# zenon manual

## Batch Control

**v.7.10**

**COPA-DATA**
do it your way

© 2013 Ing. Punzenberger COPA-DATA GmbH

# Contents

# 1. Welcome to COPA-DATA help

## GENERAL HELP

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

## PROJECT SUPPORT

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

## LICENSES AND MODULES

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

# 2. 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 know as ANSI/ISA–S88.

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

- ▶ PFC editor: (on page 125) For complex recipes with sequence selection.

- ▶ Matrix editor: (on page 165) For simple, sequentially or parallel running recipes.

Depending on the license, both or only one editors is available to you.

▶ **License information**

*Must be licensed for Editor and Runtime (single-user, Server, Standby and Client).*
*Licensing distinguishes after use of:*

▸ PFC editor

▸ Matrix editor

▸ both editors

*Editor:*

*For the engineering in the zenon Editor one of the Batch editors must be licensed.*

*Runtime:*

*For the execution in the Runtime one of the two Batch editors must be licensed in order to fill the list of recipes. Creating and editing recipes is only possible for the licensed editor.*

*Network:*

*In the network the license of the Server counts. Stand-alone licenses of the Clients are ignored in network operation.*

⚠ **Attention**

*Hints in the property help about VBA keywords and XML export are not functional at the moment.*

## 2.1     Introduction

The module Batch Control consists of three parts:

1.   <u>The engineering environment (on page 13) in the zenon Editor:</u>
     There all units (ISA nomenclature, chapter 4.2.5: units) 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) which is called process action according to ISA.

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. <u>PFC editor (on page 125) and Matrix editor (on page 165):</u>

   With the help of these editors, master recipes in zenon Runtime (ISA: master recipes) are created. They form the base of the control recipes which can be executed (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. <u>Recipe Execution Engine (on page 205) (REE):</u>

   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: Au`tomatic`, S`emi-automatic` and M`anual`.

**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 strictly follows the division between the processing of the Batch recipe (ISA: Procedural Control Model) and die execution of the process actions (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 cryptic label, an additional abstraction level was introduced. It contains the tags which are available in two types:

▶ Command tag: They are used to transfer set values to the PLC when a phase is executed.

▶ Return tags: They are used to return values from the PLC for evaluation.

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

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 seamlessly on the Standby!*

## 2.2    Terminology

In the zenon module Batch Control the following terms are used:

| Term | Definition |
|---|---|
| Unit | Physically available machine or equipment part with which phases can be carried out. |
| 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 manager | Internal management mechanism which manages the unit allocation for all REE's in the Runtime. |
| Action | Used in Batch Control: all commands which are used for editing a recipe e.g. inserting basic functions, testing recipes etc. |
| Begin simultaneous sequence | Element which ensures the separation of the recipe process in two or more branches. |
| Begin sequence selection | Element which 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. |
| Begin element | Element of module Batch Control with which every recipe begins. |
| 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 command "next step" 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. |
| End simultaneous sequences | Element which brings together the branches to one branch. |
| End sequence selection | 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 |

| | tags, a phase done condition, event reactions, etc... |
|---|---|
| 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. |
| Simultaneous sequence | Area of module Batch Control. A parallel branch starts with a begin parallel branch element and is brought together with an end parallel branch element to one execution branch. Between there are at least two branches which are executed parallel. |
| 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. These data can be transferred from the control system to the control and vice versa (teaching) in one step. We differentiate between standard recipes and RGM recipes.Im Modul Batch Control ist in einem Rezept zusätzlich zu den Daten auch der Ablauf definiert. Dabei wird zwischen Matrix-Rezept und PFC-Rezept unterschieden. |
| 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. Parameters of the applied basic functions 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 of the element. |
| Connection point | Part of the connection element in the Batch Control module. Connects |

| | |
|---|---|
| | two elements with eachother (e.g.  basic funtion with basic funtion or basic function with line). It changes color when the mouse pointer is on it. |
| `Connection element` | A possibility in the Batch Control of connecting elements with one another. It consists of a connection point and a connection line. |
| `Sequence selection` | Area of module Batch Control which ensure 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 a begin branch element and ends with an 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: basic functions, transitions,parallel circuits etc (ISA 88:  Master recipe). Template recipes serve as templates for control recipes. |
| `Branch` | Execution area for the Batch Control Module Phases, transitions and jump targets can be placed on it. |

## 2.3    Procedure

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

1.    Engineering in the zenon Editor

2.    Recipe creation in the Batch editor (PFC or matrix) in the Runtime

3.    Recipe execution in the Runtime

### ENGINEERING

The engineer depicts the existing physical world in the Editor. He defines and creates units (on page 14) and assigns phases (on page 19) with tags (on page 34) to them.

### RECIPE CREATION

In the Runtime the recipe creator defines the master recipes on base of the presets from the Editor. They define the process. At this only the units, phases and tags defined in the engineering environment

can be used.

**Exception:** If you activate property `Changeable in recipe` in group `Write set value` in the Editor, you can modify the corresponding tag. This is only possible for command tags.

**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. Recipe either run automatically, semi-automatically or manually.

The operator cannot influence the recipes.

**Exception:** If the recipe creator activates option (on page 144) `Changeable in the control recipe`, the operator can edit tags in the Runtime.

## 2.4 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.

## 2.5 Function authorizations

For changes in the Editor and in Runtime, the corresponding function authorizations can be issued to users. A warning is displayed if operations are executed for which there are no corresponding rights. Exception: No warning is displayed when editing.

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.

**Hints:**

- ▶ 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.

## 2.6 Engineering in the Editor

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

▶ create units (on page 14), phases (on page 19) and reactions (on page 25)

▶ create a screen (on page 43) of type `Batch Control`

▶ create a screen switch function (on page 53) for the screen of type `Batch Control`

**DETAIL VIEW**

In module Batch Control the detail view divided in two:

▶ The left part features the unit tree. The entry `Unit` is the starting point of the tree.
On the next level the existing units are displayed. The phase belonging to each unit follow. Each phase has the subitem reactions.

▶ In the right part a flat list of the units, phases, tags or reactions is displayed depending on what level is chosen on the left side.

## 2.6.1 Job variables

Job variables can be linked in the Editor. Job variables make it possible to assign job IDs to Batch productions. 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.

Job variables defined in the Individual job variable property can be allocated using a function (on page 92). Otherwise the global Job variable is used.

**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 property, the type of transfer to 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 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, five decimal points are taken into account. Zeros at the end are cut off.

4. You can also define a Individual job variable if you wish

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

**INDIVIDUAL JOB VARIABLE**

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 zenon functions (on page 92), select the desired variable and the type of allocation

## 2.6.2    Units

To create a new unit:

1. in the project manager go to node `Batch Control`

2. in the detail view select `Unit`

3. in the context menu select menu item **New unit**

4. a new unit is created in the detail view

## Tool bar unit tree

The tool bar corresponds to the complete unit tree. Depending in the selected element symbols are available or deactivated.

| Parameter | Description |
|---|---|
| **Rename** | Opens the name filed of the selected element for renaming.<br><br>Not available for main node **Units**. |
| **Delete** | Deletes selected element after confirmation message.<br><br>Not available for main node **Units**. |
| **Expand all** | Displays the entire tree structure.<br><br>By clicking on the arrow you receive a drop-down list in which you can select on of the following commands:<br><br>▶ Expand all: expands all nodes<br><br>▶ Collapse all: collapses all nodes<br><br>▶ Expand selected: expands all selected nodes<br><br>▶ Collapse selected: collapses all selected nodes<br><br>A click on the button always expands all elements.<br><br>Via double click on the superordinate entry elements can also be expanded or collapsed. |
| **Help** | Opens the online help. |

**Project tree - context menu unit**

| Parameter | Description |
|---|---|
| New unit | Creates a new unit. |
| Editor profile | Opens the drop-down list which includes pre-defined Editor profiles. |
| Help | Opens the online help. |

**Context menu units**

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

| Menu item | Action |
|---|---|
| New unit | Creates a new unit. |
| Help | Opens the online help. |

**Context menu unit X**

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

| Menu item | Action |
|---|---|
| New phase | Creates a new phase. |
| Rename | Makes it possible to rename the selected unit. |
| Delete | Deletes selected unit from the tree after a confirmation message. |
| Help | Opens online help. |

**Detail view units**

| Menu item | Action |
|---|---|
| New unit | Creates a new unit in the detail view. |
| Copy | Copies selected entries to the clipboard. |
| Paste | Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of". |
| Delete | Deletes selected entries. |
| Remove all filters | Removes all filter settings. |
| 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. |
| Replace text in selected column | Opens the dialog for searching and replacing texts. |
| Properties | Opens the property window. |
| Help | Opens the online help. |

## Detail view unit X

| Parameter | Description |
|---|---|
| New phase | Creates a new phase in the detail view. |
| Copy | Copies selected entries to the clipboard. |
| Paste | Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of". |
| Delete | Deletes selected entries. |
| Remove all filters | Removes all filter settings. |
| 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. |
| Replace text in selected line | Opens the dialog for searching and replacing. |
| Properties | Opens the property window. |
| Help | Opens the online help. |

## Information in Runtime

Information on individual units can be called up and displayed 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 was the first to allocate the unit. Variables are always only filled in the event of a change.

The variables for the information desired in 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.

- ▶ Control recipe: Information on ID, name and description and job ID of of the control recipe.

- ▶ Execution: Information on the number of active recipes and execution status and mode, numerically and as a text respectively.

  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 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.

### EXECUTION STATUS VALUES

The variable linked at Execution status (numeric) is coded as follows:

| State | Return | Remark |
|---|---|---|
| Idle | 0 | |
| Running | 1 | |
| Executed | 2 | Is never displayed, because a completed recipe does not allocate the unit. |
| Stopping | 3 | |
| Stopped | 4 | Is never displayed, because a stopped recipe does not allocate the unit. |
| Pausing | 5 | |
| Paused | 6 | |
| Holding | 7 | |
| Held | 8 | |
| Aborting | 9 | |
| Aborted | 10 | Is never displayed, because an aborted recipe does not allocate the unit. |
| Restarting | 11 | |

The variable linked at Execution mode (numeric) is coded as follows:

| Mode | Return | Remark |
|---|---|---|
| Ignore | 0 | No recipe runs |
| Automatical | 1 | |
| Semi-automatic | 2 | |
| Manual | 3 | |

## 2.6.3    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 in the tank.

2. A temperature sensor which measures the actual temperature in the tank. Connect this sensor with a control.

3. In the control a program which 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 - which must be reached and which is implemented via a return 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 25) and conditions (on page 22) for the response.

A phase therefore consists of:

| Type | Description |
|---|---|
| Command tags (on page 35) | The set values which should be transferred to the control |
| Return tags (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 25) | 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). |
| Conditions (on page 22) | Are used for the evaluation of the return tags: The state of the `process action` in the control is evaluated. |
| Times (on page 24) | 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 for the phase or first create the desired unit

2. in the context menu select menu item `New phase`

3. a new phase is shown in the detail view

4. the subitem `Reactions` is automatically added to the phase

> ### 💡 Info
>
> *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.*

## Context menu phase

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

| Menu item | Action |
|---|---|
| `New phase` | Creates a new phase. |
| `Rename` | Makes it possible to rename the selected unit. |
| `Delete` | Deletes selected unit from the tree after a confirmation message. |
| `Help` | Opens online help. |

### CONTEXT MENU PHASE X

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

| Parameter | Description |
|---|---|
| `New command tag` | Creates a new command tag (on page 35). |
| `New return tag` | Creates a new return tag (on page 36). |
| `Rename` | Makes it possible to change the name of the currently selected phase. |
| `Delete` | Deletes the currently selected phase after a confirmation message. |
| `Help` | Opens the online help. |

## Detail view phase

| Parameter | Description |
|---|---|
| `New command tag` | Creates a new command tag in the detail view. |
| `New return tag` | Creates a new return tag in the detail view. |
| `Copy` | Copies selected entries to the clipboard. |
| `Paste` | Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of". |
| `Delete` | Deletes selected entries. |
| `Remove all filters` | Removes all filter settings. |
| `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. |
| `Edit text in selected line` | Opens the dialog for searching and replacing. |
| `Properties` | Opens the property window. |
| `Help` | Opens the online help. |

## 2.6.4   Conditions

The conditions are used to inform the REE about the status of the `process action` in the control. For evaluating the conditions formulas are used which where created with the formula editor (on page 238).

Hint: Use a single status tag (return tag) which takes on different values in order to transmit the status of the `process action` in the control to the phase. You can find an example in chapter Example for status parameter (on page 36).

**ENGINEERING**

To create a new condition:

1. click on the corresponding phase

2. in the properties select the desired conditions from node `General` or`Condition transient states`

3. click in the field for the value or on button ...

4. the formula editor is started

5. define the formula (on page 238) for the condition

Note: The counterpart to the conditions are the reactions (on page 25). With them the status of the REE is transmitted to the process actions in the control.

**TRANSIENT STATES**

As transitions the following properties are available

▶ `Paused`
Within the phase the process stops at:
- Waiting for `Finished`
- Waiting for allocation
- Waiting for interlocking
- Waiting for `Phase finished`
- Check for parallel execution

▶ `Held`
Within the phase the process stops at:
- Waiting for `Finished`
- Waiting for allocation
- Waiting for interlocking
- Waiting for `Phase finished`
- Check for parallel execution

▶ `Stopped`

▶ `Aborted`

▶ `Restarted`
Phase is completely restarted.

▶ Escape condition
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 226) chapter.

**VARIABLES**

Transition conditions can be linked with a binary variable which defines when the phase changes its status. The status changes is delayed until:

- the value of the variable is defined

- the value of the variable is 1

- the status of the variable is not invalid.

If no variable 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 values for a variable which define a status change are 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.

**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 27) is triggered. With the reactions (on page 32) 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.

## 2.6.5 Reactions

Reactions are the most important piece to influence the recipe process and to communicate with the control. Reactions are always based on events. This can be events of the REE (e.g.: `Phase started`) and also general events (e.g. `Exit Runtime initiated`). With the help of reactions you can e.g. tell the control when a phase has been started or finished in the REE and when all command tags have been written.

Likewise you must transfer the status of the phase to the control with the help of the reactions. 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. As reaction you can transfer this status change as set value input to the control. Only then can the control react and stop the `process action`. You can find an example in chapter Example for status parameter (on page 36).

**Note**: The counterpart to the reactions are the conditions (on page 22). With them the states of the process action in the control is transferred to the REE.

**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. in the context menu select menu item `New reaction`

3. a new reaction is shown in the detail view

4. in the detail view click on the entry in column `Event`

5. select the desired event from the drop-down list and define the desired reaction type (on page 32) in the property window.

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. Reactions of the same type can only be re-sorted using the toolbar or context menu (on page 26).

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.

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.

Values of the reactions are logged in the CEL.

## Context menu reactions unit tree

| Tag | Description |
|-----|-------------|
| New reaction | Creates a new reaction in the detail view. |
| Help | Opens online help. |

## Detail view reactions



| Parameter | Description |
|-----------|-------------|
| New reaction | Creates a new reaction in the detail view. |
| Execution order: Earlier | For reactions of the same type:<br>Moves the reaction forward in the execution order. |
| Execution order: Later | For reactions of the same type:<br>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. |
| Copy | Copies selected entries to the clipboard. |
| Paste | Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of". |
| Delete | Deletes selected entries. |
| Remove all filters | Removes all filter settings. |
| 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. |
| Edit text in selected line | Opens the dialog for searching and replacing. |
| Properties | Opens the property window. |
| Help | Opens the online help. |

**Events**

Each `reaction` is a reaction to an event. The event is defined in property `Event`. For each event several reactions can be defined. The execution order can be defined in the detail view.

When validating the recipe in Runtime, the name of the event is displayed in the event of an error. Syntax: **(event name.x)** whereby **event name** corresponds to the `Event`. **x** is a number that indicates the position in the execution sequence.

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

| Event | Description |
|---|---|
| `Unit allocation not possible` | Is triggered if the unit was not allocated successfully at first try. |
| `Escape condition fulfilled` | Is triggered if the Escape condition for exiting from a phase is met. |
| `Escape condition started` | Is triggered if the Escape condition for exiting from a phase is started. |
| `Finished writing command tags` | Is triggered if all command tags have been written. It cannot be guaranteed however that really all tags arrived at the control. It depends on the communication and the respective driver. It can however be assumed.<br><br>**Recommendation:** Use this event to tell the PLC that the phase has written all command tags and the PLC can start processing the process actions. |

| Command tag without value | Is triggered if the command tag should be toggled and the variable linked to the tag doe not have a valid initial value. |
|---|---|
| Input interlocking blocked | Is triggered if the input interlocking was blocked at the first check.<br><br>Makes only sense if property `Interlocking condition` was configured. |
| Phase deactivated | Is triggered if the phase was started and now is finished. |
| Phase activated | Is the first event which is triggered.<br><br>With this event, you tell the PLC that the phase has been activated in the REE and that the phase is expected to be started soon. |
| Phase started | With this event you tell the PLC that the phase has been started in the REE and that it is likely that the command tags will be written soon.<br><br>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. |
| Phase started multiple times | A phase can only be active once. If it is activated several time in parallel, this event is triggered. |
| Communication reestablished | This event reports that communication is functional again after an interruption. |
| Loss of communication | This event reports that communication has been interrupted. |
| Loss of communication fixed | This event reports that the communication failure has been rectified. |
| REE mode change: Automatical | The REE switched to mode `Automatic`. |
| REE mode change: Manual | The REE switched to mode `Manual`. |
| REE mode change: Semi-automatic | The REE switched to mode `Semi-automatic`. |
| Exit Runtime initiated | Is triggered if the Runtime is exited. This is an especially critical state for module Batch Control 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 230) |
| | At this event no reaction types of group `Influencing the recipe` are possible. |
| `Runtime restart` | The Runtime was restarted. |
| `Status change` | When the phase changes its status, the correspondingreaction is activated. |
| `Status change: Aborting` | The phase is aborted at the moment. |
| `Status change: Aborted` | The phase was aborted. |
| `Status change: Executed` | The phase is finished. |
| `Status change: Held` | The phase was stopped. |
| `Status change: Holding` | The phase is held at the moment. |
| `Status change: Resuming` | The phase is resuming after a break. A status change in the object from `Paused` to `Running` triggers the events `Resuming` and `Running`. |
| `Status change: Stopped` | The phase was stopped. |
| `Status change: Running` | The phase is executed. |
| `Status change: Restarting` | The phase is restarting at the moment. |
| `Status change: Pausing` | The phase is switched to `Paused` at the moment. |
| `Status change: Paused` | The phase is paused. |
| `Status change: Stopping` | The phase is stopping at the moment. |
| `Linked variable interrupted` | ▸  If the value of a variable with status 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 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 status change at transient states allowed |
| | ▶ Note: INVALID events are not processed in the sequence that they occur. If an INVALID event occurs whilst processing another event, this event can go before the one that is being executed. |

| | |
|---|---|
| `Waiting period unit allocation exceeded` | Is triggered if the waiting duration for the unit allocation runs. Can also occur during `Paused` and `Held`. |
| `Waiting period input interlocking exceeded` | Is triggered if the waiting duration (time out) for the input interlocking expired.<br><br>Makes only sense if a `Interlocking condition` was defined. |
| `Waiting period following condition exceeded` | Is triggered if the phase was not finished within the scheduled waiting duration (time out) although the phase done condition was fulfilled. |
| `Maximum waiting period phase done condition exceeded` | Is triggered if the waiting duration (time out) for waiting for the phase done condition (`Finished`) was exceeded. |
| `Phase done condition completed` | Is triggered if the phase is finished. This event is the last reaction of the phase and independent of the reason of the finishing. With this the phase done condition is fulfilled.<br><br>This event is also triggered at a restart. |

**ALLOW EXECUTION BEFORE "PHASE STARTED"**

Events can be also be approved before the `phase started` event. To do this, the Allow execution before start event property must be activated for the corresponding event. This property can only be configured for events that are possible both before and after `phase started`. The value is automatically set according to type for all other events.

The following events are approved before and after "phase started":

- ▶ REE mode change

- ▶ Status change

- ▶ Exit from a phase

- ▶ Exit Runtime initiated

- ▶ Restart Runtime

- ▶ Linked variable interrupted

- ▶ Phase deactivated (can occur before if the phase was exited before `phase started`)

- ▶ Waiting period for subsequent conditions (can occur before if the phase was exited before `phase started`)

- ▶ Communication failures (however only from the point at which the values for the Loss of communication property are waited for)

The following events are only approved before "phase started":

- ▶ Phase activated

- ▶ Unit allocation

- ▶ Phase started more than once (exclusive execution)

All other events are only approved after the `Phase started` event.


**Reaction types**

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

| Reaction type | Description |
|---|---|
| Tag set value | Influences command and return tag directly. All tag data types can be used.<br>**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. |
| CEL entry | Creates entries in the CEL and log files. With this the reaction can be documented and the recipe process can be tracked later. For this property `Create CEL entry` must be activated. The text for the CEL is defined in property `CEL message text`. |
| Function | Makes it possible to link any zenon function.<br><br>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.<br><br>**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`. You can use both command tags and return tags. The data type of source and target tag must be identical otherwise an error is displayed at the validation of the recipe. |
| Recipe influence | Make it possible to:<br><br>▸ change the REE mode<br><br>▸ execute REE commands<br><br>▸ execute phase commands<br><br>With this you can react on serious events such as `Waiting duration exceeded` or `Linked variable invalid`.<br><br>**Note:** Use this reaction type carefully as this reaction type influence the entire recipe process.<br><br>For each event you can only once:<br><br>▸ set the `REE mode` and<br><br>▸ write a single `Commando`<br><br>Because e.g. it does not make sense to pause and hold the recipe at the same time with the same event.<br><br>You can read more about commands in the commands and actions (on page 109) chapter. |

## 2.6.6    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 an unique name and a description. With this the engineer can give the recipe creator or user a description for what the tag is used or which effects it has.

You can add any number of tags to a phase. We distinguish between command tags (on page 35) and return tags (on page 36). Each tag can be switched between command and return tag at any time.

**Context menu tag unit tree**

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

| Parameter | Description |
|---|---|
| New command tag | Creates a new command tag (on page 35). |
| New return tag | Creates a new return tag (on page 36). |
| Rename | Makes it possible to change the name of the currently selected phase. |
| Delete | Deletes the currently selected phase after a confirmation message. |
| Help | Opens the online help. |

**Detail view tag**

| Parameter | Description |
|---|---|
| `New command tag` | Creates a new command tag in the detail view. |
| `New return tag` | Creates a new return tag in the detail view. |
| `Copy` | Copies selected entries to the clipboard. |
| `Paste` | Pastes the contents of the clipboard. If an entry with the same name already exists, the content is pasted as "Copy of". |
| `Delete` | Deletes selected entries. |
| `Remove all filters` | Removes all filter settings. |
| `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. |
| `Edit text in selected line` | Opens the dialog for searching and replacing. |
| `Properties` | Opens the property window. |
| `Help` | Opens the online help. |

## Command tags

Command tags contain the set values which should be transferred to the control. When executing the phase, all command parameters are saved (on page 228) in the control unit. You can find the exact description in chapter: Process of a phase in detail (on page 220).

Command tags can also be used in transitions (on page 22), conditions (on page 22) and reactions (on page 25).

Command parameters have a number of properties which can be defined via the property window. Each tag must be linked with a variable. The data type of the variable must comply with the data type of the tag and the set value limits of the parameters 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 `Changeable in 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.   in the context menu select menu item **New command tag**

3.   a new command tag is created in the detail view

### Return TAGs

The return tags contain the return values with which the process action of the REE communicate its status. Normally the value is set by the control and evaluated by the REE. Return tags can be evaluated in transitions (on page 22) and conditions (on page 22).

Likewise they can be used in reactions (on page 25) and can also be written there. For this the are listed as target tags at `Allocate tag` and `Tag set value`.

#### ENGINEERING

To create a new return tag:

1.   select the desired phase for which you want to create a new return tag

2.   in the context menu select menu item **New return tag**

3.   a new command tag is created in the detail view

### 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. We recommend that you execute both parameters as return parameters. This may seem illogic for commands in write direction at first glance but has the following background: At the execution of the phase, all command tags are written. With this they are visible in the list of command tags and can therefore be deleted 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 status of the phase in the the recipe.

Especially at the writing of a command tag it makes sense to not simply inform the PLC about the writing but the status when all values have been written and the PLC therefore can start to process the process action.

For this it is best to use the reaction to event `Finished writing command tags`. At reactions to an event you can also write values to an return tag. Therefore it is better to use return tags for both status tags.

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 | Finished writing command tags |
| 3 | Phase finished: Phase done condition fulfilled and `Minimum execution duration` reached (if engineered) |
| 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 | REE mode change: Automatical |
| 21 | REE mode change: Semi-automatic |
| 22 | REE mode change: Manual |
| 23-29 | Reserve |
| 30 | Exit Runtime initiated |
| 31 | Runtime restart |
| 32 | Unit allocation not possible |
| 33 | Waiting period unit allocation exceeded |
| 34 | Input interlocking blocked |
| 35 | Waiting period input interlocking exceeded |
| 36 | Maximum waiting period phase done condition exceeded |

| 37 | Waiting period following condition exceeded |
|---|---|
| 38 | Phase started multiple times |

You can find the exact meaning of the events in chapter Event type (on page 27).

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 actionfinished | `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 | `Abborted` |

Link the values with a formula in the respective property.

**Hint:** You can copy the formula and just change the respective value. If you make this setting at the beginning of the engineering of the first phase, you can copy the entire phase and with that have this settings for all phases.

### Execution period

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.

## 2.6.7 Keyboards

Parameters can be amended in Runtime. Adapted keyboards are available for this.

**KEYBOARDS FOR BATCH CONTROL**

You define keyboards for use in the Batch Control module in general in the properties for the module in the Edit tag/Keyboards group. Define the desired keyboards for:

▶ Binary tay

▶ Numeric tag

▶ String tag

▶ Time period tag

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.

## KEYBOARD FOR WRITING SET VALUE FOR PARAMETER

You define keyboards for writing set values for individual parameters in the properties of the respective parameter in the `Write set value`/Keyboards group. 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. In the Screen Keyboard property, define which keyboard screen is to be called up in Runtime

## 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 this. 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:

▶ 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 this 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 names are sent to the system variables again.

▶  For the `Changeable in control recipe property`, the keyboard for Binary tay 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 the parameter 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.

## 2.6.8    Input interlocking

With the help of an input interlocking the phase is only executed in the Runtime when the condition for the input interlocking is fulfilled.

The input interlocking is configured via property `Interlocking condition`. Via the formula editor (on page 238) the condition is defined which the input interlocking must fulfill. The formula can consist of one or more command tags and return tags of the phase. Value and status of the variables can also be used. The formula returns `TRUE` or `FALSE` as result. The condition can be displayed in the Runtime but cannot be changed there. The waiting period for the input interlocking is configured with the help of property `Waiting period input interlocking`.

If an input interlocking was defined, the phase is executed as soon as the following conditions are fulfilled:

▶  the phase is active

▶  the unit is allocated

▶  the phase is not active twice

▶  the phase is not already executed

▶  the input interlocking is fulfilled

If no input interlocking is linked, the phase is executed when:

▶  the phase becomes active

▶  the unit is allocated

▶   the phase is not executed twice

▶   the needed variables all have a value

> 💡 **Info**
>
> *You can find more information about input interlockings in chapter Processing a phase in detail (on page 220).*

## 2.6.9     Create screen of type Batch Control

In screen of the type Batch Control master recipes (on page 120) and/or control recipes (on page 187) can be:

▶   created
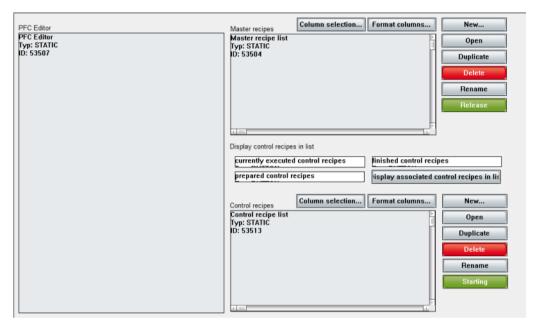
▶   managed

▶   executed in the Runtime

The display size of the screen depends on the selected template size.

To create a screen of type Batch Control:

1.   in node `Screens` in the context menu select menu item `New screen`

2.   in column `Screen type` select `Batch Control` from the drop-down list

3.   in the menu bar `Control elements` select one of the three default settings:

  •   Insert template: provides templates for different resolutions and selection of only master recipes or default (master and control recipes)

  •   Default master recipe: adds selected elements for a master recipe

  •   Default control recipe: adds selected elements for a control recipe

4.   the default elements for the desired setting are created

5.   if needed you can add additional control elements from menu `Control elements`

6. create a screen switch (on page 53) in order to display the screen in the Runtime

| Element | Description |
|---|---|
| `Insert template` | Inserts control elements for master recipes and/or control recipes on predefined locations on the screen. These control elements can be supplemented, reduced and positioned newly. |
| `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. |
| `Recipe control` | Control elements for the recipe control. |
| `Execution commands of the recipe` | Control elements for recipe commands: <br><br>▸ Start recipe <br><br>▸ Recipe pausing <br><br>▸ Recipe resuming <br><br>▸ Recipe holding <br><br>▸ Restart recipe <br><br>▸ Recipe stopping <br><br>▸ Recipe aborting |
| `Execution commands of the phases` | Control elements for phase commands: <br><br>▸ Phase pausing <br><br>▸ Phase resuming <br><br>▸ Phase holding <br><br>▸ Restart phase <br><br>▸ Escape phase |
| `Switch execution mode` | Control elements for control REE modes. |

| | ▶ Switch to `automatic mode` |
|---|---|
| | ▶ Switch to `semi-automatic mode` |
| | ▶ Switch to `manual mode` |

| | |
|---|---|
| `Execution navigation` | Control elements for navigation in recipes:<br><br>▸ Continue recipe only at selected execution position<br><br>▸ Continue recipe at all execution positions<br><br>▸ Skip active condition |
| `General` | General control elements:<br><br>▸ Check recipe for errors<br><br>▸ Edit element<br><br>▸ Display grid<br><br>▸ Colors for background and grid |
| `Partial recipe navigation` | Control elements for navigation in operations:<br><br>▸ Change to partial recipe template<br><br>▸ Switch to main recipe |
| `Master recipes` | Control elements for master recipes. |
| `Master recipes list` | In this list all master recipes can be displayed. The display can be limited by filters to an individual selection.<br><br>The filtering can be preset in the zenon Editor in the screen switch function (on page 53). Online filtering is also possible. These filters are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.<br><br>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.<br><br>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 62). |

| | |
|---|---|
| | **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 65). **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 121). |
| `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 changed the description of the control recipe.<br><br>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 description is opened.<br><br>The copy of the recipe automatically receives status `Editable` and can be edited further.<br><br>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.<br><br>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`.<br><br>If recipes must not be deleted - e.g. FDA regimented environment - we recommend not to engineer this |

| | |
|---|---|
| | button or to give it an appropriated `Authorization level`. When deleting 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. |
| **Switch master recipe to edit mode** | Changes the master recipe status of the selected recipes to `Editable`. In this status recipe 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 recipe to `Test mode`. Only faultless recipes can be switched to test mode. If error occur during the validation (on page 175), 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 118). |
| **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 175), 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 118). 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. |
| **Display associated 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.** | Deactivate the button **Display associated control recipes in list**. 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: <br><br> ▸ **Currently executed control recipes**: Displays only control recipes which are currently executed. Only takes effect as soon as you click on **Show associated control recipes in list**. <br><br> ▸ **Prepared control recipes**: Display only control recipes which are prepared for execution. Only takes effect as soon as you click on **Show associated control recipes in list**. <br><br> ▸ **Completed control recipes**: Displays only control recipes which have already been executed. Only takes effect as soon as you click on **Show associated control recipes in list**. <br><br> ▸ Outdated control recipe |
| Control recipe | Control elements for control recipes. |
| **List control recipes...** | In this list all control recipes can be displayed. The display can be limited by filters to an individual selection. <br><br> Per default the list is empty. For filling the list you must: <br><br> ▸ select master recipes <br><br> ▸ Set the **currently-executed control recipes**, **prepared control recipes** and **completed control recipes** filters <br><br> ▸ click button **display associated control recipes in list** <br><br> In addition to the filters mentioned above, you can filter the list itself. The filtering can be preset in the zenon Editor in the screen switch function (on page 53). Online filtering is also possible. These filters are discarded when the screen is called up again. A permanent definition is only possible 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 recipe...` | Opens a dialog in order to determine which columns should be displayed (on page 62).<br>**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 recipe...` | Opens a dialog to edit the column settings (on page 65).<br>**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 187) 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.<br><br>When creating a control recipe, a CEL entry is created. |
| `Rename 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.<br>Recipes can only be renamed if they are in status `Prepared`.<br>Also use this function in order to changed the description of the control recipe. |
| `Duplicate control recipe` | Only active if exactly one recipe is selected.<br>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.<br><br>The copy of the recipe automatically gets the status `Prepared` and can therefore be edited and started. The REE status (on page 212) of the duplicate is set to `automatic`.<br><br>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. FDA regimented environment - we recommend not to engineer this button or to give it an appropriate `Authorization level`. When deleting a recipe, a CEL entry is created. |
| **Open control recipe** | Opens the selected control recipe 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 defined REE mode. The recipes are executed invisibly at the Server. It is not necessary that the recipe is opened in the recipe editor. |
| **Tag lists** | List box for the display of parameters. Two list boxes can be created. These are configured in the screen switching (on page 90). |

## 2.6.10 Screen switch Batch Control

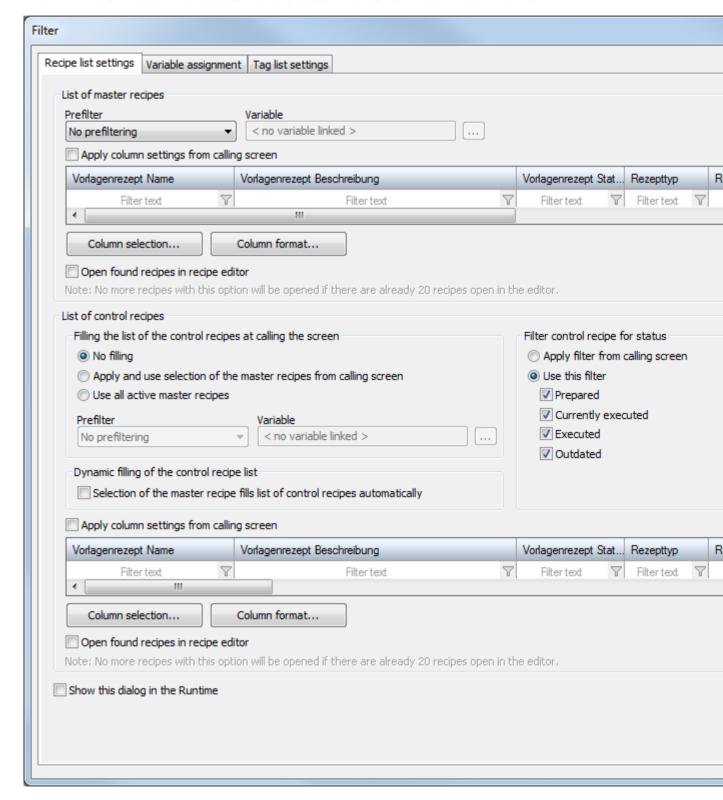To use Batch Control in the Runtime, engineer a screen switch function to a screen of type `Batch Control`:

1. in node **Functions** in the context menu select menu item **New function**

2. select function **Screen switch**

3. select the screen of type `Batch Control`

4. the filter dialog (on page 56) is displayed

5. configure the

a) Settings (on page 56) for the list of the master recipes/control recipes including prefiltering (on page 67)

b) Variable allocations (on page 70)

c) Tag list settings (on page 90)

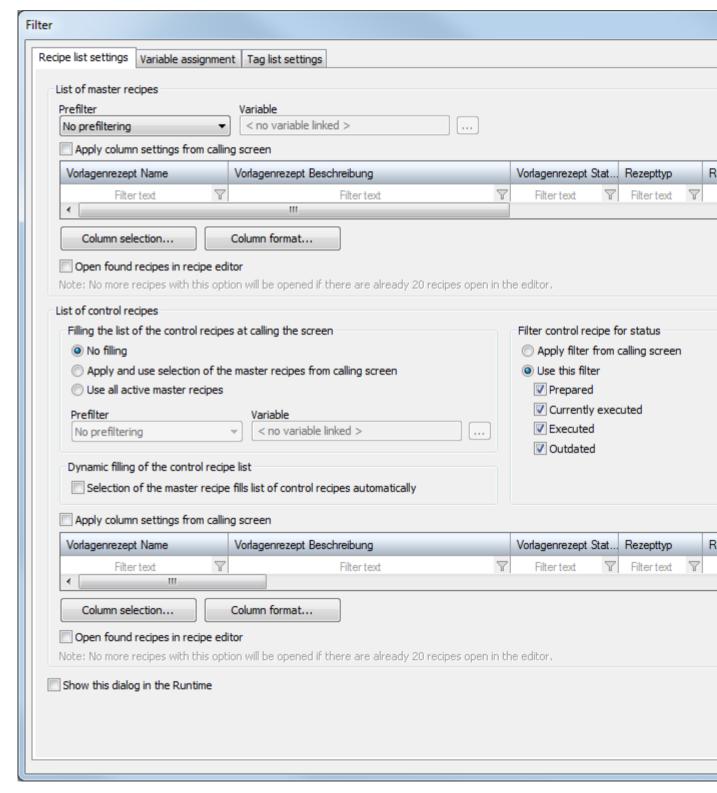6. link the function with a button on the screen in order to switch in the Runtime

**Recipe list settings**

The settings are configured for:

- ▶ List of master recipes

- ▶ List of control recipes

▶ Prefilter (on page 67)

Filter

Recipe list settings | Variable assignment | Tag list settings

**List of master recipes**

Prefilter
No prefiltering ▼

Variable
< no variable linked > | ...

☐ Apply column settings from calling screen

| Vorlagenrezept Name | Vorlagenrezept Beschreibung | Vorlagenrezept Stat... | Rezepttyp | R |
|---|---|---|---|---|
| Filter text ▽ | Filter text ▽ | Filter text ▽ | Filter text ▽ | |

[ Column selection... ] [ Column format... ]

☐ Open found recipes in recipe editor

Note: No more recipes with this option will be opened if there are already 20 recipes open in the editor.

**List of control recipes**

Filling the list of the control recipes at calling the screen
◉ No filling
○ Apply and use selection of the master recipes from calling screen
○ Use all active master recipes

Prefilter
No prefiltering ▼

Variable
< no variable linked > | ...

Dynamic filling of the control recipe list
☐ Selection of the master recipe fills list of control recipes automatically

Filter control recipe for status
○ Apply filter from calling screen
◉ Use this filter
  ☑ Prepared
  ☑ Currently executed
  ☑ Executed
  ☑ Outdated

☐ Apply column settings from calling screen

| Vorlagenrezept Name | Vorlagenrezept Beschreibung | Vorlagenrezept Stat... | Rezepttyp | R |
|---|---|---|---|---|
| Filter text ▽ | Filter text ▽ | Filter text ▽ | Filter text ▽ | |

[ Column selection... ] [ Column format... ]

☐ Open found recipes in recipe editor

Note: No more recipes with this option will be opened if there are already 20 recipes open in the editor.

☐ Show this dialog in the Runtime

| Parameter | Description |
|---|---|
| **List of master recipes** | Configuration for master recipes. |
| Prefilter | Select from a drop-down list whether master recipes should be pre-filtered when called up. Possible settings:<br><br>▸ No prefiltering: Recipes are not pre-filtered.<br><br>▸ Recipe ID: Recipes are filtered according to ID. Filter condition is defined in the Variable property.<br><br>▸ Recipe name: Recipes are filtered according to recipe name. Filter condition is defined in the Variable property. |
| Variable | Definition of the variables that provide the values for the prefiltering. Click the button ... and a dialog opens to configure the variables. |
| Apply column settings from calling screen | Active: The column settings are accepted by the screen that is calling them up in Runtime. The corresponding properties can no longer be configured in the Editor. |
| **List box** | Display of the configured columns. |
| **Column selection** | Opens the dialog for selecting the columns |
| **Column format** | Opens a dialog (on page 156) to format the columns. |
| Open found recipes in recipe editor | Active: The recipes found are opened in the recipe editor.<br><br>Note: The first 20 recipes found is the maximum that can be opened. |
| **List of control recipes** | Configuration for control recipes. |
| **Filter control recipe for status** | 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. Several statuses can be selected by selecting the corresponding checkbox:<br><br>▸ Prepared<br><br>▸ Currently executed<br><br>▸ Executed |

| | ▸ Outdated |
|---|---|

| | |
|---|---|
| **Filling the list of the control recipes at calling the screen** | Settings for the filling of the list when called up. Select an option.<br><br>▸ No filling<br><br>▸ Apply and use selection of the master recipes from calling screen<br><br>▸ Use all active master recipes |
| Prefilter | Select from a drop-down list whether control recipes should be prefiltered when called up. Possible settings:<br><br>▸ No prefiltering: Recipes are not pre-filtered.<br><br>▸ Recipe ID: Recipes are filtered according to ID. Filter condition is defined in the Variable property.<br><br>▸ Recipe name: Recipes are filtered according to recipe name. Filter condition is defined in the Variable property.<br><br>Job ID: 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 the button ... and a dialog opens to configure the variables. |
| **Dynamic filling of the control recipe list** | Settings for dynamic filling of the list. |
| Selection of the master recipe fills list of control recipes automatically | Active: When calling up in Runtime, the master recipes selected in the calling screen fill the list of the control recipes. |
| Apply column settings from calling screen | Active: The column settings are accepted by the screen that is calling them up in Runtime. The corresponding properties can no longer be configured in the Editor. |
| **List box** | Display of the configured columns. |
| **Column selection** | Opens the dialog for selecting the columns |
| **Column format** | Opens a dialog (on page 156) to format the columns. |
| Open found recipes in recipe editor | Active: The recipes found are opened in the recipe editor.<br><br>Note: The first 20 recipes found is the maximum that can be opened. |
| Show this dialog in | Active: This dialog is opened in Runtime when the screen is called |

| | |
|---|---|
| `the Runtime` | up. Changes to the settings can be made. |
| **OK** | Applies all changes in all tabs and closes the dialog. |
| **Cancel** | Discards all changes in all tabs and closes the dialog. |
| **Help** | Opens the 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.

## Column selection

### OPTIONS

| Button | Function |
|--------|----------|
| 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. |
| Down | Moves the selected entry downward. This function is only available for unique entries, multiple selection is not possible. |
| OK | Applies settings and closes dialog. |
| Cancel | Discards settings and closes the dialog. |
| Help | Opens online help. |

Note: These settings are only used in Runtime for dockable windows (on page 103) 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 120):
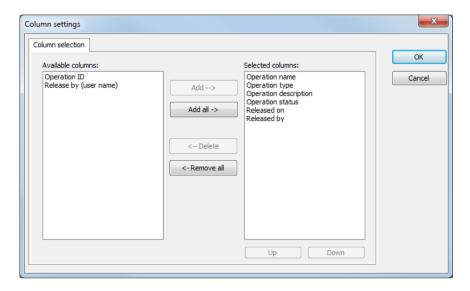
Column selection for list of the control recipes (on page 187):
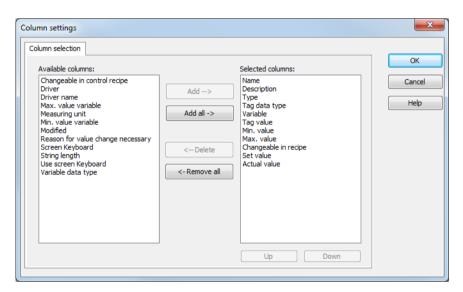
Column selection for list of the operations (on page 175) (only available in Runtime):



Column selection for Tag lists (on page 90)

## Column format

In this dialog you define the column format:

| Parameter | Description |
|---|---|
| **Available columns** | List of columns available using **column selection**. The column selected here is configured using the settings in the **Parameters** section. |
| **Parameter** | Settings for selected column. |
| Labeling | Name for column title. The column title is online language switchable. For this you must enter character @ in front of the name. |
| Width | Width of the column in characters.<br>Calculation: Number time average character width of the selected font. |
| Alignment | Alignment.<br><br>Possible settings:<br><br>▸ Left-justified: Text is justified on the left edge of the column.<br><br>▸ Centered: Text is displayed centered in the column.<br><br>▸ Right-justified: Text is justified on the right edge of the column. |
| Block column filter in Runtime | Active: The filter for this column cannot be changed in Runtime.<br><br>Note: Only available for:<br><br>▸ Batch Control<br><br>▸ Extended Trend<br><br>▸ Message Control<br><br>▸ Recipegroup Manager |
| **OK** | Applies settings and closes dialog. |
| **Cancel** | Discards settings and closes the dialog. |
| **Help** | Opens online help. |
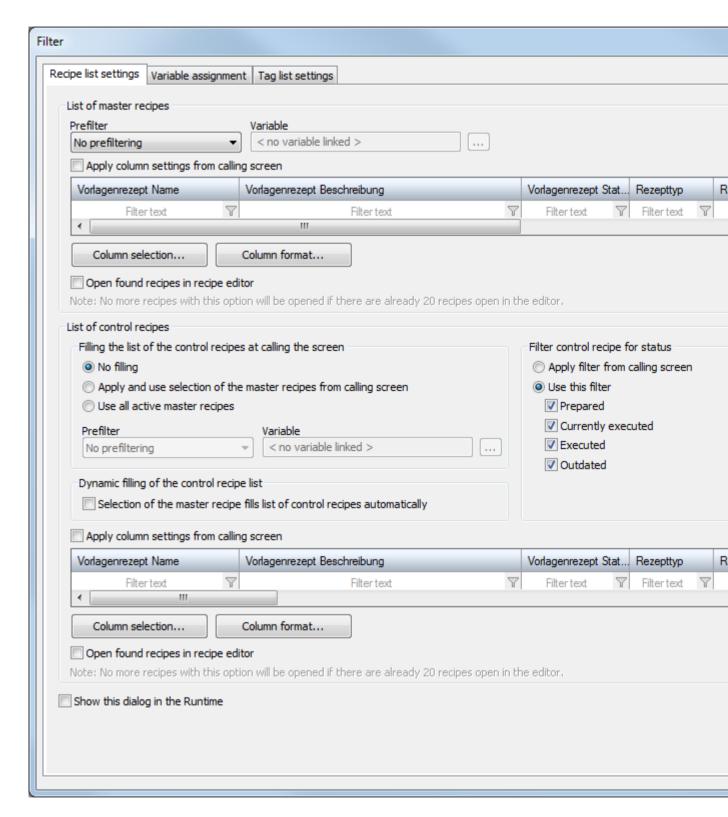
Note: These settings are only used in Runtime for dockable windows (on page 103) if there is no Runtime profile available for the user who is logged in.

**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.

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 65) dialog and activate the `Block column filter in the process screen` option. With this the user cannot modify the filters in the Runtime and therefore does not get the recipes, which e.g. he is not allowed to modify, not displayed.

### FILTERING FOR MASTER RECIPES

Configure:

1. Prefilter

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

2. Column settings

   a) in the screen switch function click on column filters for the **List of the master recipes** or the **List of control recipes**

   b) enter the desired filter text; wildcards (*) are allowed

   c) confirm the filter text with `Return` 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.

---

**Info**

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 state

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

2. <u>Filling the recipe list</u>

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

3. <u>Prefilter</u>

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

4. <u>Column settings</u>

   a) in the screen switch function click on column filters for the `List of the master recipes` or the `List of control recipes`

   b) enter the desired filter text; wildcards (*) are allowed

   c) confirm the filter text with `Return` for it to be applied.

5. <u>Automatic recipe switching</u>

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.

---

### Info

*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 Runtime are displayed (master recipes, status, column filter).*

---

**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 75) 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
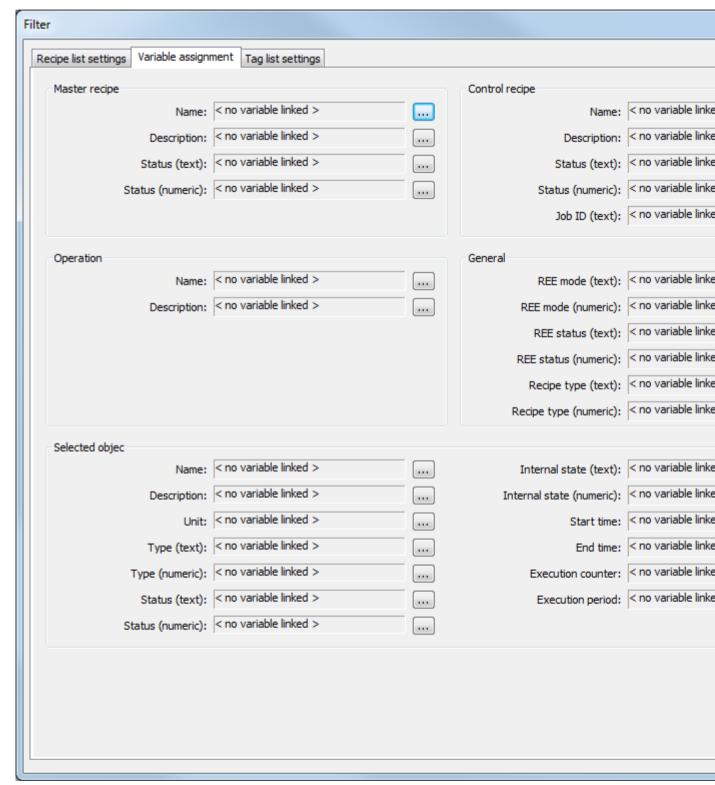
3. Select the desired variable

## DISPLAY

The following are displayed:

► For the recipe that is in focus:
  - Name
  - Description
  - State
  - REE mode
  - REE status
  - Recipe type

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

- Execution period

**Filter**

| Recipe list settings | Variable assignment | Tag list settings |

**Master recipe**

Name: < no variable linked >

Description: < no variable linked >

Status (text): < no variable linked >

Status (numeric): < no variable linked >

**Control recipe**

Name: < no variable linke

Description: < no variable linke

Status (text): < no variable linke

Status (numeric): < no variable linke

Job ID (text): < no variable linke

**Operation**

Name: < no variable linked >

Description: < no variable linked >

**General**

REE mode (text): < no variable linke

REE mode (numeric): < no variable linke

REE status (text): < no variable linke

REE status (numeric): < no variable linke

Recipe type (text): < no variable linke

Recipe type (numeric): < no variable linke

**Selected objec**

Name: < no variable linked >

Description: < no variable linked >

Unit: < no variable linked >

Type (text): < no variable linked >

Type (numeric): < no variable linked >

Status (text): < no variable linked >

Status (numeric): < no variable linked >

Internal state (text): < no variable linke

Internal state (numeric): < no variable linke

Start time: < no variable linke

End time: < no variable linke

Execution counter: < no variable linke

Execution period: < no variable linke

| Parameter | Description |
|---|---|
| **Master recipe** | Variable linkings for the master recipe (on page 77). <br><br>Status is displayed if the recipe has the focus or a phase or operation is highlighted. <br><br>The following can be linked: <br><br>▸ Name <br><br>▸ Description <br><br>▸ Status (text) <br><br>▸ Status (numeric) |
| **Control recipe** | Variable linkings for the control recipe. (on page 78) <br><br>Status is displayed if the recipe has the focus or a phase or operation is highlighted. <br><br>The following can be linked: <br><br>▸ Name <br><br>▸ Description <br><br>▸ Status (text) <br><br>▸ Status (numeric) <br><br>▸ Job ID (text) |
| **Operation** | Variable linkings for the operation. (on page 79) <br><br>Status is displayed if the recipe has the focus or a phase or operation is highlighted. <br><br>The following can be linked: <br><br>▸ Name <br><br>▸ Description |
| **General** | Variable linkings for general information. (on page 79) <br><br>Status is displayed if the recipe has the focus or a phase or operation is highlighted. <br><br>The following can be linked: <br><br>▸ REE mode (text): |

| | |
|---|---|
| | ▸ REE mode (numeric): |
| | ▸ REE status (text): |
| | ▸ REE status (numeric): |
| | ▸ Recipe type (text) |
| | ▸ Recipe type (numeric) |
| **Selected object** | Variable linkings for the selected object (phase or operation) (on page 81).<br><br>Status is displayed if a phase or an operation in the recipe is highlighted.<br><br>The following can be linked:<br><br>▸ Name<br><br>▸ Description<br><br>▸ Unit<br><br>▸ Type (text)<br><br>▸ Type (numeric)<br><br>▸ Status (text)<br><br>▸ Status (numeric)<br><br>▸ Internal state (text)<br><br>▸ Internal state (numeric)<br><br>▸ Start time<br><br>▸ Time of the end<br><br>▸ Execution counter<br><br>▸ Execution period |
| **OK** | Applies all changes in all tabs and closes the dialog. |
| **Cancel** | Discards all changes in all tabs and closes the dialog. |
| **Help** | Opens the online help. |

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

**Coding of the execution status**

### EXECUTION STATES

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

- ▶ Byte 0: Execution states

- ▶ 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.

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |

| | | |
|---|---|---|
| Idle | 1 | 1 |
| Running | 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 |
| Stopped | 14 | 1110 |
| Abborting (starting from state: running) | 15 | 1111 |
| Abborting (starting from state: Paused) | 16 | 10000 |
| Abborting (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.

| Value for numeric variable | | Meaning |
|---|---|---|
| **Decimal** | **Binary** | |
| 1 | 1 | Recipe |
| 2 | 10 | Phase |
| 3 | 11 | Operation object in the recipe |

**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.

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **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 running | 5 | 101 |
| Terminated with error | 6 | 110 |

**Control 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 or the recipe must be open and have the focus. Is filled in the control recipe and in the operation instance in the control recipe.

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | Decimal | Binary |

| | | |
|---|---|---|
| no control recipe active | 0 | 0 |
| Creation (not visible) | 1 | 1 |
| Prepared | 2 | 10 |
| Running | 3 | 11 |
| Executed | 4 | 100 |
| Terminated with error | 5 | 101 |

**Operation**

### NAME

Name of the operation 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.

**General**

### REE MODE (TEXT AND NUMERIC)

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

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |
| Automatical | 1 | 1 |
| Semi-automatic | 2 | 10 |
| Manual | 3 | 11 |

### REE STATUS (TEXT UND NUMERIC)

Status of the recipe

**BYTE 0**

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |
| Idle | 1 | 1 |
| Running | 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 |
| Abborting (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

| Value for numeric variable | | Meaning |
|---|---|---|
| **Decimal** | **Binary** | |
| 1 | 1 | Recipe |

### RECIPE TYPE (TEXT AND NUMERIC)

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

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **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 |

**Selected object**

The variables for the selected object always contain data if a single phase or a single operation was 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)

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **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

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |

| | | |
|---|---|---|
| `Idle` | 1 | 1 |
| `Running` | 2 | 10 |
| `Executed` | 3 | 11 |
| `Pausing`<br><br>(Starting from state: `running`) | 4 | 100 |
| `Paused` | 5 | 101 |
| `Holding`<br><br>(Starting from state: `running`) | 6 | 110 |
| `Holding`<br><br>(Starting from state: `Paused`) | 7 | 111 |
| `Holding`<br><br>(Starting from state: `Restarting`) | 8 | 1000 |
| `Held` | 9 | 1001 |
| `Restarting`<br><br>(Starting from state: `Held`) | 10 | 1010 |
| `Stopping`<br><br>(Starting from state: `running`) | 11 | 1011 |
| `Stopping`<br><br>(Starting from state: `Paused`) | 12 | 1100 |
| `Stopping`<br><br>(Starting from state: `Held`) | 13 | 1101 |
| `Stopped` | 14 | 1110 |
| `Aborting`<br><br>(Starting from state: `running`) | 15 | 1111 |
| `Aborting`<br><br>(Starting from state: `Paused`) | 16 | 10000 |

| | | |
|---|---|---|
| Aborting<br><br>(Starting from state: `Held`) | 17 | 10001 |
| Aborted | 18 | 10010 |

## BYTE 1:

Always empty.

## BYTE 2:

Object type. For the numerical value only

| Value for numeric variable | | Meaning |
|---|---|---|
| **Decimal** | **Binary** | |
| 2 | 10 | Phase |

## VALUE FOR A OPERATION INSTANCE

## BYTE 0

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |

| | | |
|---|---:|---:|
| Idle | 1 | 1 |
| Running | 2 | 10 |
| Executed | 3 | 11 |
| Pausing<br><br>(Starting from state: running) | 4 | 100 |
| Paused | 5 | 101 |
| Holding<br><br>(Starting from state: running) | 6 | 110 |
| Held | 9 | 1001 |
| Restarting<br><br>(Starting from state: Held) | 10 | 1010 |
| Stopping<br><br>(Starting from state: running) | 11 | 1011 |
| Stopping<br><br>(Starting from state: Paused) | 12 | 1100 |
| Stopping<br><br>(Starting from state: Held) | 13 | 1101 |
| Stopped | 14 | 1110 |
| Aborting<br><br>(Starting from state: running) | 15 | 1111 |
| Aborting<br><br>(Starting from state: Paused) | 16 | 10000 |
| Aborting<br><br>(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

| Value for numeric variable | | Meaning |
|---|---|---|
| **Decimal** | **Binary** | |
| 3 | 11 | Operation in the recipe. |

### INTERNAL STATUS (TEXT AND NUMERIC)

Is filled with the internal execution status of the element.

Composition of the internal status:

- ▶ Byte `0`: Status

- ▶ Byte `1`: Status Bits

- ▶ Byte `2`: Object type

## VALUE FOR A PHASE

### BYTE 0

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

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |

| Idle | 1 | 1 |
|---|---|---|
| Waiting for phase to be ready for starting | 2 | 10 |
| Waiting for unit allocation | 3 | 11 |
| Waiting for the unit allocation - timeout | 4 | 100 |
| Waiting for exclusive execution | 5 | 101 |
| Waiting for input interlocking | 6 | 110 |
| Waiting for the input interlocking - 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 period | 11 | 1011 |
| Wait until recipe has status "running" | 12 | 1100 |
| Waiting for following conditions | 13 | 1101 |
| Waiting for following condition - timeout | 14 | 1110 |

**BYTE 1**

Status. For numeric variable only

| Value for numeric variable | | Meaning |
|---|---|---|
| Binary | Decimal | |
| 00000001 | 1 | There is an execution error. |
| 00001000 | 8 | There is a communication error. |
| 00010000 | 16 | Loss of communication fixed. |
| 01000000 | 64 | Communication reestablished. |

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

| Value for numeric variable | | Meaning |
|---|---|---|
| Decimal | Binary | |
| 3 | 11 | Operation in the recipe. |

## POSSIBILITIES FOR PHASE

### BYTE 0

All possibilities.

### BYTE 1

All possibilities.

## BYTE 2

| Value for numeric variable | | Meaning |
|---|---|---|
| **Decimal** | **Binary** | |
| 2 | 10 | Phase |

## POSSIBILITIES FOR OPERATION INSTANCES

### BYTE 0

| Value for string variable | Value for numeric variable | |
|---|---|---|
| | **Decimal** | **Binary** |
| Idle | 1 | 1 |
| Execution of the internal objects | 8 | 1000 |
| Waiting for following conditions | 13 | 1101 |

### BYTE 1

Always empty.

### BYTE 2

| Value for numeric variable | | Meaning |
|---|---|---|
| **Decimal** | **Binary** | |
| 3 | 11 | Operation in the recipe. |

### START TIME

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

### TIME OF THE END

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

**EXECUTION PERIOD**
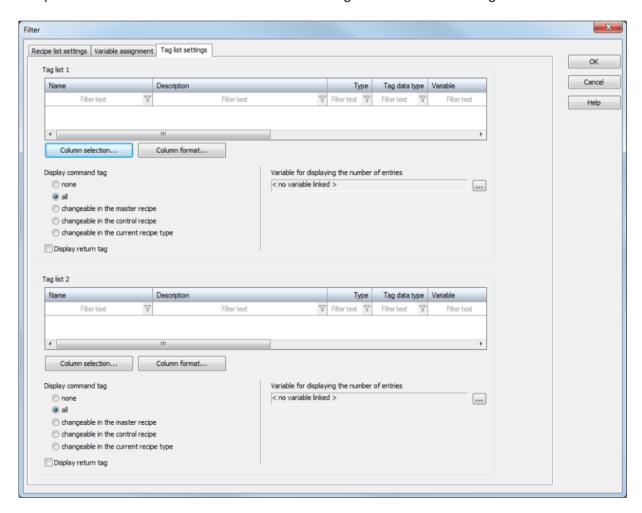
Time period that has expired during the execution.

**EXECUTION COUNTER**

Number denoting how often the element was executed

### 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.

| Parameter | Description |
|---|---|
| **Tag list 1** | Display of the configured lists. The columns can: <br><br> ▸ be filtered <br><br> ▸ have their width changed by dragging the column title with the mouse <br><br> ▸ arranged by dragging & dropping with the mouse |
| **Column selection** | A dialog for choosing the columns which shall be displayed is opened |
| **Column format** | Opens a dialog to format the columns. |
| Display command tag | Selection of the command parameters which should be displayed. Possible selection: <br><br> ▸ none: No command parameters are displayed. <br><br> ▸ all (default): All command parameters are displayed. <br><br> ▸ Changeable in the master recipe: Only command parameters for which the `changeable in the recipe` option has been set in the Editor are displayed. <br><br> ▸ Changeable in the control recipe: Only command parameters for which the `changeable in the recipe` option has been set in the master recipe are displayed. <br><br> ▸ changeable in the current recipe type: Only command parameters that can be changed in the current recipe type are displayed. |
| Display return tag | `Active`: Return parameters are displayed: |
| Variable for displaying the number of entries | Displays the maximum number of entries that is possible with the current pre-filtering of the screen filter. It is independent of the filtering in the columns and groupings. <br><br> A click on button `...` opens the dialog for selecting variables. |
| **Tag list 2** | Parameter list 2 with identical configuration possibilities to **Parameter list 1**. |

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

The parameter lists are recreated in Runtime if:

▶ The phase was edited

▶ The recipe was saved (even if the client has saved it)

## 2.6.11 zenon functions

With zenon functions, control commands can be sent to the batch execution and pre-defined control recipes can be created:

▶ Execute recipe command or mode switch (on page 92): sends control commands to batch execution

▶ Create control recipe function (on page 97): creates, in Runtime, a recipe that has been pre-defined in the Editor
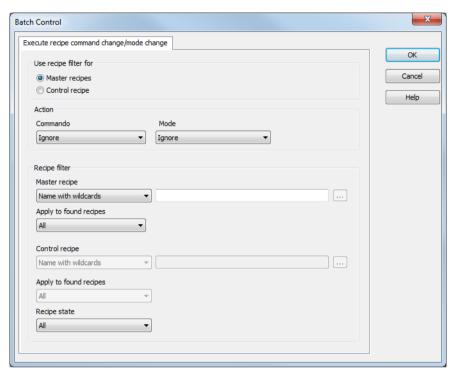
**Execute recipe command change or mode change**

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 change/mode change`

5. the dialog for configuring functions is opened

| Parameter | Description |
|---|---|
| `Use recipe filter for` | Selection of what the recipe filter is applied to: <br><br> ▶ Master recipe <br><br> ▶ Control recipe |
| `Master recipe` | `Active`: It is filtered on Master recipes. |
| `Control recipe` | `Active`: It is filtered on control recipes. <br><br> **Note:** The attendant master recipes must also be selected. If no master recipe has been selected for the control recipe, th filer cannot find the recipe being searched for in Runtime. <br><br> **Hint:** If the master recipe is not known, filtering of all master recipes with a placeholder is recommended. |
| `Action` | Selection of the action to be executed: <br><br> ▶ Command <br><br> ▶ Mode |
| `Command` | Selection of the command to be executed from drop-down list: <br><br> ▶ Ignore <br><br> ▶ Start recipe <br><br> ▶ Recipe pausing <br><br> ▶ Recipe resuming <br><br> ▶ Recipe holding <br><br> ▶ Restart recipe <br><br> ▶ Recipe stopping <br><br> ▶ Recipe aborting |
| `Mode` | Selection of the mode to which the recipe is to be switched: <br><br> ▶ Ignore <br><br> ▶ Automatic |

| | ▸ Semi-automatic |
|---|---|
| | ▸ Manual |

| Recipe filter | Configuration of the recipe filter |
|---|---|
| Master recipe | Parameters for the selection of the master recipe. Select from drop-down list:<br><br>▸ Name with wildcards:<br>    A name with placeholder can be entered into the input field. Filtering according to this name is carried out.<br><br>▸ Name from variable:<br>    The name of the master recipe is defined by a variable in Runtime. A click on button . . . opens the dialog for selecting variables.<br><br>▸ ID from variable:<br>    The ID of the master recipe is defined by a variable in Runtime. A click on button . . . opens the dialog for selecting variables. |
| Apply to found recipes | Definition of which master recipes the filter is applied to. Select from drop-down list:<br><br>▸ All<br><br>▸ Only oldest ID<br><br>▸ Only newest ID |
| Control recipe | Parameters for the selection of the control recipe. Select from drop-down list:<br><br>▸ Name with wildcards:<br>    A name with placeholder can be entered into the input field. Filtering according to this name is carried out.<br><br>▸ Name from variable:<br>    The name of the control recipe is defined by a variable in Runtime. A click on button . . . opens the dialog for selecting variables.<br><br>▸ ID from variable:<br>    The ID of the master recipe is defined by a variable in Runtime. A click on button . . . opens the dialog for selecting variables.<br>    Precisely one recipe can be found if the variable value at the time of execution is a valid ID of a control recipe. |

| | |
|---|---|
| | ▸ <u>Job ID from variable</u><br><br>Finds control recipes that belong to the master recipes already found and which have the given job ID. Any type of variable can be linked. The value is automatically converted into STRING. |
| `Apply to found recipes` | Definition of which control recipes the filter is applied to. Select from drop-down list:<br><br>▸ `All`<br><br>▸ `Only oldest ID`<br><br>▸ `Only newest ID` |
| `Recipe state` | Definition of which recipe status the filter is applied to. Select from drop-down list:<br><br>▸ `All`<br><br>▸ `Prepared`<br><br>▸ `Running` |
| **OK** | Applies settings and closes the dialog. |
| **Cancel** | Discards changes and closes dialog. the function is created however. |
| **Help** | Opens online help. |
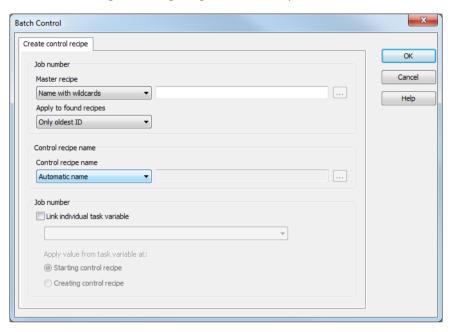
## Create control recipe function

With the help of the function `Create control recipe`, a pre-defined control recipe can be created in the Editor by means of a button in Runtime. 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 `Create control recipe`

5. the dialog for configuring functions is opened

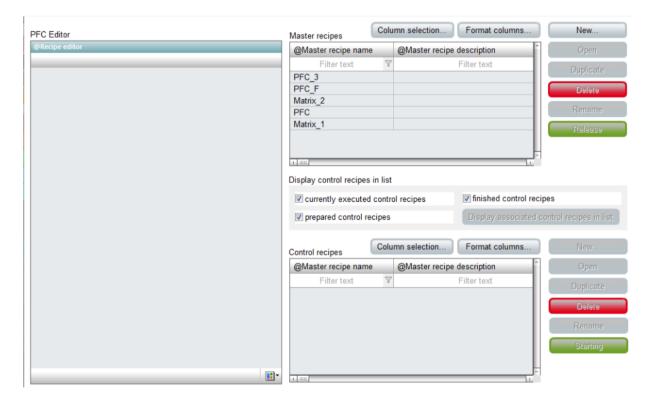| Parameter | Description |
|---|---|
| **Recipe filter** | Configuration of the recipe filter |
| Master recipe | Parameters for the selection of the master recipe. Select from drop-down list: <br><br> ▸ Name with wildcards: <br> A name with placeholder can be entered into the input field. Filtering according to this name is carried out. <br><br> ▸ Name from variable: <br> The name of the master recipe is defined by a variable in Runtime. A click on button `...` opens the dialog for selecting variables. <br><br> ▸ ID from variable: <br> The ID of the master recipe is defined by a variable in Runtime. A click on button `...` opens the dialog for selecting variables. |
| Apply to found recipes | Definition of which recipes the filter is applied to. Select from drop-down list: <br><br> ▸ All <br><br> ▸ Only oldest ID |
| **Control recipe name** | Configuration of the name of the control recipe. |
| Control recipe name | Selection of the naming from the drop-down list: <br><br> ▸ Automatic name: Name is automatically issued on creation <br><br> ▸ Name from variable: Name is taken from a variable. Click on button ... in order to open the dialog for selecting a variable. <br> If there is already a recipe with the name that has been transferred from the variable, no new control recipe is created. |
| **Job ID** | Configuration of the Job ID. |
| Link individual job variable | `Active`: A job variable (on page 13) can be linked. The variable must already be configured. Selection of the variable from the drop-down list: |
| Apply value from job variable | 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 changes and closes dialog. the function is created however. |
| **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.

## 2.7    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.



**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 194) chapter.
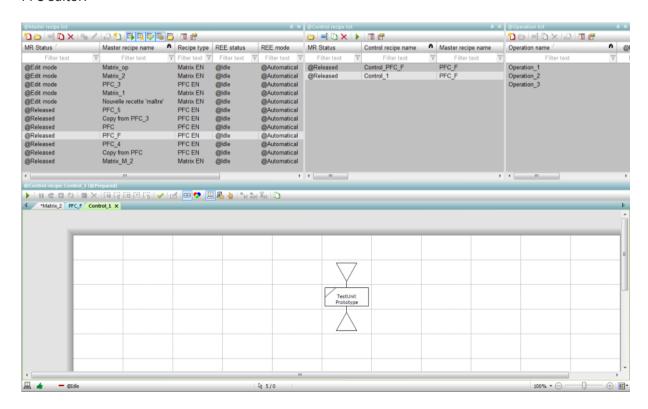
**SELECTION PROCEDURE IN LISTS:**

- ▶  `Ctrl+A`: selects all elements

- ▶  `Ctrl+mouse click`: adds master recipe to the existing selection

- ▶  `Key Shift+mouse click`: Extends selection from the currently selected master recipe to the clicked master recipe

## 2.7.1    User interface

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

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

**Editor operating elements**

The following are available for the editor:

▸ Tool bars (can be hidden)

▸ Tabs (on page 102)

▸ Dockable windows (on page 103)

### TOOLBARS

There are independent toolbars available in the Editor for each type of recipe and the different status. For details, see the Matrix recipe toolbars (on page 166) and Toolbars and PFC recipe context menu (on page 126) chapters. All actions of the individual symbols can also be engineered using their own buttons in the screen. If the tool bars are thus not needed, they can be displayed or hidden using the settings in the zenon Editor.

To show/hide tool bars:

1. In the zenon Editor, highlight the `Recipe editor` comment area in the `Batch Control` screen

2. go to property group `Representation`\Display editor control elements

3. Activate or deactivate the Toolbars checkbox

**Tabs in the Editor**

If several recipes are open in the 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. Select, in the context menu of the recipe, 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. |
| `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. <br><br> ▸ Move display: The recipe is moved and the upper group is removed. <br><br> ▸ Move display in 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. <br><br> ▸ Move display: The recipe is moved and removed from the left group. <br><br> ▸ Open display in parallel: The recipe is displayed in parallel in both groups. |

**Dockable windows**

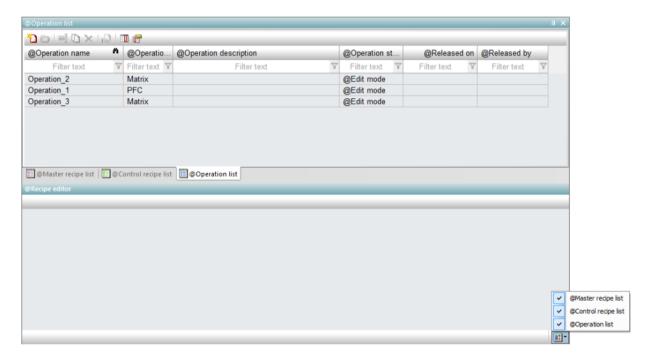In the recipe editor, three windows can be shown, positioned and docked as desired:

▶ List of master recipes (on page 120)

▶ List of control recipes (on page 187)

▶ List of the operations (on page 175)

For details on the selection and positioning, see the Selection and positioning (on page 107) chapter.

The settings are saved individually for each computer and user.



**SHOW/HIDE LIST**

The list of dockable windows can be displayed or hidden by means of settings in the zenon Editor.
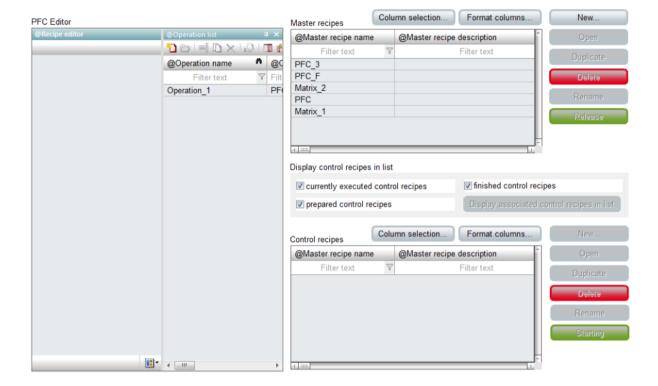
To show/hide the list:

1. In the zenon Editor, highlight the `Recipe editor` comment area in the `Batch Control` screen

2. go to property group `Representation`\Display editor control elements

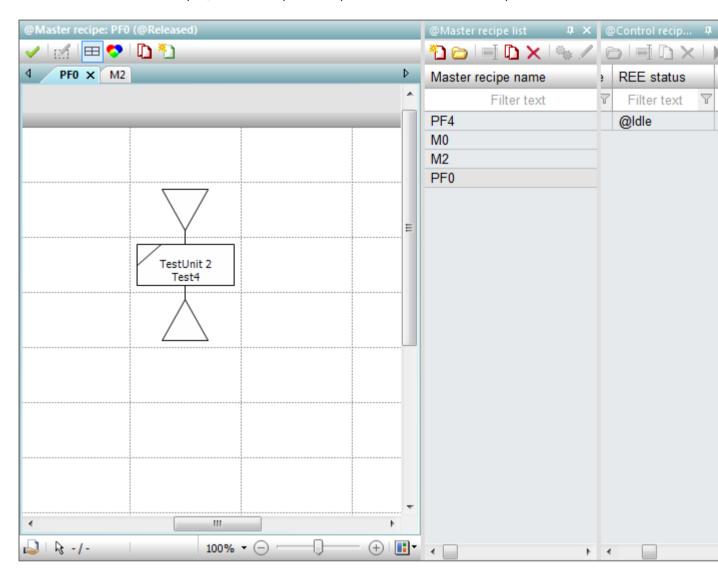3. Activate or deactivate the Dockable windows checkbox

## EXAMPLES

Additional list of the operations shown in the Recipe Editor:

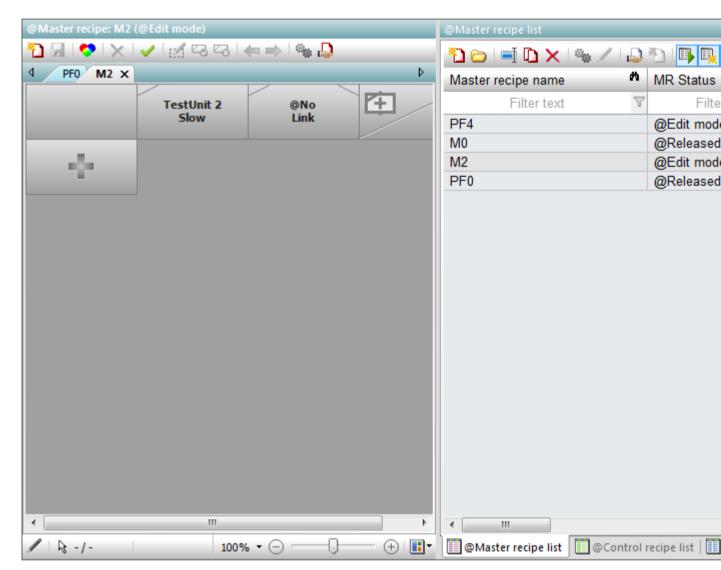Additional list of master recipes, control recipes and operations shown in the Recipe Editor:

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



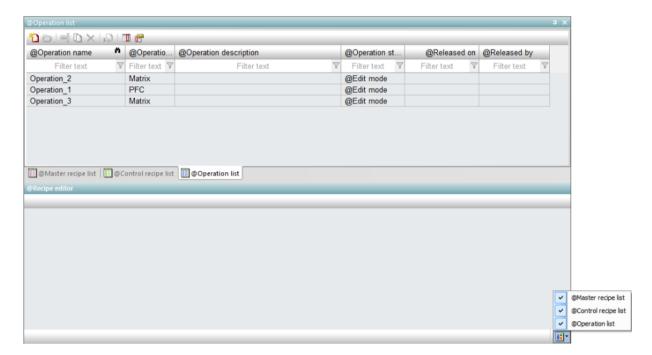**Selection and positioning**

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 means of activation/deactivation of the checkboxes.



## 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.

## 2.7.2    Commands and actions

In the Runtime the following commands and actions are available:

▶   Commands effect the recipe process.

▶   Actions make it possible to edit recipes.

**COMMANDS**

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.

| Command | Description |
|---|---|
| `Start recipe` | Starts the recipe process. |
| `Recipe pausing` | Pauses the recipe process. |
| `Recipe resuming` | Resumes a held or paused recipe. |
| `Recipe holding` | Holds the recipe process. |
| `Restart recipe` | Restarts the recipe. |
| `Recipe stopping` | Stops the recipe. |
| `Recipe abborting` | Aborts the recipe. |
| `Phase pausing` | Pauses the phase. |
| `Phase resuming` | Resumes the phase. |
| `Phase holding` | Holds phase. |
| `Restart phase` | Restarts the phase. |
| `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

| 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:** This overview has been taken from the ANSI/ISA-S88 standard (illustration 18).

### ACTION ON `STOP` COMMAND

After a `stop` command, the **phases**, **transitions** and end **simultaneous sequence** 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.

**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. |
| Show/hide grid | Toggles the grid for the PFC editor on/off. |
| Change background color and grid color | Opens the dialog (on page 131) for changing the background color and grid color. |
| 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 121) 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 136) 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 134). It can be added to any allowed, free location via click. |
| Insert transition | Occupies the mouse cursor with a transition (on page 159). It can be added to any allowed, free location via click. |
| Insert begin simultaneous sequence | Occupies the mouse cursor with a begin simultaneous sequence (on page 162). It can be added to any allowed, free location via click. |
| Insert end simultaneous sequence | Occupies the mouse cursor with an end simultaneous sequence (on page 162). It can be added to any allowed, free location via click. |
| Insert begin sequence selection | Occupies the mouse cursor with a begin sequence selection (on page 159). It can be added to any allowed, free location via click. |

| `Insert end sequence selection` | Occupies the mouse cursor with an end sequence selection (on page 159). It can be added to any allowed, free location via click. |
|---|---|
| `Insert unit allocation` | Occupies the mouse cursor with a unit allocation (on page 131). It can be added to any allowed, free location via click. |
| `Insert jump target` | Occupies the mouse cursor with a jump target (on page 164). It can be added to any allowed, free location via click. |
| `Switch recipe to test mode` | Switches recipe to the test mode (on page 172). |
| `Release recipe` | Releases (on page 174) the recipe. With this a control recipe can be created. |

## 2.7.3    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 target element. Here at least the output connection point and one input connection point must be connected.

▶ 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 must 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 an **begin sequence selection** must be a transition.

▶ The individual branches which start at **begin sequence selection** must all end in an **end sequence selection** never in an **end simultaneous sequence**. Outside of a simultaneous sequence a branch can also end in a jump target element. Any element can be placed between begin and end of a branch even simultaneous sequences as long as they are closed before the **end sequence selection** element.

▶ It is not necessary to have an **end sequence selection** for each **begin sequence selection**. You can for example have two **begin sequence selection** elements ending in one **end sequence selection** or the other way round.

▶ It is not necessary to have an **end sequence selection** for a **begin sequence selection**. It can simply end in a line. If for example you have a **begin sequence selection** element with two paths and one of the paths ends in a jump target, it does not make sense to have an **end sequence selection**.

▶ **End sequence selection** elements can be used in the same way as jump target elements. Limitation: The element must not be followed by a transition.

## SIMULTANEOUS SEQUENCES

▶ Each simultaneous sequence must contain at least one phase.

▶ The first element after a **Begin simultaneous sequence** must no be a transition.

▶ The individual branches that start at a **begin simultaneous sequence** must all end in **end simultaneous sequence**, but must never end in a **end sequence selection**. You may use any elements between **begin simultaneous sequence** and **end simultaneous sequence** even branches as long as they are closed before the **end simultaneous sequence**.

▶ Not all branches which were started in a `begin simultaneous sequence` must end in an `end simultaneous sequence`. It is enough when all branches converge over an `end parallel objects`. Equally branches from different `begin simultaneous sequence` objects may converge in a single `end simultaneous sequence`.

▶ Simultaneous sequences can be embedded in simultaneous sequence. For this, the following applies:

- each embedded simultaneous sequence must unite to the superordinate simultaneous sequence

- embraced interlacing is not supported

## 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 175) an error message is displayed.

## JUMP TARGETS

▶ Jump targets only exist in PFC recipes.

▶ Jump targets are used to

- jump between branches,

- jump out of branches

- engineer loops

For this, the following applies: A path which ends in a jump target must have started with a `begin sequence selection`. Otherwise the end is not reached.

▶ Jump targets consist of tree inputs and one output. At least one input and the output must be connected. At this it makes no difference which input connection point is connected.

▶ Jump targets may be interlaced consecutively.

▶ Jumps are prohibited:

- between simultaneous sequences

- to jump out of a simultaneous sequence

- to jump in a simultaneous sequence.

**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.

▶ Variables which are used in the tags of the reaction as source are registered at the driver for reading. If a value is needed at 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.

▶ If as an reaction to a status change a command was triggered which triggers an immediate status change (command to REE), the command of the second reaction is processed directly after processing the first. Commands processed.

**EXECUTION PERIOD**

▶ Property `Minimum execution duration` defines how long it is waited after writing the command tag independent of the check of the phase done condition.

▶ During the execution the minimum execution duration is checked. If the value is not `0`, an event is triggered after it has expired.

Read more about execution duration in chapter Execution duration (on page 39).

## 2.7.4    Status line

The status line is automatically adapted to each recipe that contains the focus and initialized with its data. The 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 status of the REE (on page 212) during the execution

▶ Mouse cursor position and changes such as moving the connection line

▶ Deleting, adding or selecting of elements

▶ 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.

## 2.7.5    Recipe types and recipe states

Basically we distinguish between the following recipe types:

▶ PFC recipe for free, complex processes. They are created with the PFC editor (on page 125).

▶ Matrix recipe for simple sequential/parallel processes. They are created with the matrix editor (on page 165).

For both recipe types its true that they are available in the following characteristics:

▶ Master recipes:
Form the basis for the control recipes. The created in status `Editable`. If they are in status `Released`, they can no longer be modified.

▶ Control recipes:
Are recipes which can be executed once. Each control recipe is based on a master recipe. Control

recipes can be modified concerning the process. The process is defined by the master recipe. Changeable are only command tags for which option `Changeable in control recipe` was activated.

### 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 172): In this status the recipe behaves similar as a control recipe. As them the process 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 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 taken over after a reload or after a Runtime restart.

- `Released` (on page 174): In this status no changes to the recipe are possible
  **Note:** Also changes to the phases or to the reactions in the Editor are not transferred to a master recipe. The status at the release are frozen.
  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 175) is done automatically during the release and 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.

▶ `running`: The control recipe was started and is processed. It remains in this status until it is `Completed`, `Stopped` or `Canceled`. Via the REE Status (on page 212) you can learn the exact state of the recipe.
In this status command parameters, for which option `Changeable in control recipe` 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.
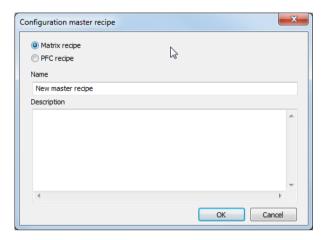
## 2.7.6    Master recipes

Master recipe are the basis of control recipes. With the help of master recipes you define and test the recipe process.  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. Depending on the license only one of them or both can be used.

## Create master recipe

A click on button `Create master recipe` opens the dialog `Configuration master recipe`.

| Parameter | Description |
|---|---|
| Matrix recipe (on page 165) | Activate this radio button if you want to create a matrix recipe (on page 165).<br><br>Note: Only possible if the corresponding license is available. |
| PFC recipe (on page 124) | Activate this radio button if you want to create a PFC recipe (on page 124).<br><br>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 (`*`).<br><br>Maximum length: 256 characters.<br><br>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.<br><br>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.<br><br>The recipe names can be changed later but only as long as the recipe is in status Editable. |
| Description | Optional description for the recipe which should be created.<br><br>The description can be changed later but only as long as the recipe is in status Editable. To change the description select the symbol **Rename master recipe**. |
| OK | Applies all settings and created a new recipe. |
| Cancel | Closes the dialog without creating a recipe. |

## Tool bar 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. |
| Rename master recipe | Opens dialog to rename 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. |
| 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. |
| 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 that belong to approved control recipes. |
| Column selection | Opens the dialog for selecting the columns which should be displayed. |
| Column format | Opens the dialog for configuring the column formats. |

## CONTEXT MENU
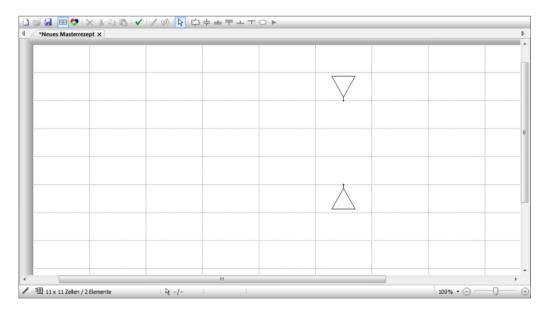
| Command | Description |
|---|---|
| New master recipe | Opens the dialog for creating a new master recipe. |
| Open in Recipe Editor | Opens the selected recipe in the recipe editor. |
| Rename | Opens dialog to rename the selected recipe. |

| Duplicate | Creates a copy of the selected recipe and opens the dialog to rename the duplicate. |
|---|---|
| Delete | Deletes selected recipes. |
| 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. |
| New control recipe | Opens the dialog for creating a new control recipe. |
| Display associated control recipes in list | Shows all control recipes that belong to selected approved control recipes. |

**PFC recipe**

If you selected PFC recipe in dialog `Configuration master recipe` and exited the dialog with OK, the newly created recipe opens on a new tab in the PFC editor (on page 125).
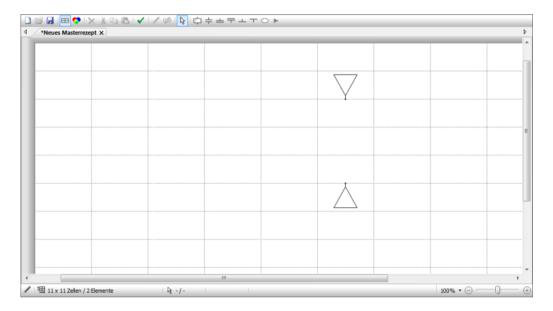


As each recipe needs a begin and an end element, these two elements already exist in the recipe and cannot be deleted from it.

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

**PFC editor**

In the PFC editor you can create your recipes graphically.



**TECHNICAL DETAILS**

- ▶ Sheet size:
  - • Default: 10 x 10 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 via mouse wheel: up and down or if you press and hold `Shift` left and right.

- ▶ Zooming: `Ctrl + mouse wheel`

- ▶ Selecting elements: `left mouse click`

- ▶ Multiple selection: `Ctrl + 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.

## Tool bar and context menu PFC recipe

### TOOL BAR 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. |
| `Show/hide grid` | Shows/hides the grid. |
| `Change background color and grid color` | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| `Delete` | Deletes the selected elements from the diagram.<br><br>Multiple selection via `Ctrl+mouse click`. |
| `Check recipe for errors` | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed.<br><br>The 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.<br><br>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 simultaneous sequence` | Adds a begin simultaneous sequence. |
| `Insert end simultaneous sequence` | Adds an end simultaneous sequence. |
| `Insert begin sequence selection` | Adds a begin sequence selection. |
| `Insert end sequence selection` | Adds an end sequence selection. |
| `Insert unit allocation` | Adds a unit allocation. |
| `Insert jump target` | Adds a jump target. |
| `Insert operation` | Adds an operation (on page 175). |

| 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.
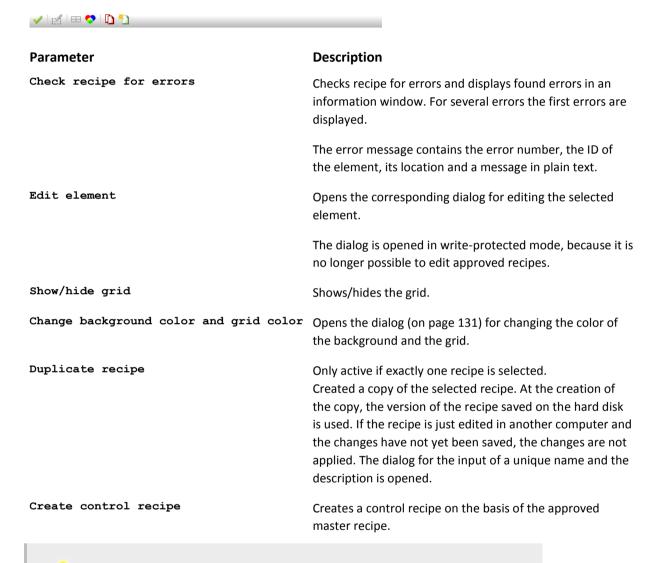
## TOOL BAR PFC RECIPE TEST MODE

| 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 abborting` | 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. |
| `Escape phase` | Starts process to exit from the phase. |
| `Check recipe for errors` | Starts recipe validation (on page 175). |
| `Edit element` | Opens the corresponding dialog for editing the selected element. |
| `Show/hide grid` | Shows/hides the grid. |
| `Change background color and grid color` | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| `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. |
| --- | --- |

## PFC RECIPE TOOL BAR: APPROVED

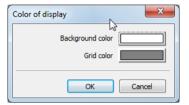| Parameter | Description |
| --- | --- |
| Check recipe for errors | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed. |
| | The error message contains the error number, the ID of the element, its location and a message in 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. |
| Show/hide grid | Shows/hides the grid. |
| Change background color and grid color | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| Duplicate 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. |
| Create control recipe | Creates a control recipe on the basis of the approved master recipe. |

### Info

*The functions of the individual symbols can also be configured using buttons and thus be made touch-operable. Tool bars can therefore also be hidden (on page 102).*

## Settings diagram

A click on the symbol for the color configuration in the tool bar opens the dialog for configuring the color for the background and grid.



| 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. |
| Grid color | Defines the line color of the grid. Click on the color in order to open the palette for selecting a color. |
| OK | Applies all settings and closes the dialog. |
| Cancel | Discards all changes and closes the dialog. |

## Begin element

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

## Allocate and configure unit

To allocate a unit:

1. in the tool bar select the symbol for `Insert unit allocation`

2. move the mouse to the desired position

3. place the unit

4. the unit is added

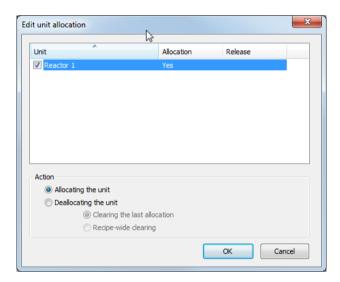To configure the unit allocation:

1. double click the unit

2. The dialog for configuration is opened

## CONFIGURATION

In a configuration dialog you can select the units which exist in the batch recipe for the Runtime. It is distinguished between allocation and release.  Releases can be set selectively and globally. Units which were selected before and no longer exist in the recipe, are displayed with an appropriate note.

| Parameter | Description |
|---|---|
| `List units` | Displays existing units and their assigned actions. |
| `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: <br> ▸ Release of the last allocation: The last allocation is released. <br> ▸ Recipe-spanning release: All allocations in the recipe are released. |
| `OK` | Applies settings and closes the dialog. |
| `Cancel` | Discards changes and closes dialog. |

**ALLOCATE AND RELEASE UNITS**

The allocation of a unit by element unit allocation or by a phase is only possible if the unit is not yet allocated or only in the same recipe. Allocation and release of units is always done in a cycle. It is always waited until all units which should be allocated are released. Then all allocations and releases are down at the same time. Phases which are located in front of an element unit allocation remain active until the allocation was successful.

Unit allocations remain in place as long as the phase is active. When the recipe is finished and there are still allocations of elements `Unit allocation` active, they are released implicitly.

Note: 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.

**ACTIONS**

For each unit which is used in the recipe, you can define an action:

▸ no action

▸ allocate

▸ release

**TOOLTIP**

In the tooltip of element `Unit allocation` all units are displayed which are marked for allocation or release. During the execution the Execution duration (on page 39) 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.

**Add and configure phase**

To add a phase:

1. in the tool bar select the symbol for `Insert phase`

2. move the mouse to the desired position

3. locate the phase

4. the dialog for selecting a phase (on page 136) is opened

5. select the desired phase

To configure a phase:
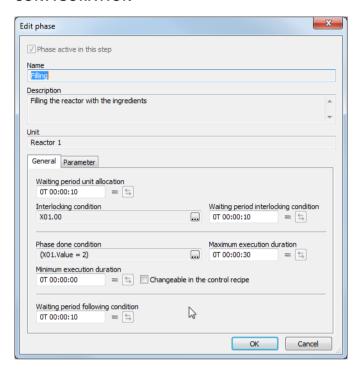
1. double click the phase

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



| Parameter | Description |
|---|---|
| `Phase active in this step` | `Active:` Phase is active in this step |
| `Name` | Name of the phase.<br>Only display. |
| `Description` | Comment about the phase.<br>Only display. |
| `Unit` | Unit on which the phase is carried out.<br>Only display. |
| `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 137): Display and configuration of the settings for the phase

▶ Tag (on page 143): 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**
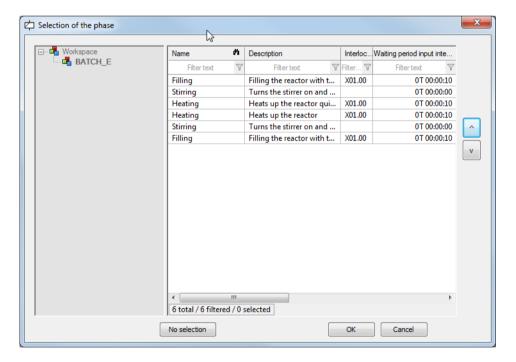
A phase can be edited:

▶ 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.

## Selection phase

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

| Parameter | Description |
|---|---|
| `Project tree` | Displays the current project from which the phases can be selected. |
| `List field phases` | In the list all phases engineered in the Editor are displayed.<br><br>This list can be filtered. The filtering is case-sensitive. Placeholders `*` and `?` can be used. |
| `Arrow keys` | Move selected phases 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. |

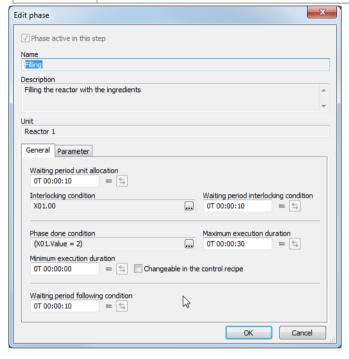All settings of the dialog are saved user-specifically when the dialog is closed.

**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:

| Parameter | Description |
|---|---|
| = | Value in the dialog and the value in the Editor match. |
| <> | Value in the dialog and the value in the Editor do not match. |
| ↹ | 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. |



For information about the base data on these tabs see chapter Add and configure phase (on page 134).

| Parameter | 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 status `Editable`.<br><br>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.<br><br>Maximum: `9999d 23:59:59`<br>Default: `0d 00:00:00`<br><br>Note: A unit can only be allocated by a single recipe with status `In execution` at a time.<br><br>Value is predefined in the Editor and can be changed here as long as the recipe is not released. |
| `Interlocking condition` | Defines conditions for input interlocking. Click on button `...` or an entry to open the formula editor (on page 238) for defining the condition.<br>If an input interlocking is configured, the phase is only executed in the Runtime when the condition for the input interlocking is fulfilled. The formula can consist of one or more command tags and return 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 there.<br>The waiting period for the input interlocking is configured with the help of property `Waiting period input interlocking`. |
| `Waiting period input interlocking` | Period in days, hours, minutes and seconds in which the condition defined in property `Interlocking condition` must return value `TRUE`. The waiting period begins with the beginning of the check of the input interlocking. If the condition is not fulfilled within the waiting period, the event `Waiting period input interlocking exceeded` is triggered and the waiting is continued. If no reaction was defined for the event which forces another behavior, it is waited until the condition is fulfilled.<br><br>If `0d 00:00:00` is defined as waiting period, the event is not triggered. |

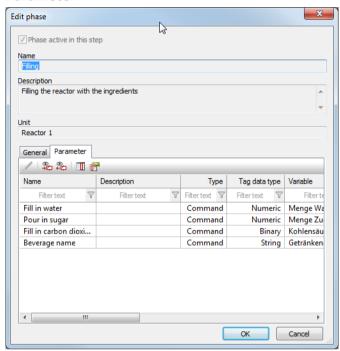| | |
|---|---|
| | ▸ Minimum value: `0d 00:00:00` |
| | ▸ Maximum value: `9999d 23:59:59` |
| | ▸ Default: `0d 00:00:00` |
| | Value is predefined in the Editor and can be changed here as long as the recipe is not released. |

| Phase done condition | Defines the condition for phase done. Click on button ... or an entry to open the formula editor (on page 238) for defining the condition.<br><br>The condition can be displayed in the Runtime but cannot be changed there. The period which is waited for the fulfillment of the condition is defined via property `Maximum execution duration.` |
|---|---|
| Maximum execution duration | 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 when writing of the command parameter. 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 for the event which forces another behavior, it is waited until the condition is fulfilled.<br><br>**Note:** Time keeps running when the recipe/the phase is paused.<br><br>If `0d 00:00:00` is defined as waiting period, the event is not triggered.<br><br>▸ Minimum value: `0d 00:00:00`<br><br>▸ Maximum value: `9999d 23:59:59`<br><br>▸ Default: `0d 00:00:00`<br><br>Value is predefined in the Editor and can be changed here as long as the recipe is not released.<br><br>Makes only sense if property `Phase done condition` was configured.<br><br>Read more about execution duration in chapter Execution duration (on page 39). |
| Minimum execution duration | Minimum execution duration of the phase.<br><br>▸ `>0`: Period which is at least waited after the writing of the command tag regardless of whether the phase done condition is fulfilled.<br><br>▸ `0`: Execution duration is not checked |

| | |
|---|---|
| | Minimum execution duration can exceed `Maximum execution duration`. Value is predefined in the Editor and can be changed here as long as the recipe is not released.<br><br>Changeable in the control recipe if option `Changeable in the control recipe` was activated in the master recipe.<br><br>Read more about execution duration in chapter Execution duration (on page 39). |
| `Changeable in the control recipe` | `Active:` Property can be changed in the control recipe. |
| `Waiting period following condition` | 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 for the event which forces another behavior, it is waited until the condition is fulfilled.<br><br>If `0d 00:00:00` is defined as waiting period, the event is not triggered.<br><br>‣ Minimum value: `0d 00:00:00`<br><br>‣ Maximum value: `9999d 23:59:59`<br><br>‣ Default: `0d 00:00:00`<br><br>**Note:** If the following condition exists depends on the recipe structure. Therefore the configuration of a waiting period should not be done 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.<br><br>For more information see chapters **Times** (on page 24) and **Following conditions** (on page 225). |
| **OK** | Applies all changes on all tabs and closes the dialog. |
| **Cancel** | Discards all changes on all tabs and closes the dialog. |

## Parameter



For information about the base data on these tabs see chapter Add and configure phase (on page 134).

| Parameter | Description |
|---|---|
| List tag | Displays the tag configured in the Editor. Tags can be filtered and sorted according to columns.<br><br>Click on symbol `Edit tag`, double click the tag, menu item in the context menu or press `Return` to open the dialog (on page 144) for editing a tag. |
| `OK` | Applies all changes on all tabs and closes the dialog. |
| `Cancel` | Discards all changes on all tabs and closes the dialog. |

**TOOL BAR**

| Symbol | Meaning |
|---|---|
| `Edit tag` | Opens the dialog (on page 144) for editing the tag. |
| `Display all command tags` | Toggles between the display of the changeable tags and all tags. |
| `Display return tag` | In addition to the command tags also displays the return tags or hides them. |
| `Column selection` | Opens the dialog (on page 155) for selecting the columns which should be displayed. |
| `Column format` | Opens the dialog (on page 156) for formating the columns. |

**Edit tag**

To edit a tag in the Runtime:

1. in dialog Edit phase select tab `Tag`

2. highlight the desired tag

3. open the dialog for editing the tags via a click on symbol `Edit tag`, the context menu, a double click on the tag or press `Return`
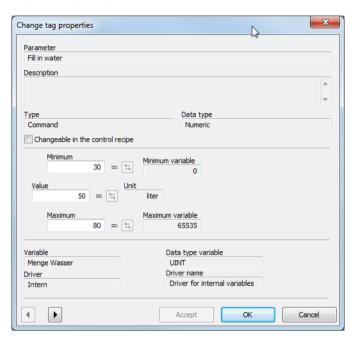
4. the dialog for editing is opened

For each data type an own dialog is opened:

▶   Numerical

▶   Binary

▶   String

▶   Time period

The properties are normally configured in the Editor and only displayed in the Runtime. Exceptions are values of the data type. They can be adapted if property `Changeable in recipe` was activated in the Editor.
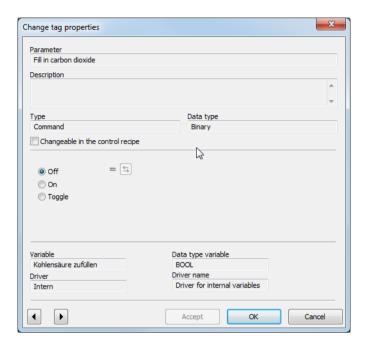
## NUMERIC TAG

| Change tag properties | | |
|---|---|---|
| **Parameter** | | |
| Fill in water | | |
| **Description** | | |
| | | |
| **Type** | **Data type** | |
| Command | Numeric | |
| ☐ Changeable in the control recipe | | |
| **Minimum** | **Minimum variable** | |
| 30 = | 0 | |
| **Value** | **Unit** | |
| 50 = | liter | |
| **Maximum** | **Maximum variable** | |
| 80 = | 65535 | |
| **Variable** | **Data type variable** | |
| Menge Wasser | UINT | |
| **Driver** | **Driver name** | |
| Intern | Driver for internal variables | |
| ◄ ► | Accept | OK Cancel |

| Parameter | Meaning |
| --- | --- |
| `Parameter` | Name of the tag. Only display. |
| `Description` | Free description of the tag. Only display. |
| `Type` | Type of the tag: `Command` or `Return`. Only display. |
| `Data type` | Data type of the tag. Only display. |
| `Changeable in the control recipe` | `Active`: Value of the property can be changed in the control recipe.<br><br>Only available if:<br><br>▸ the tag is of type `command`<br><br>▸ it was configured in the Editor that the tag can be changed in the Batch recipe (property: ). `Changeable in recipe`). |
| `Minimum` | Minimum value for the tag.<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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. |
| `Minimum variable` | Allowed minimum value of the variable. |
| `Value` | Name of the tag.<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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. |
| `Unit` | Unit of the value. |
| `Maximum` | Maximum value for the tag. Only display.<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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. |
| Maximum variable | Allowed minimum value of the variable. Only display. |
| Variable | Variable which is linked to the tag. Only display. |
| Data type variable | Data type of the variable. Only display. |
| Driver | Driver of the variable. Only display. |
| Driver name | Description of the driver of the variable. Only display. |
| 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. |
| **Accept** | 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**
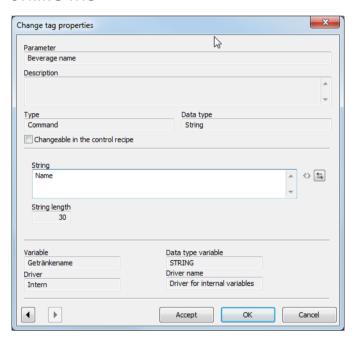
| Parameter | Meaning |
|---|---|
| `Parameter` | Name of the tag. Only display. |
| `Description` | Free description of the tag. Only display. |
| `Type` | Type of the tag: `Command` or `Return`. Only display. |
| `Data type` | Data type of the tag. Only display. |
| `Changeable in the control recipe` | `Active`: Value of the property can be changed in the control recipe.<br><br>Only available if:<br><br>▸ the tag is of type `command`<br><br>▸ it was configured in the Editor that the tag can be changed in the Batch recipe (property: ). `Changeable in recipe`). |
| `Off` | Status: Off.<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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. |
| `On` | Status: On. |
| `Toggle` | Toggles between the states. |
| `Variable` | Variable which is linked to the tag. Only display. |
| `Data type variable` | Data type of the variable. Only display. |
| `Driver` | Driver of the variable. Only display. |
| `Driver name` | Description of the driver of the variable. Only display. |
| Arrow keys | Navigating through the tags.<br><br>They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping.<br><br>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 |

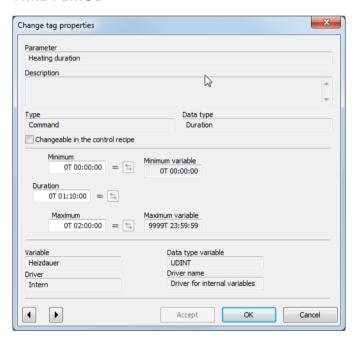| | before advancing. |
|---|---|
| **Accept** | Applies all changes if the check of the changes was successful. The dialog remains open for further editing. |
| **OK** | Applies all changes and closes the dialog if the check of the changes was successful. |
| **Cancel** | Discards all changes which have not been taken over yet and closes the dialog. |

## STRING TAG

| Parameter | Meaning |
|---|---|
| Parameter | Name of the tag. Only display. |
| Description | Free description of the tag. Only display. |
| Type | Type of the tag: `Command` or `Return`. Only display. |
| Data type | Data type of the tag. Only display. |
| Changeable in the control recipe | `Active`: Value of the property can be changed in the control recipe.<br><br>Only available if:<br><br>▸ the tag is of type `command`<br><br>▸ it was configured in the Editor that the tag can be changed in the Batch recipe (property: ). `Changeable in recipe`). |
| String | Alphanumeric character string.<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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.<br><br>Possible length is limited by the `String length` engineered in the variable. |
| String length | Defines possible length of the string. Only display. |
| Variable | Variable which is linked to the tag. Only display. |
| Data type variable | Data type of the variable. Only display. |
| Driver | Driver of the variable. Only display. |
| Driver name | Description of the driver of the variable. Only display. |
| Arrow keys | Navigating through the tags.<br><br>They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping.<br><br>If changes were done, there is a prompt before you can change to another tag whether the changes should be applied or |

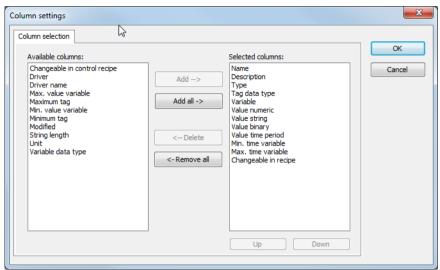| | discarded. If changes should be applied, the input is checked before advancing. |
|---|---|
| `Accept` | Applies all changes if the check of the changes was successful. The dialog remains open for further editing. |
| `OK` | Applies all changes and closes the dialog if the check of the changes was successful. |
| `Cancel` | Discards all changes which have not been taken over yet and closes the dialog. |

## TIME PERIOD

| Parameter | Meaning |
|---|---|
| `Parameter` | Name of the tag. Only display. |
| `Description` | Free description of the tag. Only display. |
| `Type` | Type of the tag: `Command` or `Return`. Only display. |
| `Data type` | Data type of the tag. Only display. |
| `Changeable in the control recipe` | `Active`: Value of the property can be changed in the control recipe.<br><br>Only available if:<br><br>▸ the tag is of type `command`<br><br>▸ it was configured in the Editor that the tag can be changed in the Batch recipe (property: ). `Changeable in recipe`). |
| `Minimum` | Minimum value for the tag in format:<br>**`d hh:mm:ss`**<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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. |
| `Minimum variable` | Allowed minimum value of the variable. |
| `Duration` | Value of the parameter in the format:<br>**`T hh:mm:ss.`**<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. If the value is changed, it is marked with the symbol (on page 137) 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. |
| `Unit` | Unit of the value. |
| `Maximum` | Maximum value for the tag in format:<br>**`d hh:mm:ss`**.<br><br>Default from the Editor can be changed when property `Changeable in recipe` was activated in the Editor. |

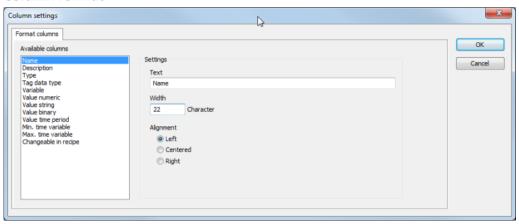| | If the value is changed, it is marked with the symbol (on page 137) on the right side of the value. Changed values can be overwritten with the default from the Editor with the help of the button right next to it. |
|---|---|
| `Maximum variable` | Allowed minimum value of the variable. Only display. |
| `Variable` | Variable which is linked to the tag. Only display. |
| `Data type variable` | Data type of the variable. Only display. |
| `Driver` | Driver of the variable. Only display. |
| `Driver name` | Description of the driver of the variable. Only display. |
| Arrow keys | Navigating through the tags.<br><br>They are displayed in the order of the list. At this only tags are displayed which are visible with the current filter and grouping.<br><br>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. |
| **Accept** | 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. |

## Column selection



| Button | Function |
|---|---|
| **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. |
| **Down** | Moves the selected entry downward. This function is only available for unique entries, multiple selection is not possible. |
| **OK** | Applies settings and closes dialog. |
| **Cancel** | Discards settings and closes the dialog. |
| **Help** | Opens online help. |

## Column format



| Parameter | Description |
|---|---|
| **Available columns** | List of columns available using **column selection**. The column selected here is configured using the settings in the **Parameters** section. |
| **Parameter** | Settings for selected column. |
| Text | Name for column title. The column title is online language switchable. For this you must enter character @ in front of the name. |
| Width | Width of the column in characters.<br>Calculation: Number time average character width of the selected font. |
| Alignment | Alignment.<br><br>Possible settings:<br><br>▸ Left-justified: Text is justified on the left edge of the column.<br><br>▸ Centered: Text is displayed centered in the column.<br><br>▸ Right-justified: Text is justified on the right edge of the column. |
| **OK** | Applies settings and closes the dialog. |
| **Cancel** | Discards settings and closes the dialog. |

**Insert operation**

To insert an operation:

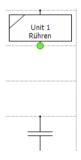1. in the tool bar select the symbol for **Insert operation**

2.  move the mouse to the desired position

3.  place the operation

4.  the dialog for selecting a template (on page 183) for the operation is opened

5.  select the desired operation

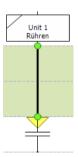6.  The operation is inserted

**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.

**USING TREND CURVES**

Lines:

- ▶ are dragged with the mouse

- ▶ can be moved (press and hold key `Ctrl`)
  At this all existing connections are separated and it is tried to reconnect the line if there are
  objects with connections points in the right direction at the target.
  If several lines are highlighted, the line, in whose cell the mouse cursor is, is moved.

- ▶ can be deleted by highlighting them an pressing `Del`

- ▶ 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

**Transition**

Transitions are used after phases in order to ensure a defined transition from one phase to another. Transitions display their internal status during the process and inform via a tool tip about status and process duration.

For details about transitions see section Engineering in the Editor (on page 13) in chapter Transitions (on page 22).

**Sequence selections**

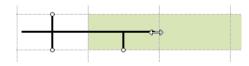Recipes can branch[1] and run in simultaneous sequences (on page 162).

Branches and simultaneous sequences consist of:

> ▶    single/double horizontal lines

> ▶    Connection pieces (consisting of connection line and connection point)

### CREATE A SEQUENCE SELECTION

To create a sequence selection:

1. select the symbol `Insert begin sequence selection`

2. put the sequence selection on the desired location

3. connect the input connection point with a output connection point of the preceding object

4. connect both output connection points with the desired following objects

5. close a sequence selection with object `Insert end sequence selection`



---

[1] 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 sequence selection can only be followed by a transitions.Procedure:The path is chosen for which the transition is TRUE first. Then it is waited until all transitions have a value.If several transitions are TRUE at the same time, always the leftmost path for which the transition is TRUE is selected.For begin an ...

**INSERT SIMULTANEOUS SEQUENCE**

To create a simultaneous sequence:

1.  select the symbol `Insert begin simultaneous sequence`

2.  put the sequence selection on the desired location

3.  connect the input connection point with a output connection point of the preceding object

4.  connect both output connection points with the desired following objects

5.  close a simultaneous sequence with object `Insert end simultaneous sequence`

**MODIFY AND MOVE**

Branches and simultaneous sequences can be moved and changed in size.

**MOVE**

To move an object:

1.  click on the object

2.  keep the mouse button pressed

3.  move the object to the desired position

**CHANGE SIZE**

In this way object `Begin/end sequence selection/simultaneous sequence` can be extended and shortened. To change their size:

1.  move the mouse cursor over the object until it turns into a double arrow

2.  press and hold the left mouse button and move the mouse in the desired direction:

    *   away from the object to extend it
    *   into the object to shorten it

3. at 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. at shortening all corresponding connection pieces are deleted
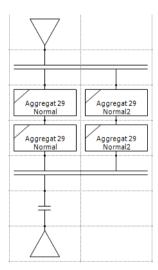
**Sequence sequence selection**

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 sequence selection` can only be followed by a transitions (on page 22).

Procedure:

▶ The path is chosen for which the transition is TRUE first.

▶ Then it is waited until all transitions have a value.

▶ If several transitions are TRUE at the same time, always the leftmost path for which the transition is TRUE is 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 sequence selection the objects are processed sequentially. Each sequence selection processes its objects independent of other branches.

### Parallel branch

At the parallel branch an execution path parts into several execution paths which are executed in parallel during the process.  For the activation of the different elements within a parallel branch you cannot define a certain order.

In the process the respective intermediate area of the **end parallel branch** is also colored. The color matches the coloring (on page 206) of the phase.

**Phase completed** is displayed as active as soon as the first previous element has been completed. This means that a phase is Finished or a transition is passed. Transitions are marked as completed as soon as they are passed. Phases wait at **Phase completed** until the **end parallel branch** is completed. Completed means that either the following phase is active or the following transition is inactive.

During the execution the status is color-coded.
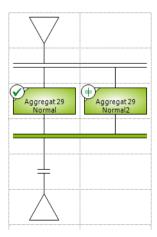
### Split up and combine sequence selection

For simultaneous sequences the sequence selection splits up a **Begin simultaneous sequence** and combines the single sequence selections at **End simultaneous sequence**. The paths of the parallel objects are independent branches. Only at **End simultaneous sequence** all branches are synchronized.

The possible branches are defined by the engineering. If the separate branches are allocated or released, arises in the Runtime. A branch is active as long as an object on it is active.

The object types Begin sequence selection, End sequence selection and Jump target do not allocate and release branches as these objects are processed in the same branch. Combining branches is not allowed.
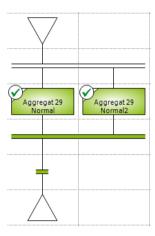
#### END SIMULTANEOUS SEQUENCES

An **End simultaneous sequence** 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.
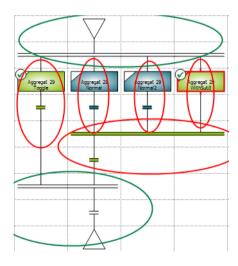
The left path is ready. Therefore `End simultaneous sequence` is already active. The transition after `End simultaneous sequence` is not yet active as the right branch is not yet completed.
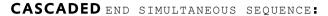
When the right branch is also completed:



The transition after `End simultaneous sequence` is activated. All objects which were active before are still active. Instead of the transition there could also be another `End simultaneous sequence`.
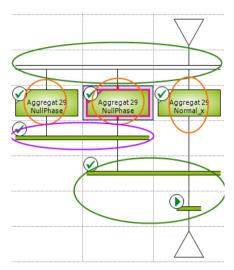
## A LITTLE MORE COMPLEX ALTERNATIVE:



The areas highlighted in green are a branch.

**CASCADED** END SIMULTANEOUS SEQUENCE:



The areas highlighted in green are a branch.

**Jump target**

Adds a jump target.

Jump targets make it possible to

- jump between branches
- 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 sequence selection`. Otherwise the end is not reached. Jump targets are not allowed for simultaneous sequences.

During the editing all connection points are visible. In the checking mode only the connection points which are connected are displayed.
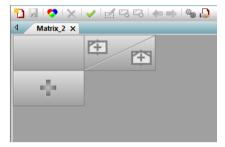
**End element**

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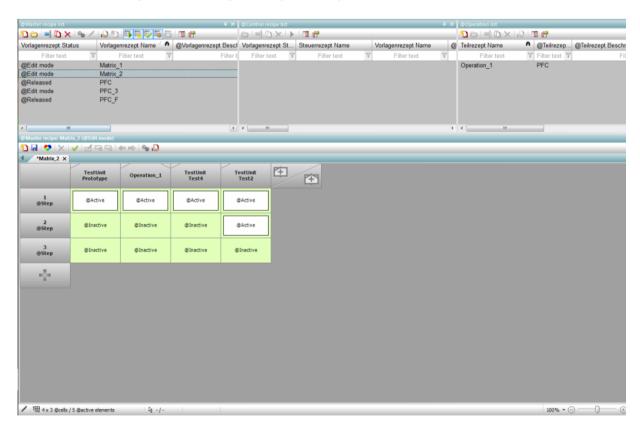
## Matrix recipe

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 165).



## Matrix editor

In the matrix editor you can create your recipes and operations in a matrix.



### TECHNICAL DETAILS

▶ Matrix:

- Columns contain phases (on page 170) and operations (on page 172):
  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.

▶ Insert column/row

Clicking on the plus sign inserts a new line or column with a phase or operation into the matrix.

▶ Delete column/row Press `Del` to delete the highlighted line 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 170) for configuring the phase.

- `Double click` on step: opens the dialog for labeling (on page 170) the step.

▶ Selection of lines/columns:

- `left mouse click` in border cell: selects a line/column.

- `Ctrl+mouse click` in border cell: selects additional lines/columns.

▶ Cells:

- `left mouse click`: selects empty cell.

- `Shift + click`: activates/deactivates phase.

- `Ctrl+click`: selects several cells.

- `Double click` on cell: opens the dialog for configuring (on page 170) the phase.

▶ Scroll bar: Is displayed if the document is larger than the frame.

▶ Scrolling via mouse wheel: up and down or if you press and hold `Shift` left and right.

**Tool bars matrix recipe**

**TOOL BARS MATRIX RECIPE: EDIT MODE**

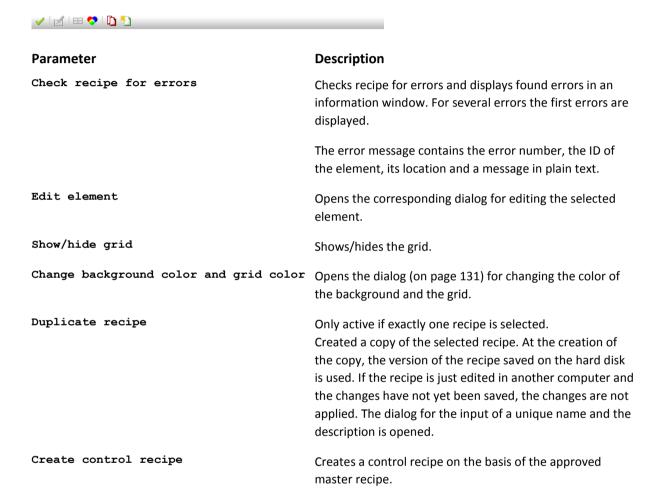| | |
|---|---|
| **New master recipe...** | Opens the dialog for creating a new master recipe. |
| **Save master recipe** | Saves the master recipe which is open for editing.<br><br>If another recipe is opened, the current recipe is saved automatically. |
| **Change background color and grid color** | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| **Delete** | Deletes the selected elements from the diagram.<br><br>Multiple selection via `Ctrl+mouse click`. |
| **Check recipe for errors** | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed.<br><br>The 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. |
| **Activate selected phase** | Activates the selected phase. |
| **Deactivate selected phase** | Deactivates the selected phase. |
| **Move selected column to the left or selected step up** | Moves the selected column to the left by one position or moves selected step up by one position. |
| **Move selected column to the right or selected step 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. |

## TOOL BARS 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 abborting` | 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. |
| `Escape phase` | Starts process to exit from the phase. |
| `Check recipe for errors` | Starts recipe validation (on page 175). |
| `Edit element` | Opens the corresponding dialog for editing the selected element. |
| `Show/hide grid` | Shows/hides the grid. |
| `Change background color and grid color` | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| `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.<br><br>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 TOOL BAR: APPROVED

| Parameter | Description |
|---|---|
| **Check recipe for errors** | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed. |
| | The 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. |
| **Show/hide grid** | Shows/hides the grid. |
| **Change background color and grid color** | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| **Duplicate 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. |
| **Create control recipe** | Creates a control recipe on the basis of the approved master recipe. |

### 💡 Info

*The functions of the individual symbols can also be configured using buttons and thus be made touch-operable. Tool bars can therefore also be hidden (on page 102).*

## Name steps

Steps in the matrix editor can be named individually. Double click on the cell to open the dialog for entering an individual name.



The name can have up to 100 characters and is language switchable is preceded by a @.
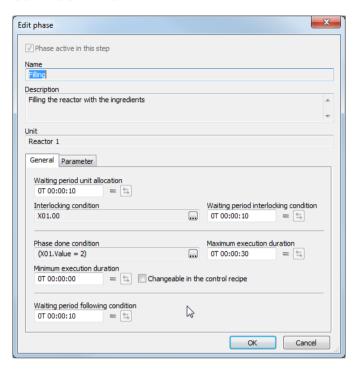
## Add and configure phase

To add a phase:

1. Click on the plus sign above the diagonal in the last column of the matrix

2. the dialog for selecting a phase is opened

3. select the desired phase

To configure a phase:

1. double click the matrix filed with the desired phase

2. The dialog for configuration is opened

## CONFIGURATION



| Parameter | Description |
|---|---|
| **Phase active in this step** | `Active:` Phase is active in this step |
| **Name** | Name of the phase.<br>Only display. |
| **Description** | Comment about the phase.<br>Only display. |
| **Unit** | Unit on which the phase is carried out.<br>Only display. |
| **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 125)/chapter:

▶ General (on page 137): Display and configuration of the settings for the phase

▶ Tag (on page 143): 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.

**Insert operation**

To insert an operation (on page 175):

1. Click on the plus sign below the diagonal in the last column of the matrix

2. the dialog for selecting a template (on page 183) for operation is opened

3. select the desired templates

4. A new operation is inserted

**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.

| Parameters | 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 abborting` | 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. |
| `Check recipe for errors` | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed.<br><br>The 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. |
| `Show/hide grid` | Shows/hides the grid. |
| `Change background color and grid color` | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| `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. |

**Release master recipe**

You can release a master recipe by selecting it and clicking button `Release master recipe`. Several recipes can also be selected and approved together. Approval must be confirmed by means of a dialog.

As soon as a recipe has been approved, the dialog to allocate the unit can no longer be opened. All information about the unit is displayed in the tooltip. Dialogs for transitions and phases can continue to be opened as write-protected.

> **Info**
>
> Only master recipes without errors can be released. A released master recipe can no longer be edited.
>
> For each recipe you can create a copy of the released master recipe by clicking on button `Duplicate recipe`. This copy can then be edited.
>
> Recipes can only be approved if all operations contained therein have also been approved.

**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

## 2.7.7 Validate recipe

Recipes can be checked for error during the engineering. To validate a recipe click on the corresponding symbol in the tool bar of the recipe editor in the Runtime (green tick - Check recipe for errors). With this the recipe is checked for functionality according to internal rules. The following is especially checked:

▶ Syntax (all lines connected, processable from begin to end, etc.)

▶ Variables

▶ Data types

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.

Rules which must be adhered to during the engineering can be found in chapter Engineering rules for recipes (on page 114).

## 2.7.8 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 183) and as an instance in existing matrix recipes or PFC recipes.
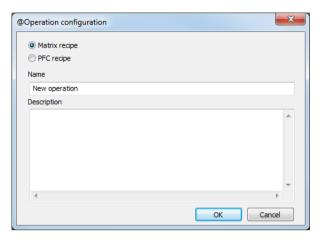
**CREATING AN OPERATION**

To create a new operation:

1. In the Recipe Editor (on page 101) switch the list of operations to visible (activate checkbox)

2. Select, in the tool bar or context menu (on page 179) of the list, the **New operation** command

3. The dialog for configuring an operation is opened

| Parameter | Description |
|---|---|
| Matrix recipe | Activate this radio button if you would like to create an operation on the basis of a matrix recipe (on page 165). |
| | Note: Only possible if the corresponding license is available. |
| PFC recipe | Activate this radio button if you would like to create an operation on the basis 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 status 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 status editable. To change the description select the symbol **Rename operation**. |
| **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 124) and Matrix recipes (on page 165).

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 tool bar

## 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 editor using the `Insert operation` (on page 156) symbol

▶ In the matrix recipe using the `Add operation` (on page 172) 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 is saved, the attendant master recipe is also saved automatically.

## TOOLTIP

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. Each object is always only used for one error.

▶ Exit from phase:

Number of objects that are being exited from or have been exited from

▶ Overall duration with time when it was deactivated and deactivated

▶ Information about the status of the internal objects: Number of objects with the respective status (except `idle`)

**Toolbar and context menu operations**

**TOOL BAR LIST**

| Parameter | Description |
|---|---|
| New operation | Creates a new operation. |
| Open operation in Editor | Opens the selected operation in the recipe editor. |
| Rename the operation | Opens the dialog to name a recipe. |
| Duplicate operation | Duplicates the selected operation. |
| Delete operation | Deletes the selected operation. |
| 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 62). |
| Column format | Opens the dialog to format a column (on page 62). |

## CONTEXT MENU

| Parameter | Description |
|---|---|
| New operation | Creates a new operation. |
| Rename | Opens the dialog to name a recipe. |
| Duplicate | Duplicates the selected operation. |
| Delete | Deletes the selected operation. |
| Open in Editor | Opens the selected operation in the recipe editor. |
| Release | Checks the selected operation and approves it if no errors were found. |

## TOOL BAR 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. |
| `Show/hide grid` | Shows/hides the grid. |
| `Change background color and grid color` | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| `Delete` | Deletes the selected elements from the diagram. Multiple selection via `Ctrl+mouse click`. |
| `Check operation for errors` | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed. The 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. |
| `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+double click` |
| `Edit mode` | Toggles between insert mode and edit mode. |
| `Insert phase` | Adds a phase. |
| `Insert transition` | Adds a transition. |
| `Insert begin simultaneous sequence` | Adds a begin simultaneous sequence. |
| `Insert end simultaneous sequence` | Adds an end simultaneous sequence. |
| `Insert begin sequence selection` | Adds a begin sequence selection. |
| `Insert end sequence selection` | Adds an end sequence selection. |
| `Insert unit allocation` | Adds a unit allocation. |
| `Insert jump target` | Adds a jump target. |
| `Release operation` | Releases the operation. 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.
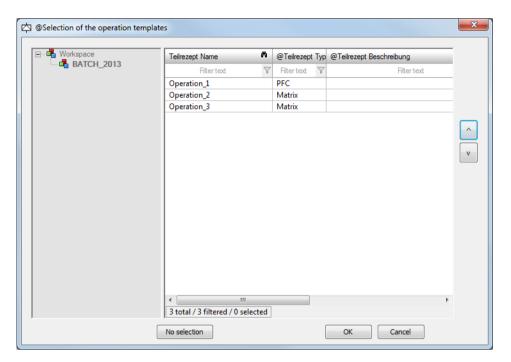
## PFC RECIPE TOOL BAR: APPROVED

| Parameter | Description |
|---|---|
| **Check operation template for errors** | Checks recipe for errors and displays found errors in an information window. For several errors the first errors are displayed.<br><br>The 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.<br><br>The dialog is opened in write-protected mode, because it is no longer possible to edit approved recipes. |
| **Show/hide grid** | Shows/hides the grid. |
| **Change background color and grid color** | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| **Duplicate operation** | Only active if exactly one recipe is selected.<br>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. |

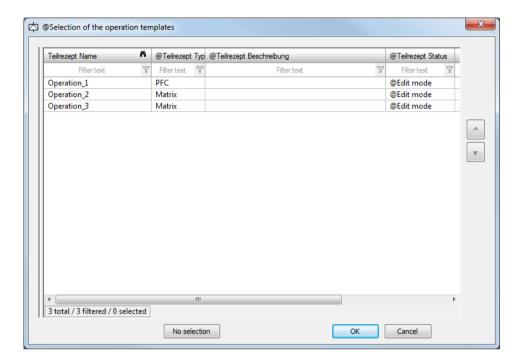## Selection of the template for an operation

Operations can be inserted into recipes as an instance. When inserting an operation into a matrix recipe (on page 172) or a PFC recipe (on page 156), the dialog to select a template for an operation is opened. This contains all previously-configured (on page 175) operations:

| 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.<br><br>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).<br><br>Entries can:<br><br>▸ be sorted by clicking on the column title; another click inverts the sorting<br><br>▸ be filtered into filter rows by alphanumeric entries (wildcards can be used) |
| `Arrow keys` | Move the selection:<br><br>▸ up<br><br>▸ down |
| `Status line` | Display:<br><br>▸ Number of entries<br><br>▸ Number of filtered entries displayed<br><br>▸ 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:

## 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

**Symbol for execution**

The symbols correspond to the symbols (on page 206) generally used the REE. Operations are symbolized in the REE by triangles in the left and right corner.

| Symbol | Description | Tooltip |
|---|---|---|
| No symbol | `Idle:`<br><br>No object is active in the operation. | Idle |
| | `Running:`<br><br>A symbol is always displayed whilst an operation is running.<br><br>If objects are still running in the operation, the progress symbol is shown. | Execution of the internal objects. |
| | `Phase finished:`<br><br>If objects are already active after the operation, the symbol for `following condition` is displayed. | Waiting for `Following conditions.` |
| | If an object has a different status within the operation, then a symbol is also displayed on the right.<br><br>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`. | |
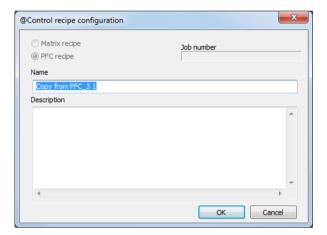
## 2.7.9 Control recipe

Control recipes control the progress of a recipe in the Runtime.

**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....`

| Parameter | Description |
|---|---|
| Matrix recipe | Active: A matrix control recipe is created. |
| | Only display. |
| PFC recipe | Active: A PFC control recipe is created. |
| | Only display. |
| Job ID | The job ID provided by the job variables (on page 13). |
| Name | Unique name for the control recipe. The name must not contain a dot (.), a question mark (?), a @ or an asterisk (*). |
| | 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 | Accepts configuration and closes dialog. |
| Cancel | *Discards entries 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.

**Tool bar and context menu for control recipe list view**

**TOOLBAR**

| Symbol | Description |
|---|---|
| `Open control recipe in Editor` | Opens the selected recipe in the recipe editor. |
| `Rename control recipe` | Opens dialog to rename 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. |
| `Start control recipe` | Starts selected control recipe. |
| `Column selection` | Opens the dialog for selecting the columns which should 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` | Creates a copy of the selected recipe and opens the dialog to rename the duplicate. |
| `Delete` | Deletes selected recipes. |
| `Starting` | Starts selected control recipe. |

## Control recipe edit mode tool bar

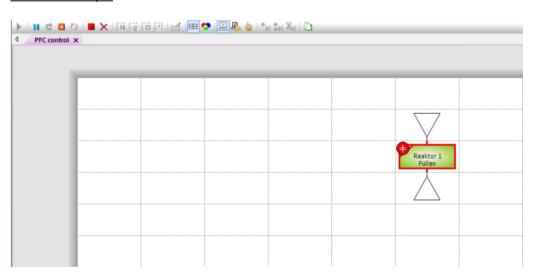| Parameter | Type | Description |
|---|---|---|
| `Start recipe` | Command | Starts the recipe process. |
| `Recipe pausing` | Command | Pauses the recipe. |
| `Recipe resuming` | Command | Resumes paused recipe. |
| `Recipe holding` | Command | Holds recipe. |
| `Restart recipe` | Command | Restarts held recipe. |
| `Recipe stopping` | Command | Stops the recipe. |
| `Recipe abborting` | Command | Aborts the recipe process. |
| `Phase pausing` | Command | Pauses the phase. |
| `Phase resuming` | Command | Resumes the process of a paused phase. |
| `Phase holding` | Command | Holds phase. |
| `Restart phase` | Command | Restarts held phase. |
| `Escape phase` | Command | Starts process to exit (on page 226) from the phase. |
| `Check recipe for errors.` | Action | Starts recipe validation (on page 175). |
| `Edit element` | Action | Opens the corresponding dialog for editing the selected element. |
| `Show/hide grid` | Action | Shows/hides the grid. |
| `Change background color and grid color` | Action | Opens the dialog (on page 131) for changing the color of the background and the grid. |
| `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 only at selected execution position` | Command | Continues a recipe at the selected position. |
| `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`. |

| Duplicate recipe | Action | 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.<br><br>The copy of the recipe automatically gets the status Prepared and can therefore be edited and started. The REE status (on page 212) of the duplicate is set to automatic.<br><br>When duplicating a recipe, a CEL entry is created. |
|---|---|---|

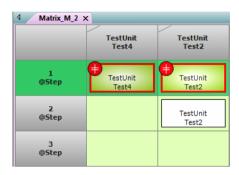## Execute control recipe

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 tool bar if the control recipe is opened

PFC control recipe:

Matrix control recipe:



## CHANGE VALUES

If in the master recipe property `Changeable in the control recipe` was activated, certain values can be adapted in the control recipe as long as the phase i snot 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.
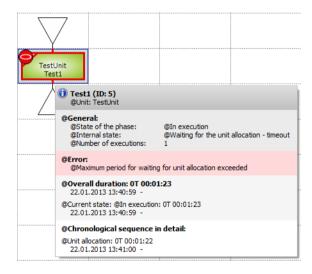
## MEANING OF THE SYMBOLS NEXT TO THE VALUES:

| Symbol | Description |
|---|---|
| = | Value in the control recipe and in the master recipe match. |
| ◇ | Value in the control recipe and in the master recipe are different. |
| ⇆ | 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. |

### TOOLTIP

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



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

## 2.7.10   Synchronization

**CHECKING FOR CHANGES IN THE EDITOR**

When loading, opening, duplicating or approving a recipe, a check is made to see if the configuration of the unit, phase, 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. When the recipe was changed, it is shown via an asterisk (*).

At reloading the recipe is also checked.

> **Info**
>
> *Only recipes in edit mode or test mode are updated. Recipes in test mode that are currently running are only updated after execution has ended.*

If a changed recipe should be release which was not yet saved, a prompt is displayed with the following possibilities:

- ▶ to approve the current recipe

- ▶ to approve the saved recipe

- ▶ cancel the release process

> **Attention**
>
> *Changes to phases in the Editor are taken over without warning message when the recipe is releases 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 the four following levels:

- ▶ in the zenon Editor

- ▶ In the template of the operation

- ▶ in the master recipe

- ▶ in the control recipe

When instancing, the data form the level above is always used. Synchronization and comparison in the editing dialogs also always relates to the previous level.

**ORIGIN OF DATA FOR COMPARISON OR SYNCHRONIZATION:**

| Position of phase | Phase that provides comparison data |
|---|---|
| in the zenon Editor | No data to compare. |
| in the master recipe | Phase configured in the zenon Editor. |
| in the control recipe | The phase from the master recipe. |
| In the template of the operation | Phase configured in the zenon Editor. |
| In the instance of the operation in the master recipe | The phase from the template of the operation. |
| In the instance of the operation in the control recipe | The phase from the instance of the operation in the master recipe. |

You can find information on the origin of the comparison data in the tooltips of the control elements for synchronization.

**RECIPES AND OPERATIONS**

The following is always the case when synchronizing operations:

▶   Master recipes are always synchronized with the editor data

▶   Then the instances of operations are synchronized with the data from their templates

Synchronization is carried out if:

▶   A recipe is opened,

▶   loaded,

▶   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.

## 2.7.11 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 and/or control recipes.

In the screen different control elements (buttons, lists, editors) are available for different tasks. The screen is separated in three main areas which can be used and engineered in part completely independent of each other:

- ▸ Area master recipes (on page 196): Consists of a list and buttons for managing. The area can be used completely independently.

- ▸ Area control recipe (on page 201): 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 125) or the matrix editor (on page 165) is used. The recipe editor needs either `List master recipe` or both lists for a recipe to be opened in it.

**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 53). Online filtering is also possible. These filters are discarded when the screen is called up again. A permanent definition is only possible 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.<br><br>**Hint for ideal configuration of the list:**<br><br>The list can be designed diversely concerning content and look:<br><br>▸ **Content:** The displayed columns (on page 62) can be selected, the column format (on page 65) (column width, alignment, label) can be changed and you can define filters (on page 67). These settings can be edited in Editor and Runtime.<br><br>▸ **Look:** At the settings of the list in the Editor you can find diverse setting possibilities in areas `Representation`, `Scroll bars` and `Color`. With these properties you can even design the list ready for touch operation. |
|---|---|

## ACTIONS FOR LIST MANAGEMENT

| Action | Description |
|---|---|
| `Column selection master recipe...` | Opens a dialog in order to determine which columns should be displayed (on page 62). <br> 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 65). <br> 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 completely empty master recipe in status `Editable`. The dialog (on page 121) for entering a unique name and a free description is displayed. The uniqueness of the name is also checked in the zenon network. <br> Depending on the license you can also select the recipe type: Matrix recipe (on page 165) or PFC recipe (on page 124). If only one of these recipe types is licensed, the licensed recipe type is fixedly set. The selected recipe type cannot be changed afterwards. <br><br> When creating a master recipe, a CEL entry is created. |
| `Duplicate master recipe` | Only active if exactly one recipe is selected. <br> 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. <br> The copy of the recipe automatically receives status `Editable` and can be edited further. <br><br> 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. <br> Deleting is only possible if there are no control recipes which are based on the master recipe. First you must delete all control recipes. <br> 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`. <br><br> If recipes must not be deleted - e.g. FDA regimented environment - we |

| | |
|---|---|
| | recommend not to engineer this button or to give it an appropriated `Authorization level`.<br>When deleting a recipe, a CEL entry is created. |
| **Rename master recipe** | Only active if exactly one master recipe was selected.<br>The dialog for the input of a unique name and the description is opened.<br>Recipes can only be renamed if they are in status `Editable`.<br>Also use this function in order to changed the description of the control recipe.<br><br>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 occur during the validation (on page 175), you must first fix them.<br>Only recipes in status `Test mode` and `Editable` can be released.<br><br>Released recipes can no longer be edited. Control recipes can only be created from released recipes.<br>For details about the states see chapter Recipe types and recipe states (on page 118).<br><br>When releasing a recipe, a CEL entry is created. |
| **Test master recipe** | Changes the master recipe status of the selected recipe to `Test mode`. Only faultless recipes can be switched to test mode. If error occur during the validation (on page 175), you must first fix them.<br><br>Recipes in the test mode can be executed but no longer reengineered.<br>For details about the states see chapter Recipe types and recipe states (on page 118). |
| **Edit master recipe** | Changes the master recipe status of the selected recipes to `Editable`. In this status recipe can again be edited completely.<br>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 187) 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. Therefore control recipes cannot be opened automatically. They must be called up manually and via filters:

1.  Provide the fitting filter options.

2.  Select the desired master recipes.

3.  Click on button `Display associated control recipes in list`.

4.  All control elements complying with the filters and the selection are displayed in the list of the control recipes.

| Action/filter | Description |
|---|---|
| `currently executed control recipes` | Opens only control recipes that are currently being executed. Control recipe status: `Running` |
| `prepared control recipes` | Opens only control recipes which are prepared for execution. Control recipe status: `Prepared` |
| `finished control recipes` | Opens only control recipes which have already been executed. Control recipe status: `Executed` |
| `Display associated control recipes in list` | Displays all control recipes that are based on the selected master recipe and that comply with the set filter criteria. |

## 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.<br><br>Per default the list is empty. For filling the list you must:<br><br>▸ select master recipes<br><br>▸ Set the **currently-executed control recipes**, **prepared control recipes** and **completed control recipes** filters<br><br>▸ click button **display associated control recipes in list**<br><br>In addition to the filters mentioned above, you can filter the list itself. The filtering can be preset in the zenon Editor in the screen switch function (on page 53). Online filtering is also possible. These filters are discarded when the screen is called up again. A permanent definition is only possible in the zenon Editor.<br><br>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.<br>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.<br><br>#### Hint for ideal configuration of the list<br><br>The list can be designed diversely concerning content and look:<br><br>▸ **Content:** The displayed columns (on page 62) can be selected, the column format (on page 65) (column width, alignment, label) can be changed and you can define filters (on page 67). These settings can be edited in Editor and Runtime.<br><br>▸ **Look:** At the settings of the list in the Editor you can find diverse setting possibilities in areas `Representation`, `Scroll bars` and |

| | |
|---|---|
| | `Color.` With these properties you can even design the list ready for touch operation. |

## ACTIONS FOR LIST MANAGEMENT

| Action | Description |
|--------|-------------|
| Column selection master recipe... | Opens a dialog in order to determine which columns should be displayed (on page 62).<br>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 65).<br>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.<br>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.<br><br>The copy of the recipe automatically gets the status Prepared and can therefore be edited and started. The REE status (on page 212) of the duplicate is set to automatic.<br><br>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.<br>Deleting is only possible if all selected recipes are not executed (control recipe status: In execution). First they must be finished, stopped or canceled.<br><br>If recipes must not be deleted - e.g. FDA regimented environment - we recommend not to engineer this button or to give it an appropriated Authorization level.<br>When deleting a recipe, a CEL entry is created. |
| Rename control recipe | Only active if exactly one control recipe was selected.<br>The dialog for the input of a unique name and the description is opened.<br>Recipes can only be renamed if they are in status Prepared.<br>Also use this function in order to changed the description of the control recipe. |
| Open control recipe | Opens the selected control recipe 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 defined REE mode. The recipes are executed invisibly at the Server. It is not necessary that the recipe is opened in the recipe editor. |

**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` the master recipe can 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. For this column `Control recipe ID` must be visible. 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 regards 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` 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 folder for the command recipes are sub folder 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.

## 2.7.12   Recipe Execution Engine (REE)

The REE (Recipe Execution Engine) executes recipes in the Runtime. You can start any number of recipes.

## 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 simultaneous sequence.

### SYMBOLS AND WHAT THEY MEAN:

| Symbol | Meaning |
|--------|---------|
| ⏳ | Phase starts. |
| ⊖ | Waiting for unit allocation. The unit of the phase is already being used in another recipe. |
| ⊕ | Whilst waiting for the input interlocking. |
| ▶ | ▸ During the execution of a phase and the waiting for `Reaction finished.`<br><br>▸ With transitions: Whilst running and waiting for transition condition.<br><br>▸ With end simultaneous sequence: Waiting for all branches combined. |
| ↻ | In preparation for start. After a restart when waiting for the restart conditions. |
| ✔ | ▸ `Reaction finished` recognized; waiting for `Reaction completed`.<br><br>▸ With transitions: Waiting for transition condition met.<br><br>▸ With end simultaneous sequence: Waiting for simultaneous sequence finished and waiting for following condition. |
| ◷ | Waiting for `Minimum execution duration` procedure. |
| ⊗ | Waiting for exclusive execution. |
| ⊖ | Time out for unit allocation expired. |
| ⊕ | Time out for input lock expired. |
| ▤ | Waiting too long for value for writing. |
| ⬇ | Time out for `Reaction completed` expired. |
| ▶ | Time out for `Reaction ended` expired. |

If an error occurs during a phase, the phase is marked as faulty until it is restarted.

## STATE

The execution status (on page 212) of `phases, transitions` and `end simultaneous sequence` is signaled in color:

| State | Color |
|---|---|
| Idle: | white |
| Running: | green |
| Finished: | blue |
| Stopping: | Two colors:<br><br>▸ yellow<br><br>▸ Original color |
| Stopped: | yellow |
| Pausing: | Two colors:<br><br>▸ orange<br><br>▸ Original color |
| Paused: | orange |
| Holding: | Two colors:<br><br>▸ grey<br><br>▸ Original color |
| Held: | grey |
| Aborting: | Two colors:<br><br>▸ red<br><br>▸ Original color |
| Aborted: | red |
| Restarting: | Two colors:<br><br>▸ green<br><br>▸ Original color |
| Force:<br><br>(phases or transitions only) | Violet border |

| Timeout: | red border |
|----------|------------|

## ACTION ON STOP COMMAND

After a stop command, the **phases**, **transitions** and end **simultaneous sequence** 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.

## Create image

At closing the Runtime an image file Image of the running recipe is created. This image contains the images of the REEs, the order of the allocation and all reactions which are needed for the restart. The image can also be created cyclically during Runtime operation.

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 image to be loaded and remains as a file after restarting. All other files with the extension **.REE** are deleted.

### ENDING THE IMAGE IN RUNTIME

If the Trigger end of Runtime event is triggered, the ending of Runtime is prevented until the Batch Control module has backed up all data. A process screen 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. Find out more information in the chapter Exit and restart Runtime (on page 230)

### CYCLIC IMAGE WRITING

An image can also be written cyclically during Runtime. As soon as Activate cyclic writing is activated, an image is created in the given cycle.

To activate cyclical writing:

1. Go, in the properties of the Batch Control module, to the Cyclic image writing group.

2. Activate the  propertyActivate cyclic writing

3.  In the Cycle time [s] property, create the time period for the writing of the image; the minimum time period is 30 seconds

    Note: If the cycle time is changed and reloaded, then the next time to which an image is written is recalculated.

With cyclical writing, the last two image files are always retained. Older ones are deleted. Writing is a two-stage process:

1.  The 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.

### SAVING THE CURRENT RECIPE TO THE IMAGE

With cyclically-created images, all recipes that are currently running are saved in the image. This ensures that the recipe is appropriate to the execution status on restarting. To do this, the recipe from the 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 image has been created.

**Behavior of elements in Runtime**

### 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 `next step`, then all phases are paused and there are no more active elements.

**Solution:** Resuming of the paused phases.

## PAUSE AND RESUME

The following applies to pausing and resuming:

▶ Pausing and resuming with active element: A paused phase that is active for an active element is not continued.

▶ Switching from manual REE mode to automatic: All phases that are active for an active element are resumed.

▶ If a phase is `paused` in manual mode and the REE is switched to `automatic`, the phase remains paused. A global resumption would also not put this phase back to the status of `resume`, because the phase was paused in `manual` mode.

- A `Pause phase` command, followed by a global pause and a global resumption, sets phases that were paused using the `Pause phase` command to the status `resume`.

- Phases that were paused in `manual` mode can be set to the status of `resume` with the `Resume phase` command.

▶ 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.

## SEQUENCE SELECTIONS

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**

▶ A transition before an end simultaneous sequence remains active, including the phase before the transition after it has been run through, if the transition is active after the end simultaneous sequence but has not yet been run through.

▶ In simultaneous sequences, 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.

## 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 (forced).

To execute a recipe manually or semi-automatically, you can use the operation types (on page 214) **Step-by-step execution of the recipe** and **Jump**.

When switching to `automatic` mode, all execution positions are removed. Global commands are only executed in branches that have no execution positions.

> 💡 **Info**
>
> *To react on serious events, you can change the mode during the running process via reaction type (on page 32)* `influencing the recipe`*.*

## The REE states

The following states are possible:

| State | Description |
|---|---|
| Idle | The REE is in idle state. |
| Running | When starting a control recipe, it changes to the status `running`. |
| Executed | As soon as the execution is finished, the recipe changes to status `Finished`. In this status execution is not possible. |
| Pausing | The recipe changes to status `Paused`. |
| Paused | Within the phase the process stops at:<br><br>▸ Waiting for `Finished`<br><br>▸ Waiting for `Allocation`<br><br>▸ Waiting for `Interlocking condition`<br><br>▸ Waiting for `Phase finished`<br><br>▸ Check for parallel execution |
| Holding | The object changes to `Held` and does not carry out any allocations anymore. When restarted, the object is restarted and changes to `running`. |
| Held | Within the phase the process stops at:<br><br>▸ Waiting for `Finished`<br><br>▸ Waiting for `Allocation`<br><br>▸ Waiting for `Interlocking condition`<br><br>▸ Waiting for `Phase finished`<br><br>▸ Check for parallel execution |
| Restarting | Phase is restarting. |
| Restarted | Phase is completely restarted. |
| Stopping | Stops the process and changes to `Stopped`. |
| Stopped | The object was stopped. |
| Aborting | Aborts the process and changes to `Aborted`. |
| Aborted | Recipe process was aborted. |

| | If a recipe cannot be restarted in the image at the restart, its status automatically changes to `Aborted`. |
|---|---|
| `Prepared` | Prepared for execution. |

## EXECUTION POSITION AND JUMP TARGETS

| State | Description |
|---|---|
| **Resumption** | If an object is paused and after the object an execution position is located, `Resume` has the same effect as `Next step`. This also includes jumps.<br><br>At a phase command the command only effects a jump in the same branch. |
| **Held** | Removed<br><br>   ▸   With **phase**: execution positions in the branches<br><br>   ▸   With **global**: all execution positions |
| **Pause** | Has now effects for jump targets. Already defined targets remain. |
| **Others** | Always causes the deletion of the jumps.<br><br>For a phase command only the jump in the area of the phase is deleted. |

## 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 held as soon as an element is finished with its execution. The holding is done via command `Pause` to the concerned execution path. As soon as all active elements in this path reached status `Paused`, the execution position is marked by an 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.
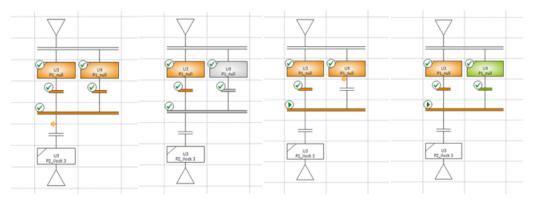
**Exceptions:** `Pause` and `Resume`. The execution positions remain as they are.

## PHASE COMMANDS

> ▶ `Hold` for a phase  The execution position is deleted from the execution path of the phase.

> ▶ `Resume`: If an execution position exists, a selective next step is executed in this execution path.
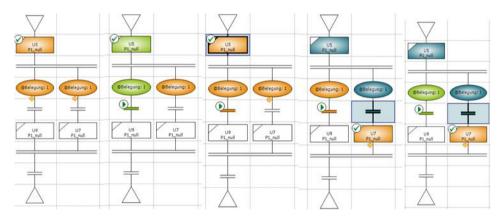
## SPECIAL CASES：

### PHASE COMMAND `HOLD` AND `RESTART` IN A SIMULTANEOUS SEQUENCE FOR ACTIVE END SIMULTANEOUS SEQUENCE

If you hold in this example (images from left to right) a simultaneous sequence and restart it, then you will reach after a step-by-step execution the already paused **end simultaneous sequence**. To resume the execution from here, the left phase must receive command Resume.

## ONLY ONE PATH IN A SIMULTANEOUS SEQUENCE WITH AN ACTIVE PHASE BEFORE STEP-BY-STEP EXECUTION



If, in a simultaneous sequence with a phase before, only one path is executed completely and waiting is taking place in one of the other simultaneous sequences (phase before is running), no active element is displayed in the simultaneous sequence. 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.

If then however the simultaneous sequence with the execution position is still executed, the phase before the simultaneous sequence is deactivated. With this the left path remains in Paused and without execution position. 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 execution position with the mouse cursor

2.  move it to one of the offered targets
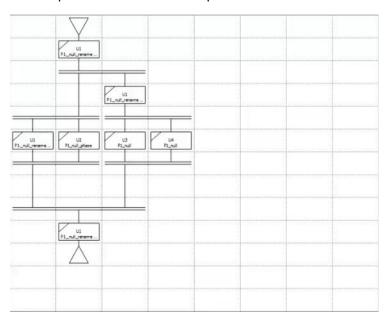
3.  execute the next step

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 sequence selection objects), this jump is ignored and a simple step is executed.
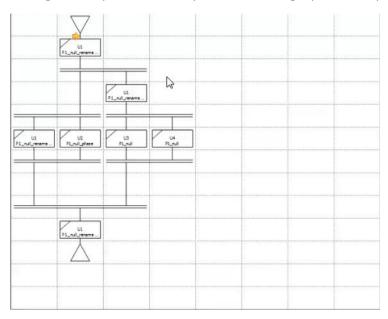
## JUMPING IN SIMULTANEOUS SEQUENCES

If, in parallel sequences, a jump is made in first simultaneous sequence via the phase, then the first phase before the simultaneous sequence is deactivated. Therefore no phase is active. Phases cannot be skipped if this means that no phase would be active in the recipe. The following behavior when skipping leads to an error:
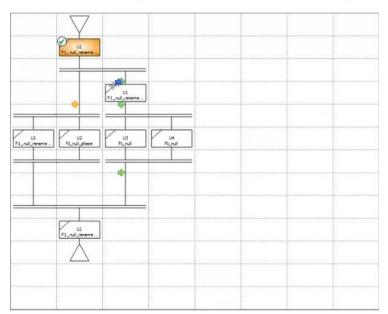
▶ Recipe with simultaneous sequence

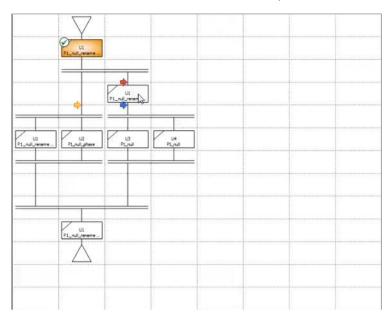▶ A global step activates the phase before begin parallel sequence.



▶ After the next step, the active element is before the phase in the simultaneous sequence.
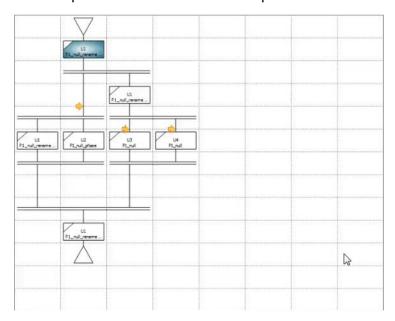
▶ The active element is moved behind this phase.



**Rule when jumping:** Objects before a begin parallel sequence become inactive as soon as objects become active after this.

Thus the first phase becomes inactive when jumping. The skipped phase in the simultaneous sequence never became active, nor did the following phases.

Thus no phase is still active in the recipe:

## 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. We give the phase the name **Phase 1**. You can find additional special process behavior with following condition in chapter Following condition (on page 225).

| Phase | Transition | Event |
|---|---|---|
| Phase is activated<br> | | `Phase activated` |
| All variables of the phase are registered at the drivers. | | |
| `Unit allocation` is started and `Waiting period unit allocation` is started.<br> | | If the unit allocation was not successful in the first try: `Unit allocation not possible` |
| Optional: Unit allocation possible within the waiting period.<br> | | `Waiting period unit allocation exceeded` |
| Check is started whether phase isn't already executed. This can happen if the phase is already active in a simultaneous sequence or if the unit allocation was skipped manually and the phase is already executed in another recipe.<br> | | `Phase started multiple times` |
| Start phase | | `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 interlocking blocked` |

| | | |
|---|---|---|
| Unit 1 Phase 1 | | |
| Optional: Condition of the input interlocking not TRUE within the waiting period. Unit 1 Phase 1 | | `Waiting period input interlocking exceeded` |
| Writing the command tag Unit 1 Phase 1 | | When all command tags were written: `Finished writing command tags` |
| Checking of the phase-done condition is started and time for `Minimum execution duration` and `Maximum execution duration` is started. Unit 1 Phase 1 | | |
| Optional: Phase-don condition fulfilled but `Minimum execution duration` not reached. Unit 1 Phase 1 | | |
| Optional: Condition of the `Phase-done condition` not TRUE within the `Maximum execution duration`. Unit 1 Phase 1 | | `Maximum waiting period phase done condition exceeded` |

| | | |
|---|---|---|
| Optional: Waiting for restart of the whole execution. If the execution is still restarting (for other phases in the recipe the restart condition is not yet fulfilled), it is waited here. This guarantees that the following element is activated after the recipe changes to **running**. | | |
| `Phase-done condition` is `TRUE` and minimum execution duration is reached or exceeded. Waiting period following condition is started. | Transition is activated | `Phase finished` |
| | All variables of the transition are registered at the drivers. | `Waiting period following condition exceeded` |

| | | |
|---|---|---|
| Optional: `Following condition` not within waiting period `TRUE`<br><br>Unit 1 Phase 1 | The transition condition is checked. | |
| 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 is fulfilled.<br><br>Unit 1 Phase 1 | Transition condition is `TRUE`. | `Phase deactivated` |
| 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. | |

### RULE FOR VALUES OF TRANSITIONS

If a transition has value `TRUE` for the phase-done condition during the waiting period, it is marked as `finished`. If its value should later change to `FALSE`, the execution of the recipe is not influenced.

### GLOBAL PAUSING AND CONTINUATION IF THERE IS A COMMUNICATION ERROR

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.

**Following condition**

A phase is active as long as the following condition is fulfilled. Only once the following condition is fulfilled is the phase is deactivated. The `phase completed` event is triggered and the phase is deactivated. Before it is deactivated, the event reactions are executed. Se also Process of a phase in detail (on page 220)

The following condition can be very different. Here some examples:

▶ Phase 1 followed by a phase 2:
As soon as the phase done condition (and optional the `Minimum execution duration`) is fulfilled for phase 1, phase 1 is completed and phase 2 is activated.
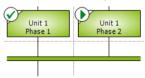


▶ Phase 1 followed by a transition:
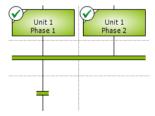Only when the transition condition is fulfilled, phase 1 is completed.



▶ Phase 1 and phase 2 parallel followed by an end simultaneous sequence:
Not before the phase done condition (and optional the `minimum execution duration`) is fulfilled for both phases, both phases are completed.



▶ Phase 1 and phase 2 parallel followed by an end simultaneous sequence followed by a transition:
Only when the transition condition is fulfilled, both phases are completed.

**Escape phase**

It is possible to exit a phase during execution. Execution after the phase is then continued 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 states`/Escape condition property.

It is also possible to exit if the phase is in this status:

- ▶ `Running`

- ▶ `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: All steps between instigating the exit and *Wait until the recipe has the status* `"running"` are not executed.

- ▶ Running:  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.

> 💡 **Info**
>
> *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`.

### 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 `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.

### Secure writing of the command parameters

The command parameters are written 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. Command parameters are written.

   - Waiting occurs until the parameters written have been confirmed.
   - If the confirmation is not made within the waiting period, the values are rewritten.
   - If a write is confirmed as negative, the values are written again. There is a maximum of 3 repetitions.

3. The actual values are compared to the written values.

   - After a positive write confirmation, waiting occurs until the variables to which they were written can be read again. All written values must be active at the same time. If the waiting period has expired, writing starts again. There is a maximum of 3 repetitions.

4. If all attempts have been unsuccessful, the `Command parameter without value` event is triggered.

5. If secure writing is unsuccessful, a `communication failure` event is triggered. That means:

   - The error must 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 Runtime is ended whilst writing the command parameters, these are rewritten when Runtime is started. 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

> 💡 **Info**
>
> *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 standard 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.

**Exit and restart Runtime**

At closing the Runtime an image file 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. A time out is written in the log file.

Variables which don't access a driver are always written without an acknowledgment even if an acknowledgment 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.

It is waited for the values of the internal driver 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 251).

## 2.8      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 seamlessly on the Standby!*

For using Batch Control in a network the following is true:

**ALLOCATION**

> ▸      The forcing of allocations can be carried out be Server and 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.

### RECIPES

- ▶ Recipes can be started and controlled from Server and Client.

- ▶ If parameters in a recipe are changed whilst the recipe is saved on a different client, the change to the parameters is refused and not carried out.

- ▶ 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 tool bar, all symbols that are not permitted are deactivated; it is not possible to select the corresponding objects.

The web client PRO is not affected by these restrictions.

## 2.9 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 233)

▶ 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

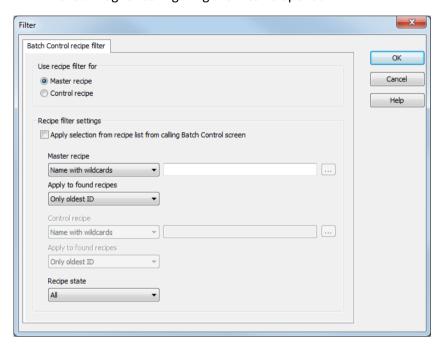### 2.9.1 Batch Control recipe filter

When screen switching to a report viewer type 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`

4. the dialog for configuring the filter is opened

| Parameter | Description |
|---|---|
| **Use recipe filter for** | Selection of the recipe type that is applied to the filter:<br><br>  ▸  Master recipe<br><br>  ▸  Control recipe |
| Master recipe | `Active`: It is filtered on Master recipes. |
| Control recipe | `Active`: It is filtered on control recipes.<br><br>Note: The attendant master recipes must also be selected. If no master recipe has been selected for the control recipe, th filer cannot find the recipe being searched for in Runtime.<br><br>Hint: If the master recipe is not known, filtering of all master recipes with a placeholder is recommended. |
| **Recipe filter settings** | Options for the recipe filter |
| Apply selection from recipe list from calling Batch Control screen | `Active`: In 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.<br><br>`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:<br><br>  ▸  <u>Name with wildcards:</u><br>     A name with placeholder can be entered into the input field. Filtering according to this name is carried out.<br><br>  ▸  <u>Name from variable:</u><br>     The name of the master recipe is defined by a variable in Runtime. A click on button `...` opens the dialog for selecting variables.<br><br>  ▸  <u>ID from variable:</u><br>     The ID of the master recipe is defined by a variable in Runtime. A click on button `...` opens the dialog for selecting variables. |
| 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: |
|---|---|
| | ▸ <u>Name with wildcards:</u><br><br>A name with placeholder can be entered into the input field. Filtering according to this name is carried out.<br><br>▸ <u>Name from variable:</u><br><br>The name of the control recipe is defined by a variable in Runtime. A click on button `...` opens the dialog for selecting variables.<br><br>▸ <u>ID from variable:</u><br><br>The ID of the master recipe is defined by a variable in Runtime. A click on button `...` opens the dialog for selecting variables.<br>Precisely one recipe can be found if the variable value at the time of execution is a valid ID of a control recipe.<br><br>▸ <u>Job ID from variable</u><br>Finds control recipes that belong to the master recipes already found and which have the given job ID. Any type of variable can be linked. The value is automatically converted into STRING.<br>**Note:** If the variable does not have a value, no recipe is sent to the Report Viewer. |
| Filter result for | Define which ID is to be selected when filtering for names or job ID by selecting from the drop-down list:<br><br>▸ Only oldest ID<br><br>▸ Only newest ID |
| Recipe state | Selection of the recipe status from the drop-down list:<br><br>▸ All<br><br>▸ Prepared<br><br>▸ In execution<br><br>▸ Executed<br><br>▸ Terminated with error<br><br>▸ Outdated |

| OK | Applies all changes, creates filter and closes the dialog. |
|---|---|
| Cancel | Discards all changes and closes the dialog. |
| Help | Opens the 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.

## 2.10 Formula editor

The formula editor is automatically opened if you need to enter or edit a formula. Above all:

Editor:

- ▶ Phases
- ▶ Interlocking conditions
- ▶ all conditions for transitions
- ▶ Phase done condition

Runtime:

- ▶ Phase done condition and interlocking when editing a phase
- ▶ Editing transitions

**Note:** If the phase referenced in the formula is removed and a new phase is added, the operands are reassigned in the case of operands. To do this, the same phase must be reinserted. Parameters from a different phase are not automatically linked.
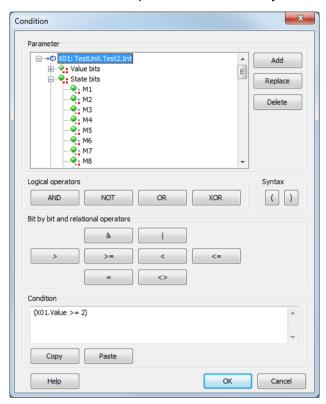
**ENTER FORMULA**

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

| Parameter | Description |
|---|---|
| **Tag list** | List of the tags which can be used for the formula.<br><br>Each entry contains of:<br><br>▸ a basis node for the label<br><br>▸ a value<br><br>▸ a status<br><br>▸ the bits for value and status<br><br>A symbol at the first node displays whether it is a command or return tag.<br><br>The short indetifier at the beginning of the name is used for the formula. |
| **Add** | Opens the dialog for adding a parameter. For this, the following applies:<br><br>▸ The following can be added: numeric and binary tags and tags for time duration. Values for duration are converted to seconds<br><br>▸ For conditions of the phase only the tags created for it can be added.<br><br>▸ Tags can be added multiple times. |
| **Replace** | Makes it possible to replace a tag. The short identifier remains the same at replacing. |
| **Remove** | Removes the highlighted tag. For a tag to be deleted:<br><br>▸ the formula must be correct<br><br>▸ 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 than or equal |
| < | less than |
| <= | Less then or equal |
| = | equal |
| <> | less or greater |
| **Condition** | Configuration and display of the formula. |
| **Copy** | Copies the whole formula:<br><br>▸ All configured tags from the tag tree<br><br>▸ 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.<br><br>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.<br><br>For this the formula must be correct. |
| **Cancel** | Discards all changes and closes the dialog. |

> 💡 **Info**
>
> *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 (von 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. |
| `State 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 status` | In the formulas, all values (value bits and status bits) are treated as binary values and can be logically linked with AND, OR, etc.<br> The total value and overall status are an exception to this. In order to get a Boolean result this total value has to be ORed with a constant `bitwise`. For this, we use the operator `&`.<br>For the result 0 (false) of this logical ORing we get the binary value 0 (false), otherwise 1 (true).<br><br>Example: see chapter Example bit by bit ORing |

> 💡 **Info**
>
> *The status bits NORM and N_NORM are only available in the formula editor and cannot be engineered via the status.*

> 💡 **Info**
>
> Formulas with binary x values and bit-by-bit link can be used with a maximum of 2 binary values. If more values are necessary the link must be done 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

## 2.10.1    List of status bits

| Bit number | Short term | Long name | zenon Logic label |
|---|---|---|---|
| 0 | M1 | User status1 | _VSB_ST_M1 |
| 1 | M2 | User status2 | _VSB_ST_M2 |
| 2 | M3 | User status3 | _VSB_ST_M3 |
| 3 | M4 | User status4 | _VSB_ST_M4 |
| 4 | M5 | User status5 | _VSB_ST_M5 |
| 5 | M6 | User status6 | _VSB_ST_M6 |
| 6 | M7 | User status7 | _VSB_ST_M7 |
| 7 | M8 | User status8 | _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 | Runtime exceedance | _VSB_RTE |
| 12 | MAN_VAL | Manual value | _VSB_MVALUE |
| 13 | M14 | User status14 | _VSB_ST_14 |
| 14 | M15 | User status15 | _VSB_ST_15 |
| 15 | M16 | User status16 | _VSB_ST_16 |
| 16 | GI | General query | _VSB_GR |
| 17 | SPONT | Spontaneous | _VSB_SPONT |
| 18 | INVALID | Invalid | _VSB_I_BIT |
| 19 | T_CHG_A | Daylight saving time/winter time announcement | _VSB_SUWI |
| 20 | OFF | Switched off | _VSB_N_UPD |
| 21 | T_EXTERN | Real time external | _VSB_RT_E |
| 22 | T_INTERN | Realtime internal | _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<br><br>If no value was transferred, the defined alternate value is used otherwise the last valid value is used. | _VSB_AVALUE |
| 28 | RES28 | Reserved for internal use (alarm flashing) | _VSB_RES28 |
| 29 | N_UPDATE | Not updated | _VSB_ACTUAL |
| 30 | T_STD | 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 |
| 38 | N_CONF | Negative acceptance of Select by device  (IEC 60870) | _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 | Normal 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 | Time invalid | not defined |
| 50 | CB_TRIP | Breaker tripping detected | not defined |
| 51 | CB_TR_I | Breaker tripping detection inactive | not defined |
| 52 | RES52 | reserved | not defined |
| 53 | RES53 | reserved | not defined |
| 54 | RES54 | reserved | not defined |
| 55 | RES55 | reserved | not defined |
| 56 | RES56 | reserved | not defined |
| 57 | RES57 | reserved | not defined |
| 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 |

### Info

*In formulas all status bits are available. For other use the availability can be reduced.*

You can read details on status processing in the Status processing chapter.

## 2.10.2    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 |
| 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.*

## 2.10.3   Bit formula

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 |

**Example bit by bit OR function**

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 `256`; 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 (|) in a similar manner to this example.

## 2.10.4 Comparison operators

Comparison operators serve 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 then or equal |
| >= | Greater than 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. (the constants can only be integers; a comparison to a floating point number is not possible.)
These constants are entered as hexadecimal values or decimal values in the combined element.
Hexadecimal figures are automatically converted to decimal values by clicking on oĸ (for example, 0x64 is in decimal figures 100).

---

### ▣ 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)*
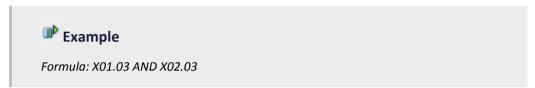
*(X01.value = 0x64) OR (X01.value = 0x65)*
*The result is 1, if the value of X01 is exactly equal to the numeric value 100 or 101 (= hex 0x64 and hex 0x65)*
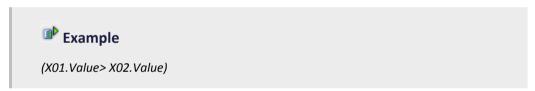
---

## 2.10.5    Examples for formulas

### SIMPLE LOGICAL AND LINKING BETWEEN TWO BIT VALUES
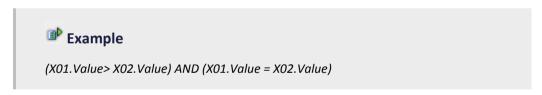
**Example**

*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.
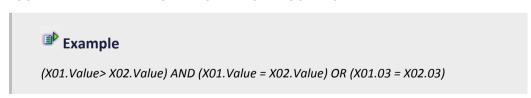
### COMPARISON OF AN ANALOGUE VALUE OR STATUS OF A VARIABLE

**Example**

*(X01.Value> X02.Value)*

### COMPARE ANALOG VALUES WITH EACH 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 oĸ. 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.*

## 2.11 communication errors

If communication errors occur, these can be detected in 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). In addition, a reaction to the `communication error` (on page 27) event can be configured.

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.

To configure the detection of communication errors:

1. highlight the desired phase

2. Click, in the `communication error` property group, on the Loss of communication property

3. In the formula editor (on page 238) 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

    - Phase holding

    - Phase pausing

## CHECKING 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 Runtime, the waiting for values can be in many steps, because the communication is 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 one cannot occur until the value of the formula for the Loss of communication property has not changed 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 above the symbol of the error display and its tooltip. 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.

## 2.12    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 there messages see manual `SYSDRV` (`sysdrv.chm::/22853.htm`) in chapter `Topic – Batch Control` (`sysdrv.chm::/34270.htm`).

## RECONSTRUCTION OF INDEX FILES

If the index file 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.