND, OR, etc. The total value and overall status are an exception to this. In order to arrive at a Boolean expression, this total value has to be ORed *bitwise* with a constant. For this, we use the operator *&*. For the result *0* (*FALSE*) of this logical ORing, we get the binary value *0*

Parameter	Description
	(*FALSE*), otherwise *1* (*TRUE*). Example: See the bitwise ORing example chapter

Note: The status bits **NORM** and **N_NORM** are only available in the formula editor and cannot be engineered via the status.

Information

Formulas with binary X values and bitwise linking can be used with a maximum of 2 binary values. If more values are required, the linking must be carried out without binary X values.

Example:

X01.Value & X02.Value -> works

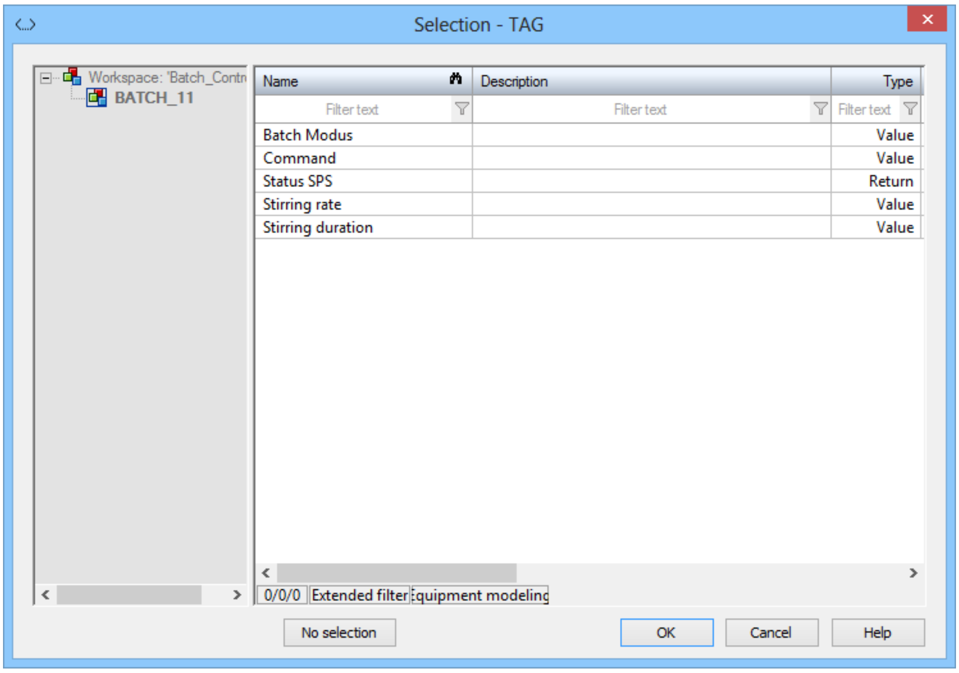
X01.Value & X02.Value & X03.Value -> does not work

But:

X01.00 AND X02.00 AND X03.00 AND X04.00 AND X05.00 -> works

12.1 Adding parameters

Clicking on the **Add** button in the formula editor (on page 303) opens the dialog to select parameters that are to be used for a formula.



Name	Description	Type
Batch Modus		Value
Command		Value
Status SPS		Return
Stirring rate		Value
Stirring duration		Value

Parameter	Description
Project list	Display of the active project. Only parameters that have been created in the active project for the phase to be configured can be selected.
TAG list	List of the parameters available for the selected phase. Apply by selecting and clicking on the OK button or by double clicking on a parameter.
No selection	Deletes parameters already set. Only effective for replacement of parameters. If a parameter is highlighted in the formula editor and this dialog is opened by clicking on the **Replace** button, then clicking on the **No selection** button deletes the parameter from the list in the formula editor. The short identifier remains the same at replacing.
OK	Inserts the selected parameter into the parameter list of the formula and closes the dialog.
Cancel	Discards selection and closes dialog.

Parameter	Description
Help	Opens online help.

12.2 List of status bits

Bit number	Short term	Long name	zenon Logic identifier
0	M1	User status 1; for Command Processing: Action type "Block"; Service Tracking of the IEC 850 driver	_VSB_ST_M1
1	M2	User status 2	_VSB_ST_M2
2	M3	User status 3	_VSB_ST_M3
3	M4	User status 4	_VSB_ST_M4
4	M5	User status 5	_VSB_ST_M5
5	M6	User status 6	_VSB_ST_M6
6	M7	User status 7	_VSB_ST_M7
7	M8	User status 8	_VSB_ST_M8
8	NET_SEL	Select in the network	_VSB_SELEC
9	REVISION	Revision	_VSB_REV
10	PROGRESS	In operation	_VSB_DIREC
11	TIMEOUT	Command "Timeout exceeded" (command runtime exceeded)	_VSB_RTE
12	MAN_VAL	Manual value	_VSB_MVALUE
13	M14	User status 14	_VSB_ST_14
14	M15	User status 15	_VSB_ST_15
15	M16	User status 16	_VSB_ST_16
16	GI	General interrogation	_VSB_GR
17	SPONT	Spontaneous	_VSB_SPONT

Bit number	Short term	Long name	zenon Logic identifier
18	INVALID	Invalid	_VSB_I_BIT
19	T_STD_E	External standard time (standard time) **Caution:** up to version 7.50, this was the status bit T_CHG_A	_VSB_SUWI
20	OFF	Switched off	_VSB_N_UPD
21	T_EXTERN	Real time - external time stamp	_VSB_RT_E
22	T_INTERN	Internal time stamp	_VSB_RT_I
23	N_SORTAB	Not sortable	_VSB_NSORT
24	FM_TR	Error message transformer value	_VSB_DM_TR
25	RM_TR	Working message transformer value	_VSB_RM_TR
26	INFO	Information for the variable	_VSB_INFO
27	ALT_VAL	Alternate value	_VSB_AVALUE
28	RES28	Reserved for internal use (alarm flashing)	_VSB_RES28
29	N_UPDATE	Not updated (zenon network)	_VSB_ACTUAL
30	T_STD	Internal standard time	_VSB_WINTER
31	RES31	Reserved for internal use (alarm flashing)	_VSB_RES31
32	COT0	Cause of transmission bit 1	_VSB_TCB0
33	COT1	Cause of transmission bit 2	_VSB_TCB1
34	COT2	Cause of transmission bit 3	_VSB_TCB2
35	COT3	Cause of transmission bit 4	_VSB_TCB3
36	COT4	Cause of transmission bit 5	_VSB_TCB4
37	COT5	Cause of transmission bit 6	_VSB_TCB5

Bit number	Short term	Long name	zenon Logic identifier
38	N_CONF	Negative confirmation of command by device (IEC 60870 [P/N])	_VSB_PN_BIT
39	TEST	Test bit (IEC870 [T])	_VSB_T_BIT
40	WR_ACK	Writing acknowledged	_VSB_WR_ACK
41	WR_SUC	Writing successful	_VSB_WR_SUC
42	NORM	Default status	_VSB_NORM
43	N_NORM	Deviation normal status	_VSB_ABNORM
44	BL_870	IEC 60870 status: *blocked*	_VSB_BL_BIT
45	SB_870	IEC 60870 status: *substituted*	_VSB_SP_BIT
46	NT_870	IEC 60870 status: *not topical*	_VSB_NT_BIT
47	OV_870	IEC 60870 status: *overflow*	_VSB_OV_BIT
48	SE_870	IEC 60870 status: *select*	_VSB_SE_BIT
49	T_INVAL	External time stamp invalid	not defined
50	CB_TRIP	Breaker tripping detected	not defined
51	CB_TR_I	Breaker tripping detection inactive	not defined
52	OR_DRV	Value out of the valid range (IEC 61850)	not defined
53	T_UNSYNC	ClockNotSynchronized (IEC 61850)	not defined
54	PR_NR	Not recorded in the Process Recorder	not defined
55	T_DEV	Configured time difference between internal and external timestamp reached.	not defined
56	RES56	reserved	not defined
57	RES57	reserved	not defined

Bit number	Short term	Long name	zenon Logic identifier
58	RES58	reserved	not defined
59	RES59	reserved	not defined
60	RES60	reserved	not defined
61	RES61	reserved	not defined
62	RES62	reserved	not defined
63	RES63	reserved	not defined

Information

In formulas all status bits are available. For other use the availability can be limited.

You can read details on status processing in the Status processing chapter.

12.3 Logical Operators

Logical links: Variables will only be checked for the logical value '0'; if the value does not equal '0', it will be considered as '1'.

In contrast to bit formulas, the technical range can be modified by a stretch factor -> (not equal '0' or '1').

Operator	Meaning
AND	logical 'AND'
NOT	Negation
OR	logical 'OR'
XOR	logical 'EXCLUSIVE OR'

The operators have the following priority in the formula calculation:

Priority	Operator
1	& (operator for bit formulas)
2	NOT

Priority	Operator
3	AND
4	XOR/OR

Info

Up to 99 variables can be linked in one formula. X01 to X99.

Info

The status bits NORM and N_NORM are only available in the formula editor and cannot be engineered via the status.

12.4 Bit formulas

Bit formulas only have a logical high or low state. In contrast to logical formulas, the raw value is already predefined (**0**,*1*).

Operator	Description
&	AND
|	OR

12.4.1 Example: ORing bitwise

You want to find out if one of the user status bits 1-8 (M1 ... M8) of the variable X01 is set.

USUAL FORMULA:

X01.M1 OR X01.M2 OR X01.M3 OR X01.M4 OR X01.M5 OR X01.M6 OR X01.M7 OR X01.M8
This query can be made much easier by the logical ORing of the overall status.

LOGICAL ORING

X01.Status & 0xFF

The constant can be entered in hexadecimals, as described above:

0xFF corresponds to decimal *255*; these are the first eight status bits (binary 11111111). If one of these bit is set to *1*, the result of this bitwise ORing is *1* (true), otherwise it is *0* (false).

If, for example, all user status bits except the user status bit M7 should be queried, the binary statement for this would be: 10111111. Bit 7 is not of interest and is thus set to *0*. This corresponds to 0xBF in hexadecimal. The expression for the formula is then: **X01.Status & 0xBF**.

Instead of ORing bitwise with a constant, the value can also be directly compared to a decimal number. If the comparison is wrong, the binary value is *0* (false) otherwise it is *1* (true).

Example:

You want to find out if the value is equal to the constant *202*: The formula is:

X01.value = 202

If the value is equal to the constant *202*, the result of the comparison is *1* (*True*) otherwise it is *0* (*False*).

Note: The bitwise ORing works with the OR character (**|**), the same as in this example.

12.5 Comparison operators

Comparison operators are for the direct comparison of two numeric values. The result of this comparison is a binary value. "*0*" if the condition is not fulfilled and „*1*" if the condition is fulfilled.

Operator	Description
<	less
>	greater
<=	Less than or equal
>=	greater or equal
=	Equal
<>	unequal

To the left and to the right of the comparison operator, there has to be a (total) value or a (total) status, single bits cannot be used with these comparison operators.

There can also be a constant to the right of the comparison operator.
These constants are entered as hexadecimal values or decimal values in the combined element. Hexadecimal numbers are automatically converted to decimal numbers by clicking on **OK**. For example, *0x64* corresponds to the numerical value *100*.

Note: The combined element is not available in the **Batch Control** module.

Example

X01.value >= X02.value
The result is 1, if the value of X01 is higher than or equal to the value of X02

X01.value = 0x64
The result is 1, if the value of X01 is exactly equal to the numeric value 100 (= hex 0x64)

(X01.value = 0x64) OR (X01.value = 0x65)
The result is 1, if the value of X01 is exactly equal to the numeric value 100 or 101 (= hex 0x64 and hex 0x65)

12.6 Examples for formulas

SIMPLE LOGICAL AND LINKING BETWEEN TWO BIT VALUES

Example

Formula: X01.03 AND X02.03

This formula has the status TRUE, if both **bit 3** of variable *1* and **bit 3** of variable *2* both have the value *1*.

COMPARISON OF AN VALUE OR STATUS OF A VARIABLE

Example

(X01.Value> X02.Value)

COMPARE COMPARISONS TO ONE OTHER ON A LOGICAL BASIS

Example

(X01.Value> X02.Value) AND (X01.Value = X02.Value)

COMPARE WITH VALUE BITS AND STATUS BITS

Example

(X01.Value> X02.Value) AND (X01.Value = X02.Value) OR (X01.03 = X02.03)

COMPARE A VALUE WITH A DECIMAL OR HEXADECIMAL VALUE

Example

Formula: (X01.Value = 111)

Formula: (X01.Value = 0x6F)

If a hexadecimal values is used, this is later transferred to decimal by clicking on **OK**. If a decimal value is entered and confirmed, the value continues to be displayed as a decimal value after reopening.

Info

It is not possible to use a comma or a period when entering values.

13 Import - Export

In the Editor and Runtime, color palettes can be exported as XML files and imported from these. When exporting, unit classes, unit instances and phases for placeholders are also exported.

13.1 Editor

All objects in the Editor can be exported as an XML file and it is possible to import back from these. Import and export are started via the respective context menu of a node or a list. Subordinate objects are automatically exported too.

IMPORT RULES

When importing, new objects with their own ID are created. Any pre-existing objects are overwritten. In doing so, the pre-existing ID is used.

This ensures:

- Inheritance between classes and instances in the Editor.
- Correct linking of batch recipes in the Runtime for the editor configuration.

If a unit class is imported, all instances of this class are enhanced with the subobjects that are added. This is also applicable for phases in unit classes.

IMPORT OF UNIT INSTANCES

The following is applicable when importing unit instances:

- If a unit instance for which there is no class is imported, this is imported as a normal unit.
- If there are subobjects in the unit class that are not present in the XML file, these remain part of the unit instance.
- If there are subobjects that have been inherited by the class in the XML file, but which are not present in the class, these subobjects are not imported.
- If there are local subobjects in the XML file, these are imported as local objects. If there is a naming conflict, the local object is imported under another name.
- If a unit instance or a phase in a unit instance is imported and data is overwritten in the process, only the local information from the XML file and the unit class is taken into account. Local data in the unit instance to be overwritten is lost. Local objects that are missing in the XML file are deleted.
- If the unit class is modified before the import of the unit instance, the following is applicable:
  - Newly-created elements are also present in the added instance.
  - Amended elements are also changed in the added instance.
  - Deleted elements are also removed from the added instance.
  - Local data of the unit instance is retained when adding or importing.

EXECUTION SEQUENCE OF REACTIONS

Special rules when importing the execution sequence:

- The execution sequence of a reaction is always exported too. This also applies if it has been overwritten locally.
- If the execution sequence has not been overwritten locally and there are more or fewer reactions of a type than in the XML file, all local reaction objects from the XML file are moved forwards or backwards accordingly. In this case:
  - They are added after the inherited reactions.
  - The sequence of the local reactions remains the same.
- If the execution sequence has been overwritten locally and inherited reactions are no longer present in the class, all reactions in the imported phase are moved forwards with the sequence remaining the same.

- If the execution sequence has been overwritten locally and there are reactions in the class that are not part of the XML file, these reactions are placed at the end in the imported phase. The sequence of the new reactions in relation to one another remains the same.

13.2 Runtime

You can do the following in the Runtime using configurable buttons:

- Export master recipes (on page 161)
- Export control recipes (on page 232)
- Import recipes from XML files.

Functions for recipes:

Recipes can be exported or imported with the help of functions:

- Export Batch recipes (on page 125)
- Import Batch recipes (on page 131)

You can find information on the structure of the XML files in the **XML structures** (on page 318) section.**Units, phases and recipes** (on page 318).

13.2.1 XML structures: Units, phases and recipes

XML structure when exporting units, unit classes, phases and recipes:

13.2.1.1 General recipe properties in the XML file

PROPERTIES OF MASTER RECIPES

Properties of master recipes are also contained in exported control recipes.

node	Property	Description	Possible value
Recipes			*MasterRecipe* *ControlRecipe* *OperationTemplate*
	MrId	ID of the corresponding master recipe.	
	MrName	Name of the corresponding master recipe.	

node	Property	Description	Possible value
	MrDescription	Description of the corresponding master recipe.	
	MrVersion	Version of the corresponding master recipe.	
	MrSourceVersion	Original version of the master recipe.	
	MrStatus	Status of the corresponding master recipe.	- *1: Edit mode* - *2: Released* - *3: Test mode* - *4: Test in execution* - *8: Outdated*
	RecipeType	Type of the recipe.	- *Matrix* - *Pfc*
	ApprovalTime	Time stamp of the corresponding master recipe.	
	ApprovalUserName	Name of the user who approved the recipe.	
	ApprovalUserID	ID of the user who approved the recipe.	
	OutdatedTime	Outdated time for the recipe.	
	OutdatedUserName	Name of the user who set the recipe to obsolete.	
	OutdatedUserID	ID of the user who set the recipe to obsolete.	
	Structure	Nodes for the structure of the recipe.	

PROPERTIES OF CONTROL RECIPES

Property	Description	Possible value
CrId	ID of the control recipe.	
CrName	Name of the control recipe.	
CrDescription	Description of the control recipe.	
CrStatus	Status of the control recipe.	- *2: Prepared* - *3: In execution* - *4: Executed* - *7: Outdated*
CrJobID	Job ID of the control recipe	
CrJobIDVar	Variable for the job number.	
CreationTime	Time stamp of the creation of the control recipe.	
CreationUserName	Name of the user who created the recipe.	
CreationUserID	ID of the user who created the recipe.	
StartingTime	Time stamp of the start of the control recipe.	
StartingUserName	Name of the user who started the control recipe.	
StartingUserID	ID of the user who started the control recipe.	

PROPERTIES OF OPERATIONS

Property	Description	Possible value
OpId	ID of the Operation template.	
OpName	Name of the Operation template.	
OpDescription	Description of the operation template.	

Property	Description	Possible value
OpType	Type of operation	- *Matrix* - *Pfc*
OpStatus	Status of the operation template.	- *1: Edit mode* - *2: Released*
ApprovalTime	Time stamp of the approval of the operation template.	
ApprovalUserName	Name of the user who approved the operation template.	
ApprovalUserID	ID of the user who approved the operation template.	
Structure	Nodes for the structure of the recipe.	

13.2.1.2 Matrix properties in the XML file

MATRIX STRUCTURE

node	Property	Description	Possible value
Structure			
	Column	Column number in the matrix recipe.	- *Type 1: Phase* - *Type 2: Operation*
	ObjectName	Name of the linked phase or the linked operation.	
	StepInfo	Step number in the matrix recipe.	
	StepDescr	Description of the step in the matrix recipe.	

MATRIX CELL PROPERTIES

node	Property	Description	Possible value
Cell			- *Type 1: Phase* - *Type 2: Operation*
	CellActive	*TRUE*if the matrix cell is active at this point.	- *TRUE: Matrix cell active* - *FALSE: Matrix cell inactive*
	ChartId	ID of the matrix cell	
	Phase	Properties node for the phase of a matrix cell. You can find details in the General properties (on page 330) chapter.	
	ControlStrategy	Properties node for the mode of operation used. **Note:** With a phase, this mode of operation is used in a matrix cell. You can find details in the General properties (on page 330) chapter.	

13.2.1.3 PFC properties in the XML file

STRUCTURE OF THE PFC RECIPE

node	Property	Description	Possible value
Structure			
	LastObjId	Last-used ID of the PFC recipe.	
	ColCount	Number of columns in the PFC recipe.	

node	Property	Description	Possible value
	RowCount	Number of rows in the PFC recipe.	
	CenterColOffset	Start coordinates of the column.	
	CenterRowOffset	Start coordinates of the row.	

ELEMENT PROPERTIES

node	Description	Possible value
Chart object	Element for the PFC recipe.	*TYPE: 1=start element* *TYPE: 2=end element* *TYPE: 3=phase* *TYPE: 4=transition* *TYPE: 5=Begin branch* *TYPE: 6=end branch* *TYPE: 7=start parallel branch* *TYPE: 8=end parallel branch* *TYPE: 9=unit allocation* *TYPE: 10=operation* *TYPE: 11=line* *TYPE: 12=jump target*

BEGIN ELEMENT

node	Property	Description	Possible value
ChartObject		Element for the PFC recipe.	*TYPE: 1*
	ChartId	ID of the element in the	

node	Property	Description	Possible value
		PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	

END ELEMENT

node	Property	Description	Possible value
ChartObject		Element for the PFC recipe.	*TYPE: 2*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	

PHASE

node	Property	Description	Possible value
ChartObject			*TYPE: 3*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	
	Unit	Unit name of the linked phase.	
	Phase	Properties node for the phase of a PFC element.	

TRANSITION

node	Property	Description	Possible value
ChartObject			*TYPE: 4*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	
Condition		Condition for the selected transitions.	
	Expression	The conditions are used to inform the REE of the status of the technological function in the controller.	
OperandTag		Addressing of the tag conditions.	
	PhaseChartId	ID of the phase in the condition.	
	TagAddressing	Tag of the phase.	

INITIAL BRANCH

node	Property	Description	Possible value
ChartObject			*TYPE: 5*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	
Cell		Number of branches from left to right.	

node	Property	Description	Possible value
	Connector	Linking points of the branches.	*TRUE: Branch downwards.* *FALSE: No sequence selection.*
	ArrayOffset	Initial cell of the branch.	

END BRANCH

node	Property	Description	Possible value
ChartObject			*TYPE: 6*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	
Cell		Number of branches from left to right.	
	Connector	Linking points of the branches.	*TRUE: Branch downwards.* *FALSE: No sequence selection.*
	ArrayOffset	End cell of the branch.	

BEGIN PARALLEL BRANCH

node	Property	Description	Possible value
ChartObject			*TYPE: 7*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	

node	Property	Description	Possible value
Cell		Number of branches from left to right.	
	Connector	Linking points of the branches.	- *TRUE: Branch downwards.* - *FALSE: No sequence selection.*
	ArrayOffset	Initial cell of the branch.	

END PARALLEL BRANCH

node	Property	Description	Possible value
ChartObject			*TYPE: 8*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	
Cell		Number of branches from left to right.	
	Connector	Linking points of the branches.	- *TRUE: Branch downwards.* - *FALSE: No sequence selection.*
	ArrayOffset	End cell of the branch.	

UNIT ALLOCATION

node	Property	Description	Possible value
ChartObject			*TYPE: 9*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	

node	Property	Description	Possible value
	ChartCol	Column coordinates of the element.	
Allocations		Name of the unit allocation.	- *Allocations* - *Deallocations*
	Unit	Name of the allocated unit.	
	Global	Options for the clearing of the unit.	- *FALSE=clearing the last allocation* - *TRUE=recipe-wide clearing*

OPERATION

node	Property	Description	Possible value
ChartObject			*TYPE: 10*
	ChartId	ID of the element in the PFC recipe.	
	ChartRow	Row coordinates of the element.	
	ChartCol	Column coordinates of the element.	
Operation		Operation	
	OpId	ID of the operation	
	OpName	Name of the operation.	
	OpType	Type of the operation.	
	OpDescription	Description of the operation.	
	OpStatus	Status of the operation	
	ApprovalTime	Time stamp for approved recipes.	
	ApprovalName	Name of the approved recipe.	

node	Property	Description	Possible value
	ApprovalUserId	ID of the user who approved the recipe.	
Structure		Structure of the recipe.	- *PFC_Structure* - *Matrix_Structure*

LINE

node	Property	Description	Possible value
ChartObject			*TYPE: 11*
	ChartId	ID of the line.	
	ChartRow	Row coordinates for the starting point of the line.	
	ChartCol	Column coordinates for the starting point of the line.	
	FirstCol	Column coordinates of the cell where the line starts.	
	FirstRow	Row coordinates of the cell where the line starts.	
	SecondCol	Column coordinates of the cell where the line ends.	
	SecondRow	Row coordinates of the cell where the line ends.	
	LineSegments	Column, row, type coordinates of the cells, separated by # where the complete line runs.	- *0: Straight line from top to bottom* - *1: 90° from the top to the right* - *2: 90° from the top to the left* - *3: Straight line from left to right* - *4: 90° from the right*

node	Property	Description	Possible value
			to the bottom - *5: 90° from the left to the bottom*

JUMP TARGET

node	Property	Description	Possible value
ChartObject			*TYPE: 12*
	ChartId	ID of the jump target.	
	ChartCol	Column coordinates for the starting point of the jump target.	
	ChartRow	Row coordinates for the starting point of the jump target.	

13.3 Editor and Runtime: XML structure parameter properties

PHASES

node	Property	Description	Possible value
Phase		General properties of the phase.	
	PhaseName	Name of the phase.	
	PhaseDescr	Freely definable string for detailed description of the phase.	
	TOAllocation	Time period, in days, hours, minutes and seconds, that is waited until the unit is allocated.	
	TOInterlocking	Time period in days, hours, minutes and seconds in which the	

node	Property	Description	Possible value
		condition defined in the input lock property must return the value *TRUE*.	
	MinExecTime	Minimum execution duration of the phase.	
	ExplanationMinExecTimeNeeded	Reason for minimum execution duration change necessary	*TRUE: When entering value changes for this phase in Runtime, a reason for the change must be entered by the user.* *FALSE: No reason necessary.*
	MaxExecTime	Time period in days, hours, minutes and seconds in which the condition defined in the phase done condition property must return the value *TRUE*.	
	TOFollowingCond	Time period in days, hours, minutes and seconds in which the phase must be deactivated.	
	MinExecTimeCrModifiable	Minimum execution duration of the phase in the control recipe can be modified.	- *TRUE: Minimum execution duration of the phase in the control recipe can be modified.* - *FALSE: Minimum execution duration of the phase in the control recipe cannot be modified.*
	RtLocalFlags	Flags whose properties are no longer linked to the Editor configuration. They are not overwritten when reloading.	*These properties are a bit-coded DWORD in that each bit stands for a property.In a 0-based index, the coding is as follows:*

node	Property	Description	Possible value
			- *Timeout phase done condition: Bit 0 (decimal 1)* - *Timeout input lock: Bit 1 (decimal 2)* - *Timeout subsequent condition: Bit 2 (decimal 4)* - *Timeout unit allocation: Bit 3 (decimal 8)* - *Minimum execution duration: Bit 7 (decimal 128)*
	CSActive	Active control strategies	- *TRUE: Control strategies active* - *FALSE: Control strategies inactive*
	CSTag	control strategy tag	
	CondInterlocking	Node for the condition of the input lock.	
	CondDone	Node for phase done condition.	
	CondEscaping	Node for the condition of leaving the phase.	
	CondPausing	Node for the change from *pausing* to *paused*.	
	CondHolding	Node for the change from *holding* to *held*.	
	CondStopping	Node for the change from *stopping* to *stopped*.	
	CondAborting	Node for the change from *aborting* to *aborted*.	
	CondRestarting	Node for the condition	

node	Property	Description	Possible value
		for the change from *starting* to *running*.	
	CondFailure	Node for the condition to detect a communication fault.	
	CondConnReconnect	Node for the condition to acknowledge a communication fault.	
	CondPlcError	Node for the condition for PLC error.	

PARAMETERS

Node	Property	Description	Possible value
Tag			
	TagName	TAG name	
	TagDescr	Description of the tag.	
	TagType	Tag type	*0: Value TAG* *1:Return / PLC sync TAG* *2: Initial TAG*
	DataType	Data type	*0: Bool* *1: String* *2: Numeric* *3: Time period*
	Variable	Variable which is linked to the tag.	
	VariableDataType	Data type of the variable.	
	VariableDriver	Name of the EXE file of the driver of the variable.	
	VariableDriverDescr	Name of the driver of the variable.	
	TagValue	Value of the tag.	

Node	Property	Description	Possible value
	VariantType		
	ValueMin VariantType	Minimum value of the tag.	
	ValueMax VariantType	Maximum value of the tag.	
	VariableMin	Minimum value of the variable.	
	VariableMax	Maximum value of the variable.	
	MeasUnit	Unit of the variable.	
	MaxStringLength	String length as engineered at the variable.	
	EditableInRecipe	Changeable in the master recipe	- *TRUE:* Tag values can be changed in the master recipe. - *FALSE:* Tag values cannot be changed in the master recipe.
	ExplanationNeeded	Reason for value change necessary	- *TRUE:* When entering value changes for these parameters in Runtime, a reason for the change must be entered by the user. - *FALSE:* Reason not required.
	TagModified	Tag value changed.	- *TRUE:* Tag value changed - *FALSE:* Tag value not changed.
	EditableInCr	Tag value can be changed in control recipe	- *TRUE:* Tag value can be changed in the control recipe.

Node	Property	Description	Possible value
			- *FALSE:* Tag value cannot be changed in the control recipe.
	UseKeyboard	Use screen Keyboard	- *TRUE:* Keyboard screen is used for this parameter. - *FALSE:* Keyboard screen is not used for this parameter.
	KeyboardPictureName	Name of the selected keyboard screen. Only available if the keyboard screen property has been activated.	
	RtLocalFlags	Flags whose properties are no longer linked to the Editor configuration. They are not overwritten when reloading.	Bit-coded DWORD. One bit each stands for a property. Coding in a *0*-based index: - Minimum: *Bit 0* (decimal *1)* - Maximum: *Bit 1 (decimal 2)* - Tag value: *Bit 2 (decimal 4)* - Changeable in the control recipe: *Bit 6 (decimal 64)*
	ValueVariableMask	The field configured in the Editor for the selection of the variable that provides the parameter value in the Runtime.	
	ValueVariable	Variable selected in the Runtime that provides the parameter value.	

REACTIONS

node	Property	Description	Possible value
Reaction		Contains general properties for reactions.	
	EventType	Selection of the event type when the reaction is to be executed. For each event several reactions can be defined.	*0: Exit Runtime initiated* *1: Input lock blocked* *2: Waiting period input lock exceeded* *3: Phase activated* *4: Maximum execution duration exceeded* *5: Phase deactivated* *6: Waiting period following condition exceeded* *7: Linked variable interrupted* *8:State change: Resume* *9:State change: In execution* *10:State change: Pausing* *11:State change: Paused* *12:State change: Executed* *13:State change: Restarting* *14:State change: Holding* *15:State change: Held* *16:State change: Stopping*

node	Property	Description	Possible value
			- *17:State change: Stopped* - *18:State change: Aborting* - *19:State change: Aborted* - *20: Mode change: Automatic* - *21: Mode change: Semi-automatic* - *22: Mode change: Manually* - *28: Waiting period unit allocation exceeded* - *29: Phase started multiple times* - *30: Unit allocation not possible* - *31: Value TAG fully written* - *32: Phase done condition completed* - *33:Value TAG without value* - *34: Runtime restart* - *35: Escape condition started* - *36: Escape condition met* - *37: Loss of communication* - *38: Loss of communication fixed* - *39: Loss of communication*

node	Property	Description	Possible value
			acknowledged ▶ *40: Phase started* ▶ *41: PLC error* ▶ *42: PLC error fixed* ▶ *43: PLC error rectified by deactivation of the phase.* ▶ *44: Input lock checked successfully*
	ReactionDescr	Description per reaction.	
	ReactionPrio	Displays the execution order when several reactions were defined for the same event. The order is defined by the position in the list and is only displayed here.	
	ModeCommand	This reaction can be controlled in the Runtime of the execution mode as a reaction to the event.	▶ *0: Ignore* ▶ *1: Automatic* ▶ *2: Semi-automatic* ▶ *3: Manually*
	StateCommand	Recipe or phase command that is executed in the Runtime when the event occurs.	▶ *0: Ignore* ▶ *2: Pause recipe* ▶ *3: Resume recipe* ▶ *4: Hold recipe* ▶ *5: Restart recipe* ▶ *6: Stop recipe* ▶ *7: Abort recipe* ▶ *8: Pause phase* ▶ *9: Hold phase* ▶ *10: Restart phase* ▶ *11: Resume phase*

node	Property	Description	Possible value
	Function	Function that is to be carried out.	
	CelEnabled	Create CEL entry	*TRUE:* The text defined in the CEL message text property is entered in the Chronological Event List, CEL. *FALSE:* No entry in the CEL.
	CelMsg	CEL message text. Only available if CEL entry property is active.	
	CelGroup	Allocation of a pre-existing alarm/event group to CEL messages for the selected event.	
	CelClass	Allocation of a pre-existing alarm/event class to CEL messages for the selected event.	
	SourceTag	Tag whose value is written to the tag selected in the destination tag property.	
	DestinationTag	Tag to which the value of the tag defined in source tag property is transferred.	
	SetTag	Tag to which the defined set value should be written as reaction to the event.	
	SetValueNum	Set value for numeric tag.	
	SetValueStr	Set value for *string* TAGs.	
	SetValueBool	Set value for a binary tag.	- *0: Off* - *1: On*

node	Property	Description	Possible value
			- *4: Toggle*
	SetValueDuration	Set value in days, hours, minutes and seconds for time period tags. The time span is written as value in seconds to the variable linked for the tag.	
	ExecuteBeforeStartEvent	Allow execution before start event.	*TRUE:* The event can be executed before the phase started event. *FALSE:* The event can only be started if the phase started event has been executed.

CONTROL STRATEGIES

node	Property	Description	Possible value
ControlStrategy		Contains properties for the configuration of the control strategies.	
	CSName	Name of the control strategy.	
	CSDescription	Description of the control strategy.	
	ActiveCSNumber	Unique number of the control strategy for identification within this phase.	
	CSTag	Node for the definition of a parameter linked for this control strategy.	

CONTROL STRATEGIES

node	Property	Description	Possible value
ConfiguredControlStrategy		Contains properties for the configured control	

node	Property	Description	Possible value
		strategy in the recipe.	
	CSName	Name of the control strategy.	
	CSDescription	Description of the control strategy.	
	ActiveCSNumber	Unique number of the control strategy for identification within this phase.	

14 CEL

Messages, information and errors for recipes, units, commands, reactions, events etc. are saved and displayed in the **Chronological Event List** (**CEL**).

GROUPS AND CLASSES

CEL information can be allocated to groups and classes. These correspond to the **alarm/event groups** and **alarm/event classes** created in the project. The respective group or class is also entered in the CEL and can be used for filtering.

Groups and classes are allocated for actions for:

- Recipes
- Reactions
- Commands
- Value changes
- Jumping, forcing and step-by-step execution

RECIPES

The set group or class is entered into the CEL for the following recipe actions:

- Master recipes list
  - New
  - Create new version

  - Delete
  - Duplicate
  - Rename
  - Release
  - Highlight as outdated
  - New control recipe
- Control recipe list
  - Rename
  - Duplicate
  - Delete
- Operation list
  - New
  - Rename
  - Duplicate
  - Delete
  - Release

COMMANDS

The set group or class is entered for all recipe commands, phase commands, mode switches and restart messages.

VALUE CHANGES

The set group or class is entered in the event of value changes to parameters in control recipes.

JUMPING, FORCING AND STEP-BY-STEP EXECUTION

The set group or class is entered for all manual steps, when jumping steps and when executing steps.

ALLOCATION

Groups and classes are allocated

- General: In the properties of the **Batch Control** node in the **CEL groups/classes** group
- Reactions: In the properties of the event of a reaction in the **Reactions**/**CEL entry** group *System* is entered as a user.

15 Failure handling

If communication failures or PLC errors occur, these can be detected in the Runtime using a formula configured in the editor. In the event of a communication error, the phase concerned is paused or held (depending on configuration).

15.1 communication errors

Detection of problems can be specially configured for each phase, because phases can also run on different PLCs. The reaction to a communication error is defined globally for all phases. In addition, a reaction to the *communication error* (on page 50) event can be configured.

To configure the detection of communication errors:

1. highlight the desired phase
2. Click, in the **Loss of communication** property group, on the **Loss of communication** property
3. In the formula editor (on page 303) that opens, define the condition for detecting communication errors
4. click on property **Loss of communication acknowledged**
5. Define the condition to detect the reestablishment of communication
6. Navigate to the properties for the Batch Control module
7. In the **General**/**Loss of communication** group, open the **Action for loss of communication** property
8. Select the desired reaction to a communication error from the drop-down list
    - Hold phase
    - Pause phase

CHECK IN RUNTIME

When starting a phase or when restarting Runtime, 60 seconds is waited for values for the formula to check communication. If no values are received within this waiting time, a communication error is assumed.

Another check for communication errors is made if the phase is in a new step.

When restarting or reloading Runtime, the waiting for values can be in many steps, due to the phase being started again in the middle of the process. A check for communication failures is also made at areas where a check for faults is made.

REESTABLISHMENT

CHECKING THE FORMULA FOR REESTABLISHING COMMUNICATION

A loss of communication to the PLC is determined if the value of the formula for the **Loss of communication** property is *TRUE*. Waiting is carried out until communication has been reestablished, i.e. the value of the formula is *FALSE*. After this, waiting is continued until the value of the formula for the **Loss of communication acknowledged** property is TRUE or empty. The execution of the object can be continued from this time only.

In the time between the loss of communication and reestablishment of communication:

- No phase commands can be sent to the phase concerned
- If the status is changed directly, transient conditions are not checked
- No reactions are triggered for a status change
- the procedure path of the phase concerned remains the same when a *Continue* or *Restart* global command is executed
- The subsequent execution positions are offset until the communication to the phase has been reestablished

SKIPPING THE REESTABLISHMENT OF COMMUNICATION

In the event of a communication error, this can be skipped both whilst the error is active and whilst communication is being reestablished. In both cases, the communication is considered reestablished. All reactions to the skipped steps are triggered. The skipping is displayed in the tooltip.

If a communication error has been skipped, then a new communication error only occurs if the value of the formula for the **Loss of communication** property switches to *FALSE*.

COUNTER FOR THE RECIPE AND OPERATION FOR REESTABLISHED COMMUNICATION

In the status line of the recipe editor, there is information about errors in the recipe available with the symbol of the error display and the attendant tool tip information. Only the information with the highest priority is shown. Active errors and historical errors are not shown at the same time.

Prioritization and coding of the error display in the tooltip:

Priority	Color	Description
1	red	Currently *x* errors active.
2	red	Currently *x* communication errors not yet acknowledged.
3	yellow	Errors were present.
4	green	faultless

Note: "**Currently x errors active**" counts both procedure errors and communication errors.

15.2 PLC error

The detection of PLC errors can be configured for each phase.

To configure the detection of PLC errors in the Runtime:

1. Configure a variable that reports the PLC error in zenon
2. Highlight the desired phase in the Editor in the Batch Control module.
3. Click, in the **Loss of communication** property group, on the **PLC error** property
4. In the formula editor (on page 303) that opens, define the condition for detecting PLC errors using the configured variables.
   Note: The formula can be displayed in the report (on page 298).

To display the number of active and rectified PLC errors in a screen:

1. Select the unit for whose phases PLC error detection has been configured.
2. Go to the property group **Runtime information**/**Error**.
3. Configure the variables for the properties.
   - **Number of active PLC errors**
   - **Number of rectified PLC errors**
4. Configure the evaluation and display of variables in the Runtime.

CHECK IN RUNTIME

If a formula is linked for PLC errors, this is checked whilst the phase is active. Checking starts once communication with the PLC has been established and lasts until the phase has been deactivated. The connection is considered established if all values for the formula for detecting communication problems have been received.

If a communication error (on page 343) occurs, the PLC error is not checked as long as the communication error has not been acknowledged. Once it has been acknowledged, checking continues with the currently-pending value.

PLC errors and the rectification of these are displayed with their own symbol in the phase or in the operation and each displayed with an entry in the tooltip.

If there is a PLC error when a phase is ended, this is amended to *rectified*. In contrast to a normal deactivation of a PLC error, a separate event is sent: *PLC error rectified by deactivation of the phase*. This only applies for the actual closing, but not for a restart. When restarting, no new event is sent for PLC errors that continue to be active.

16 Error treatment

Warnings and error messages are written in log files and can be analyzed with the Diagnosis Viewer. For this you must activate module Batch Control in the filter settings.

SYSTEM DRIVER VARIABLE

Batch Control provides the system driver **SYSDRV** with information via system driver variables. For information about their messages see manual **SYSDRV** in chapter **Topic - Batch Control**.

RECONSTRUCTION OF INDEX FILES

If the index file (**recipe.unique**) is not read when Runtime is started or it does not exist, it is reconstructed upon starting. During this time, requests from clients that need an ID or are processing an ID cannot be processed. Modifications are rejected, queries are answered after the index has been created.