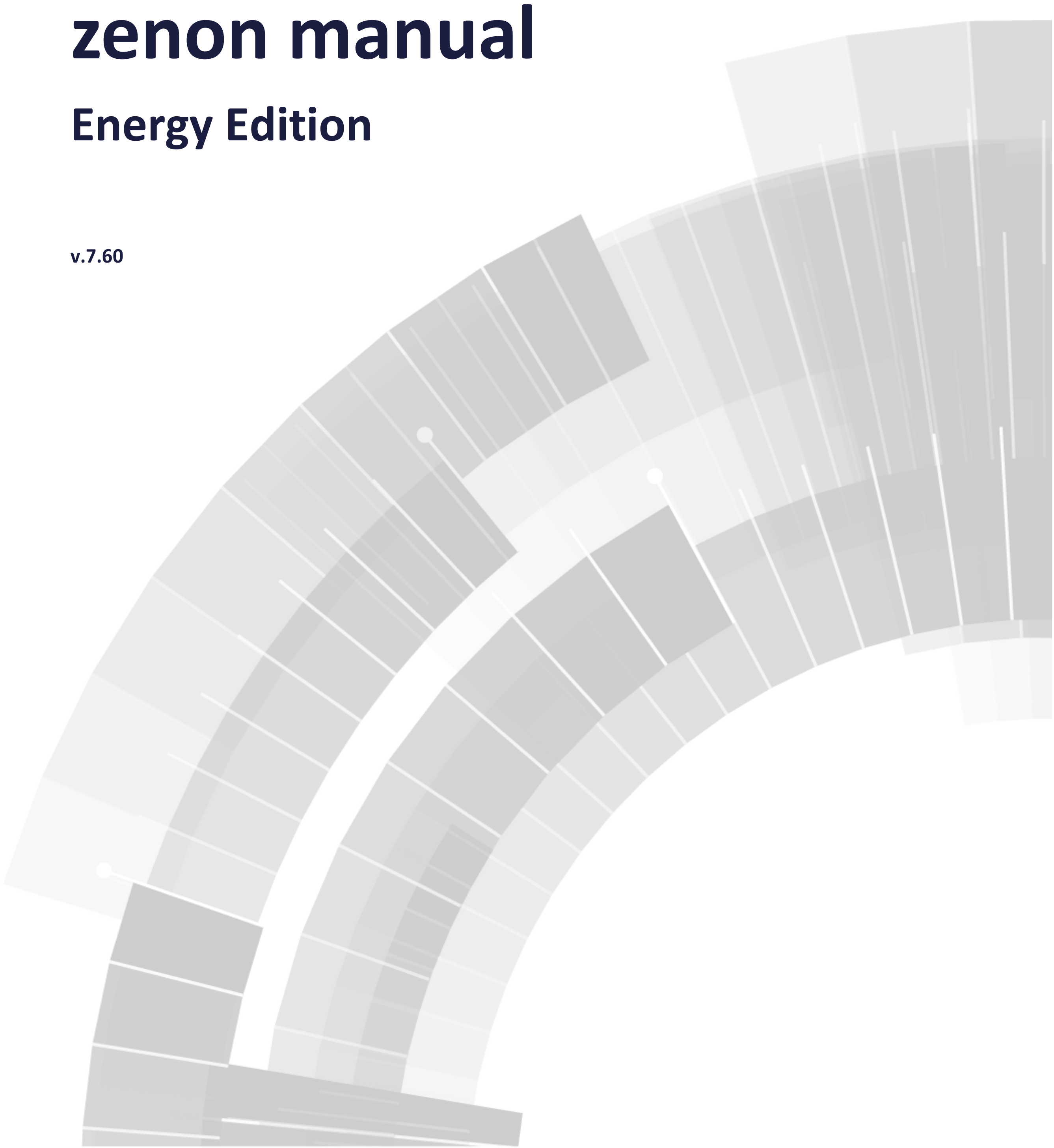
zenon manual

Energy Edition

v.7.60



Contents

1. Welcome to COPA-DATA help

ZENON VIDEO-TUTORIALS

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

GENERAL HELP

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

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

License information

Must be licensed in Editor and Runtime.

The following is available for the Energy Edition:

- Command Processing
- ALC (Automatic Line Coloring): Already included in the license for Energy Edition, provides basic properties for line coloring.
- Command sequences
- Topological element transformer
- Topology package: Requires additional licensing on the server (not on the client) and expands ALC by:
  - Multiple supply
  - Secured supply
  - Topological interlockings
  - Topological element disconnector
  - Error detection and ground fault search

3. Automatic Line Coloring (ALC) - Topology

The topological coloring of lines allows easy automatic dynamizing of tubes in technology (for media) as well as in the energy distribution (for electricity). So process controlled coloring of topological nets can easily be realized.

Because the tube structure is designed in the screen with all its technological elements (e.g. tanks and valves, or generators, switches and consumers), it is internally emulated as a model and the media flow is displayed in the Runtime.

In order to allow screen-overlapping models the entire design and configuration is always project-wide. You therefore have one entire topological model per project, which is used for the calculation of the tube statuses and ultimately for the coloring of the tubes.

The whole topology is created automatically from the graphic design. No other engineering actions are necessary.

Information

The ALC algorithm only runs through once from a source starting from each switch.

DETAIL SCREENS

To display individual screens, a partial area can be taken from the topological network and displayed individually by means of alias. A detail screen (on page 40) can be displayed with the data from different equipment parts, for instance outputs or partial networks.

License information

Must be licensed for Editor and Runtime (single-user, server, standby).

No need to be licensed for Runtime client.

Licensing is carried out using the zenon Energy Edition.

- ALC: Included in the license for Energy Edition; provides basic properties for line coloring.
- Topology package: Requires additional licensing on the server (not on the client) and expands ALC by:
  - Multiple supply
  - Secured supply
  - Topological interlockings
  - Transformer and separator topological elements
  - Error detection (version 6.50 and above)

3.1 ALC elements

Automatic Line Coloring (ALC) makes it possible to color lines depending on the process status. The combined element is used as the process element. Automatic line coloring allows easy automatic dynamizing of tubes in technology (for media) as well as in the topological networks (for electricity).

ENGINEERING

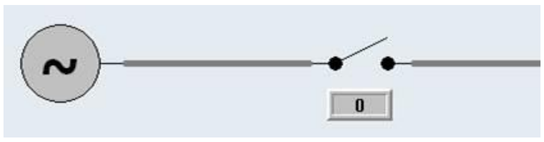
For the design two types of screen elements with different functions are distinguished. On the one hand these are procedural elements (on page 9) (source, switch/disconnector, drain, transformer or link) and on the other hand lines (on page 20).

In doing so, the technical elements have a function and a color (source and transformer). If the procedural elements are active, the connected lines take on the color of these elements at the source and transformer or they take on the color of the element's input line for the switch and the link. If the procedural elements are inactive, the color of the lines is taken from the definition in the editor.

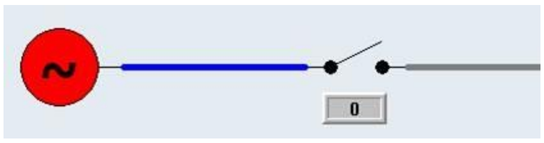
The different functions of the elements are assigned in the properties of the combined element.

EXAMPLE

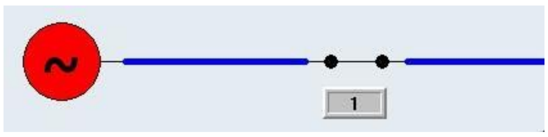
A source has a connected line. A switch is connected to the line. And a second line is connected there. If the source is active, the first line is colored with the color of the Automatic Line Coloring defined in the source up to the valve. The other line is not colored before the switch is closed.



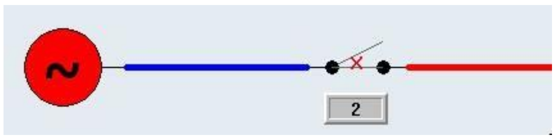
Source inactive



Source active



Switch closed



Undefined or invalid

Information

If the procedural element status is `undefined` *or* `malfunction`*, this is automatically detected. All connected lines and all further elements are displayed in the color of the predefined source* `undefined`*' for both states.*

NUMBER OF CLOSED SWITCHES IN A SERIES

For the correct functioning of the ALC algorithm, the number of connected switches in a series plays a role.

Recommendation: Arrange a maximum of 256 closed switches in a series between the source and the drain.

3.1.1 Procedural elements

Procedural elements are created in zenon with a **combined Element**. Their state determines the coloring of the connected line.

The following settings are available:

Property	Description
Function type	Defines the technological type of the Combined element.
`Terminator`	For bus bar ends. Blocks the error message "**Line only connected on one side**" when being compiled in the Editor.
`Generator`	A generator generally behaves like a source, but it is considered as an independent and not net-synchronous.
`No function`	The element has no function in the ALC. **Note:** The "`no function`" function type is the default value.
`Link`	With a link a line can be continued on some other place. If a link is supplied by a line, all other links with the same link number also are supplied by this line. Here it does not matter, whether the links are in the same screen or on different screens in the project. So screen independent lines can be defined. It is possible to have more than two links with the same link number in one project. Links can be supplied by several lines at the same time or can themselves supply several lines. In principle there is no difference between inputs and outputs. The source information is passed on to all connected lines. **Attention:** Two link elements cannot be connected directly to one line. In between, there has to be at least one other procedural element (switch/disconnector or transformer). A link cannot be switched active or inactive: it always is active.
`Source`	Passes on its color. If the source is active (value: `1`), all connected lines that have **Color from ALC** option set in the element properties are allocated the color of the source. The color is defined in the project properties as the source color. (e.g. tanks or generators). A source is a single pole with a static source number assigned to it. The source is switchable over the state of its main variable. Generally, sources are considered as net-synchronous and detachable. You can find details on the source in the configuration of the sources (on page 28) chapter.
`Switch`	With this lines can be split. If the switch is `closed/active` (value: `1`), then the connection between the two lines is closed and the line is colored up to the next switch with the defined source color. In this case a switch forwards the source color of the input line to the output line. If the status of the switch is `invalid` (value: `3`) or `undefined` (value: `2`) or the status of the main variable is INVALID, the line colors itself in the color `undefined` from the ALC configuration in the project properties. A switch thus delivers source number `0` (undefined) to its output (connection `2`) instead of the incoming source number. **Example:** see **Switch example - colors from ALC** (on page 14) section. **Note**: If the **Switch input/output** property is active, the input and output of this element are reversed for the ALC.
`Valve`	A slider (a valve) acts in a similar manner to a `switch`, but it is used for water and gas lines.

	Value of the main variable: - Switch OFF: Value `0` -> Slider closed-> No forwarding - Slider ON: Value `1` -> Slider open completely-> Water flow - Slider DIF: Value `2` -> Slider partially open-> Water flow - Slider STO: Value `3` -> Slider malfunction **Note**: If the **Switch input/output** property is active, the input and output of this element are reversed for the ALC.
`Drain`	This defines the end of the line. The drain does not influence the coloring; it is only used so that the model can be displayed in full. If an external program (e.g. VBA) should access the model, then the drain probably is needed for further calculations, and so has to be inserted. In Energy projects, the drain is used for representing consumers. These are used to calculate the ALC - topological interlockings (in the **command processing**) '**Device would not be supplied**'.
`Check valve`	The `check valve` only forwards information in one direction. Value of the main variable: - `Value 0`: The forwarding is not active (= the valve is closed) - `Value 1 or 2`: Forwarding is only possible in one direction. In doing so, the color of the source is only forwarded from the input to the output. Forwarding in the opposite direction is not envisaged. This also affects the forwarding of the ground. - `Value 3`: Forwarding is undefined. This then occurs, for example if the `check valve` is faulty. In this case the status is only forwarded at the output. **Note**: If the **Switch input/output** property is active, the input and output of this element are reversed for the ALC. The `check valve` is also taken into account by the topological interlockings (on page 33).

`Transformer`	A transformer is a drain and a source at the same time. SO with a transformer the input color (input source) can be transformed to a new output color (transformer source color). The output line is only switched to active once the transformer has an active input line. However the output line does not get the color of the input line as with a switch, but instead the color of the transformer's own source. So a source has to be defined for each transformer. A transformer cannot be switched active or inactive, it always is active. Note: If the **Switch input/output** property is active, the input and output of this element are reversed for the ALC. `Reverse-feed-compatible transformer:` To have a transformer capable of reverse feed, you must select, for **Source for reverse feed**, a different source than `UNDEFINED [0]`. This means that the transformer behaves the same for both directions - from the input to the output (forward) and also from the output to the input (backward). The only difference is that the **Source for reverse feed** property and not the **Source** property is used for further distribution of the source number. Note: Defective network statuses or missing configurations, such as a feed from the input and output at the same time or a short circuit from input and output are not specially colored. This means that the transformer capable of taking a reverse feed behaves like two transformers switched to run antiparallel that are not capable of taking a reverse feed.
`Disconnector`	A disconnector generally behaves like a `switch`. However, a disconnector in the topological model must not be switched when live - topological interlocking "**Disconnector under load**" in the command processing. As with the switch, the main variable determines the status: `On, off, intermediate position, malfunction`. Note: If the **Switch input/output** property is active, the input and output of this element are reversed for the ALC.

The source numbers given - for the source and transformer function types - are forwarded via closed switches (disconnnectors, sliders etc.) up to the devices (drains). The colors of all connected lines and process-related elements are calculated from the higher-level sum of the supplying source numbers.

SOURCE AND LINK NUMBER

Parameter	Description
Source	Here a source is assigned to an element. In this selection box all sources defined in the ALC configuration (in the project properties) are available. All source names are listed. This property is only active if the function type '`source`', '`transformer`' or '`generator`' has been selected. You can find details on the source in the configuration of the sources (on page 28) chapter. **Attention:** the pre-defined system sources (ID 0..9) are not suitable for your own sources in the project; exception - ground.
Link number	Only the link number is entered for a `link` function. All identical link numbers in a project correlate with each other. Detailed description in the function type `Link`. This property is only active, if the function type `link` has been selected.

VARIABLES OF PROCESS-RELATED ELEMENTS

In order for a switch (and disconnector, slider, etc.) to be given the status - open, closed, invalid - a BOOL data type or integer variable must be linked in the respective combined element as the main variable.

Example:

- IEC870 driver: Variables with **Typ ID** `T01..T37`
- IEC850 driver: Variables `*/Pos/stVal[ST]`
- DNP3 driver: Input Variables

Pre-requisite: the **DPI/DPC mapping** has not been deactivated in the driver.

Information

For the position of a switch, only the first two bits of the main variable are taken into account.

- The first bit is the actual switching; 0 is OFF and 1 is ON.
- The second bit is the error bit. There is no error if it is 0.

For a source, the status - "present" (ON) / "not present" (OFF) - must also be evaluated using the linked main variable. The internal BOOL data type variables are best suited to this. The source can then be linked to the rest of the topology using a switch or a disconnector (as is common in reality), so that only the position of the switch forwards the color of the source.

Note: for the main variable of a source that is connected to the network by means of a switch/disconnector, such as ground for example, create an **internal driver** variable, configure it with **calculation** `network` and **initial value** `1` ("always present").0 Alternatively, you can also link a source to a process variable directly (the source and its switch in one) if you do not need any **topological interlockings** when switching the source.

STATES

- A switch and a source are switched on (closed) if the value of the linked variable is `1`.
- A switch is invalid if the value of the linked variable is `>1` or has an `INVALID` status bit.
  An invalid switch provides the source number `0` (undefined) at its exit (connection 2) instead of the source number entering. In the direction towards the input the switch behaves as normal.

  Note: if the main variable has the status `INVALID`, the whole subsequent network is `INVALID`, because the status of the network is not known. The status `INVALID` is forwarded using subsequent closed switches.

  Attention

  If, in the individual statuses of the combined element, the color and the fill color from ***ALC*** *is activated, it is not just the line, but also the process-related elements that are colored in Runtime.*

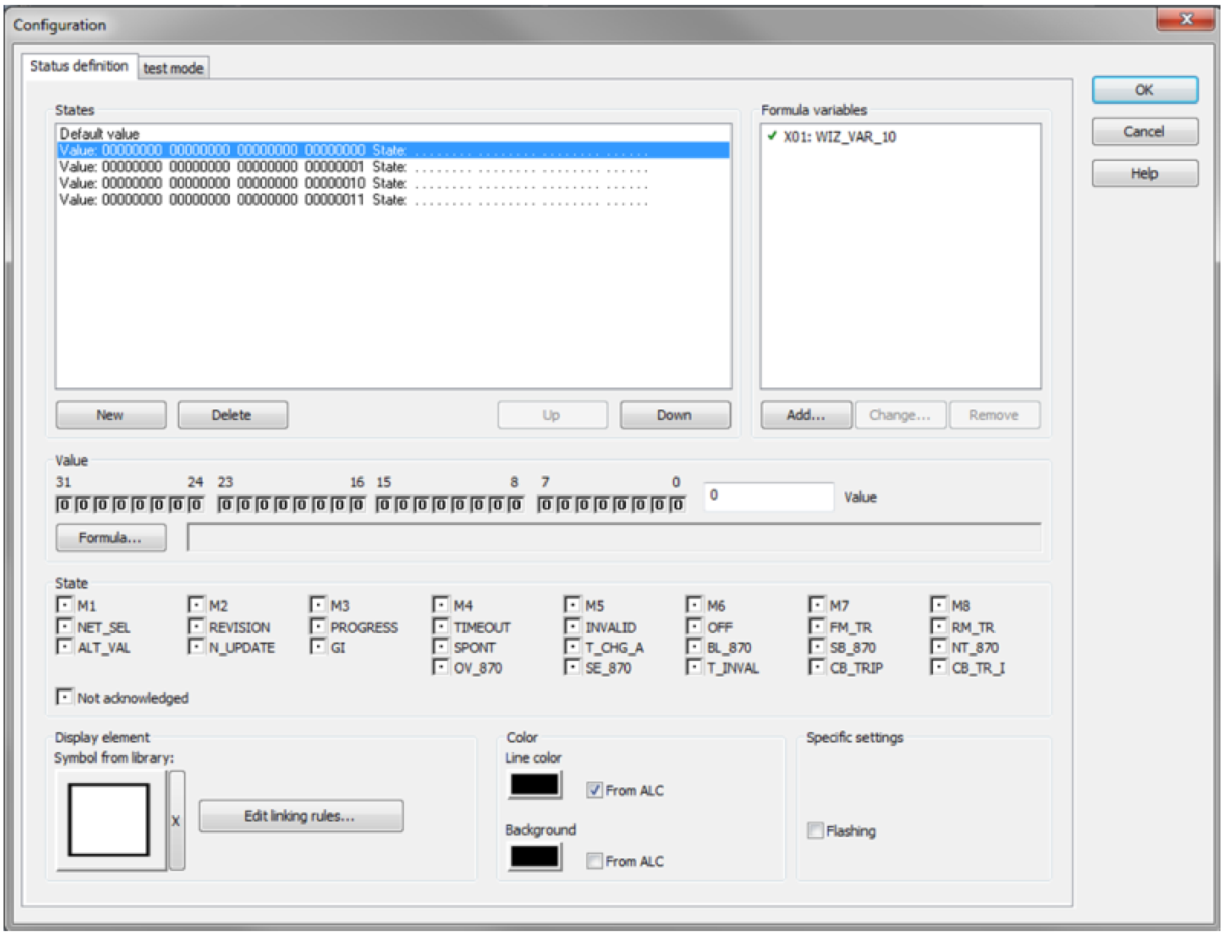
Switch example - colors from ALC

EXAMPLE 1

Combined element with value status `00` and line color from ALC:

1. Configuration in the Editor:
   - Combined element with value status `00`

- Line color from ALC active



Configuration

Status definition | test mode

States

```
Default value
Value: 00000000 00000000 00000000 00000000 State: ........ ........ ........ ......
Value: 00000000 00000000 00000000 00000001 State: ........ ........ ........ ......
Value: 00000000 00000000 00000000 00000010 State: ........ ........ ........ ......
Value: 00000000 00000000 00000000 00000011 State: ........ ........ ........ ......
```

Formula variables

✓ X01: WIZ_VAR_10

Value

31 24 23 16 15 8 7 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 Value

State

M1	M2	M3	M4	M5	M6	M7	M8
NET_SEL	REVISION	PROGRESS	TIMEOUT	INVALID	OFF	FM_TR	RM_TR
ALT_VAL	N_UPDATE	GI	SPONT	T_CHG_A	BL_870	SB_870	NT_870
			OV_870	SE_870	T_INVAL	CB_TRIP	CB_TR_I

Not acknowledged

Display element

Symbol from library:

Color

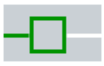
Line color ✓ From ALC

Background From ALC

Specific settings

Flashing

2. Results in the following in Runtime:
    - Source color: green
    - Color without voltage: white
    - Switch status: `off/open` (value `0`)

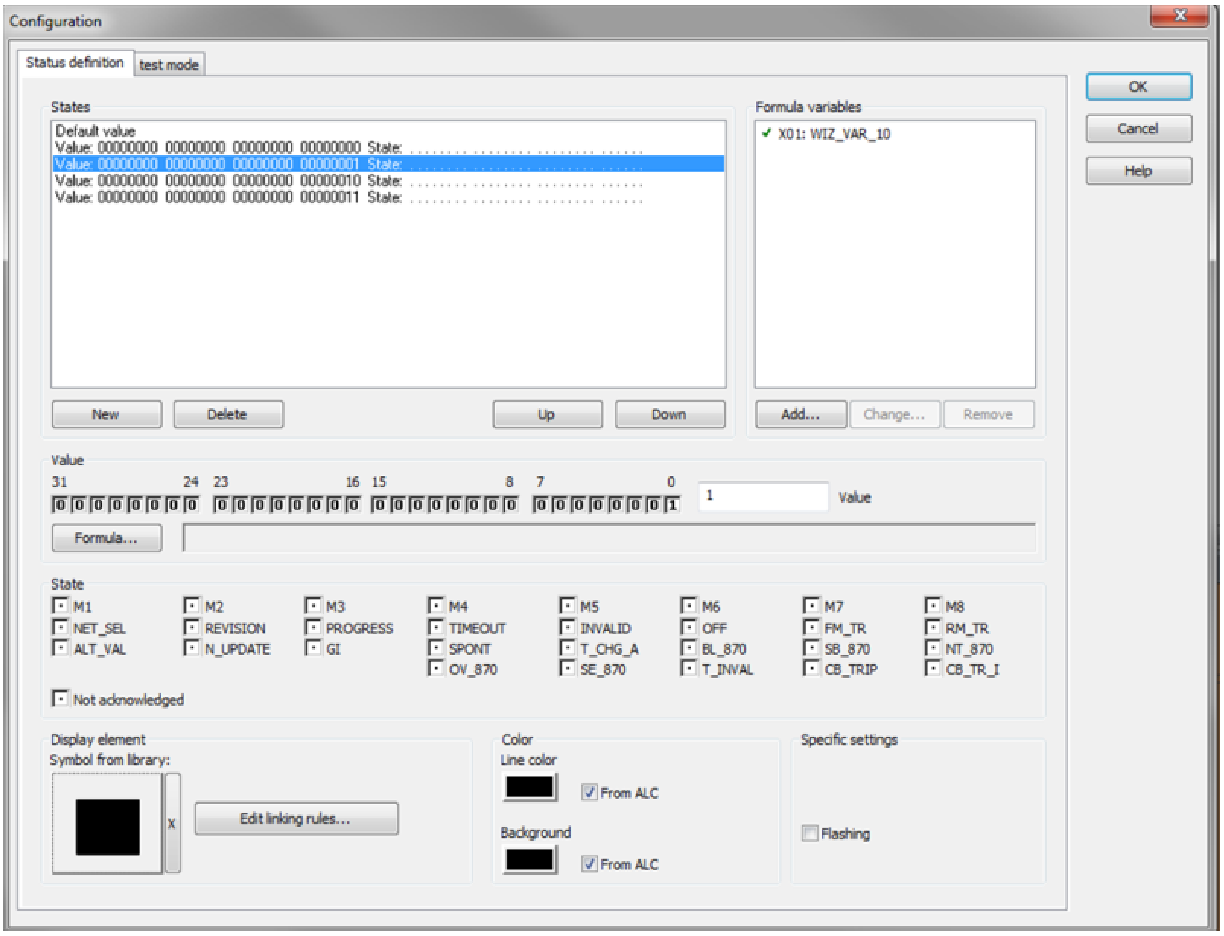


EXAMPLE 2

Combined element with value status `01` and colors from ALC:

1. Engineering in the Editor
    - Combined element with value status `01`
    - Line color from ALC active

- Fill color from ALC active


Configuration

Status definition | test mode

States

```
Default value
Value: 00000000  00000000  00000000  00000000  State: ........ ........ ........ ......
Value: 00000000  00000000  00000000  00000001  State: ........ ........ ........ ......
Value: 00000000  00000000  00000000  00000010  State: ........ ........ ........ ......
Value: 00000000  00000000  00000000  00000011  State: ........ ........ ........ ......
```

Formula variables

✓ X01: WIZ_VAR_10

Value

```
31            24  23            16  15             8  7              0
0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 0   0 0 0 0 0 0 0 1     1   Value
```

State

M1	M2	M3	M4	M5	M6	M7	M8
NET_SEL	REVISION	PROGRESS	TIMEOUT	INVALID	OFF	FM_TR	RM_TR
ALT_VAL	N_UPDATE	GI	SPONT	T_CHG_A	BL_870	SB_870	NT_870
			OV_870	SE_870	T_INVAL	CB_TRIP	CB_TR_I

Not acknowledged

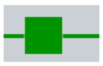
Display element
Symbol from library:

Color
Line color: ✓ From ALC
Background: ✓ From ALC

Specific settings
Flashing

2. Results in the following in Runtime:

   - Source color: `Green`
   - Color without voltage: `White`
   - Switch status: `on/closed` (value`1`)



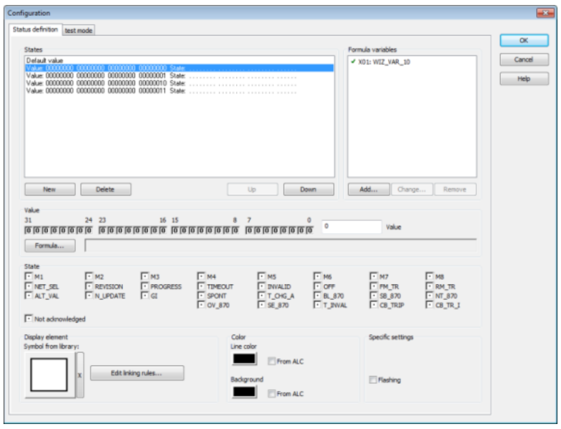
EXAMPLE 3

Combined element with value status `00` without colors from ALC:

1. Configuration in the Editor:

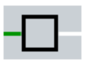
   - Combined element with value status `00`

- Line color from ALC not active



```
Configuration
Status definition | test mode

States
Default value
Value: 00000000 00000000 00000000 00000000 State:
Value: 00000000 00000000 00000000 00000001 State:
Value: 00000000 00000000 00000000 00000010 State:
Value: 00000000 00000000 00000000 00000011 State:

Formula variables
✓ X01: WIZ_VAR_10

Value
31       24 23       16 15        8 7         0
0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0   0   Value

State
M1       M2         M3        M4       M5        M6      M7       M8
NET_SEL  REVISION   PROGRESS  TIMEOUT  INVALID   OFF     FM_TR    RM_TR
ALT_VAL  N_UPDATE   GI        SPONT    T_CHG_A   BL_870  SB_870   NT_870
                              OV_870   SE_870    T_INVAL CB_TRIP  CB_TR_I
Not acknowledged

Display element          Color                      Specific settings
Symbol from library:     Line color   ☐ From ALC    ☐ Flashing
                         Background   ☐ From ALC
```

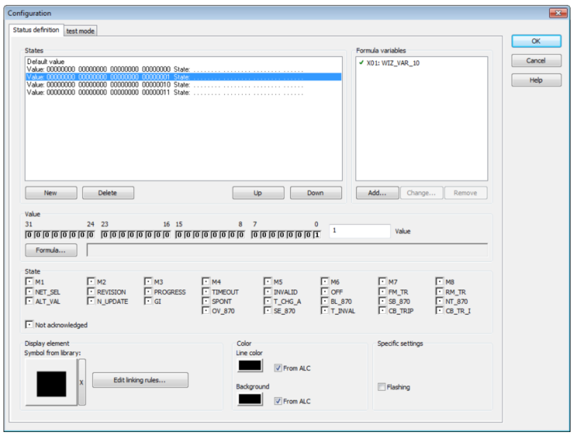
2. Results in the following in Runtime:
   - Source color: `Green`
   - Color not energized and construction color of the line: `White`
   - Defined line and fill color of the combined element: `black`
   - Switch status: `off/open` (value `0`)



EXAMPLE 4

Combined element with value status `01` without colors from ALC:

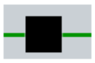
1. Engineering in the Editor
   - Combined element with value status `01`
   - Line color from ALC inactive
   - Fill color from ALC inactive



```
Configuration
Status definition | test mode

States
Default value
Value: 00000000 00000000 00000000 00000000 State:
Value: 00000000 00000000 00000000 00000001 State:
Value: 00000000 00000000 00000000 00000010 State:
Value: 00000000 00000000 00000000 00000011 State:

Formula variables
✓ X01: WIZ_VAR_10

Value
31       24 23       16 15        8 7         0
0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 1   1   Value

State
M1       M2         M3        M4       M5        M6      M7       M8
NET_SEL  REVISION   PROGRESS  TIMEOUT  INVALID   OFF     FM_TR    RM_TR
ALT_VAL  N_UPDATE   GI        SPONT    T_CHG_A   BL_870  SB_870   NT_870
                              OV_870   SE_870    T_INVAL CB_TRIP  CB_TR_I
Not acknowledged

Display element          Color                      Specific settings
Symbol from library:     Line color   ☑ From ALC    ☐ Flashing
                         Background   ☑ From ALC
```

2. Results in the following in Runtime:

- Source color = green
- Color not energized and construction color of the line: `White`
- Defined line and fill color of the combined element: `black`
- Switch status: `on/closed` (value`1`)



Connection points of procedural elements

When configuring, a line is connected to a procedural element (combined element) by overlapping drawings in the screen at connection points of the combined element. Only one line can be connected to the same connection point at the same time. All lines that start within the area defined, are connected (Topology from the graphic).

Attention

Use ALC elements only in un-rotated state because:

The calculation for the topological model for the ALC in the Editor is based on the position of the elements in un-rotated state and without considering any dynamics.

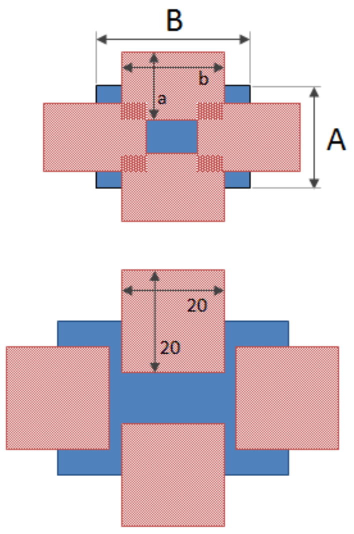
CONNECTION POINTS AND CONNECTION AREAS

- The connection area for a connection point is in the middle of each side of the combined element. Each combined element thus has four connection points.
- The size of a connection area corresponds to 2/3 of the height and width of a combined element, but no more than 20 pixels.
- Each connection area is centered in the middle of the respective element corner and stretches symmetrically inwards and outwards, to a maximum of 10 respective pixels.

Attention

If the combined element is less than 30 pixels, connection areas within an element overlap. Lines that could touch can cause errors (compilation, coloring).

You can see the possible connection points for combined elements smaller and larger than 30 pixels in the illustration.



<u>Colors</u>

- Blue: Combined element
- Red: Connection areas

<u>Dimensions:</u>

- **A**: height of the Combined element
- **B**: width of the Combined element
- **a**: Width of the connection area: 2/3 of **A**, but a maximum of 20 pixels.
- **b**: Length of the connection area: 2/3 of **B**, but a maximum of 20 pixels.

RULES

- If a line is outside the connection area, no connection is detected and there is thus no coloring of the line. So there will also be no coloring for further lines.
- With sources, drains and `Links`, all described connection points can in principle be used.
  Attention: With sources and drains, only one connection point can be used at the same time. If different connection points are used at the same time, undefined states can occur.
  Elements of the type `Link` can also use several connection points at the same time. The incoming color information is passed on to all lines.
- With switches/disconnectors/sliders and transformers, the connection `1` (supply) is on the left or on the top and connection `2` (output) is on the right or on the bottom. This sequence can be changed with the **Switch input/output** property.
  Attention: At switches and transformers it has to be cared, that only one input connection and

one output connection is used. The simultaneous use of several input or output connection points results in inconsistencies and is therefore not reliable.

- For all procedural elements the following is true: Only one line can be connected to a connection point. Junctions cannot be realized directly on an element but must be drawn with lines.

Switch input/output

If a transformer, a disconnector or a switch is configured, the input and output can be swapped. To do this:

1. Select either transformer, disconnector or switch as a **Function type**
2. Activate the checkbox**Switch input/output**

The input is then set at the bottom right and the output at the top left.

OVERVIEW

Device configuration	Input	Output
normal	Left	Right
normal	top	bottom
swapped	Right	Left
swapped	bottom	top

3.1.2 Lines

Lines are represented by the vector elements Line, Polylines and Pipe.

If the option **Color from ALC** is activated for a line, the coloring is defined by the ALC configuration. Lines are automatically colored by the system depending on the status of the procedural elements and the ALC settings.

Here the color usually comes from the highest priority source number of the media flowing through the line, or stays "empty/not energized" just as defined in the screen with static or dynamic colors.

You define the display type by means of drop-down lists:

- Priority for display
- Display multiple supply
- Display secured supply

The following options are available in the properties of the lines:

Parameter	Description
Color from ALC	Activates the automatic line coloring for this vector element. That means: If the source for the line is active and all switches/valves leading from the source to the line are closed/open, the line is accordingly colored. If the line is fed by a single source, the defined source color is used for coloring the line. The line width is not changed.
Priority for display	Defines if `multiple supply`, `secured supply` or both are displayed. Default: `Multiple supply`
`Secured supply`	The element is displayed according to the rules of the secured supply. A line is then considered to have a secure supply if it is supplied by at least two different switches or transformers with a non-system source. System sources do not contribute to secured supply, but do not exclude it.
`Multiple supply`	The element is displayed according to the rules of the multiple supply. A line is considered to have multiple supplies if it is supplied by at least two different sources. In doing do, it does not matter if they are system or user sources and from which side the line is supplied by the sources.
`No priority`	The coloring rules for `multiple supply` and for `secured supply` are applied at the same time if both criteria are met. That means: If a line - has `multiple supplies` and a `secured supply`, - The priority is set to `No priority`, - The display for multiple supply is set to `two sources with highest priority`, - The display for secured supply is set to `double width`, Then the line is twice as wide and displayed as a dashed line in two colors.
display multiple supplies	Multiple supply means that a line is supplied by multiple sources at the same time. Here you can define how lines with multiple supply are displayed. Default: `highest priority source`

`highest priority source`	The line gets the color of the source with the highest priority. **Note:** Priorities correspond to the sequence chosen in the ALC configuration.
`two highest priority sources`	Applies for lines fed by two or more different sources. The two sources with the highest priorities define the coloring. The line is displayed with these two colors (dashed). The dash length can be changed using the **Dashing length supplied multiple times** property. System sources apply for multiple supplies just as with genuine sources and color lines in two colors it they are configured accordingly.
`Alternative color`	The color defined in the **Alternative color** property is used.
Dashing length supplied multiple times	Defines the dash length (in pixels) of lines, polylines or tubes for the dashed ALC coloring for `two sources with the highest priority` for **display multiple supplies**. - Minimum: `0` (automatic dash length) - Maximum: `32767` - Default: `0`
Alternative color	Alternative color for the ALC coloring of lines, polylines or tubes with multiple supplies.
display secured supply	Secured supply means that a line gets multiple supply from one source (parallel). Here you can define how 'secured supply' is displayed. A line is always displayed as having a `secure supply` if it is supplied by at least two switches with a genuine source (not system source). Default: `normal`
`double width`	Relevant for lines fed in parallel by the same source. If this is the case, the line is displayed with double the configured width. (Example: A line with line width 5 pixels is displayed with 10 pixels if secure-fed.) If this line is fed by two or more different sources (multi-supply), the line width does not change! The color is `always` defined by the source with the highest priority!
`double brightness`	Relevant for lines fed in parallel by the same source. The line is displayed with double the original brightness. If this line is fed by two or more different sources (multi-supply), the line color does not change! If this line is multi-fed from one source (secure supply), the line is

	displayed with double the original brightness. Formula for the calculation of the double brightness: 1. The defined RGB color is transformed to the HLS system. 2. L (luminance = brightness) is recalculated with NewLuminance = 240*3/4 + L/4 3. The color value is recalculated to the RGB system with the new brightness. The color is `always` defined by the source with the highest priority!
`normal`	The element is displayed in the color of the source and with the configured width.
Use alias	Active: Alias is used.
Alias	Opens the dialog (on page 40) for selecting a model.

Information

The source color and the priorities of the sources are defined in the project properties.

User-defined sources must have a higher ID than 9. IDs up to 9 are reserved for system sources.

Information

The calculation of the color of a line in the Runtime is done with the following priority list:

1. Automatic Line Coloring (highest priority, overrules all other settings)
2. Dynamic colors
3. Static colors

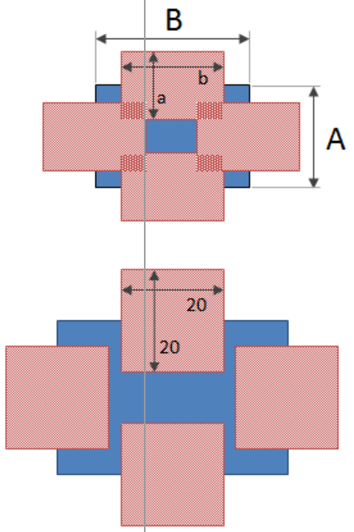
Example

In the following example Source 0 has the color blue and Source 1 has the color red. And Source 0 is the source with the highest priority.

▬ Source 0

▬ Source 1

This results in the following displays for the different options:

	Line / Polyline	Pipe
highest priority source	▬	
two highest priority sources	▬	▬
double width	▬	▬
double brightness	▬	▬

Connection points of lines

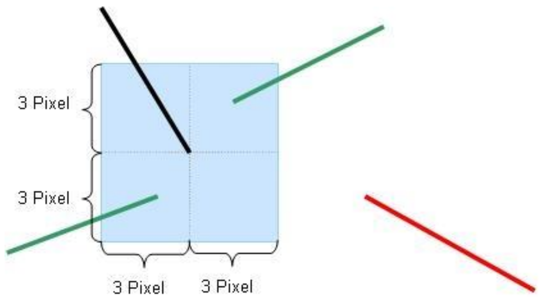
The connection of one line (line, polyline or tube) to another line is done with overlapping drawing in the screen at connection points. The connection points - either connection areas - are at the start and the end of each line and are around 3 pixels large.

Example

The start point of a line has the coordinates (start point x/start point y): 150/100 pixels. This results in a connection area (x / y): 147 - 153 / 97 - 103 pixels.

If the line start or end of this line and that of one or more other lines is within this area, the lines are automatically connected without any further engineering. A mere overlapping of the connection areas of the single lines is not sufficient!

In the following illustration the connection area is displayed graphically (the green lines are connected to the black one, the red line not.



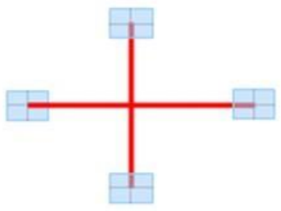
Information

Any number of lines can be connected in a connection area.

Attention

If a line is outside the connection area (e.g. the red line in the illustration), no connection is established and there is no coloring of the line. So there will also be no coloring for further lines.

Line crossings can easily be realized, if the ends of the lines are not in the connection area.



Attention

Use ALC elements only in un-rotated state because:

The calculation for the topological model for the ALC in the Editor is based on the position of the elements in un-rotated state and without considering any dynamics.

3.1.3 Checking the project

Engineer the desired procedural elements and lines in one or more screens and save these screens. Then you can check via **Create all Runtime files** or **Create changed Runtime files** whether there are any errors or conflicts in the screens. If error or conflicts should exist, corresponding error messages or warnings are displayed in the output window.

Information

Double click the corresponding line in the output window. The screen with the erroneous screen element will be opened automatically. If the erroneous screen element is part of a symbol, the corresponding symbol is automatically selected.

The following error message can be displayed.

- ALC: Screen '%s' - Two Link elements with different Link number are connected to line '%s' . (double clicking opens the screen and selects the line.)
- ALC: Screen '%s' - More than two connection points are used at element '%s'. For each element only one input and one output may be used. (double clicking opens the screen and selects the element)

The following warnings can be displayed.

- ALC: Screen '%s' - Alias line '%s' is connected to a no-alias line. (double clicking opens the screen and selects the line.)
- ALC: Screen '%s' - Alias element '%s' is connected to a no-alias line. (double clicking opens the screen and selects the element)
- ALC: Screen '%s' - No-alias element '%s' is connected to an alias line. (double clicking opens the screen and selects the element)
- ALC: Screen '%s' - Line '%s' is only connected on one side. (double clicking opens the screen and selects the line.)
- ALC: Screen '%s' - Element '%s' is not connected. (double clicking opens the screen and selects the element)
- ALC: Screen '%s' - Element '%s' is only connected on one side. (double clicking opens the screen and selects the element)

In the error messages or warnings the corresponding elements are identified using the element reference. This reference also serves as the link key for ALC aliases.

3.2 Configuration

To configure ALC:

1. In project properties, select **ALC configuration** the property in the **Automatic Line Coloring** group
2. Click on the **...** button
3. The dialog for configuration is opened
4. Configure the desired properties for:
   - Sources (on page 28)

     Create new sources - in the desired colors for the topology.

     Note: In doing so, note that the system sources (ID 0..9) already have a pre-defined meaning or are reserved for future versions.

     Note also the principles for coloring for UNDEFINED (on page 32).
   - Interlockings (on page 33)

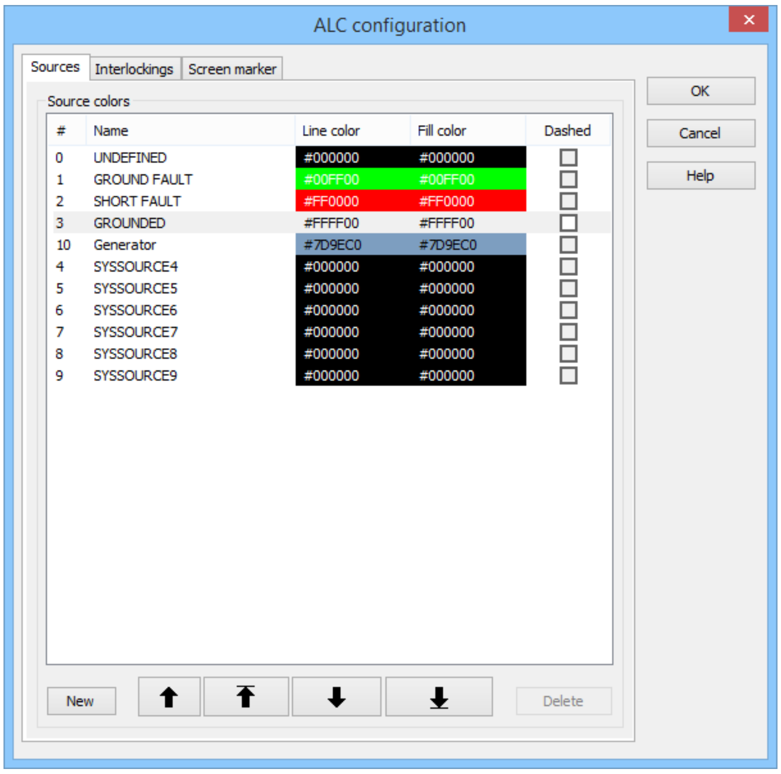
     Configure which **topological interlockings** the **Command Processing** module should take into account.
     Note: the tab is only visible with the topology package license.
   - Screen marker (on page 37)

     Configure the color table for the screen marker for **impedance-based error detection**.
     Note: the tab is only visible with the topology package license.

3.2.1 Configuration of the sources

The sources, e.g. their names and colors (sequence and priority), are configured project-specifically within the project properties under 'ALC configuration'. Sources with ID between `0` and `9` are reserved for system sources. Those that already have a function (such as GROUNDED - the color of the "earth" source) must not have their function changed. Those that do not yet have any functionality in the current zenon version remain reserved for future versions.

The source colors from ID #10 are freely available for the process-related elements, for example source: "Generator" or "110kV" etc. Add further colors for this.



#	Name	Line color	Fill color	Dashed
0	UNDEFINED	#000000	#000000	☐
1	GROUND FAULT	#00FF00	#00FF00	☐
2	SHORT FAULT	#FF0000	#FF0000	☐
3	GROUNDED	#FFFF00	#FFFF00	☐
10	Generator	#7D9EC0	#7D9EC0	☐
4	SYSSOURCE4	#000000	#000000	☐
5	SYSSOURCE5	#000000	#000000	☐
6	SYSSOURCE6	#000000	#000000	☐
7	SYSSOURCE7	#000000	#000000	☐
8	SYSSOURCE8	#000000	#000000	☐
9	SYSSOURCE9	#000000	#000000	☐

SOURCE COLORS

Parameter	Description
Number	Internal unique consecutive number, so that the source can be identified. This number is given by the system automatically and cannot be changed. **Attention:** IDs 0 to 9 are reserved for the system sources and must not be used user-specific.
Name	Logical name for the source (e.g.: 'water' or 'grounded'). This name is also used when selecting the source number for Combined elements. You can change the name by clicking it with the left mouse button. With this edit mode is switched on. The changes are accepted with Enter or by selecting another source. **Note:** The labels are not language switchable.
Line color	Line color of the respective source. This color is used for coloring lines, polylines and as the outside color of tubes.
Dashed	Check box for activation. If active, the line is drawn as dashed. **Note:** This checkbox can only be activated for **GROUNDED**. This check box is grayed out for all other sources.
New	Adds a new color.
Delete	Deletes the selected color.
Upwards (arrow symbol)	Moves selected source up one position.
Fully upwards (arrow symbol)	Moves selected source to the start of the list.
Downwards (arrow symbol)	Moves selected variable down one position.
Fully downwards (arrow symbol)	Moves selected source to the end of the list.

CLOSE DIALOG

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

The colors can be configured directly by entering the corresponding hexadecimal code or by using a color palette.

<u>For direct input:</u>

1. Click on the color description with the left mouse button.

   The field is switched to editing mode.

2. Enter the code.
3. Press the `Enter` key or select another source to apply the change.

<u>To select via a color palette:</u>

1. highlight the desired line.
2. Click on the **...** button behind the color
   Note: The **...** button is only visible if the color entry is selected with a mouse click.

   The color palette is opened in the context menu.

3. select the desired color

The hexadecimal code describes the RGB color value and consists of the following. **#RRGGBB**.

Item	Meaning
#	Identifier to indicate that a hexadecimal color code is used.
RR	2 digits are the red value of the color in hexadecimal system. `0-255` corresponds to `0-FF`.
GG	2 digits are the green value of the color in hexadecimal system. `0-255` corresponds to `0-FF`.
BB	2 digits are the blue value of the color in hexadecimal system. `0-255` corresponds to `0-FF`.

Information

The sequence in this list represents the priority of the sources, with the first element having the highest priority.

To change the priorities of the single sources, they can be moved upwards or downwards using the arrow buttons

Attention

Limitations when deleting the sources and resetting erroneous colorings:

Sources with ID between `0` and `9` are reserved for system sources. They can:

- Not be deleted:
- Not be reset as an erroneous color

Deleting sources

In order for sources to be able to be deleted, they must have an ID from `10`. Only the source with the highest ID can be deleted.

Resetting erroneous colorings

In order for erroneous colorings to be able to be reset once the cause has been rectified, no system source colors can be used. A color for IDs from `10` must be selected.

Coloring mode for UNDEFINED

Coloring in the network can be implemented in two modes with the UNDEFINED status:

- `Standard`
- `Input takes priority`

This setting is made using the **Automatic Line Coloring/Mode for coloring** project property.

STANDARD

The graph search starts with a source and goes through the whole network, so that each closed switch (switch variable has the value `1`) per direction is only gone through once, so no cycles occur. In doing so, each node visited (=line segment) is colored with the source color. The directly-related lines are marked as a node.

If the search finds a switch that has a switch variable with the following status, the UNDEFINED color is used for coloring from this point onwards:

- INVALID [values: any desired],
- is invalid [value: `3`]
- is in intermediate position [value: `2`])

The graph search is now continued in the same form. Each switch is gone through just once per direction with the UNDEFINED color. Therefore each switch can be gone through a maximum of four times per source:

1. with source number in forwards direction,
2. with source number in backwards direction,
3. with UNDEFINED in forwards direction,
4. with UNDEFINED in backwards direction,

INPUT TAKES PRIORITY

With the `Supply takes priority` setting, only lines that have a supply from at least one source but not clearly from any one source are colored as UNDEFINED. If a line is supplied with at least one source, it can no longer receive an UNDEFINED color from another source.

This search is a two-stage search:

- In the first stage, as with `Standard`, the source color is distributed in the network from each switched source, as long as the next switch is closed. The search is ended if the switch is open or invalid/undefined.
- In the second stage, the search is started at each invalid/undefined switch that receives a supply from one side and the UNDEFINED color is distributed to the unsupplied side. This search also considers the switches that are invalid/undefined as closed and thus distributes the UNDEFINED color in the network until it meets a clearly open switch. In addition, a search is ended if a line element is reached that is already supplied.

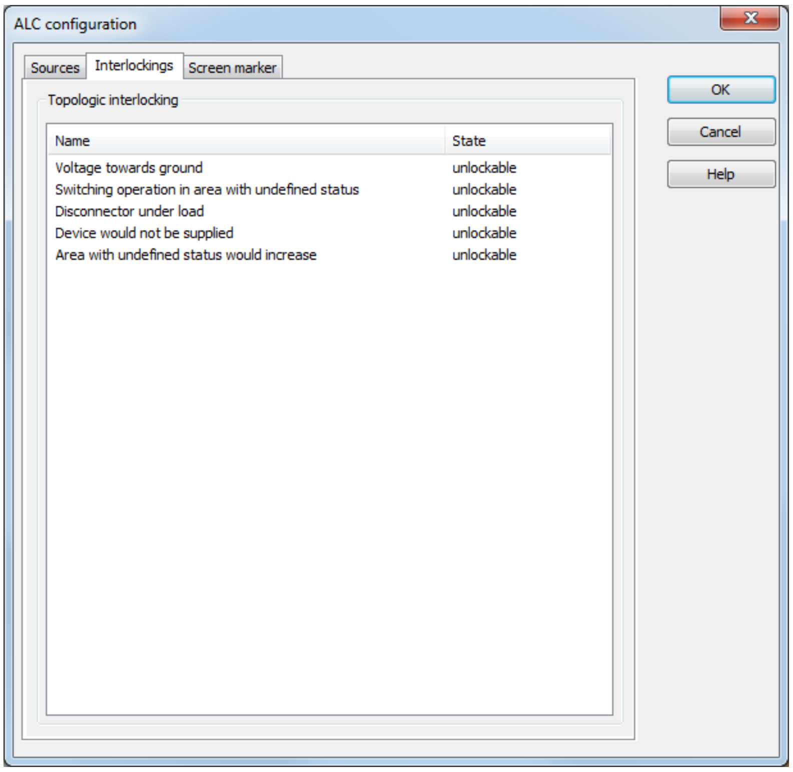
3.2.2 Configuration of topological interlockings

In Runtime, the **Command Processing** modules can calculate the interlockings that result from the dynamic status of the electrical network, the topology of which was configured with **ALC**, from scratch. Using the topology of the network configured with ALC and current statuses of the sources (`ON` / `OFF`), switches, disconnectors etc. can automatically detect the **command input** that will result in the execution of a command, for example "Voltage to ground" and then prevent the execution of the command.

The topological interlockings from the **ALC** for **command input** are configured centrally - for the respective project. In doing so, a decision is also made as to whether a user can unlock an interlocking (provided they also have the **authorization level for unlocking** for the action).

Information

This dialog is only available when both the Energy Edition and the ***Automatic Line Coloring*** *modules are licensed.*



Name	State
Voltage towards ground	unlockable
Switching operation in area with undefined status	unlockable
Disconnector under load	unlockable
Device would not be supplied	unlockable
Area with undefined status would increase	unlockable

The settings made here apply globally, for the whole Topological Model. The following conditions are available:

Parameter	Description
Voltage towards ground	Interlocking is active if a `switch`/`disconnector` is to be closed and a grounded potential is connected to its first connector and its other connector is connected or undefined.
Switching action in an area with an undefined status	Interlocking is active if a `switch`/`disconnector` is to be closed and both of its connectors are 'undefined' or 'disturbed'.
Disconnector under load	Interlocking is active if certain conditions have been met for switching the `disconnector` on or off. Conditions: See "Disconnector under load - interlocking conditions (on page 36)" section.
Device would not be supplied	Interlocking is active if a `switch`/`disconnector` is to be opened and a device that is switched on and supplied with voltage from a `source` (`drain`) then loses supply.
Area with undefined status would increase	Interlocking is active if a `switch`/`disconnector` is to be closed and one connector is 'undefined' or 'disturbed' and the other not. It is also reported if the `command` has been configured with the **switching direction** `none`.

If you click in the **Status** column in one of these interlockings, a drop-down list opens with three choices:

Parameter	Description
do not check	The selected condition is not considered in this project (topological model).
unlockable	The selected condition is considered in this project. If the condition applies, the user can unlock it with the Command Processing (in the screen of type **Command Processing**). This unlocking action is logged in the **Chronological Event List**.
not unlockable	The selected condition is considered in this project. The user cannot unlock it.

EXCEPTION TOPOLOGICAL INTERLOCKING

The topological interlocking is not carried out if:

- the variable of a switch has the status Revision
  or
- the variable is manually corrects or set to **Alternate value** and with this is set to the same variable value as the initial value; in other words if the switch:
  - Is set to OFF and then it is manually corrected to OFF or replaced.
  - Is set to On and then it is manually corrected to ON or replaced.

Disconnector under load - interlocking conditions

For the **disconnector under load** topological interlocking, a disconnector can be switched if one of the following conditions is met for the line segments that connect the disconnectors:

WHEN TURNING THE DISCONNECTOR ON:

A check is carried out to see whether the topology before switching to `ON` is in one of the following states:

- Both line segments are supplied/grounded by the same source;
- One line segment does not receive any voltage and the other line segment is grounded;
- A line segment is not under load.

WHEN TURNING THE DISCONNECTOR OFF:

A check is carried out to see whether the topology after switching to `OFF` is in one of the following states:

- Both line segments are supplied by the same source;
- One line segment stops receiving voltage, the other line segment is grounded;
- A line segment stops being under load.

Information

Meaning of "not under load"

The status `not under load` means:

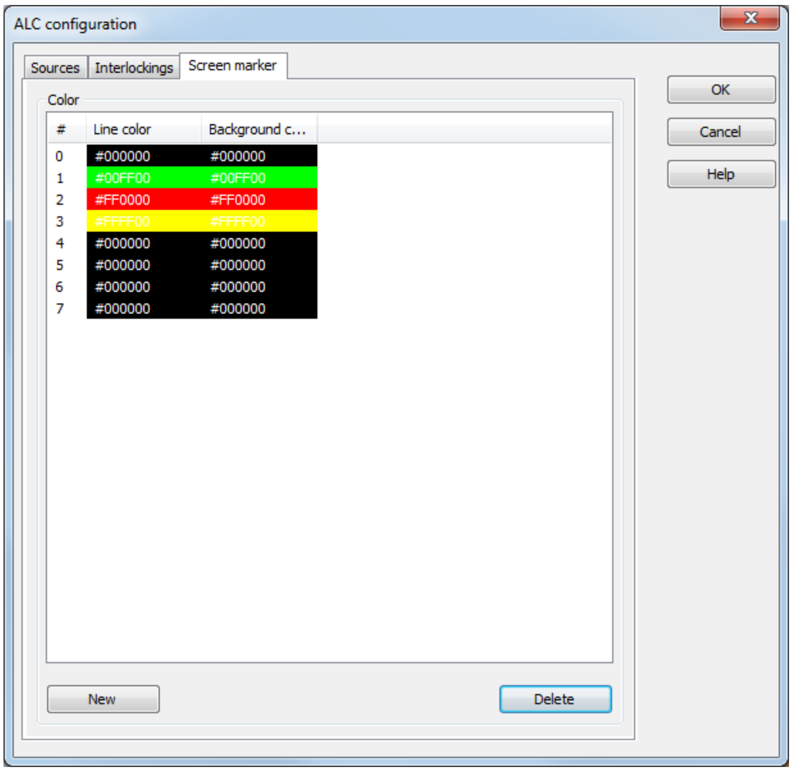
- Either:
  All switches and disconnectors connected to the line segment are open.
- Or:
  Switches and disconnectors connected to the line segment are closed but only connect to one additional segment that is also not under load.

In addition, all of the following conditions must be met for the status of `not under load`:

- All sources and consuming devices connected to the line segment are switched off.
- No transformer may be connected to the line segment.
- It must not be a line that is only connected to this disconnector (one open line).

3.2.3 Configuration of the screen marker

Here you configure the color table for the color marker for the impedance-based error detection and calculation of load distribution (on page 59). See also: **AddMarker**.



#	Line color	Background c...
0	#000000	#000000
1	#00FF00	#00FF00
2	#FF0000	#FF0000
3	#FFFF00	#FFFF00
4	#000000	#000000
5	#000000	#000000
6	#000000	#000000
7	#000000	#000000

Parameter	Description
Number	Unique internal serial number for clear assignment. This number is given by the system automatically and cannot be changed.
Line color	Line color of the screen marker.
Fill color	Fill color of the screen marker.
New	Adds a new color.
Delete	Deletes the selected color. **Note:** Only the last color in the list can be deleted. Standard colors cannot be deleted.

The colors can be configured directly by entering the corresponding hexadecimal code or by using a color palette.

For direct input:

1. Click on the color description with the left mouse button.

   The field is switched to editing mode.

2. Enter the code.
3. Press the `Enter` key or select another source to apply the change.

To select via a color palette:

1. highlight the desired line.
2. Click on the **...** button behind the color
   Note: The **...** button is only visible if the color entry is selected with a mouse click.

   The color palette is opened in the context menu.

3. select the desired color

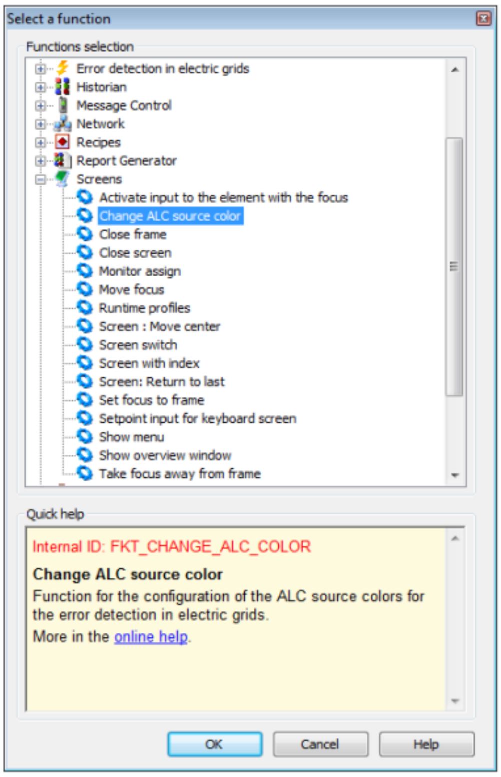
The hexadecimal code describes the RGB color value and consists of the following. **#RRGGBB**.

Item	Meaning
#	Identifier to indicate that a hexadecimal color code is used.
RR	2 digits are the red value of the color in hexadecimal system. `0-255` corresponds to `0-FF`.
GG	2 digits are the green value of the color in hexadecimal system. `0-255` corresponds to `0-FF`.
BB	2 digits are the blue value of the color in hexadecimal system. `0-255` corresponds to `0-FF`.

3.3 Function: Change ALC source color

The foreground and background color of an ALC source can be temporarily changed for the coloring in Runtime using the **Change ALC source color** function. The change remains until Runtime is ended, reloaded or the function is executed again. To create the function:

- select New Function
- Navigate to the Screens node
- select **Change ALC source color**



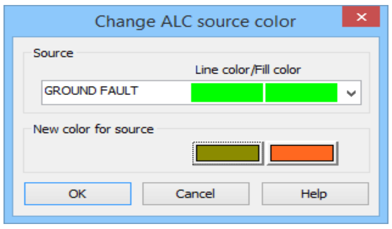
Internal ID: FKT_CHANGE_ALC_COLOR

Change ALC source color

Function for the configuration of the ALC source colors for the error detection in electric grids.

More in the online help.

- The dialog to define line colors and fill colors opens
- define the desired color



Property	Function
Source	Drop-down list to select the source and display the colors currently assigned. These colors cannot be changed here.
New color for source	Click on the color and a dialog opens to select a color.

3.4 Alias for detail screens

To display individual screens, a partial area can be taken from the topological network and displayed individually by means of alias. The screen elements in the detail screen are not included in the topological model, but do however get their ALC colors from the model. They relate to an alias of the screen elements in the overall screen.

Attention

Aliases are only valid within a project.

This means that for symbols that contain links to aliases:

If the symbol is added to the ***general symbol library*** *or the* ***library in the global project*** *and edited there, all ALC alias information is lost without notice!*

CREATE ALIAS

Aliases can be created for the elements:

- Line
- Polyline
- Pipe
- Combined element

Attention

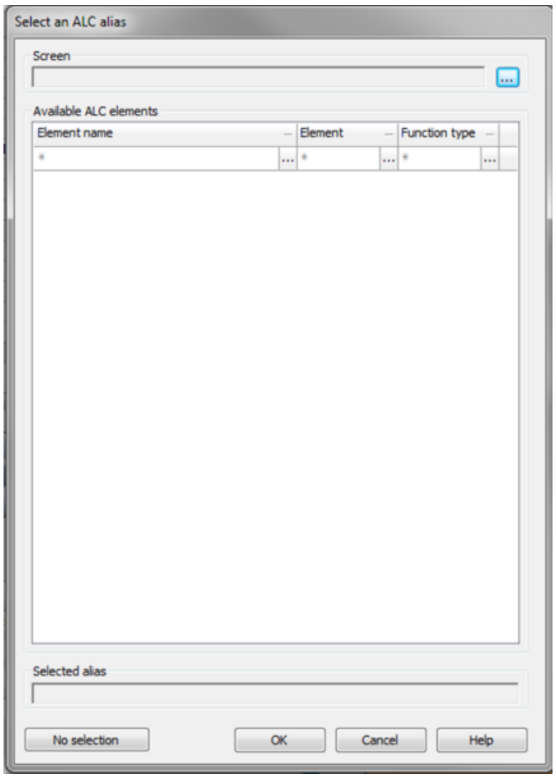
An ALC alias cannot be created if a period (**.**) is contained in the name of the selected screen.

Solution: Replace the period in the screen name with a different character, such as an underscore for example (_).

To create a source element as an alias:

- Activate it in the element's properties **Use alias**.

  To do this, ALC must be licensed and the **Color from ALC** property active.
- Click on the ... button in the **Alias** property
- The dialog to select the element opens.



Parameter	Description
Screen	Click the ... button and a dialog opens to select a screen.
Available ALC elements	Shows the elements that belong to a screen with the element name, type of element and function type. Clicking on an element selects an alias. **Filter** The elements can be sorted according to all columns. When setting a filter, the options offered from all other filters are reduced to values that can be sensibly combined. ▸ **Name**: Input of a standard search term with wild cards (*). The last 12 search terms are offered in the list until the Editor is ended. ▸ **Element**: Select from drop-down list. ▸ **Function type**: Select from drop-down list. Clicking on **...** opens saved search or drop-down list. If a filter is active, clicking on the **X** deletes the filter.
Selected alias	Shows the selected element in the field of **Available ALC elements**.
No selection	Removes selected element.
OK	Saves selection and closes dialog.
Cancel	Discards changes and closes dialog.
Help	Opens online help.

Information

When selecting an element for a new alias, only elements and screens from the same project that the alias was defined in can be selected. Elements from subprojects or parallel projects are not available.

REPLACING ALIAS NAMES

Aliases can be changed when switching screens with Replace link. A detail screen can therefore be displayed with the data from different equipment parts, for instance lines or partial networks. Alias names are replaced along the lines of variables and functions. It is also possible to replace in elements that are used in symbols. The same dialog as is opened for the target as the **Alias** property.

Note: Substitution using index variables is not possible.

3.5 Fault locating in electric grids

Error detection marks grid parts that are subject to ground faults or short circuits by means of special colors in ALC. Starting points for error detection are called ground fault or short circuit reporters (such as a protective device) that are assigned to a circuit breaker. It is assumed that the ground fault and short circuit reporters are always at the output of the circuit breaker element. For this reason, the corresponding variables (with reports from the protective device) are linked to ALC `switch` elements.

The reports from protective devices are shown by means of special coloring - with the source colors ID 1 and 2, but only if the report is received whilst the lines are live. At the same time as this, the reports are set to the additional variables for display. Graphic error displays can thus also be displayed in the screen, for example with further combined elements that are only visible if there is a display active.

The display must be reset manually (acknowledged) once the protective devices have retracted the reports.

Information

This function is only available when both the "Energy Edition" and the "Automatic Line Coloring" modules are licensed.

ERROR DETECTION

Error detection runs locally on each computer in the zenon network. Each client in the network has its own independent model and can therefore search for ground faults and short circuits in different parts of the topology.

Error detection in the electrical network is divided into:

- Search for ground fault (on page 44)
- Short circuit search (on page 51)

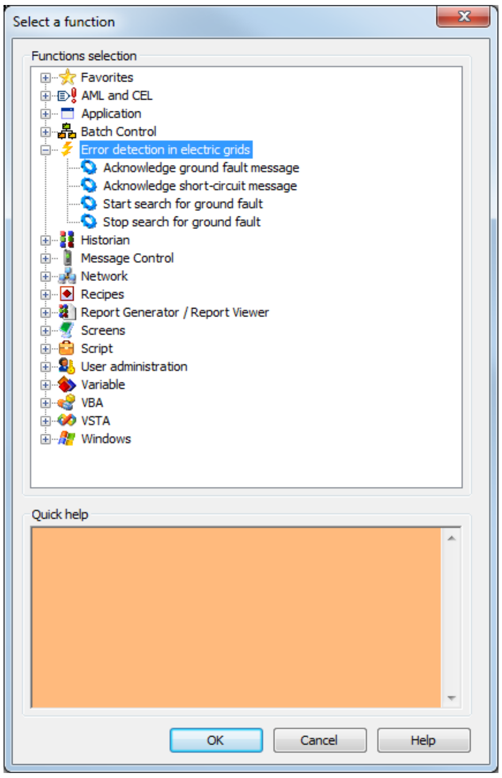
To configure error detection

- You require a license for ALC and zenon Energy Edition
- configure the appropriate screens
- Configure (on page 9) ALC to the corresponding combined elements with the `switch` function
- configure (on page 20) the lines so that they are colored by ALC

Special functions are available in Runtime for error detection:

- Start search for ground fault (on page 48)
- acknowledge (on page 49) ground fault message (on page 49)
- Stop search for ground fault (on page 50)

- Acknowledge short-circuit message (on page 53)



COLORINGS

Errors can be displayed with special coloring of the lines in the ALC if the notifications are received whilst the lines are live. In Runtime, the color assigned by ALC changes automatically as soon as the status of the line changes. The colorings configured can be changed in Runtime via the Change ALC source color (on page 39) function.

Messages are processed in the order in which they arrive. In the event of conflicts

- The colors for displaying errors take priority
- short circuit messages have priority over ground fault messages

3.5.1 Search for ground fault

The search for a ground fault serves to highlight the network parts that may have a ground fault by coloring these. The color is taken from the engineering of ALC source colors (on page 28) for the `GROUND FAULT` (ID 1) source. At the same time as this, the notifications are set to the additional variables for graphical **display**.

The network parts that may have a ground fault are derived from the ground fault reports from ground fault detection devices (ground indicators, protective device that records ground faults). The following is applicable for ground faults:

- Each device can have one to three ground fault reports.
- Ground fault reports are handled either by permanent message processing or by wiper message processing.
- For directional ground fault detection devices, the direction can be lagging or leading in relation to triggering.
  - Leading: First the notification of the direction comes in **forwards** and/or **backwards**, then via the **trigger**.
  - Lagging: First comes the **trigger**, then the solution.

> **Information**
>
> *A network component that may have a ground fault is then no longer considered to have a ground fault if this has been successfully connected.*

ENGINEERING

To configure a search for a ground fault:

1. assign the combined element that represents the switching element to the **Function type** switch (on page 46)
2. Define the mode of search for ground fault (on page 45), ground fault trigger (on page 47) and ground fault display (on page 47).
3. Create the functions for start search for ground fault (on page 48), acknowledge ground fault report (on page 49) and end search for ground fault (on page 50)

> **Information**
>
> *In order to also be able to limit ground faults in mixed networks, only one area with ground faults is searched per path, starting with a source.*

Mode of the search for ground faults

The short circuit search can either:

- color the network part potentially subject to a short circuit
  or
- the whole network where the short circuit is located

The coloring mode is defined via the **Mode of the search for ground faults** property.

To configure the property:

- navigate to the **Automatic Line Coloring** node in properties
- select the desired mode in the **Mode of the search for ground faults** property drop-down list
  - Color grid part: colors only the grid parts that are potentially subject to a short circuit
  - Color whole grid: colors in the whole linked grid where the short circuit is located

This setting can be changed in Runtime via the zenon API object model. In doing so, the short circuit search is recalculated once again.

Ground fault detection type

The direction and type of message processing for the combined element are determined by means of the **Type** setting. For project configuration:

1. navigate to the **Automatic Line Coloring** node in the combined element properties
2. open the **Ground fault recognition** node
3. Select the desired type with direction and type of message processing in the **Type** property
   - Direction:
     indicates if the raising edge of trip alarm or if the raising edge of a direction comes before
   - leading: The current direction status is used for the raising edge of the trip alarm
   - lagging: after a raising edge of the trip alarm, the first raising edge of a direction is waited on; if this does not occur within 2 seconds, the earth fault device is considered non-directional
   - Information processing:
     states which information can be processed
   - none: normal switch; information is not processed
   - Permanent message processing: Newly received messages are considered a new ground fault trip
   - wiper message processing: Messages that are received during a current Search (on page 48) are suppressed

Note: The distinction between permanent message processing and wiper message processing is only how the message is processed, not its type. Wiper message processing thus does not need to relate to a wiper bit.

Attention

To suppress intermittent ground faults, ground fault messages that occur in intervals of less than 2s are ignored.

Ground fault display

The variable linked at **Display** is an output variable for error detection and displays the recorded status of the ground fault identification device. This is necessary because all messages remain saved internally until they are acknowledged, i.e. they do not necessarily conform to the current status of the message variables.

Each time a recording is made, a set value is sent to this variable. In doing so, the values are as follows:

Value	Meaning
0	no ground fault
1	ground fault forwards
2	Ground fault backwards
3	non-directional ground fault
4	Error status - > both directions have activated

Information

To reduce problems in network operation, the variable linked here should be a linked variable.

Earth fault triggering

The alarm to report an earth fault is defined by the **Triggering** variable It can contain information on the presence of an earth fault and the direction of the earth fault from the point of view of the earth fault recognition device. In doing so, a distinction is made between:

- non-directional earth fault alarms
- Directional earth fault alarms with a trip alarm
- Directional earth fault alarms with a trip alarm

To configure the variable for the **Triggering**:

1. navigate to the **Automatic Line Coloring** node in the combined element properties
2. open the **Ground fault recognition** node
   a) for non-directional earth fault alarms
      Click on the ... button in the **Triggering** property

select the variable you wish to import in the dialog that opens

The properties for the direction remain empty

b) for directional earth fault alarms with a trip alarm

link the variable with **Triggering** and add the appropriate direction:

Forwards: link a variable to the **Forwards** property

Backwards: link a variable to the **Backwards** property

c) for directional earth fault alarms without a trip alarm

Link the variable with the corresponding direction:

Forwards: link a variable to the **Forwards** property

Backwards: link a variable to the **Backwards** property

The **Triggering** property remains empty

Note: If you address a directional identification device with **Forwards** in both directions, this is then considered erroneous and ignored.

Start search for ground fault

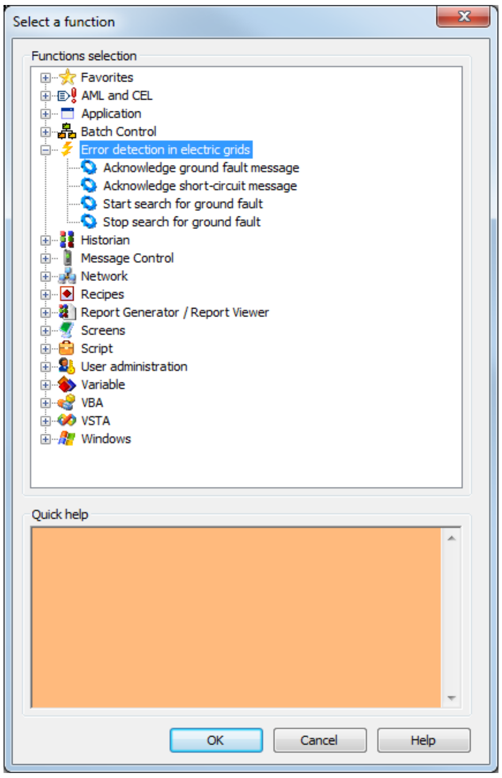
The function **Start search for ground fault** serves to localize a ground fault and has two effects in Runtime:

1. Fault reports from all ground fault identification devices that were configured with wiper message processing are ignored.
2. The search algorithm is changed: Switch actions can only reduce the area subject to a ground fault further. Newly received messages do not therefore increase the area potentially subject to a ground fault.

To configure the **Start search for ground fault** function:

- create a new function
- navigate to the error detection node in the electrical network

- Select the **Start search for ground fault** function



- link the function to a button

Acknowledge ground fault message

With the **Acknowledge ground fault message** function, an internally recorded ground fault from a ground fault indication device can be acknowledged. In doing so, the internally-latched ground fault status is reset if the status is still pending, or highlighted as acknowledged. A recorded ground fault message is only deleted internally if this has been acknowledged and is no longer pending.

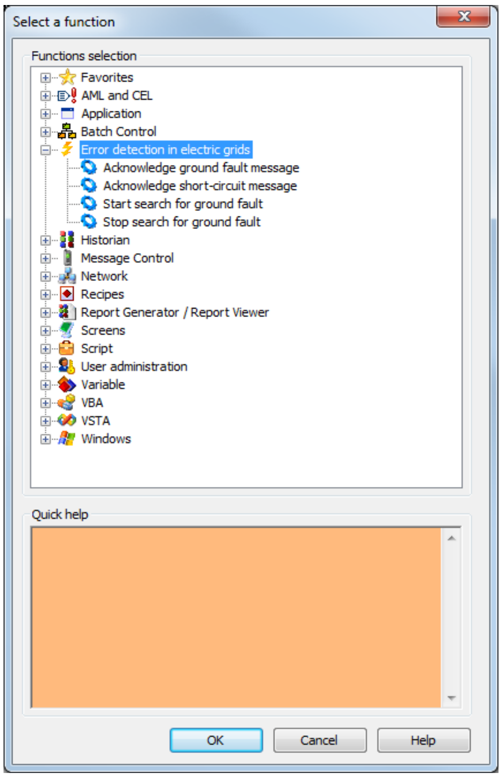
Rules when acknowledging:

- If a variable that corresponds to a triggering or direction variable of a ground fault recognition device is linked, this special ground fault message is acknowledged.
- If no variable has been linked, all ground fault messages are acknowledged.
- Acknowledgment can also take place via the zenon API object model.

To configure the **Acknowledge ground fault message** function:

- create a new function
- navigate to the error detection node in the electrical network

- Select the **Acknowledge ground fault message** function



- the dialog to select a variable opens
- link the desired variable to the function
- link the function to a button

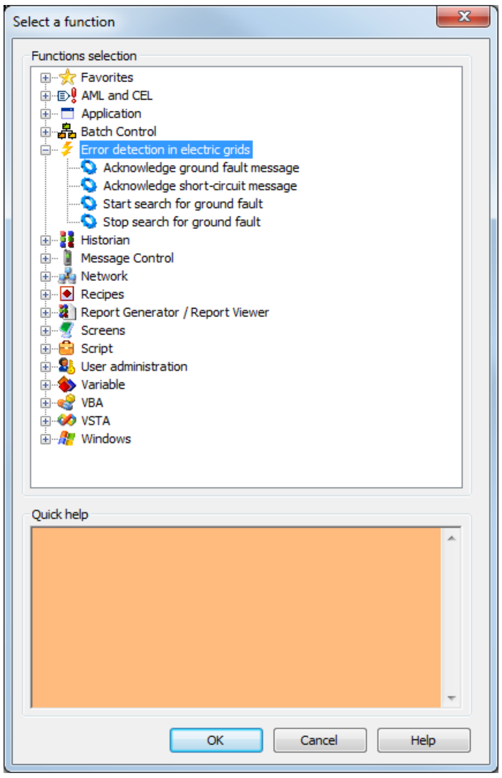
Stop search for ground fault

You end the ground fault search with the **Stop search for ground fault** function in Runtime.

To configure the function:

- create a new function
- navigate to the error detection node in the electrical network

- Select the **Stop search for ground fault** function



- link the function to a button

3.5.2 Short circuit search

The short circuit search serves to highlight the network parts that potentially have a short circuit by coloring these. The color is taken from the configuration of ALC source colors for the `SHORT FAULT` source.

The network parts that are potentially subject to short circuits are deduced from short circuit reports. A short circuit identification device (short circuit indicator, protective device) can have one to three short circuit messages. For directional short circuit indication devices, the direction can be lagging or leading in relation to triggering. A network component that potentially has a short circuit is then no longer considered to have a ground fault if this has been successfully connected.

ENGINEERING

To configure the short circuit search:

1. assign the combined element that represents the switching element to the **Function type** switch (on page 52)

2. Define ground fault display (on page 52) andtriggering of ground fault detection (on page 53)
3. Set up the function for acknowledgment of ground fault message (on page 53)

Ground fault detection

The direction and type of message processing for the combined element are determined by means of the **Type** setting. For project configuration:

1. navigate to the **Automatic Line Coloring** node in the combined element properties
2. open the **Short-circuit detection** node
3. Select the desired type in the **Type** property
   - `Direction:`
     indicates if the raising edge of trip alarm or if the raising edge of a direction comes before
   - `Leading:`
     The current direction status is used for the raising edge of the trip alarm
   - `lagging:`
     after a raising edge of the trip alarm, the first raising edge of a direction is waited on; if this does not occur within 2 seconds, the short circuit device is considered non-directional
   - `Information processing:`
     states which information can be processed
   - `none:`
     normal switch; information is not processed
   - `Permanent message processing:`
     Newly received messages are considered a new ground fault trip

Ground fault display

The variable linked for **Display** is an output variable for error detection and displays the recorded status of the ground fault detection device. This is necessary because all messages remain saved internally until they are acknowledged, i.e. they do not necessarily conform to the current status of the message variables.

Each time a recording is made, a set value is sent to this variable. In doing so, the values are as follows:

Value	Meaning
0	No short circuit
1	Short circuit forwards
2	Short circuit backwards
3	Non-directional short circuit

Ground fault detection triggering

The variable for the message from the short circuit identification device is defined by the **Triggering** variable It can contain information on the presence of a short circuit and the direction of the short circuit from the point of view of the ground fault recognition device. In doing so, a distinction is made between:

- non-directional short circuit reporters
- directional short circuit reporters with a trip alarm
- directional short circuit alarms with a trip alarm

To configure the variables for:

1. navigate to the **Automatic Line Coloring** node in the combined element properties
2. open the **Short-circuit detection** node
   a) for non-directional short circuit detection devices

      Click on the ... button in the **Triggering** property

      select the variable you wish to import in the dialog that opens

      The properties for the direction remain empty

   b) for directional short circuit detection devices with a trip alarm

      link the variable with **Triggering** and add the appropriate direction:

      Forwards: link a variable to the **Forwards** property

      Backwards: link a variable to the **Backwards** property

   c) for directional short circuit detection devices without a trip alarm

      Link the variable with the corresponding direction:

      Forwards: link a variable to the **Forwards** property

      Backwards: link a variable to the **Backwards** property

      The **Triggering** property remains empty

Acknowledge short-circuit message

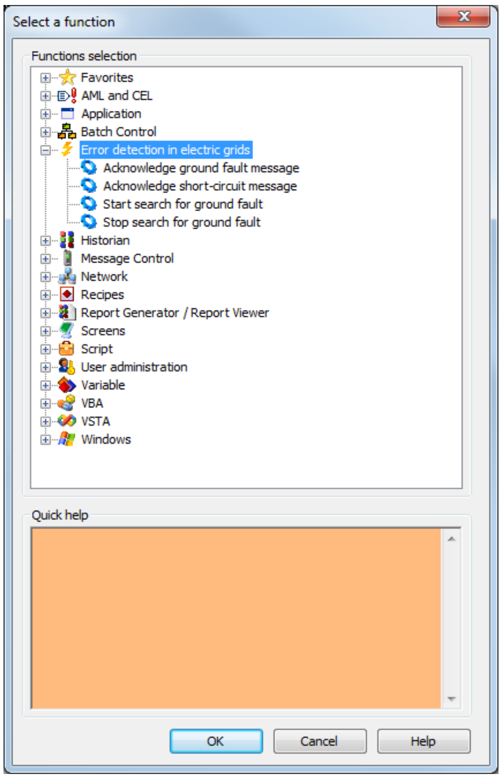
With the **Acknowledge short-circuit message** function, an internally recorded short circuit from a short circuit indication device can be acknowledged. In doing so, the internally-latched ground fault status is reset if the status is still pending, or highlighted as acknowledged. A recorded short circuit message is only deleted internally if this has been acknowledged and is no longer pending.

Rules when acknowledging:

- If a variable that corresponds to a triggering or direction variable of a short circuit recognition device is linked, this special short circuit message is acknowledged.
- If no variable has been linked, all short circuit messages are acknowledged.
- Acknowledgment can also take place via the zenon API object model.

TO CONFIGURE THE ACKNOWLEDGE SHORT-CIRCUIT MESSAGE **FUNCTION:**

- create a new function
- navigate to the error detection node in the electrical network
- Select the **Acknowledge short-circuit message** function



- select the variable you wish to import in the dialog that opens
- link the function to a button

3.6 Impedance-based error detection and calculation of load distribution

Impedance based error detection and calculation of load distribution expands ALC. Whereas ALC identifies nodes and beams, this model also detects lines and their parameters. Fault locating from protection is possible by means of configuration in the zenon Editor.

The model provides properties and methods for external evaluation of the fault location and load distribution via API.

PROPERTIES FOR ALC AND THE EXTENDED TOPOLOGICAL MODEL

The ALC elements **combined element** and **line** (line, polyline, tube) have special properties for impedance-based fault location and to calculate the load distribution. The properties for load distribution are not evaluated in zenon, but are available via the zenon API as algorithms to be created by users.

The simple topological model for the coloring is supplements by an expanded topological model that includes all lines as separate beams. The extended topological model is stored as **ALC.xml** and can be read by external applications this way. **ALC.xml** contains two sections:

- **GraphElements**:
  contains the extended topological model without aliases
- **GraphAliases**:
  contains only the aliases

3.6.1 Impedance-based fault location of the short circuit

With impedance-based fault location, an error marker is set at the location of the problem in the topology. The impedance values measured by protective devices are evaluated by the **ALC** module. Based on the topology, the error markers are positioned in the screen correctly in a zenon screen.

If a short circuit occurs and the reactance is not equal to zero, the search for the location of the short circuit starts:

- Short circuit:
  Reported by a linked variable for the **Triggering** property
- Reactance:
  Value of the variable (from the REAL data type), that is linked to the **Reactance value from protection** property.

POSITION OF THE MARKER

All lines are run through in the corresponding direction. The direction results from negative or positive reactancy. The respective reactancy of the line run through is deducted and the search continues until the residual reactancy is less than the reactancy of the next line. A marker is drawn in the line. The position of the marker corresponds to the residual reactancy.

If there is no reactancy value, no marker is set in the event of a short circuit notification. In order for the marker to be drawn correctly, the area must not be under load during the short circuit notification. With lagging short-circuit notifications, the reactancy is only evaluated if the notification of direction has been received or the timeout of 2 seconds has expired.

The search is canceled if an open shift element or another ALC element has been found. Each part of the network and each individual line therein must only run once per trigger, there are thus less markers that occur in the line network than would be possible.

When reloading, markers that already exist are drawn at the same point as before reloading. Changes to the configuration of the fault locating are only evaluated after another short circuit.

If a short circuit notification is removed and acknowledged, all markers of this short circuit trigger are deleted.

Note: Depending on the order of the rectification of the short circuit and switching on again, marker can remain drawn in, although the line is no longer colored as a short circuit.

Engineering in the Editor

With impedance-based fault location, an error marker is set at the location of the problem in the topology. The location is calculated from impedance, on the basis of the topology.

To configure the impedance-based fault location in the zenon Editor, carry out the following steps:

1. Activate impedance-based fault location:
    a) To do this, click on the project in your **Workspace**.
    b) Click on the **Automatic Line Coloring** project property group.
    c) Activate the **Fault location based on impedance** property.
2. Configure the display of the screen markers with the project properties:
    a) **Screen marker size**
    b) **Line width of the screen marker**
    c) **Display type of the screen marker**
3. Create a zenon screen.
4. Position the **combined element** on the zenon screen.
   The variable selection dialog is opened.
5. Configure the ALC settings for the combined element:
    a) Ensure that the combined element has been selected.
    b) Switch to the **Automatic Line Coloring** property group.
    c) In the **Function type** property, select the `Switch` entry from the drop-down list.

d) Link the **Reactance value from protection** property (in the **Fault location from protection/load distribution** properties section) to a `REAL` data type variable with the value of the measured impedance.

e) Select the type of **Short-circuit detection** in the drop-down list of the **Type** property.

f) Configure the color of the marker in the **Marker color** property.

3.6.2 Expanded topological model

Each object has a unique ID, via which it is referenced in the file. The attributes correspond to a subset of the zenon screen elements that have created the elements.

GRAPHELEMENT

ID	Description
Picture	Screen name
ElementID	Screen element ID
ElementRef	Screen element reference
Type	Screen element -type (see "element")
SourceID	Source number
ReverseSourceID	Source name in reverse direction
Variable	Status variable
VarProtReact	Reactance variable
MaxIType	Type of maximum current
MaxIVal	Maximum current constant value
VarMaxI	Maximum current variable
VarCurI	Instantaneous current variable
VarCalcI	Calculated current variable
VarCurP	Instantaneous power variable
LoadType	Type of load
LoadVal	Load constant value
VarLoad	Load variable
React	Reactance
Resist	Resistance
Length	Line length
Node1IDs	List of all element IDs connected with Node1
Node2IDs	List of all element IDs connected with Node2

GRAPHALIAS

ID	Description
Picture	Screen name
ElementID	Screen element ID
ElementRef	Screen element reference
Type	Screen element -type (see "element")

OrigElemRef	Screen element - reference to the original screen element
OrigGraphElemID	ID of the original elements in "GraphElements"

3.6.3 API

In the object model of the zenon API, the objects `ALCGraphElement` and `ALCGraphAlias` are available for the model. These contain the same information as the XML file. These objects can be accessed in the ALC engine via:

- `GraphElemCount()`
- `GraphAliasCount()`
- `GraphElemItem()`
- `GraphAliasItem()`

USER-SPECIFIC TOPOLOGICAL INTERLOCKINGS

If a topological interlocking is checked, the following event is called up at the ALC engine:

- `void CheckInterlocking(IALCEdge* pALCEdge, long nNewState, tpLockResult* LockResult, BSTR* bsText, VARIANT_BOOL* bUnlockable);`

The switch/disconnector to be switched and the new status is transferred. The event can fill `LockResult`, `bUnlockable` and `bsText` in order to display a violated interlocking condition. If the event handler returns `tpBusy` in `LockResult`, the event handler is queried until it no longer provides `tpBusy`, however for a maximum of `10 seconds`. The interlocking is active after `10 seconds`. The interlocking text and unlockability are reported back in `bsText` and `bUnlockable`.

SCREEN MARKER

Marker elements can be inserted into screens via the zenon API. These marker elements are available for the following elements:

- Line
- Polyline
- Pipe

These are added or deleted via the API functions in `DynPictures`:

- `BSTR AddMarker(BSTR bsScreenName, long nElementID, short nPosition, short nLineColorIndex, short nFillColorIndex);`
- `VARIANT_BOOL DelMarker(BSTR bsID);`

The GUID of the marker, which is supplied by AddMarker(), identifies the marker uniquely and serves as both the element name (with the prefix "**$MARKER_**") as well as the key for deletion via DelMarker(). The markers inserted via API are saved in the project according to the screen. **Attention:** Saving is not remanant, i.e. only until Runtime is restarted.

The markers set there are displayed regardless of the monitor on which the screen is opened. The markers are treated internally as normally operable screen elements. Mouse events are called up for this.

The appearance of the markers is set using the project settings in the **Automatic Line Coloring** area of the project configuration:

- **Display type of the screen marker**: Triangle, circle, square, cross
- **Screen marker size**: Size in pixels:
- **Line width of the screen marker**: Width in pixels
- Marker color: is defined via the index in the marker color table (on page 37), that is located in the properties of the screen elements in the **Automatic Line Coloring** group

4. Command Processing

Command processing serves primarily for the secured switching of variables in energy technology. 'Secured' means that there is a check whether the switching operation is allowed, according to the configured interlocking condition and the dynamically updated topology (current physical state of the topological network). The configuration of the topology and the topological commands is done via the ALC (Automatic Line Coloring) (on page 7) module.

Note: You can find step-by-step instructions for the creation of a configuration of simple **command processing** in the Project configuration in the Editor (on page 66) chapter.

License information

Included in the license for Energy Edition.

The functions documented in this manual are only available if zenon Energy Edition is licensed.

Command groups always contain a set of defined actions, which are usually adjusted to a specific data point (a specific device) . For example, different command groups can be defined individually and centrally for different topological elements (switch/disconnector etc.) .

A data point for the command processing always consists of 2 physical variables:

- Response variable:
  The response variable is defined centrally for the whole command group. It represents the status of the topological element, for example whether the switch is open or closed.

- Command variable:
  A defined command variable is assigned to every action inside a command group. The driver uses this variable to write commands to the controller.

Depending on the action to be executed, these commands are executed on one of the two variables.

Example

`Switching command on`
Sends the command/the new value to the command variable. The success of the triggered action can be checked by means of the response variable.

`Status input off`
Resets status bits of the response variable configured in the action. The command variable is not relevant to this action.
*The **action variable** is the same as the **name of the response**.*

Note: You can find a description of the command actions in the action types (on page 99) section.

NAME REPLACEMENT

To simplify or to generalize the definition of the variables, the variable references (for command variables, response variables and condition variables) can be defined using a name replacement. In doing so, wildcards '*' can be used. Wildcards are only permitted as a prefix or suffix; e.g. `*xxx` or `xxx*`.

As a result of this flexible definition, generally-valid procedures can be defined, which are then applicable for several data points. The number of command groups that must be defined is thus reduced considerably.

Example

- Definition of the command variables - **action variable** property = `*_CO`
- Definition of the response variables - **name of the response** = `*_RV`

In Runtime, the Command Processing automatically adds the name of the response variable, which is shown/selected in the process screen, to the name of the command variable. The names of both variables differ only in their endings.

This is also applicable for condition variables: `X01: *_ClsEna`.

Other variables - that have been linked to dynamic elements in the command processing screen - can also be replaced in Runtime.

You can find further details on this in the Substitution of additional variables (on page 84) chapter.

EXECUTION

In general, the single-step operations are executed by means of the **context menu** of an element in the topology (such as a switch). A further typical use is the opening of a `Command Processing` screen instead of a **write set value** dialog.

The two-stage operations are executed by means of a **context menu** or the `Command Processing` screen type.

Specific control elements are available for this screen type. They enable an individual optical and functional design of the command processing. This way, individual actions, for example, can be assigned to **action buttons** directly. After this, these actions can be selected by the user directly. This screen type also includes the necessary requirements in order to carry out functions such as unlocking, two-step execution, two-hand operation, locking etc.

Note: You can find detailed information about the process in the `Command Processing` screen in the Process in the Command Processing screen (on page 156) chapter.

Such a screen is called up on the screen element of the response variable by means of its context menu or instead of the **write set value** dialog. The call can also be by means of the **Screen switch** function that is linked to a button.

An action (switching operation) can be the following in Runtime:

- Permitted
  If there is no locking condition applicable.
- Not permitted
  If there is an unlockable condition applicable.
- Permitted after unlocking
  If an unlockable condition is applicable.

This results from the command groups and the current status of the topological model **ALC configuration** - **Interlockings** tab.

Note: You can find additional information on the procedure of a command in the Execution of a command (on page 146) chapter.

In the zenon network, there is synchronization for actions from the command that concerns a certain response variable, by activating the `NET_SEL` status bit. The simultaneous execution on the same object (same variables) by different users is thus precluded. Parallel execution on different response variables is supported.

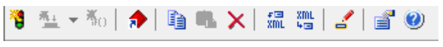
4.1 Command Processing

Menu item	Action
New command group	Creates a new command group.
Export all as XML	Exports all entries as an XML file.
Import XML...	Imports measuring units from an XML file.
Editor profile	Opens the drop-down list for selecting a Editor profile.
Help	Opens online help.

Information

Command groups can be exported, imported and copied and pasted using the clipboard. The same applies for actions and their interlocking conditions, even different command groups.

4.2 Command processing detail view toolbar and context menu



COMMAND PROCESSING AND COMMAND GROUP CONTEXT MENU

Menu item	Action
New command group	Creates a new command group.
Export XML all...	Exports all entries as an XML file.
Export selected XML...	Exports selected entries as an XML file.
Import XML...	Imports from an XML file.
Copy	Copies the selected command group to the clipboard.
Paste	Pastes command groups from the clipboard.
Delete	Deletes the selected command group after requesting confirmation.
Rename	Enables renaming of a command group.
Properties	Opens the properties window for the selected command group.
Help	Opens online help.

CONTEXT MENU GROUP ACTIONS

Menu item	Action
Command new	Creates a new command and opens the properties.
New auto/remote command	Creates a new auto/remote command and opens the properties.
New forced command	Creates a new mandatory command and opens the properties.
New set value input	Creates a new set value input and opens the properties.
New status input	Creates a new status input and opens the properties.
New replace	Creates a new replace action and opens the properties.
New revision	Creates a new revision and opens the properties.
New manual correction	Creates a new manual correction action and opens the properties.
New block	Creates a new block action and opens the properties.
New release	Creates a new manual correction and opens the properties.
Check response value	Creates a new Check response action and opens the properties.
New lock	Creates a new lock and opens the properties.
Paste	Pastes action from the clipboard.
Help	Opens online help.

CONTEXT MENU INDIVIDUAL ACTION

Menu item	Action
New interlocking condition	Creates a new interlocking condition. **Note:** Grayed out for `mandatory command` command processing actions.
Copy	Copies the selected action to the clipboard.
Paste	Pastes action from the clipboard.
Delete	Deletes the selected action after requesting confirmation.
Help	Opens online help.

CONTEXT MENU CONDITION

Menu item	Action
Remove interlocking condition	Deletes selected condition.
Copy	Copies the selected condition.
Paste	Pastes the condition from the clipboard.
Properties	Opens the property window for the selected element.
Help	Opens online help.

CONTEXT MENU GROUP VARIABLES

Menu item	Action
Add variable...	Opens the dialog for selecting a variable.
Paste	Pastes variable from the clipboard.
Help	Opens online help.

CONTEXT MENU INDIVIDUAL VARIABLE

Menu item	Action
Remove variable	Deletes the selected variable from the group after requesting confirmation.
Copy	Copies selected variables to the clipboard.

Paste	Pastes variable from the clipboard.
Properties	Opens the property window for the selected element.
Help	Opens online help.

4.3 Engineering in the Editor

The **Command Processing** module is a comprehensive module with many possibilities for expanding the behavior in Runtime and amending it individually.

Please also note, for your project configuration in the zenon Editor, the information in the introduction for this manual. (on page 60)

EXAMPLE OF AN INITIAL, SIMPLE COMMAND CONFIGURATION

1. Create a `command input` screen.
   Add the **control elements** from this template for this screen.

   Alternative project configuration for operation in Runtime by means of a context menu:
   Create a context menu with the following parameters:

   - **Action type** Command Processing
   - **Text** ALL
   - **Menu ID** ID_CDM_AUTO

2. Create two variables.
   It is recommended that these variables are created with an Energy driver (IEC870, for example).
   Attention: The **internal driver** is not suitable for response variables.

   - Command variable
     Variable to write a command (open/close) to the controller, for example **IEC870** Variable `T46`;
   - Response variable
     Variable that displays the status (position: open/close/invalid etc.) of the relevant object in the same controller, for example `T03`.

   Name these variables.
   You can name these variables so that both names have a joint description at the start, for example `switch_Q0_CO` and `switch_Q0_RV`. This allows a command group to be used for several variable pairs.

3. Create a command group.
   Configure the following properties of the command group:

- **Variable name of response**:
  With name substitution: `*_RV`
  Without name substitution: Name of the response variable from step 2
- **Screen**
  Linking to a `command input` screen from step 1

4. Create actions in the command screen.
   a) Configure a **Action type** `switching command` action with the following parameters:
   - **Action variable**
     With name substitution: `*_CO`
     Without name substitution: Name of the command variable from step 2.
   - **Return state/switching direction**: `ON`
   - **Command**: `1`
   - **Action button** `Action 1/Button: On`
     For execution without a context menu and without **two-stage**

   a) Configure a second action with the following different parameters:
   - **Return state/switching direction** : `OFF`
   - **Command**: 0
   - **Action button**: Action 2/Button: Off

   Hint: Copy the first configured action and amend the parameters.
5. Link the variables to the command group.
   Note: This is always required, regardless of whether name substitution has been configured or not.
   a) Select the two variables created in step 2.
   b) In the **Write set value** properties group, configure the **Command Group** property.
   To do this, select the three created command groups created in step 3 from the drop-down list.
6. Configure a trigger for the created processing:
   a) Create a new zenon screen. The type of the screen can be any you want except `command input`.
   b) Place a dynamic screen element on this screen.
   c) Link this screen element to the response variable.
   d) In the drop-down list of the **Write set value via** property, select the `Command` entry.
   You can find this property in the **Write set value** properties group of the screen element.

   Alternative project configuration for operation in Runtime by means of context menu:
   In the drop-down list of the **Context Menu** property, select the name of the context menu created in step 1.
   You can find this property in the **Runtime** properties group of the screen element.

The user can now click on the configured screen element in Runtime (right-click for context menu) to trigger the actions of the command.

Information

*For tests too, use a driver that supports the evaluation of the COT (Cause of Transmission - **Cause of transmission**) in full, for example the IEC 60870-5-101_104 driver. COT evaluation is an enhanced functionality to monitor communication during a command using the **Watchdog timer** settings. The status bits `COTx` of the command variables can also be evaluated in the reaction matrices **multi-numerical** and **multi-binary**.*

4.3.1 Creating a screen of the type Command Processing

A **command processing** screen allows control in Runtime and an overview of the command processing. The command processing can be controlled in Runtime using buttons.

The command processing screen is created in the Editor configuring a new **command processing** screen. (You will find more information on the pre-defined screen types in the manual Screens/Pre-defined screen types'.)

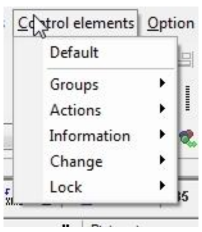
The screen `Command Processing` is used for user interaction via command during the runtime (one and two-step command). It allows the user to perform all activities that are necessary for command execution. This can be, for example, the unlocking of an active action or the confirmation of the execution of a two-step command.

Information

When using one-step command processing, a context menu can also be used. The screen type command processing is then not required in the project.

You can use specific control elements (on page 155) for this screen type, which allow all user actions necessary for command processing and which visualize current information about the status of the action to be executed (e.g. display of the switching direction).

It is opened as an empty one after a new screen has been created. You add the default control elements via menu **Control elements**/**Add template**.



ENGINEERING

Steps to create the screen:

1. Create a new screen:

   In the tool bar or the context menu of the **Screens**node, select the **New screen** command.
   An empty `Standard` screen is created.

2. Change the properties of the screen:

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

   b) Select `Command Processing` in the **Screen type** property.

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

3. Configure the content of the screen:

   a) select menu item **Control elements** from the menu bar

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

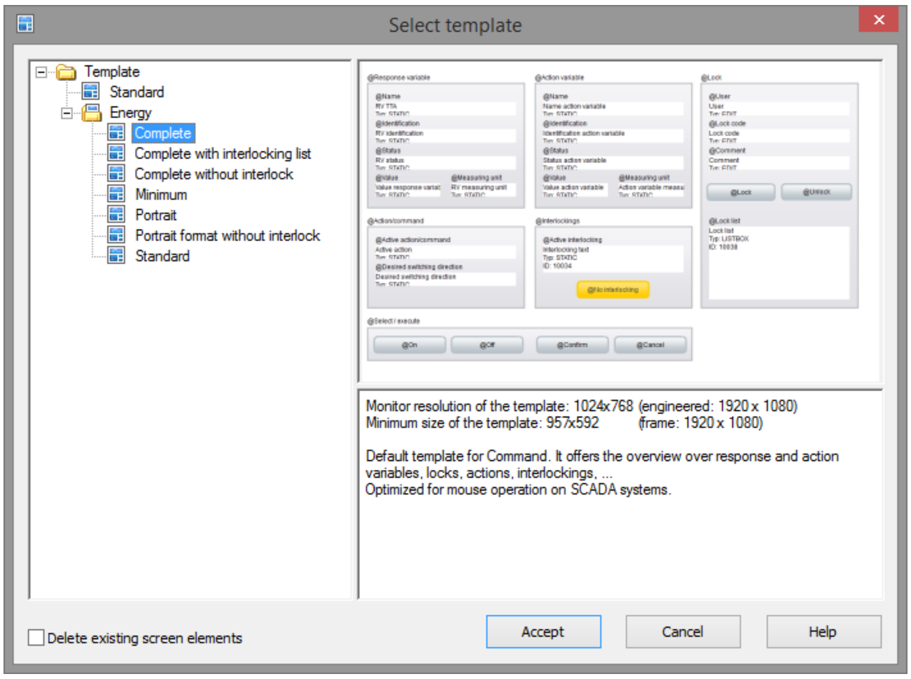
   c) Remove elements that are not required from the screen.

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

4. Create a screen switch function.

Template

Several pre-defined templates are available for **Command Processing** screens.



Monitor resolution of the template: 1024x768 (engineered: 1920 x 1080)
Minimum size of the template: 957x592 (frame: 1920 x 1080)

Default template for Command. It offers the overview over response and action variables, locks, actions, interlockings, ...
Optimized for mouse operation on SCADA systems.

Template	Description
List field templates (left)	Displays all pre-defined and user-defined template.
Preview and description (right)	Shows preview and description of the selected template.
Standard	Compact display of the command processing with visualization of: ▶ Actions ▶ Interlockings ▶ Buttons for selection and execution

ENERGY

Template	Description
Portrait Format	Display of command processing in portrait format, optimized for placing next to an overview screen: ▶ Actions ▶ Interlockings ▶ Buttons for selection and execution ▶ Lock
Portrait format without interlock	Simplified display of command processing in portrait format: ▶ Actions ▶ Interlockings ▶ Buttons for selection and execution
Complete	Enhanced display of the command processing with visualization of: ▶ Response variables ▶ Action variables ▶ Lock ▶ Actions ▶ Interlockings ▶ Buttons for selection and execution
Complete with interlocking list	Enhanced display of command processing including all interlockings: ▶ Response variables ▶ Action variables

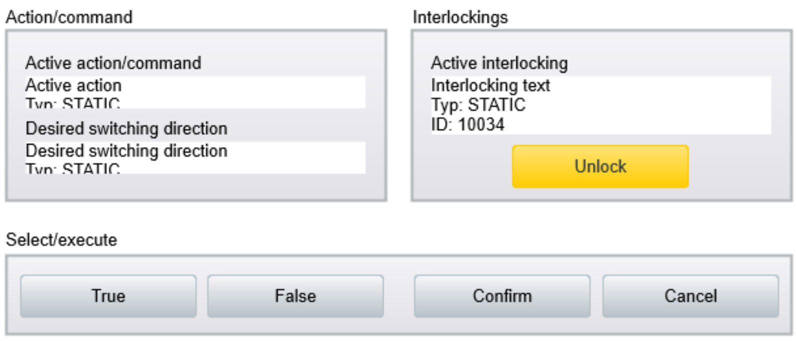
	▶ Interlocking (list) ▶ Actions ▶ Buttons for selection and execution
Complete without interlock	Enhanced display of command processing. With interlockings, only the currently-pending interlocking is visualized: ▶ Response variables ▶ Action variables ▶ Actions ▶ Interlockings (Text) ▶ Buttons for selection and execution
Minimum	Only contains visualization of buttons for selection and execution (on page 164).

CLOSE DIALOG

Parameter	Description
Delete existing screen elements	Behavior when applying the template for configuration in the Editor. `Active`: Already existing elements in the screen are deleted when taking over the template. Default: `inactive`.
Apply	Inserts the elements of the selected template in the screen and closes the dialog.
Cancel	Closes the dialog without inserting elements.
Help	Opens online help.

Screen template - standard

The **Standard** template only contains the most important control elements for `command` actions. It is suitable for actions that are executed using a **context menu** or from a **command sequence**.



ACTION/COMMAND

Control element	Description
Active action/command	Displays the pending action of the command group.
Switching direction	The switching direction configured for the active action. The texts are documented with the "switching direction" setting. Depending on the active action, the following text is shown: ▸ Command, revision, correction, replace: Text from limit value, depending on switching direction. ▸ Status: On or Off ▸ Other: empty

INTERLOCKINGS

Control element	Description
Active interlocking	Interlocking text of the active interlocking.
Unlock	If an unlockable interlocking is upcoming, it can be unlocked with this button. **Note:** This control is shown only when the screen is in the step 'Unlock'. The Control is locked when the upcoming interlocking is not unlockable.

SELECT/EXECUTE

Control element	Description
On	Command button for switching command, to close a switch for example.
Off	Command button for switching command, to open a switch for example.
Confirm	Confirms for the pending two-step action. A two-step switching command, for example, is only executed after clicking on this button.
Cancel	▸ Closes the command processing screen. The pending action is not executed. ▸ Cancel for pending two-stage action (Cancel instead of Execute). ▸ Cancellation of the execution of an action (depending on the configuration, for example: cancel Operate if it has still

	not been terminated). The button is grayed out if the screen is in 'Step 1'.

Control elements - complete overview

The following elements are available in the **Control elements** menu bar in zenon for the `command input` screen:

Name	Control type		Default
Action buttons	`Text`	Buttons, which can have an action assigned to them. By clicking in the screen, the assigned action is activated and the screen changes to the step "Release" `The button is not shown when:` ▶ No action is assigned to the button in the current command group. ▶ The variable, with which the screen was loaded, is the command variable, and the action assigned to the button does not use the command variable as action variable. However, if the action 'Lock' was assigned to the button, it is visible. The button is shown as locked when: ▶ The screen is not in 'Step 1'. ▶ The response variable has set one of the status bits I_KENNUNG(18), OFF(20) or NICHT_AKTUELL(29) and writes the assigned action to the command variable. ▶ The response variable has the status REVISION(9) active and the assigned action writes to the command variable. ▶ The response variable has the status REVISION(9) active and the assigned action is 'Correct'. ▶ The assigned action is 'Release' and the response variable does not have the status Alternativevalue(27) active. ▶ The assigned action is 'Correct' and the value of the response variable matches the switching direction. ▶ The assigned action is 'Replace' and the value of the response variable matches the switching direction. ▶ The response variable has the status REVISION(9) active and the assigned action is 'Replace'. ▶ The assigned action is 'Revision' and the value of the response variable matches the switching direction.	`Action1` `Action2`

RV TTA	Text	Name of the response variable	X
RV identification	Text	Name of the response variable	X
Action variable unit	Text	Unit of the current action variable.	X
Action variable set status	List	Defines the status to be set for the action 'Status default' for the switching direction 'None'. The statuses are set to the current status and updated when changes occur. Is locked when the active action is not 'Set status'.	
Switching direction	Text	The switching direction configured for the active action. The texts are documented with the setting 'Switching direction'. Depending on the active action, the following text is shown: Command, revision, correction, replace: Text from limit value, depending on switching direction. Status: On or Off Other: empty	X
Execute Step 2	Button	Delivers the actions to execution. This control element is visible only when the screen is in 'Step 2'. The control element is locked when: ▸ Two handed operation was configured and the key 'Ctrl' is not pressed. ▸ The status REVISION(9) of the response variable is set and the assigned action is 'Command', 'Set value', 'Replace' or 'Correct'. ▸ The button was already clicked.	X
Action variable minimum	Numeric	Minimum value of the action variable. Not visible if the action variable is of data type 'String'.	
Action variable maximum	Numeric	Minimum value of the action variable. Not visible if the action variable is of data type 'String'.	
Scrollbars	Numeric	Setpoint input with scroll bar Sets the value in the control element 'Set value' or is set by	

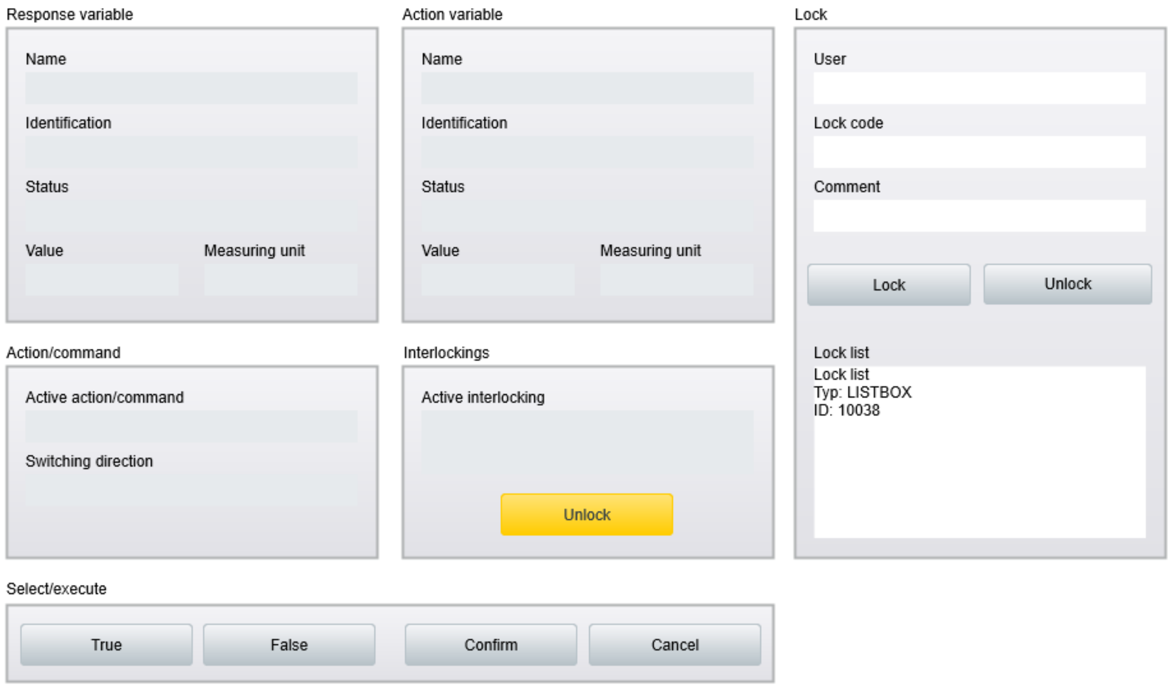
		this value. Not visible if the action variable is of data type 'String'. `The control element is locked when:` - No action is active. - The screen is not in 'Step 1'.	
Set value	`Numerical, Text`	Allows the input of the set value. By clicking the control element, it is switched to edit mode and the setpoint input is possible. The edit mode can be left again with "Enter". The new value is set only after clicking the control element 'Execute'. The desired value for the action 'Set value' is provided with this control element. `The control element is locked when:` - The status REVISION(9) of the response variable is set. - No action is active. - The screen is not in 'Step 1'.	
RV value	`Text`	Value of the response variable	X
RV status	`Text`	Contains the status of the response variable in the short form.	X
RV unit	`Text`	Unit of the response variable	X
Interlocking text	`Text`	Text of the upcoming interlocking. Text is online language switchable	X
No interlocking		If an unlockable interlocking is upcoming, it can be unlocked with this button. **Note:** This control element is shown only when the screen is in the step 'Unlock'. The control element is locked when the upcoming interlocking is not unlockable.	X
Exit	`Button`	Closes the screen without action execution. The button is only visible in a modal screen. This button is important for modal screens, because it is required to leave the screen in case of an error!	

Cancel	Button	Aborts the execution of the Command Processing and returns to 'Step 1'. The button is locked when the screen is in 'Step 1'.	X
Lock list	List	Contains the locks that were activated at the response variable. Is locked when no action 'Lock' was configured for the command group. Text is online language switchable	
User identification	Input field	For entering the user identification for the lock. Is locked when no action 'Lock' was configured for the command group.	
Lock code	Input field	For entering the user-specific lock code. Is locked when no action 'Lock' was configured for the command group.	
Execute lock	Button	Activates a lock for the user entered in the control element 'User identification'. Is locked when no action 'Lock' was configured for the command group. This user action is logged in the CEL, if not suppressed by the engineering.	
Unlock	Button	Removes the lock by the user entered in the user identification. Is locked when no action 'Lock' was configured for the command group. This user action is logged in the CEL, if not suppressed by the engineering.	
Execute	Button	Takes over the value of the control element 'Set value' or 'Set status' This control element is visible only when the screen is in 'Step 1'. The control element is locked additionally to the general lock, when: - The active action is not 'Set status', 'Set value' or 'Correct set value'. - The value in the control element 'Set value' for the action variable is invalid.	
Comment	Input field	Comment about the lock.	

Action variable Status	`Text`	Status of the active action variable in short form.	X
Action variable Name	`Text`	Name of the active action variable.	X
Action variable Identification	`Text`	Identification of the active action variable.	X
Action variable value	`Text`	Value of the active action variable.	X
Active action	`Text`	Name of the active action.	X

Screen Template - complete

The **complete** template contains all control elements for `command` and `lock` actions.



Parameter	Description
Response variable	
Name	Name of the response variable
Identification	Name of the response variable
Status	Contains the short description of the status bits for the response variable.
Value	Current value of the response variable
Measuring unit	Measuring unit of the response variable
Action variable	
Name	Name of the action variable
Identification	Identification of the action variable
Status	Contains the short description of the status bits for the action variable. Example: ▶ Bits for COT ▶ Status SE_870 during Select ▶ Status PN bit in the event of a negative response from the PLC **Note:** The status bits contain the response variable for the "status input" action.
Value	Current value of the action variable or input field for **setpoint input** command processing action. **Note:** This value changes during the course of the action from an existing to a current value. The display of the value is only refreshed with COT=7 (COT_actcon) or WR-SUC . ▸ The following is applicable for a configured setpoint input: The value to be set for the 'Set value' action is stipulated by this control element. By clicking the control element, it is switched to edit mode and the setpoint input is possible. It is possible to leave the editing mode again by pressing the Enter key. However the new value is only set when the **"Execute"** control element is clicked on. **The control element is blocked if:** - The response variable has set the status REVISION(9). - No action is active.

	- The screen is not in the "Step 1" stage.

Measuring unit	Measuring unit of the action variable
Lock	Control elements from the **Lock** group are locked if no "Lock" action is configured in the command group.
User	For entering the user identification for the lock.
Lock code	For entering the user-specific lock code.
Comment	Optional text that can be entered by the user for the lock.
Lock	Activates a lock by the user entered in the `User` control element. **Note:** This user action is logged in the CEL, if not suppressed by the engineering.
Unlock	Removes a lock that has already been activated. In doing so, only locks that the user themselves have activated can be deactivated. As a result, it is ensured that only people's own locks are removed. The user is visualized in the "`User`" control element. If there is no `Lock` action configured in the command group, this button is grayed out. **Note:** This user action is logged in the CEL, if not suppressed by the engineering.
`Lock list`	List of active locks: ▶ User Name of the user who has activated the lock. ▶ Locking time Time stamp of the interlocking ▶ Note Text for the interlocking.
Action/command	
Active action/command	Type of active command processing action such as dual command, for example.
Switching direction	The switching direction configured for the active action. The texts are documented with the setting 'Switching direction'. `Depending on the active action, the following text is shown:` Command, revision, correction, replace: Text from limit value, depending on switching direction. Status: On or Off Other: empty

Interlockings	
Active interlocking	The active interlocking (on page 125) according to the configuration or texts from ALC - topological interlocking (on page 33).
Unlock	This button unlocks an active, unlockable interlocking. **Note:** This control element is shown only when the screen is in the step 'Unlock'. The Control element is locked when the upcoming interlocking is not unlockable.
Select / execute	
On	First-step command button, to close a switch for example. **Note:** Only visible in Step 1.
Off	First-step command button, to open a switch for example. **Note:** Only visible in Step 1.
Confirm	Second-step command button. **Note:** Only visible in Step 2.
Cancel	Second-stage command button. Aborts the execution of the command processing and returns to 'Step 1'. The button is grayed out if the screen is in 'Step 1'.
Close	Closes the Command Processing screen.

Substitution of additional variables in the screen

For command input, in addition to variable substitution of zenon and the use of placeholders in response and command variables, further substitution rules can be configured for each command group and command action. You can thus place further dynamic elements in the command screen, which are linked to additional variables, whose names are then also automatically substituted in Runtime. Substitution is carried out in accordance with the response and command variables.

When configuring a project in the zenon Editor, you can find the **Replace in screen** property for each command group or command action. This properties are in the **Command Processing screen** property group.

REQUIREMENTS FOR SUBSTITUTION

Requirements for use are:

- The response variables and command variables were configured in the command processing with the * (star) placeholder.
  Example: `*_RV, *_CO`
  Response variable: **Variable name of response**
  Command variable: **Action variable**
- A command processing screen is assigned in the **Screen** in the respective command group or command action.

Variables are substituted according to the following rule in the command processing screen:

- The text from the property is substituted in the variable name that is shown in the command field.
- It is substituted with a text which command found in the name of a response variable or action variable in place of the placeholder *.

Several texts to be substituted are configured separately with a semicolon (;). These phrases are substituted from left to right when calling up a screen in Runtime. The following phrases are ignored as soon as a text for replacement is applied.

If, when calling up the command processing screen in Runtime, there is no variable name with the configured text, there is also nothing substituted.

SCREEN SWITCHING AND COMMAND PROCESSING SUBSTITUTION RULES

Substitution via the screen switching function can be combined with the substitution of command processing. The following rules apply for substitution:

- If the screen is called up with a **Screen switch** function, the substitution configured in the function is used in Runtime.
- If the screen is called up using the **Command Sequencer** module or the menu, Runtime gets the screen and the substitution from the project configuration in the respective command action. If there is no substitution configured in the command action, Runtime gets the screen and the substitution from the command group. If there is also no substitution configured in the command group, there is no replacement.
- When clicking on a dynamic element that has `new set value input` configured, Runtime gets the screen and the substitution from the **setpoint input** command action.

> **Information**
>
> *You can get further information on substitution via the screen switching function in the command processing chapter in the functions and scripts manual.*

EXAMPLE

The following are configured:

- Name of the response variable: `abc_RV`.
- Configured response variable for command group: `*_RV`
- Additional variables in the screen: `xyz_lock, xy_Switch`

Result - scenario 1

- Configured **replacement in the screen**: `xyz;xy`
- Existing variables in the project: `abc_lock and abc_Switch`.
- **Result:** Display of the variables in the screen: `abc_lock and abc_Switch`.

Result - scenario 2

- Configured **replacement in the screen**: `xy;xyz`
- Existing variables in the project: `abcz_lock` and `abc_Switch`.
- **Result:** Display of the variables in the screen for `abcz_lock` and `abc_Switch`.

Result - scenario 3

- Configured **replacement in the screen**: `xy;xyz`
- Existing variables in the project: `abc_lock` and `abc_Switch`.
- **Result:** Display of the variables in the screen: `abc_lock` and `abc_Switch`.
  Because `abcz_lock` is not present.

4.3.2 Variables of the command group

Command groups use firstly the variables of the switching actions (the response variable an command variable) ans secondly, optionally, the variables of the **command conditions** and the variables of **breaker tripping detection**.

In order for the Command Processing module to be used, the respective response and command variables must be assigned to a command group. This assignment is made in the **variables** node => for the variable => in the **Write set value** properties group => in the drop-down list of the **Command Group** property.

Ensure that this assignment is configured for both response variables and command variables.

Note: Errors in project configuration are listed in the output window of the zenon Editor when compiling the project. In Runtime, invalid or incompletely-configured commands for the variables concerned are not called up.

For the response variables and command variables, the set value can only be set using the command processing; it can no longer be set directly using dynamic screen elements. For screen elements that the user triggers with command processing, the `Command` value must be selected for the **Write set value via** property or a **context menu** must be linked. To do this, it is preferable to use the screen elements that are linked to the response variable (not command variable). This guarantees the availability of all actions of command processing (provided the user is authorized).

Note: The screen element can also be used if the response variable is "read-only", from an IEC 60870 controller for example. A combined element with a circuit breaker symbol can trigger the command, although the response variable itself cannot be changed. The position of the switch (open/closed) corresponds to the value of the response variable.

Despite the linked command input, the values of the command variables also cannot be written to directly:

- via the **RGM**
- via **API**
- In zenon Logic with the **Externally visible** property activated.

The command processing is ignored in the process.

Information

If a variable is linked to a command group, it is not possible to describe the variable with the zenon ***Write set value*** *function.*

Exception: *If a* `write set value` *(on page 104) command with* ***switching direction*** `set value` *has been created, the zenon function calls up this action in the background without the* `command processing` *screen being called up. This means that the* ***command conditions*** *(on page 124) are checked. An active interlocking condition prevents the writing of a set value. During the execution of an action, the* `NET_SEL` *status bit is not set and the* ***Select Before Operate*** *variable property is ignored.*

This is also applicable for the value entry of a variable that is linked to a dynamic element if `Element` *was selected for the* ***Write set value via*** *property.*

GENERAL EXAMPLE

The command group "`DPI one stage`" was configured with the name of the response variable `*_RV` and the switching actions in this group with the name of the action variable `*_CO`.

In the SCADA project, variables with the name `ied9_100_RV` (position of the switch) and `ied9_100_CO` (command for switch) are configured. And the `ied9_100_RV` variable was linked to a screen element with **Write set value via** = `command`.

Link the two variables `ied9_100_RV` and `ied9_100_CO` in the **Command Group** command group property to the "`DPI one stage` command group". The respective wild card * is replaced with "`ied9_100` in the Runtime.

Other variables, such as `ied9_101_RV` and `ied9_101_CO` (etc.) can thus be linked to this command group. In Runtime, the command groups are then instanced several times and can be operated independently.

Furthermore, you can also define the optional variables of the **command conditions** and the **breaker tripping detection** with the placeholder `*`, for example **X01:** `*_EnableClose`.

Information

As a result of the different use of limit values/reaction matrices for the command variable/return variable, individual switching directions can be displayed for the actions. Always depends on which of the variables the desired action is to be executed.

Limit values and reaction matrices for switching direction texts

The command uses the limit value text of the command variables for the display of the **switching direction** in the command screen and in the **context menu**.

Example: during execution of a command in two-stage command processing, a corresponding text is shown for **command** actions in the **switching direction** control element. These texts can be defined via the limit values or via the states of the reaction matrices.

In the **context menu** in particular, these texts give the user a better understanding or a better overview of the actions that are available in Runtime (e.g. '`Command: Open disconnector`')

You therefore have the possibility to issue different texts for each variable that uses the same command group. Several variable pairs (each response variables and action variables) can thus only use one command group and can nevertheless be displayed in an individualized manner.

If no limit value has been created for a variable and no reaction matrices are linked, the action uses a standard text:

Switching direction of the action	Standard text
None	@NONE
OFF	@OFF
ON	@ON
DIFF	@INTER
DIST	@FAULT
DIR	@DIR

Information

As the switching direction texts are read out from the limit value settings, they are completely language switchable.

Project overlapping variables

Attention

The variables used in the command groups must be in the same project in order for the command processing to work properly.

If you do use a variable from another project (e.g. subordinate project in multi-project administration), the command processing group, the response variable, the action variable and the action-specific screen ('Command Processing' screen) is expected to also exist in the other project.

Information

You can also use project-overlapping variables for the interlockings by the process. The above limitations apply only to the variables of the command group.

4.3.3 Configure command input

Select the **Command Processing** entry in the project tree. Select **New command group** in the context menu.

After creating a new command group, it is added to the detail view of the project manager with standard name "`Command group + index`". The index is replaced by a consecutive number.

Note: This name serves for the unique identification in the system.

Information

*You can assign any name you like to the command groups. However, it must be ensured that the names are unique within the project: applies for **general interlockings** and **command groups**.*

The following parameters are available for command groups:

Parameter	Description
Name	Name of the command group. Must be unique for all interlockings in the project. This name is used later with the variable that uses this command group. The command group can be renamed at any time.
Variable name of response	This is the variable name or the mask for the replacement of the response variable. A wild card * (star) that appears in a name serves as a placeholder for the substitute text. **Example:** - `*_RV` - `*/stVal[ST]` Only one placeholder can be used in a name. **Attention:** If the name remains empty or the variable that is used here (replaced or absolute) does not exist at the time of compiling, this command group is not available in the Runtime. A corresponding message in the output window points this error out during compiling.
Set status PROGRESS	If activated, status bit **In progress** (PROGRESS) is written for actions `command` and `Manual correction`. The value that the status bit is set to depends on the switching direction of the action. The status bit is set to 1 if: - the **Return state/switching direction** of the action is `ON` or `OFF`. - The response variable does not already have the value of the set switching direction. The status bit is set when checking the interlockings and remains until the execution of the action has been completed. This also implies that, in the case of **Select Before Operate**, the status PROGRESS is only set after a successful 'Select' `(SE+COT_actcon)` and then remains set during watchdog timer or edge delay. If the execution of the action is triggered by a **context menu** or if it is a one-step action, the status bit is also set accordingly.
Watchdog timer	There is the following setting for this drop-down list: - `none`: The watchdog timer (on page 146) is deactivated. However with **Select Before Operate**, there is a wait for confirmation of the `'Select' (SE+COT_actcon)` and it is then ensured that 'Select' has ended (PLC has reacted to `'Execute' COT_act` in the envisaged time). If `'Select'` has not yet been ended, the `'Select'` is deactivated a `'Cancel' (SE+COT_deact)` is sent to do this.

	- `Response variable only`: The value of the response variable (RV) is used to check whether the process was successful. - `Cause of transmission only`: The status bits for Cause of Transmission (COT) of the command variable are used to check whether the process was successful. - `COT and RV`: Both conditions defined above.

Screen modal	If activated, the screen is displayed modally, regardless of the configuration for the **Modal dialog** property for the screen.
Screen title from response variable	The **Identification** of the response variable is shown in the screen title. This only happens when there a title was configured for the screen at the frame. Language switching is supported.
Screen	Name of the calling screen if the command is called up using a screen element of the response variable (or action variable). **Note:** Actions called up via the context menu open, for the confirmation of the second stage or interlocking text, a screen that has been defined for the action. Only if no screen has been linked for the action a defined screen is also used here in the context menu.
Breaker tripping detection	Only available if property **Set status PROGRESS** is activated. `Active`: The response variable is monitored for a change from `<> 0` to `0`. The identification only sets the status bit `CD_TRIP (50)` to 1 if: ▶ status bit `CB_TR_I` (51) is not `1`, otherwise the identification is suppressed. ▶ status bit `PROGRESS`(10) is not `1`, otherwise the value change of the response variable is considered a result of its own command. **Attention:** Value changes that are a delayed consequence of its own command can be recognized as breaker tripping. This happens if the PROGRESS bit has already been deleted or if the action does not support **Watchdog timer**.
Suppress detection	Entering the formula with which the detection of a breaker tripping can be suppressed. A click on button **...** opens the formula editor. All variables from the **Variables** node in the command group can be used for the formula. Variables from projects loaded in Runtime can be used. Name replacements with '*' - as with the definition of the interlocking conditions of an action - are possible. The suppression sets the status bit `CB_TR_I` (51) to 1. With active recognition, all variables whose status or value are used in the formula for breaker tripping detection are activated for reading when the program is started after loading all projects, and remain this way as long as Runtime is running. **Note:** Variables that are used in the formula cannot be deleted from the list of the variables linked to the command group.

Information

The response and action variables do not need to be in the list of the variables linked to the command group. Their names need only be configured for the command group and in the action.

Command Processing in Distributed Engineering

Information

*Because the **command conditions** and the **general interlockings** (standard functionality - without Energy Edition) are saved in the zenon Editor with the same structure, the check-out symbol (allow changes) is set to exactly the same for both nodes in the project tree. All actions on the command conditions also apply to the general interlockings and vice versa.*

Variables marked for deletion are considered as not existent for the compilation of the command conditions. During compiling, the respective error messages are displayed in the **output window** in the zenon Editor.

Create action

Actions define the switching commands that are possible for command groups. By selecting the element **Action** in the detail view of the command group, you can define a new action with a right mouse click. Details of the defined actions are also shown in the detail view after creation (e.g. "`switching command: *_BE [ON,1]`").

All further settings for the actions are made in the properties window. Some of the properties are inactive, depending on the action type.

The following properties are available for a command action

Parameter	Description
Action settings	
Action variable	Variables on which is written. For some actions, this is the response variable. In this case, the field is locked. The placeholder for the replacement text is the character sequence '*' within a name. Only one placeholder can be used in a name. If the variable that is used here (replaced or absolute) does not exist during compiling, the action is not available in the Runtime. An according message announces this error during compiling. Click on the **...** button to open the dialog for selecting a variable. Default: `No Allocation`
Action type	Shows the type of command. For editing, only approved for **command** action type; possible settings are `switching command` or `pulse command`. Default: `Switching command`
Return state/switching direction	Defines the expected value and the status of the response variable after action execution. Locked for the actions `block`, `lock` and `release`. Default: `Off`
Command	Defines the value that is written to the command variable with the **Command** action. **Note:** only available for the command actions `switching command`- and `pulse command`, `auto/remote command` and `forced command`. Default: `0`
Edge delay	Time in milliseconds by which the resetting of the value is delayed for a `pulse` command. **Note:** Only available for the `pulse command` action. There is no wait until until **runtime monitoring** has ended. Default: `1000 ms`
Set value	Defines the value that is written to the controller. **Note:** Only available if **Return state/switching direction** has been set to `DIR`.
Modifiable states	List of the states which can be modified with the `Set status` action. **Note:** Only available for the action '`Status input`'. Default: `None modifiable`
Command Processing screen	

Screen	`Command Processing` screen that is used when the action has been carried out using the context menu of the element. If no screen is entered, the screen, which is entered in property **Screen** for the command group, is used. An engineered screen which is not available, creates an error message when creating the Runtime files. In this case the action is not taken over. Default: `none`. **Note:** If the command processing is called up by a dynamic screen element, this property is ignored and the screen that is entered in the **Screen** property in the command processing group is always used. Not available for the `auto/remote command` action type.
Replace in screen	Substitution rule for command screens: The target of the substitution is configured in this property. The text that is to be replaced is configured. Several substitution rules are separated by a semi colon (;). In doing so, the configured substitutions are processed from left to right. If there is no * in the response variable or action variable, there is no substitution. There is no distinction between upper-case letters and lower-case letters in the project configuration.
Action button	Assignment of an action to an action button. Action buttons are configured in a command screen. If the command group is used for another screen (e.g. via function), the allocation to the action button remains nevertheless. In other words: the action is always placed on the button with the allocated action ID. If such a button is missing, the action is not available in the screen. Only the action buttons that were not allocated yet are provided in the selection list of this property. This setting is locked if no screen was allocated to the command group and for the `Lock` action type. Default: `No Allocation`
Nominal/current value comparison	If this is active, there will be a check whether the value of the response variable already matches the **Return state/switching direction**. If this is true, an unlockable interlocking variable is shown. **Note:** Only active for `command` action type Default: `Inactive`
Can only be executed if target is <>	Deactivates an action button in the command screen if the value of the response variable already matches the value of the set value. If the value of a response variable is changed, the corresponding action button is active again. The same also applies for context menu entries. The corresponding command action is not displayed in the menu here. **Note:** Only available if the **Nominal/current value comparison** property is activated and only available for the `command` (switching or pulse

	command) command action. Default: `Inactive`
two-stage	If this is active, only after operating the **Execute 2** control. **Step** that executes action. If not active, the action is executed after releasing the last interlocking or, if there is no upcoming interlocking, immediately. **Note:** Locked for the `lock` and `auto/remote command` actions. Default: `Active`
Two-hand operation	- `Active`: The **Execute 2** control element. **Step** is only unlocked if the `Ctrl` key is held down. In Multi-Touch applications, both pressure points must each be on their own screen with their own frame. Not available if no two-step execution has been configured. **Note:** For the **Execute 2nd step** control element, the **selectable with lasso** property must not be active with two-hand operation. Default: `Inactive`
Close automatically	If this is active then the screen is closed automatically after action execution. **Note:** Not available for the `auto/remote command` action type. Default: `Inactive`
Menu ID	The menu ID is used for the creation of Context menus in the Runtime. **Note:** If two actions are fitted with the same ID, they are tagged with a special symbol in the action tree. They can then not be called up by the context menu.
Action name	Freely-definable name of the command action. This can, for the **Command Processing** module, be displayed in Runtime using a screen of type Command Processing. In the **Command Sequencer** module, this name must be used for the step.
Command Sequencer	Project configurations for the **Command Sequencer** module.
Options	
Suppress CEL entry	If this is active, no entry in the CEL will be made when executing an action. Default: `Inactive`
Timeout	Timeout for the runtime monitoring in seconds for `switching command` and `pulse command` actions. This setting is also applicable as a timeout for Select.

	Unit is seconds Only available for the actions `Command`, `Auto/Remote command`, `Check response`, `Setpoint input` and `Forced command`. Default: `30`
Timeout can be canceled	Allows the cancellation of the timeout in runtime monitoring. Only available for the `command` and `Setpoint input` actions. If the command has already been executed - after **COT_actcon** has been received - the **Cancel** button cancels runtime monitoring. Buttons are therefore active and operable again. **Note:** Not all drivers support deactivation during execution. If not, no Cancel is sent to the controller; the action is canceled only.
Use Qualifier of Command	Allows commands to provide additional information (Qualifier of Command). The requirement for this is that the driver also supports this option. Possible drivers are, for example, IEC850, IEC870 and DNP3. Is only available for the actions `command`, `auto/remote command` and `forced command`. Default: `Inactive`
Qualifier of command	Entry of a numerical value that is sent to the driver as a command parameter. This input possibility is only available the **Use Qualifier of Command** property has been activated. - Input range: `0 - 127` - Default: `0`

Attention

*The identification of the action types in the **Menu ID** must be clear, so that they are clearly identifiable in the context menu (on page 115). If two actions have the same ID, they are tagged with the special symbol **M** in the action tree.*

Hint

Note:

- When selecting individual properties, you receive additional information about functionality in the embedded help.
- Defined actions and commands can be exported into XML and imported from XML. They can thus be easily archived or reused in other applications.
- The status can be set using the command **status input**.

Action types

The action types are the available command procedures. According to the command, different activities are performed.

The system provides a variety of actions. The following action types can be defined for the command groups:

Action type	Remark
`Command new` (on page 101)	Switching command or pulse command. The value of the command variable is used to write the configured command processing status to the controller. **Note:** the switching command is suitable for pulse command and dual commands with the Energy driver (IEC60870, IEC61850, DNP3).
`New auto/remote command` (on page 102)	The remote command is forwarded from the Process Gateway or the zenon API to the command processing and processed as a switching command. The action is not available in a command processing screen nor via the context menu.
`New forced command` (on page 103)	The `forced command` action type allows the setting of a command, even if the response variable is `empty`, `OFF, NT` or `INVALID` . **Note:** the action is intended for emergency shutdowns and should only be used with caution.
`New set value input` (on page 104)	Writes a desired numerical value to the command variable.
`New status input` (on page 105)	Changes the status bits of the response variable. Only applicable for status bits in the **modifiable status** list.
`New replace` (on page 106)	Changes the status of the response variable to substitute value (`ALT_VAL`) and writes an alternate value to the response variable. **Note:** The `writing of variables to substitute values` allows the visualization of the process with manually-collected data during a communication failure, for example via **automatic line coloring**.
`New revision` (on page 107)	Sets the `REVISION` status bit of the response variable. **Note:** Alarm handling is suppressed in the revision.
`New manual correction` (on page 107)	Sets the value of the selected response variable according to the switching direction. **Note:** the communication protocols in Energy (IEC60870, IEC61850, DNP3) preclude direct writing to the response variable.
`New block` (on page 108)	Switches off the response variable (`OFF` status bit). **Note:** the switched-off variables are no longer read by the connected hardware.
`New release` (on page 108)	Sets substitute value replacement value (`ALT_VAL`) to 0. **Note:** as a consequence, the response variable has the value received by the controller again.
`Check response value` (on page 109)	Checks the status of the response variable without executing an activity. **Note:** the action is intended for use in the **command sequences module**.
`New lock` (on page 109)	The response variable is locked or unlocked for further actions when a valid **locking code** is entered.

Note: The action types are listed in the above breakdown in the sequence in which the action types are offered in the zenon context menu. However the sequence in the main window is alphabetical.

In the detail view of command processing, the actions in the tree are shown with the respective selected switching direction and configured action value.

Attention

*The identification of the action types in the **Menu ID** must be clear, so that they are clearly identifiable in the context menu (on page 115). If two actions have the same ID, they are tagged with the special symbol **M** in the action tree.*

Action type command

This **Action type** is used as a `switching command` or `pulse command` depending on the configuration.

When the command is executed, a value (0 or 1) is written to the command variable. The value to be written is configured with the **Command** property.

This action type supports **Select Before Operate** and **Watchdog timer**. The Select Before Operate depends on the corresponding property of the command variable and on the driver.

The **Watchdog timer**, depending on the configured type (`via response variable` for example) can also check whether the response variable changes its value according to the command. The value which that is then expected for the response variable as a result of the command is to be defined under the **Return state/switching direction** (`on/off/none`) property.

Switching direction	Value of the response variable after a command has been executed
`None`	No specific value change is envisaged. The action is ended if the configured **Timeout** has expired. **Note:** the **Watchdog timer** can nevertheless be activated in order to take the `cause of transmission` (COT) into account.
`Off`	The action expects value 0. If the response variable already has the value 0 before the command is executed, an internal interlocking condition is reported if the **Nominal/current value comparison** property is activated. **Note:** If runtime monitoring is configured with the values `none` or `via cause of transmission`, there is no wait for the response variable.
`On`	The action expects value 1 and (in accordance with **Nominal/current value comparison**) is compared to value 1.

With `pulse commands` a value is written to the PLC twice. The second time, after the configured **Edge delay**, there is an automatic reset to `0` or `1` (depending on the configuration of the **Return state/switching direction** property). However this does not happen if the **Select Before Operate** property has been activated for the command variable. The Energy protocols do not provide any possibilities to use a Select with pulse. If the **Select Before Operate** property has been activated for the action variable, a `pulse command` acts in the same way as a `switching command`.

Please note: The pulse command is not recommended for Energy drivers. The pulse command should only be used with a PLC that expects a pulse instead of an edge.

Note

*If, during the execution of the action, the current value of the response variable is different to the one defined in the switching direction and the switching direction was defined to be `on` or `off`, the **in progress** (PROGRESS) status bit is set. To do this, activate the **Set status PROGRESS** property in the command group.*

Auto/remote command action type

The **Remote command** (via Process Gateway, VBA, etc.) is forwarded to the zenon command processing, which processes the sequence (checking of interlocking, forwarding to driver, response, etc.) like a **Switching command (on page 101)** .

The command processing is not accessible via the command screen or the context menu.

The command is only supported by a previous Select . The action variable must have the **Select Before Operate** property activated.

When Runtime is ended, or reloaded, any Select that has been set is discarded. This means: The master connected to the Process Gateway is not informed and must get this itself using the interruption to the connection.

The VBA interface can use the `IVariable::SetValueWithStatusEx` method and the status bits to be transferred decide whether writing should be either direct or via the command processing. If the status bit `NET_SEL` (bit 8) has already been set (the command processing screen is open for example), the command is not executed. If the status bit is not set, it is set and writing is executed by Command Processing or commands are forwarded to the Command Processing. The response value of the method provides information on whether command processing has been activated or whether the command has been executed.

Transfer of the status bits of the action variable to the method:

- `SE_870` + `COT_act(6)` - Select activation
  Determines the command action to be executed and activates the command processing. The response variable of the method provides information on whether this is possible.
- `SE_870` + `COT_deact(8)` - Deactivation (Cancel)
  Ongoing command processing is canceled.
- `COT_act(6)` - Activation (Operate/Execute)
  Execute for command Command Processing is executed.

In order for this method to be able to execute command processing, a **remote command** action must exist whose switching direction corresponds to the transferred set value. The actual value written to the driver, Select etc, results form the properties of the action.

Note: When an interlocking takes effect, a (language-switchable) CEL entry with the configured text is created.

Information

You can find further information in the Select before Operate chapter in the Process Gateway manual, chapter IEC870 Slave.

Mandatory command action type

The **forced command** action type allows the setting of a command, even if the response variable is `empty`, `OFF`, `Not topical` or invalid (`INVALID`). It is not intended for emergency shutdowns.

Interlocking conditions cannot be created for the forced command, because it cannot be guaranteed that the condition variables have a valid value in Runtime.

Note: The forced command corresponds to a **switching command** without conditions.

Attention

The early or erroneous configuration of a forced command in Runtime can have dramatic consequences for the equipment. Always set this command with care and protect it with user authorizations.

Action type set point input

The **setpoint input** action type offers the possibility to set any desired numerical value to the command variable. The **command processing** screen offers its own control elements for this, which also allow manual definition of the set value. With the help of property **Return state/switching direction** you can define how the set value should be written:

Switching direction	Value of the response variable
`DIR`	Set value is written directly. You define the value which should be written with the help of function **Set value** of the action. The text which should be displayed can be engineered using a limit value/rema for the state/value 5. If this is not the case, a standard text (on page 88) is used. **Nominal/current value comparison** is not yet supported! The action can be carried out several times in a row.
`Set value`	Value of the command processing screen of the **Set value** control element is written to the action variable. In one-step execution, the value is written when clicking on the **Execute** button or when clicking on the **action button** (if configured). In two-step execution, the value is written when clicking on the **Execute 2 step** button.

This action type, for `DIR`, supports **Select Before Operate.**

Information

*If a variable is linked to a command group, it is not possible to describe the variable with the zenon **Write set value** function.*

*Exception: If a `write set value` (on page 104) command with **switching direction** `set value` has been created, the zenon function calls up this action in the background without the `command processing` screen being called up. This means that the **command conditions** (on page 124) are checked. An active interlocking condition prevents the writing of a set value. During the execution of an action, the `NET_SEL` status bit is not set and the **Select Before Operate** variable property is ignored.*

*This is also applicable for the value entry of a variable that is linked to a dynamic element if `Element` was selected for the **Write set value via** property.*

For further information, read the information in the Apply actions (on page 111) chapter.

Attention

When writing the set value with the switching direction `DIR`, neither the limits of the linked variable are checked, nor is a check carried out to see whether write set value is permitted for this variable.

Action type status input

Changes the status bits of the response variable. The following is executed, depending on the definition of the **switching direction**:

Switching direction	Action
`Off`	The states configured in the **Modifiable states** list are all reset to 0.
`On`	The states configured in the list **Modifiable states** are all set to 1 (active).
`None`	The states configured in the **Modifiable states** list must be defined in Runtime in the Command Processing screen with the help of the `Set status` control element. Each status bit is defined individually using a checkbox in the control element. ▸ In single-stage execution, the status bits are set by clicking on the **Execute** button or when pressing the **action button** (if configured). ▸ In two-stage execution, the status bits are set when the **Execute 2nd stage** button is clicked on.

If you change a status bit in Runtime, the change is logged in the Chronological Event List (status including value). These language of these messages can be switched in Runtime.

Information

For all status defaults, there is always a write to the response variable.

Info

If a switch is locked using the `Lock` action, the status bit `M1` of the response variable is set.

The status bits `OFF` and `REVISION` are also handled in other actions.

*In addition, the status bits `CB_TRIP` and `CB_TR_I` reflect the results from the property of the **breaker trip detection** **command group**.*

Action type replace

The process value of a remote-controlled switch is temporarily replaced with a replacement value (due to revision, maintenance work, or an ongoing connection outage, for example).

The response variable is set to the status alternative value **Alternate value** (ALT_VAL). In addition, the value defined by the **switching direction** is placed on the response variable.

Switching direction	Alternate value
`Off`	0
`On`	1
`Diff`	2
`Fault`	3
`None`	4

The substitute value is not sent to the connected hardware. It is for the substitution of values using manually-collected information.

Hint

By switching the response variables to substitute values, it is possible to portray the current topological status of the network in **ALC** whilst the SCADA system was disconnected from the process.

Action type revision

Sets the value for the Revision status bit of the response variable. The value is defined in the **Return state/switching direction** property.

Switching direction	Status
`Off`	Set to 0
`On`	Set to 1

Action type manual correction

The **correct** action sets the value of the response variable according to the setting of the **switching direction**:

Note: the communication protocols in Energy (IEC60870, IEC61850, DNP3) preclude direct writing to the response variable. The action will be unsuccessful in these drivers! To execute a command, the setting of the value to a command variable is expected.

Switching direction	Action
`Off`	0
`On`	1
`Diff`	2
`Fault`	3
`DIR`	The set value is written directly. You define the value which should be written with the help of function **Set value**. The text to be displayed can be configured using a limit value or a reaction matrix for the state/value 5. If this is not the case, a standard text (on page 88) is used. **Nominal/actual value comparison** is not supported. The action can be carried out several times in a row.
`Set value`	Value of the **Set value** control element is written to the response variable in the Command Processing screen.

Attention

When writing the set value directly neither the limits of the linked variable are checked nor is it checked if the write set value is allowed for this variable.

Information

*The **In progress** (PROGRESS) status bit is set if:*

- When the action is carried out, the current value of the response variable is different to the value set for the **switching direction**
  and
- the **switching direction** was defined as `on` or `off`.

MANUAL CORRECTION

Manual correction is the manual correction of a non-remote switch in zenon. A variable is usually corrected without a connection to the process. There should never really be an invalid i-bit pending for such variables. It is indeed possible, but it makes no sense to correct a variable with a reference to the process! The PLC will overwrite this value again.

Behavior:

***Correction** is completely normal value setting from the perspective of the driver.*

Opposite of this - Action: Replace (on page 106)

The process value of a remote-controlled switch is primarily replaced with a replacement value (due to maintenance work, for example).

Action type block

the response variable is switched off as a result of executing this action.

The status bit of the response variable is set to OFF. The switched-off variables are no longer read by the connected hardware.

If the response variable already has the OFF status bit set, the action of the status bit is no longer set when the action is executed. Runtime receives the current value of the response variable from the driver.

Note: Can only be configured once per command group.

Action type release

The **Release** actions resets the **replacement value** (ALT_VAL) status bit to `0` (inactive). If the **Switched off** (OFF) status bit is also active, it is also set to `0` (inactive). runtime receives the current value from the driver for the response variable once the **Release** action has been carried out.

The action can only be executed in Runtime, if the **replacement value** (ALT_VAL) (value: 1). is active for the selected response variable. `1`).

Note: Can only be configured once per command group.

Check response value action type

The **Check response value** action type is to check variables for the status `ON` or `OFF`.

Whilst the **Check response value** action is executed, the standard key **Cancel** is unlocked in the **Command Processing** screen.

In doing so - depending on the setting of the **runtime monitoring** (on page 151) - there is a wait until the value of the response variable corresponds to the value of the checking direction - **switching direction** action property. If the checking value is `EIN`, this is the value 1; it is the value 0 for `OFF`.

If no runtime monitoring has been configured (**runtime monitoring**= "`none`"), the set waiting time (~24 hours) is the maximum time that is waited. Otherwise the action is ended and the `TIMEOUT` status bit is set for the response variable.

If, after execution of the action in the Command Processing screen, the other actions are not available, this is for the following reasons:

- The **timeout** for **runtime monitoring** has not yet expired.
- The response variable does not yet have the expected value (the value change has not yet been received).
- The action has not yet been canceled with the **Cancel** button.

> **Information**
>
> *The **Check response value** action only serves to read the value of the response variable without executing an activity.*
>
> *The action is intended for use in the **Command Sequencer** module.*

If the response variable already has the value of the **switching direction**, the execution of the action is recognized as completed. The other buttons in the Command Processing screen are thus immediately available.

Note: If the response variable is set to `OFF` or `Revision`, the response value can nevertheless be checked.

Action type lock

Enables the lock of a response variable for the actions of the command processing.

Note:

- Can only be configured once per command group.

- **Interlocking conditions** are not supported for the action. Locks can always be executed.

> **Information**
>
> *If a switch is locked using the `Lock` action, status bit M1 is set.*

A prerequisite for this is that users have a **Lock code** configured in the **user administration module**. Locking or unlocking a response variable can only be done with the correct input of a **Lock code**.

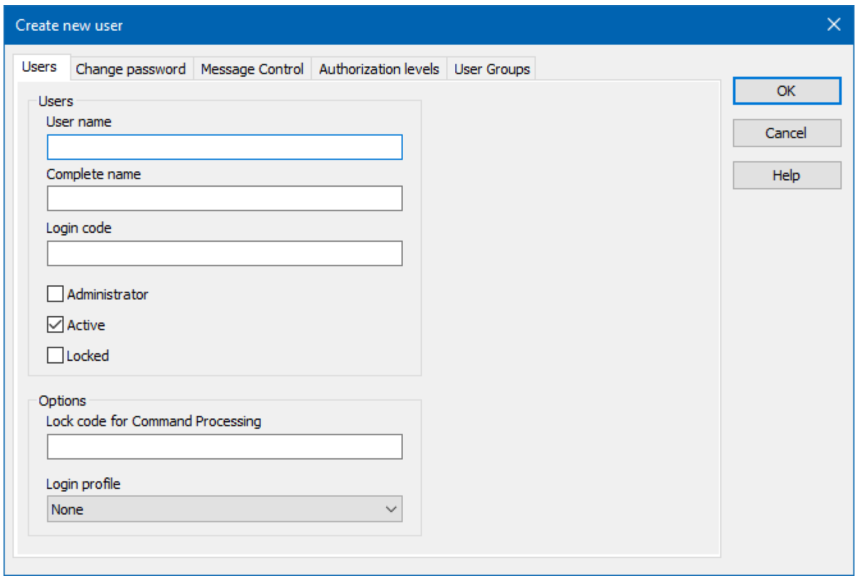
The same variable can be locked by multiple users in parallel. Actions for the response variables are possible only after alls locks have been unlocked by entering the **Lock code**.

There can be no actions executed if

- Actions of the command variables use the locked variable as response variable (e.g. `switching command`)
- Actions of the command variables use the locked variable as an action variable (e.g. `replace`)

A list of the currently-active locks can be shown in the command processing screen using a special `lock list` control element.

The **Lock code** can be defined individually for every user. These parameters are set directly for a pre-existing user with the **Lock code** property:


Create new user

Users | Change password | Message Control | Authorization levels | User Groups

Users
User name:
Complete name:
Login code:
☐ Administrator
☑ Active
☐ Locked

Options
Lock code for Command Processing:
Login profile: None

OK | Cancel | Help

You can also set the **Lock code** for an existing user in the Runtime.

In the Runtime you cannot delete users who still have an active command lock.

> **Attention**
>
> *Users can also be deleted in the development environment. This causes the loss of the defined locks after restarting or reloading.*

Users locked (activated) in the **user administration** cannot activate or deactivate command locks.

Information

Information about active locks is also synchronized in the redundant network and is therefore available after a redundancy switch.

Apply actions

Command Processing in the Energy Edition can be used in different situations. The user can choose the variant they prefer. A simultaneous use (related to an element) of the different types of use is possible at any time:

- Calling up a **screen switching** function on a **Command Processing** screen (on page 112).
- Calling up a **numeric value**, **combined element**, **dynamic text**, **bar graph**, **clock**, **universal slider**, **pointer instrument** or **status element** screen.
  For activation, the **Write set value via** property of the element must be configured with `Command Processing`.
- Call via a context menu if `Command Processing` was set for the **Action type** property. The command processing screen is opened for any possible interaction with the user (e.g. pending interlocking).
- It is called up using the **Command Sequencer module**.

Hint

Always link all screen elements or functions that call up the command input to a response variable.

Only by linking to a response variable is it ensured that all actions in the `command` screen are available for operation in in Runtime.

Linking to a command variable is expressly not recommended!

As soon as the variable is linked to a command group, direct input of set values is only possible using a zenon `command processing screen` or a `command` context menu.
Exception: If the command group contains a **setpoint input** action with **Return state/switching direction** '`set value`', this action is used for the command variables (not response variables).

This happens:

- When the **Write set value** is called up
- When calling up the screen element; also if the **Write set value via** property of the element has the value `dialog box` or `element`.
- When calling up a `set value` context menu.

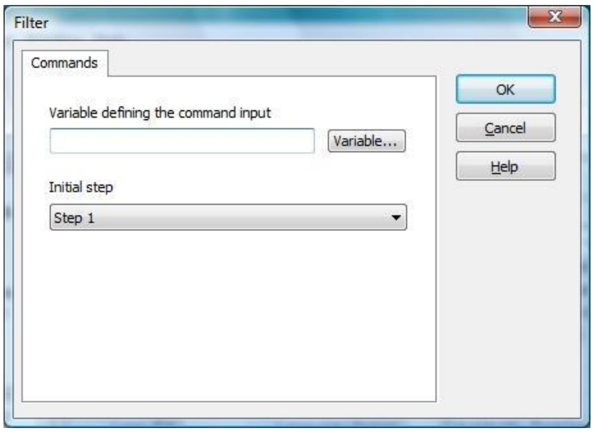
In doing so, the NET_SEL status bit of the response variable is not taken into account and no Select is executed.

Attention

With this type of execution, a pending interlocking condition in the **write set value** action prevents writing of a set value. In doing so, there is no interaction with the user.

Screen switch to screen of type Command Processing

If a Command Processing screen is selected with the **Screen switch** function, the configuration dialog for the screen switching function has the following parameters:



Variable defining the command input

Initial step: Step 1

Parameter	Description
Variable defining the Command Processing	The variable configured here defines the command group to be used. The screen determines the appropriate response variable and the associated action variable via the name of the variable.
Initial step	Defines the step (status) in which the command processing screen is loaded. ▸ `Step 1` The screen is loaded and waits for action definition and action execution. Action executions must be performed manually by the user. ▸ `Block` The screen is opened in the command step for the action block. **Note:** Note that not all configured control elements are visible with this initial step. You can find an overview of all visible control elements in the blocked or locked elements (on page 159) chapter.

Attention

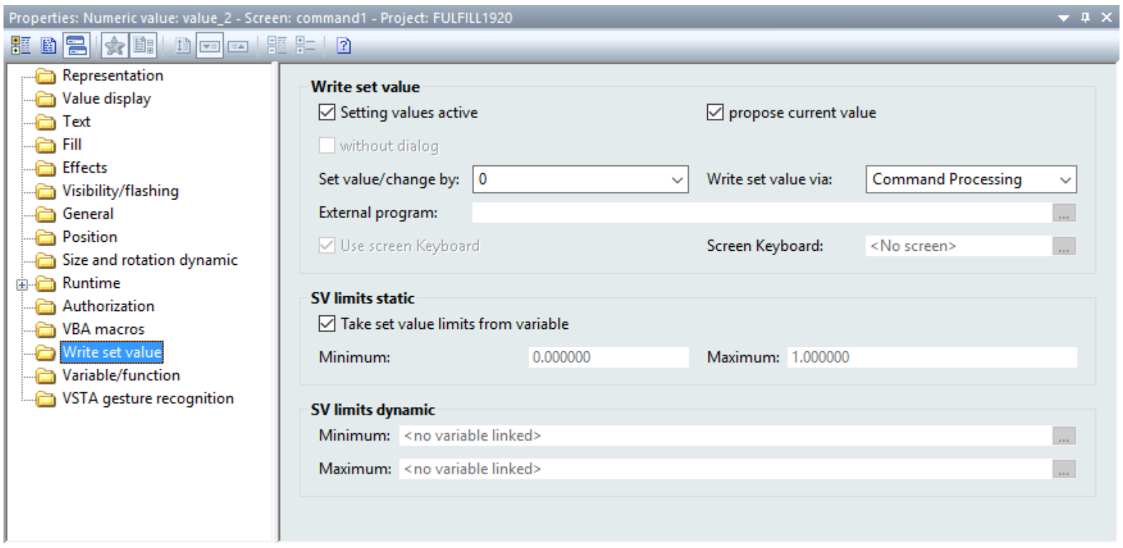
If there is no operating authorization for the command variable, screen switching to a `command` *screen is not possible.*

Command processing via dynamic screen elements

Command input can be instigated by clicking (left mouse button) on a dynamic screen element. In general, it is a combined element with a symbol of a switch in the topology or a numeric value element that displays the value of the position (`0` - off, `1` - on, `3` - invalid etc.) of the switch. The dynamic element should be:

- Configured with `command input` in the **Write set value via** property
- And linked to a response variable (an action variable for example) . That means to a variable that has been configured with a valid command group - using the **command group** property, as well as a suitable action variable (a response variable for example).

The screen to be opened - a `command input` screen - is defined at the **command group** of the variable linked to the element. The corresponding action variable (or response variable) is automatically determined from the response variable.



Properties: Numeric value: value_2 - Screen: command1 - Project: FULFILL1920

Write set value

☑ Setting values active ☑ propose current value

☐ without dialog

Set value/change by: 0 Write set value via: Command Processing

External program:

☑ Use screen Keyboard Screen Keyboard: <No screen>

SV limits static

☑ Take set value limits from variable

Minimum: 0.000000 Maximum: 1.000000

SV limits dynamic

Minimum: <no variable linked>

Maximum: <no variable linked>

Information

With the `Command Processing` *setting selected, the command processing screen is called up instead of the standard* `Write set value` *dialog.*

The following dynamic elements support the `Command` setting:

- Bar display
- Combined element
- Text element
- Clock
- Universal slider
- Numeric value
- Pointer instrument
- Status element

If no **command group** has been defined for the variable linked to the element, or the configuration of the command is invalid, an error entry for the Diagnosis Viewer is generated. The screen is not opened then.

Information

If a variable is linked to a command group, it is not possible to describe the variable with the zenon ***Write set value*** *function.*

Exception: *If a* `write set value` *(on page 104) command with* ***switching direction*** `set value` *has been created, the zenon function calls up this action in the background without the* `command processing` *screen being called up. This means that the* ***command conditions*** *(on page 124) are checked. An active interlocking condition prevents the writing of a set value. During the execution of an action, the* `NET_SEL` *status bit is not set and the* ***Select Before Operate*** *variable property is ignored.*

This is also applicable for the value entry of a variable that is linked to a dynamic element if `Element` *was selected for the* ***Write set value via*** *property.*

Command processing via context menu

The command processing can also be instigated at the element directly via the context menu - property **Runtime - context menu**. This is the most frequently used method. In this regard, the context menu is already the first step of the two-step action. For the second stage (**Execute** or **Cancel**), or an interlocking, a screen - which was linked for the action - may possibly be opened.

The menu must have an entry of the **command processing** action type. The display of the single action is defined automatically by the menu. The display of the actions can be influenced selectively, depending on the '**names**' of the menu entry.

When creating a new action in the Command Processing (on page 60), a menu ID corresponding to the action type and the switching direction for the **Action type** property is created and offered in the drop-down list. If the content corresponds to an ID defined as standard text for the action type and switching direction, the content is adapted if the action type or switching direction change.

To create a context menu for the Command Processing:

1. Create the desired actions in the command processing (on page 60)
2. In the properties of the context menu item select the **Action type** **Command Processing**
3. Select the desired action and switching direction via the drop-down menu with the **Menu ID** property
4. Give it a clear label in the **Text** property
   Note: If no entry is defined for **Text**, the field is automatically filled with the "**command processing**" label.

Attention

*The engineering of the **Text** property must be unique. If texts that are the same are given, further menu items with the same name are not displayed.*

Because automatically created menu items with the same action result in the same text, there are macros (on page 120) available for these.

The character sequence ID_CMD_AUTO is reserved for automatically created menu items. These must always be used with macros, because otherwise only the menu item is inserted.

When checking for duplicate entries the following rules apply:

- Manual menu points have priority over automatic ones.
- If it is the same type then the last entry has twice the priority.
- If a duplicate entry is found, a warning is set off in the log. This includes the menu ID and description. Automatically expandable entries have **<auto>** added to the ID.

ACTIONS FOR ACTION TYPE COMMAND PROCESSING

Action	Switching direction	Menu ID
`ID_CMD_AUTO`		This menu entry automatically shows all possible actions for an element, if no direct menu entry from the list is used already.
Pulse command	On (1)	ID_CMD_EBEF_ON
Pulse command	OFF (1)	ID_CMD_EBEF_OFF
Pulse command	NONE	ID_CMD_EBEF_NONE
Switching command	On (1)	ID_CMD_DBEF_ON
Switching command	OFF (2)	ID_CMD_DBEF_OFF
Switching command	NONE	ID_CMD_DBEF_NONE
Set value	NONE	ID_CMD_SVALUE
Set value	DIRECT	ID_CMD_SVALUE_DIR
Status input	NONE	ID_CMD_STATE
Status input	On (1)	ID_CMD_STATE_ON
Status input	OFF (0)	ID_CMD_STATE_OFF
Replace	NONE	ID_CMD_REPL_NONE
Replace	On (1)	ID_CMD_REPL_ON
Replace	OFF (0)	ID_CMD_REPL_OFF
Replace	DIST	ID_CMD_REPL_DEF
Replace	DIFF	ID_CMD_REPL_DIFF
Manual correction	NONE	ID_CMD_UPD_NONE
Manual correction	On (1)	ID_CMD_UPD_ON
Manual correction	OFF (0)	ID_CMD_UPD_OFF
Manual correction	DIFF	ID_CMD_UPD_DIFF
Manual correction	DIST	ID_CMD_UPD_DEF
Manual correction	DIRECT	ID_CMD_UPD_DIR
Block	NONE	ID_CMD_BLOCK
Release	NONE	ID_CMD_UNLOCK
Lock	NONE	ID_CMD_LOCK
Revision	OFF (0)	ID_CMD_REV_OFF
Revision	On (1)	**ID_CMD_REV_ON**
Mandatory command	On (1)	ID_CMD_FORCE_ON

Mandatory command	Off (0)	ID_CMD_FORCE_OFF
Mandatory command	NONE	ID_CMD_FORCE_NONE

NAME OF THE MENU ITEMS OF THE CONTEXT MENU

AUTOMATIC CREATION

Entries that were created using `ID_CMD_AUTO` automatically get a name according to the following pattern: 'Action name' plus 'Limit value text of the switching direction'.

MANUAL CREATION FROM TABLE

If the menu entries are created from the table, for every action under 'Display - Text' a text must be defined for the entry in the context menu.

`Names for the menu entries:`

- `Command`
- `Set value`
- `Status`
- `Replace`
- `Release`
- `Manual correction`
- `Block`
- `Lock`
- `Revision`

ACTION TEXTS

Action	Text
Pulse command Switching command	Text from the limit value text, according to the switching direction.
Manual correction Replace	If a switching direction (other than 'None') is defined, the text from the limit value text according to the switching direction is displayed.
Status	'OFF' or 'ON', depending on the set switching direction
Revision	Text from the limit value text, according to the switching direction.
Others	No special action text is displayed.

Example

Displayed text for a switching command with defined limit value:

'Command: switching direction ON'

Information

- All displayed texts are language switchable with the standard mechanisms.
  See also: Which texts are language switchable?
- All displayed menu entries are automatically sorted alphabetically.

The currently used command group is determined via the variable which is linked with the screen element. If no command group is assigned to the variable or if there is no response variable, the context menu is not displayed in the Runtime (an according error message is transferred to the Diagnosis Server).

Information

The menu entries of the command processing are displayed depending on the command group. The menu entry is showed only when the connected action exists. Consequently, if the variable of the element is the command variable, only ***the*** *actions for the command variable plus the action `Lock` can be displayed. Actions for the response variable are hidden automatically.*

AVAILABILITY CONDITIONS

The menu entries are only released when the corresponding actions are executable. The following conditions are requirements:

- All menu entries are locked if the `NET_SEL` status bit of the response variable is active.
- All menu entries are locked, when the response variable could not be determined.
- All menu entries are locked, when the response variable has no value and could not get a value within 30 seconds.
- All menu entries are locked on the Web Client without write access.
- Menu entries are locked when there is no connection to the Primary Server.
- The menu entry connected to the `Release` action is locked when the `ALT_VAL` status bit of the action variable is not active.
- The menu entry connected to the `Replace` or `Revision` action, whose switching direction matches the value of the action variable, is locked.

- All menu entries, except the one which is connected with the action `Lock`, are locked, when a change lock is active for the response variable.
- When the `REVISION` status bit of the response variable is active, the actions `Set value`, `Replace`, `Correct`, and `Command` are locked.
- As long as a watchdog timer, an edge generation or an SBO is active for the command group, all menu entries are locked. This results from the fact that the `NET_SEL` status bit also stays active.

Macros for the context menu

A macro is a defined character sequence that is replaced by a text when menu items are created in Runtime. Virtually all macros can occur more than once per menu item. They can also contain further macros as a result. In doing so, the expansion sequence must be considered. Macros are not case sensitive when configuring menus. If macros contain a macro as a result, the macro must be contained in capitals in the result. The entry is made with `$` as a prefix and suffix.

The sequence of the expansion is from left to right in the following priority.

1. **ALL**
2. **$NOTE$**
3. **TAG**
4. **$REMA<Status>$**
5. **$RDIR$**
6. **DIR**
7. **ACT**
8. **$NAME$**

Macro	Description
$ALL$	Results in **Action naming**: **Switching direction**. Corresponds to the combination of the **$ACT$:** macro **$DIR$** Note: If a context menu is created for the command processing, the default text is `$ALL$`, even if the menu already has text configured for it but the action type changes to command processing.
$NOTE$	The whole text including the macro is interpreted as a note. If the resulting text is empty, the **$ALL$** macro is used. For the last macro, the note macro is again checked and the text to the right of this including the macro is deleted. If the resulting text is empty or only consists of spaces, the menu item is not inserted.
$TAG$	Is replaced by the identification of the action variable. The identification can be translated by the online language translation function. If no translation character (@) is contained, the whole identification is highlighted for translation.
$REMA<Status>$	**<Status>** is a Rema or limit value state, the text of which is used as a replacement. If the status is not present, the menu item is not displayed. The limit value text is translated linguistically according to the placement of @ . The status can be a number between -2^{31} and 2^{31-1}. Leading characters and a prefix are permitted. If characters are contained that cannot be converted to a number, or the number is outside the given area, the menu item is not displayed.
$RDIR$	Text for the switching direction from reaction matrix/limit value as in **$DIR$ macro, with the exception of:** - Action `Write set value direct` The text is taken from the rema/limit value of the status, which corresponds to the value of the set point to be set. - Action `Status on` and `Status off` Text is taken from the rema/limit value for the `on` or `off` statuses. - Action `Correct direct` The text is taken from the rema/limit value of the status, which corresponds to the value of the set point to be set.
$DIR$	Switching direction of the action.
$ACT$	Action naming of the action.
$NAME$	The `$NAME$` macro can be used to create menus and provides the configured content of the **Action name** property, the language of which can also be switched in Runtime with a @ character.

AUTOMATICALLY CREATED MENU ITEMS

Automatically created menu items are created as a menu ID with ID_CMD_AUTO. In this case, macros must always be used, because otherwise only a menu item would be inserted.

COMPATIBILITY

Previous to version 6.51 text at automatic menu items was ignored. When converting projects that were created with versions earlier than 6.51, the macros $ALL$$NOTE$ are automatically inserted before the configured text. Therefore the menu items behave as before.

ONLINE LANGUAGE SWITCH

The labeling for the menu item in the **Text** property is translated linguistically before macro expansion from the character `@`.

Note: If, for the **$TAGS$** macro, no translation indicator (`@`) is contained, the complete text is translated.

Error messages when the context menu is called up

When menus are loaded in the Runtime environment, their content is checked for consistency. If an error occurs, corresponding error messages are issued for the **Diagnosis Viewer**. The following messages can appear:

Parameters	Description
Menu entry for command processing suppressed, because name is several times in the menu!	The menu already contains a menu entry with the name used in the command processing. Do not use that name for any other menu entries for the command processing.
Menu entry for command processing suppressed, because description is several times in the menu!	There is already a menu entry with the same description in the menu. Automatically created menu entries are not added, when a menu entry with the same description is already there.
Text for menu entry cannot be detected!	The description of an automatically created menu entry could not be determined. This most probably indicates a missing limit value text.
No command group linked to variable of the screen element!	The variable associated with the screen element has no command group or a no longer valid command group. According error messages are given during compiling.
Response variable does not exist!	The response variable used in the command group does not exist.
Select cannot be activated!	Status bit `NET_SEL` (8) could not be activated within the timeout.

Execution of actions via the context menu

After activation of a menu item for command processing, the assigned action is carried out. Execution via a menu activates the setting of the `NET_SEL` status bit in the first step. Only if this was successful is the execution of the actual action (a `switching command`, for example) started.

A command processing screen is then opened if one of the following criteria has been met:

- If the action to be executed is **Write set value**, `Status input` with input or `Correction`, the screen assigned to the action in the "Stage 1" step is opened. The status or the set value to be written can then be defined in the screen.
- If the action to be carried out is `lock`, the action-specific screen is called up with the `lock` step.
- If an active locking condition prevents execution, the screen configured in the **Unlocking** step for the action is called up. Execution is also prevented if **Select Before Operate** could not be activated without errors.
- If two-stage execution is configured for the action, the action-specific screen is called up in the "Stage 2" step.
- If no specific screen has been configured for the action, the screen that has been configured centrally for the command group is opened.

Information

If none of the above-mentioned conditions are applicable, the action is executed immediately, without further operations.

Set value context menu

If the variable assigned to a screen element is linked to a command group, the writing of a set value is also handled by the command processing. The requirement for this is that a `Write set value` action is present with **Return state/switching direction switching direction** in the Command Processing. If this is missing, the writing of the set value is not carried out.

Information

An active interlocking condition prevents the writing of a set value.

Command conditions

Command groups contain both the definition of the switch actions and the definition of the command interlocking conditions. Command conditions are optimum parameters that can be defined application-specifically.

Each action within a command group can also be supplemented with **interlocking conditions**. These process-controlled interlockings prevent unwanted execution of actions, depending on the current process state.

The following three parts are significant in a command group for the command conditions:

- The action for which the conditions were defined and for which the internal interlockings are also applicable.
  The actions define which command is executed, on which variables these actions are applied and set the parameters for the internal interlockings.
- The `condition variables`, listed in the **Variables** node of the command group.
  These define which variables can be used in the command conditions.
- The `command conditions`, created per action.
  These conditions contain the formulas on which the condition variables are based. This syntax is the same as the definition of the formulas in the Formula Editor. The execution of commands can thus also be made dependent on the current process status.

Information

Configured general interlockings have no influence on this check.

In addition to the command conditions, the following interlocking types are automatically checked before the action is executed:

- Internal interlocking conditions
- Topological interlocking conditions

INTERNAL INTERLOCKING CONDITIONS

These conditions are checked automatically before every action execution; the engineer cannot influence this. These Internal interlocking conditions (on page 127) are predefined by the system and serve as plausibility checks.

Example

Internal interlocking is applicable if:

- The response variable is already selected in the zenon network (has set `NET_SEL` status from other network client).
- The response variable already has the desired value and the action was configured with **Nominal/actual comparison**.
- In the SBO, the Select was rejected by the PLC (status bits: `SE_870` + `COT_actcon(7)` + `N_CONF`).

TOPOLOGICAL INTERLOCKING CONDITIONS

These conditions result from the current topological status during Runtime. These conditions are defined in the 'Configuration of the topological interlockings (on page 33)' settings of the project for **Automatic Line Coloring**.

Create command conditions

Any number of command conditions can be defined for every action. These conditions allow for an additional restriction of the ability to execute an action. These conditions are defined with formulas, in which you can use the variables from the active projects. The formula addresses the linked variables via the index in the condition.

Information

The condition variable is automatically replaced if a '' is used in the definition.*

DEFINE CONDITION VARIABLE

In the first step, the variables or substitute names of the variables must be configured. These are used later for the formulas of the command conditions. If the defined conditions are fulfilled by the linked process variables, the user has the respective actions available during Runtime.

Information

Variables that are used in a formula are:

- In an interlocking condition
- In the breaker tripping detection - **detection suppression**

*Cannot be removed - from the **Variables** of the command group node.*

ENGINEERING

A command condition is defined using formulas. Conditions that are not met cause the action to not be executed in Runtime or to initially have to be unlocked by the user. The user must have the corresponding **Authorization level for unlocking** for this.

The following procedure is recommended for defining a command condition:

1. Go to the **Variables** node in the detail view of command input.
   a) Select the **Insert variable...** entry in the context menu
   b) The variable selection dialog is opened
2. Select a process variable.
   This variable serves as the basis for the formulas of the command conditions.

   Optional:
   Create a replaceable definition:
   a) To do this, close the variable selection dialog by clicking on the **Cancel** button.
      An empty definition is created.
   b) In the input field of the **Interlocking Variable** property, enter the name with the ***** placeholder.
      Automatic substitution is configured as a result.
3. Go to an action that already exists.
4. Select the **New interlocking condition** entry in the context menu
5. Define the desired formula in the **Logical link** property.

Internal interlocking conditions

With the help of the internal interlocking conditions the basic requirements for the action are checked (plausibility check). The results, or the addressing of an interlocking, are displayed in Runtime in the `command processing` screen in the **interlocking text** screen element.

Parameter	Description
Status already exists	The state which should be set equals the current value of the response variable. This check is only active if the action has been activated for the **Nominal/current value comparison** property. This interlocking can be unlocked, provided the user has authorization to do this - in accordance with the **Authorization level for unlocking** property of the action.
Internal error occurred	Command Processing cannot execute the check. This happens when the data type of the action variable is not allowed for this action. Example: `Pulse command on` action for string variables. This interlocking cannot be unlocked.
No interlocking object	**Command group** cannot be determined (configuration error). This interlocking cannot be unlocked.
Action not defined	Action to be executed could not be determined (egineering error). This interlocking is not unlockable.
Differences between local and global interlocking	Pulse command parameter not consistent (configuration error). This interlocking cannot be unlocked.
One or more values are not available	Value of condition variable: - Not available (**interlocking code:** 14) - violated - status bit INVALID (**interlocking code:** 15) This interlocking cannot be unlocked.
Locking administration not valid	The administration of the `lockings` could not be loaded or is invalid. This interlocking is not unlockable.
Variable locked for changes	Command Processing locked by response variable (status bit `M1`). This interlocking is not unlockable. Note: You can both lock or unlock in the command input screen with the `Lock` action type.
SBO rejected	The activation of the Select has been rejected by the PLC. This interlocking is not unlockable. Note: Only the Energy drivers signal the rejection of the Select - the action variable gets the status bits `COT_actcon(7)` + `N_CONF` + `SE_870`.
Timeout for SBO activation	Within the configured **Timeout**, no confirmation, either positive or negative has been received for Select activation. This interlocking is not unlockable. Note: Only the Energy drivers support Select activation - the action variable gets the status bits `COT_act(6)` + `SE_870`.
Timeout for SBO deactivation	Within the configured **Timeout**, no confirmation, either positive or

	negative, was received for the deactivation (Cancel to the Select). This interlocking is not unlockable.
Timeout for execution	There was no notice for finishing the action execution within the engineered **Timeout**. The `TIMEOUT` status bit is set for the response variable. This interlocking cannot be unlocked. Note: the **Watchdog timer** of the **command group** determines what needs to be fulfilled before the action is completed.
SBO expired	The PLC has reported the expiration of the SBO activation. The second execution step will attempt to send a Select again. This interlocking cannot be unlocked. Note: The respective protocol determines whether a controller can report the Select timeout. If so, the Energy driver signalizes the process of the Select - the action variable gets the status bits `COT_actterm(10)` + `N_CONF` + `SE_870`.

Note: The numbers of the internal interlocking conditions are also shown in the **system driver** variable `[command] interlocking code`, if this variable has been created in the project.

Formula editor

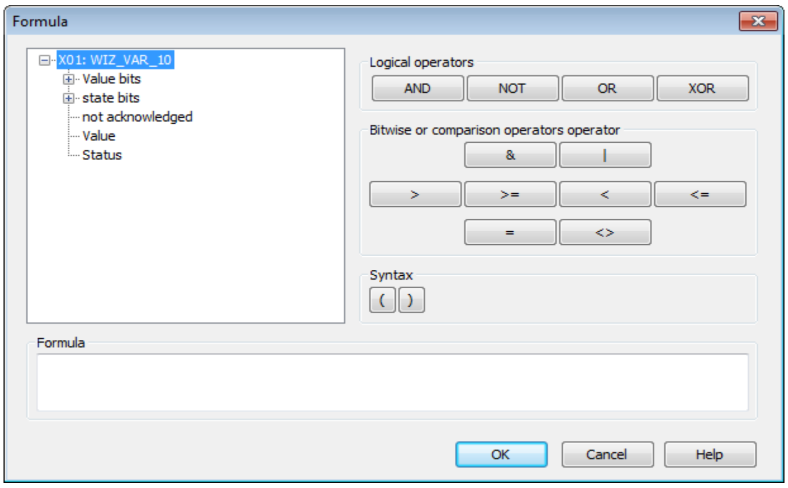
The formula editor provides support when creating formulas with logical or comparative operators with a combined element, for interlockings and command processing. If additional variables are required for a formula, create these in the **formula variables** area of the status window by clicking on the **Add** button. existing formulas are displayed in the status list with the letters**F** .

Note on the input of decimal points:

- Decimal separator: Comma (`,`) is automatically converted into a dot (`.`):
- Zero as a decimal point is removed automatically; `23,000` automatically becomes `23`

CREATING A FORMULA

Click on the **Formula** button in the status window The formula editor opens



You select the bits for your formula in the left screen.

On the right, you find the operators for logical and comparative operations.

The formula created is displayed in the **Formula** area.

Information

Up to 99 variables can be linked in one formula. X01 to X99. The length of the formula must not exceed 4096 characters.

THE MEANING OF THE BITS:

Parameter	Description
value bits	32 value bits (from 0 -31) are available. They describe the variable value bit by bit. For binary variables, only bit `0` is of importance, for SINT and USINT only the bits from `0-7`, etc. **Note:** The value refers to the raw value (signal range) of the variables and not to the converted measuring range.
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 (on page 133).
unreceipted	**Not acknowledged** is treated like a usual status bit. But here it is listed separately, because it does not belong to the classical variable statuses.
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. The total value and overall status are an exception to this. In order to arrive at a Boolean expression, this total value has to be ORed `bitwise` (on page 136) with a constant. For this, we use the operator `&`. For the result `0` (`FALSE`) of this logical ORing, we get the binary value `0` (`FALSE`), otherwise `1` (`TRUE`). Example: See the bitwise ORing example (on page 136) chapter

Info

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

If other settings outside the formula are set for the current status, they are combined with the formula with a logical AND.

Refer to the examples (on page 138) section for examples.

Information

Formulas with binary X values and bitwise linking can be used with a maximum of 2 binary values. If more values are required, the linking must be carried out without binary X values.

Example:

X01.Value & X02.Value -> works

X01.Value & X02.Value & X03.Value -> does not work

But:

X01.00 AND X02.00 AND X03.00 AND X04.00 AND X05.00 -> works

List of status bits

Bit number	Short term	Long name	zenon Logic label
0	M1	User status 1; for Command Processing: Action type "Block" (on page 109); Service Tracking (Main.chm::/IEC850.chm::/117281.htm) of the IEC 850 driver	_VSB_ST_M1
1	M2	User status 2	_VSB_ST_M2
2	M3	User status 3	_VSB_ST_M3
3	M4	User status 4	_VSB_ST_M4
4	M5	User status 5	_VSB_ST_M5
5	M6	User status 6	_VSB_ST_M6
6	M7	User status 7	_VSB_ST_M7
7	M8	User status 8	_VSB_ST_M8
8	NET_SEL	Select in the network	_VSB_SELEC
9	REVISION	Revision	_VSB_REV
10	PROGRESS	In operation	_VSB_DIREC
11	TIMEOUT	Command "Timeout exceeded" (command runtime exceeded)	_VSB_RTE
12	MAN_VAL	Manual value	_VSB_MVALUE
13	M14	User status 14	_VSB_ST_14
14	M15	User status 15	_VSB_ST_15
15	M16	User status 16	_VSB_ST_16
16	GI	General query	_VSB_GR
17	SPONT	Spontaneous	_VSB_SPONT
18	INVALID	Invalid	_VSB_I_BIT
19	T_STD_E	External standard time (standard time) **Caution:** up to version 7.50, this was the status bit T_CHG_A	_VSB_SUWI
20	OFF	Switched off	_VSB_N_UPD
21	T_EXTERN	Real time - external time stamp	_VSB_RT_E
22	T_INTERN	Internal time stamp	_VSB_RT_I
23	N_SORTAB	Not sortable	_VSB_NSORT
24	FM_TR	Error message transformer value	_VSB_DM_TR

25	RM_TR	Working message transformer value	_VSB_RM_TR
26	INFO	Information for the variable	_VSB_INFO
27	ALT_VAL	Alternate value	_VSB_AVALUE
28	RES28	Reserved for internal use (alarm flashing)	_VSB_RES28
29	N_UPDATE	Not updated (zenon network)	_VSB_ACTUAL
30	T_STD	Internal standard time	_VSB_WINTER
31	RES31	Reserved for internal use (alarm flashing)	_VSB_RES31
32	COT0	Cause of transmission bit 1	_VSB_TCB0
33	COT1	Cause of transmission bit 2	_VSB_TCB1
34	COT2	Cause of transmission bit 3	_VSB_TCB2
35	COT3	Cause of transmission bit 4	_VSB_TCB3
36	COT4	Cause of transmission bit 5	_VSB_TCB4
37	COT5	Cause of transmission bit 6	_VSB_TCB5
38	N_CONF	Negative confirmation of command by device (IEC 60870 [P/N])	_VSB_PN_BIT
39	TEST	Test bit (IEC870 [T])	_VSB_T_BIT
40	WR_ACK	Writing acknowledged	_VSB_WR_ACK
41	WR_SUC	Writing successful	_VSB_WR_SUC
42	NORM	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	External time stamp invalid	not defined
50	CB_TRIP	Breaker tripping detected	not defined
51	CB_TR_I	Breaker tripping detection inactive	not defined
52	OR_DRV	Value out of the valid range (IEC 61850)	not defined
53	T_UNSYNC	ClockNotSynchronized (IEC 61850)	not defined

54	PR_NR	Not recorded in the Process Recorder	not defined
55	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

Information

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

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

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 (on page 136))
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.

Bit formulas

Bit formulas only have a logical high or low state. In contrast to logical formulas, the raw value is already predefined (**0**,1).

Operator	Description
&	AND
|	OR

Example: ORing bitwise

You want to find out if one of the user status bits 1-8 (M1 ... M8) of the variable X01 is set.

USUAL FORMULA:

X01.M1 OR X01.M2 OR X01.M3 OR X01.M4 OR X01.M5 OR X01.M6 OR X01.M7 OR X01.M8
This query can be made much easier by the logical ORing of the overall status.

LOGICAL ORING

X01.Status & 0xFF

The constant can be entered in hexadecimals, as described above:

0xFF corresponds to decimal 255; these are the first eight status bits (binary 11111111). If one of these bit is set to 1, the result of this bitwise ORing is 1 (true), otherwise it is 0 (false).

If, for example, all user status bits except the user status bit M7 should be queried, the binary statement for this would be: 10111111. Bit 7 is not of interest and is thus set to 0. This corresponds to 0xBF in hexadecimal. The expression for the formula is then: **X01.Status & 0xBF**.

Instead of ORing bitwise with a constant, the value can also be directly compared to a decimal number. If the comparison is wrong, the binary value is 0 (false) otherwise it is 1 (true).

Example:

You want to find out if the value is equal to the constant 202: The formula is:

X01.value = 202

If the value is equal to the constant 202, the result of the comparison is 1 (True) otherwise it is 0 (False).

Note: The bitwise ORing works with the OR character (|), the same as in this example.

Comparison operators

Comparison operators are for the direct comparison of two numeric values. The result of this comparison is a binary value. "0“ if the condition is not fulfilled and „1“ if the condition is fulfilled.

Operator	Description
<	less
>	greater
<=	Less than or equal
>=	greater or equal
=	Equal
<>	unequal

To the left and to the right of the comparison operator, there has to be a (total) value or a (total) status, single bits cannot be used with these comparison operators.

There can also be a constant to the right of the comparison operator.
These constants are entered as hexadecimal values or decimal values in the combined element. Hexadecimal numbers are automatically converted to decimal numbers by clicking on **OK**. For example, 0x64 corresponds to the numerical value 100.

Note: The combined element is not available in the **Batch Control** module.

Example

X01.value >= X02.value
The result is 1, if the value of X01 is higher than or equal to the value of X02

X01.value = 0x64
The result is 1, if the value of X01 is exactly equal to the numeric value 100 (= hex 0x64)

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

Examples for formulas

SIMPLE LOGICAL AND LINKING BETWEEN TWO BIT VALUES

Example

Formula: X01.03 AND X02.03

This formula has the status TRUE, if both **bit 3** of variable `1` and **bit 3** of variable `2` both have the value `1`.

COMPARISON OF AN VALUE OR STATUS OF A VARIABLE

Example

(X01.Value> X02.Value)

COMPARE COMPARISONS TO ONE OTHER ON A LOGICAL BASIS

Example

(X01.Value> X02.Value) AND (X01.Value = X02.Value)

COMPARE WITH VALUE BITS AND STATUS BITS

Example

(X01.Value> X02.Value) AND (X01.Value = X02.Value) OR (X01.03 = X02.03)

COMPARE A VALUE WITH A DECIMAL OR HEXADECIMAL VALUE

Example

Formula: (X01.Value = 111)

Formula: (X01.Value = 0x6F)

If a hexadecimal values is used, this is later transferred to decimal by clicking on **OK**. If a decimal value is entered and confirmed, the value continues to be displayed as a decimal value after reopening.

Info

It is not possible to use a comma or a period when entering values.

4.3.4 Create menu

Command Processing can also be activated via a context menu. Context menus are created in the Editor using node **Menus** and are defined in the properties of the element they concern.

Generally there are three types of menu entries:

Parameter	Description
Action type	Sets out which type of action is to be carried out via the corresponding menu item in Runtime. Not all action types are available in the main menu, some are only available via the context menu. ▸ Acknowledge alarm (context menu only) ▸ Command processing(context menu only) ▸ Acknowledge flashing (context menu only) ▸ Function ▸ Help ▸ No action ▸ Write set value ▸ VBA macro (context menu only)
Submenu	Opens a sub-menu in Runtime.
Separator	A horizontal line divides menu entries.

Underline text: Entering a & causes the following characters to be displayed as underlined.

Plan entries

To configure a menu item in the main menu or context menu:

1. Activate the corresponding menu cell
2. In properties, select:
   - **Action type**: depending on menu type
     see also: Main menu action types and Context menu action types
   - **Menu ID**: ID of the entry
     Note: For **Command Sequencer** module entries, there are pre-defined types with prescribed IDs available.
     You can find further information about this and a list of these IDs in the Energy Edition manual in the Command processing via context menu (on page 115) chapter.
   - **Text**: clear labeling of the menu cells

> **Attention**
>
> *The engineering of the **Text** property must be unique. If texts that are the same are given, further menu items with the same name are not displayed.*

You can find details on the definition on context menus for command processing in chapter menusCommand Processing.

4.3.5 Create Runtime files

When creating Runtime files for the command groups, a check for engineering errors is performed. In addition, there is validation for the correct substitution of the variable names.

For each variable that has a command group assigned, the command group for the operation in zenon Runtime is instanced. In each instance, only the actions that can be triggered by means of this variable are now included.

Example

The command group for the command variable now contains actions for the respective command variable.

Exception: the "`Lock`" action is also available with command input.

Information

Changes to the configuration for variables require the project to be recompiled.

Substitution of variable names

In order to increase the reusabilty of the command group, there is the possibility to replace the variable references. Replacement is possible for the response, command and condition variable.

During the replacement, the placeholder (wild card *) is automatically replaced by the name of the variable that is assigned to the command processing.

EXAMPLE

The command group was configured in the **Variable name of response** property with the `*_RV` mask. In the project, the variables `xyz_RV`, `abcRV` and `bool_RV` are created and linked to the command group.

Variable	Part to be replaced	Result	Comment
`xyz_RM`	`xyz`	`xyz_RM`	Variable exists, assignment is possible
`abcRM`	`<empty>`	`_RM`	The mask is not correct, because the _ character is missing. The compiler will report an error. In Runtime, the operation of the abcRM does not trigger a command.
`bool_RM`	`bool`	`bool_RM`	Variable exists, assignment is possible

These rules are also applicable for the command variables.

Information

When compiling the command group, the text that corresponds to the part to be replaced is searched as follows:

- The text is stipulated in comparison to the mask with the name of the response variable.
- Otherwise the text is defined by the action variable. In doing so, the first action appropriate for the variable is applied.
- If the text for substitution was determined correctly, the placeholder * is replaced by this text.

Please note the following points in relation to this when naming variables:

- The names of the variables and the mask should be selected in such a way that these can be clearly assigned.
- The names of the variables that are used for the response, command and condition variables should be able to be created from the same replacement text.
- If the response variable is replaced (was defined with a mask) but not the command variable, particular care should be taken to ensure that the command group that is created for the command variable also uses the expected response variable.
- Additional validation of the response variable for the command group ensures that it only contains actions whose action variable (for its compiled command groups) uses the same response variable. Incorrectly-configured actions create a warning and are removed during compilation.

Error while creating Runtime data

At the creation of the Runtime data for the command processing, an extensive validation is carried out concerning wrong engineering and not-available references.

Information

After an `Error` the object the caused the error is not available during runtime.

If the command group has an `Error`, no command group is assigned to the variable. Consequently, during the Runtime, all user operations are locked.

A `Warning` is generated when the project would cause a problem but runs error-free.

In the error messages, the following placeholders are used:

<VERNAME>	Placeholder is replaced in the error message by the name of the command group.
<VERRM>	Placeholder is replaced in the error message by the name of the response variable.
<AUFVAR>	Placeholder is replaced in the error message by the name of the variable to which the command group is assigned.
<ACTVAR>	Placeholder is replaced in the error message by the name of the variable of the action.
<Actionname>	Placeholder is replaced in the error message by the description of the action.
<VARNAME>	Placeholder is replaced by the variable in the visualization.

The following error messages can occur during the creation of the Runtime files:

Message text	Description
<VERNAME>: Interlocking PV <VERRM> does not exist!	Condition variable for general interlocking not available.
Variable '<AUFVAR>' uses not existing command processing!	Variable uses a non-existing command processing.
(<AUFVAR>) command processing '<VERNAME>' contains no actions!	Command groups without action are not considered by the Runtime. This message can also be a follow-up error.
(<AUFVAR>) response variable '<VARRM>' does not use the command group '<VERNAME>'	A response variable using another command group is used. The response variable always has to be linked with the interlocking, which uses it as response variable.
(<AUFVAR>) response varibale '<VARRM>' for command processing '<VERNAME>' uses a driver without process linking!	The response variable must lie on a driver with process connection.
(<AUFVAR>Command processing '<VERNAME>' contains no actions after compiling!	A command group without actions does not make sense.
(<AUFVAR>) response variable '<VARRM>' of command processing '<VERNAME>' not available!	The used response variable is not present or marked as deleted.
(<AUFVAR>) command processing '<VERNAME>' uses screen '<Bild GUID>'(<BILDNAME>) which is not of the type Power!	This message is a warning. If a user action becomes necessary during execution, it cannot be performed.
(<AUFVAR>) command processing '<VERNAME>' uses not available screen '<Bild GUID>'!	The screen assigned to the command group does not exist.
(<AUFVAR>) Replaced action variable '<ACTVAR>' for action '<Action name>' of command processing '<VERNAME>' not available!	The action variable, after a replacement, is not present or marked as deleted.
(<AUFVAR>) action '<actionname>' of the command'<VERNAME>' uses the not existing variable '<ACTVAR>'	The action uses a varibale which is not present in the project or marked as deleted.
(<AUFVAR>) action variable '<ACTVAR>' for action '<Actionname>' of command processing '<VERNAME>' uses a driver without process connection!	The variable assigned to an action must not lie on an internal driver.
<VERNAME>(<AUFVAR>): Aktion '<Actionname>' already exists!	The following actions may only be configured once per action variable and command group: ▸ Switching command with the same command processing status. ▸ Correction with the same switching direction. ▸ Replacing with the same switching direction. ▸ Revision with the same switching direction.

	▸ Pulse command ▸ Setpoint input ▸ Release ▸ Block ▸ Lock
<VERNAME>(<AUFVAR>): Action '<Actionname>': Pulse and switching command not possible for the same command variable!	Pulse and switching command must not be used together.
(<AUFVAR>) Action '<Actionname>' of the command processing '<VERNAME>' uses screen '<Bild GUID>('<Bildname>') which is not of type Power!	This message is a warning. No user actions will be possible.
(<AUFVAR>) Action '<Actionname>' of the command processing'<VERNAME>' uses not existing screen '<Bild GUID>'!	The action is assigned to a non-exising screen.
<VERNAME>(<AUFVAR>): Interlocking PV '<VARNAME>' does not exist!	Replaced condition variable does not exist.
(<AUFVAR>) Variable '<VARNAME>' of action interlocking condition of the command processing '<VERNAME>' does not exist!	Variable of the interlocking condition does not exist.
(<AUFVAR>) action variable '<VARNAME>' for action '<Actionname>' of command group '<VERNAME>' uses another command group!	The action variables used for a command group may only be connected to no command group or to the command group in which they are used.
(<AUFVAR>) command variable <ACTVAR> does not have a validly compiled interlocking! Action <Actionname> removed.	The action variable used in the action has no compiled command group. This message can also be a follow-up error.
(<AUFVAR>) command variable <ACTVAR> uses response variable <VARNAME>! Action <Actionname> removed.	The compiled command group of the response variable contains actions with action variables which do not use the same response variable as <AUFVAR>. **Note**: There must not be any actions of response variables changing other response variables.

4.4 Operating during Runtime

A watchdog timer is automatically carried out in the background if a used enters commands in Runtime.

4.4.1 Execution of a command

This description for the procedure of a command with the command processing is applicable for the following action types of the Command Processing:

- Command (on page 101)
- Auto/Remote command (on page 102)
- Mandatory command (only in part) (on page 103)
- Setpoint input (on page 104)

Information

You can find information such as the execution of a command that influences the display and availability of the control elements in a `command input` screen in the Process in the command processing screen (on page 156) chapter.

The procedure of a command depends on the following parameters:

VARIABLE PROPERTIES

- **Select Before Operate** (SBO) inactive:
  In this case, a separate Select command is executed by an action.
  Note: A driver (for example: DNP3_NG or IEC850) can nevertheless execute a Select automatically. This has no effect on the command processing however. The Command Processing will, in this case, react to an unsuccessful Select in the same way as an unsuccessful Operate/Execute .
- **Select Before Operate** (SBO) active:
  In this case, a Select (`COT_act(6), SE`) is always forwarded to the driver by the command processing. In doing so, there is a wait - regardless of the type of **Watchdog timer** configured - until the complete command process (including Operate) has been completed.

  Attention: The **Select Before Operate** variable property has corresponding effects on **Watchdog timer**.
  If **Select Before Operate** is activated, the action buttons (and context menu entries) are deactivated in runtime monitoring for each configuration until `COT` contains the value `COT_accterm(10)`.
  This also applies if "`none`" or "`via response variable`" are configured for the watchdog timer.
  If no `COT_accterm` is received, only a `TIMEOUT` status bit is set.
- **Cancel Operate** active:
  If activated, with command input, after an Operate, a Cancel can also be sent to the controller, not just after a Select. Both the execution of the command in the controller and ongoing runtime monitoring can thus be ended early. The property is automatically treated like activated **Timeout can be canceled** in the action.

Info

If a variable is configured with an active **Select Before Operate** and the driver does not support a **COT** , then no reaction to a sent Select will be received from the PLC. Once the configured **Timeout** has expired, the command processing screen will inform you of the "**Timeout on SBO activation**" interlocking condition.

TYPE OF WATCHDOG TIMER:

The **Watchdog timer** property in the command group determines which conditions need to be met in order to conclude the action as successful. In addition, the **Watchdog timer** property determines how long the command input remains active in order to possibly send a deactivation (Cancel) to the PLC, to update the status bits of the response variable and to guarantee forwarding in the `auto/remote command`.

- `None`
  - without **Select Before Operate**:
    Direct command to the controller ("fire & forget") - sends the command to the controller and does not wait for a response.
  - with **Select Before Operate**:
    There is a wait in the background until the COT process has been completed in full. No `TIMEOUT` status bit is set, even if a **Timeout** has expired.
- `Via cause of transmission only (COT)`
  The action has been successfully completed after a complete COT process - if `COT_actterm(10)` has been received.
- `Only via response variable (RV)`
  The action has been successfully completed if the value of the response variable corresponds to the **response status/switching direction** in the action.
  - with **Select Before Operate**:
    There is a wait in the background until the complete COT process has been completed. Even if a **Timeout** has expired, no `TIMEOUT` status bit is set.
- `Via RV and COT`
  The action is successfully completed if both of the above-described conditions have been met.

Note: You can find further details in the runtime monitoring (on page 151) chapter.

PROPERTIES IN THE COMMAND PROCESSING ACTION:

- `One-step`:
  After a successful Select , the command processing automatically transfers the Operate/Execute to the driver.

- `Two-step`:
  A successful Select activates the buttons `Execute 2nd  Step` and `Cancel` in the command processing screen:
  - If `Execute 2nd  Step` is clicked on, the Command Processing transfers an Operate/Execute (`COT_act(6)` without `SE`) to the driver.
  - If `Cancel` is clicked on, a Cancel(`COT_deact(8)`, `SE`) is forwarded to the driver via command processing.
- **Timeout**:
  The configured value states how long is waited for a response from the PLC. Respectively:
  - After writing an Select command.
  - After writing an Execute command.
  - After writing an Cancel command.
- **Timeout can be canceled**:
  If this property has been activated, the user can also cancel a **Watchdog timer** that is still running, for example during an attempt to synchronize the frequencies of the electrical grids in the controller. The **Cancel** button in the command screen remains operable whilst the Execute command is executed. Depending on the configuration of the **Cancel Operate** variable property, it is also possible to inform the PLC of the cancellation.
  Note: not all Energy protocols offer a technical possibility to cancel an Operate. It is thus not possible for every Energy driver to forward a Cancel to the PLC.

Select before Operate

Select before Operate is a procedure in Energy protocols (such as DNP3, IEC61850 or IEC60870-101/104 for example). A "reservation/pre-assignment", the `Select` command, is first sent to the PLC. Only if this pre-assignment of the PLC is successful is the corresponding `Execute` command sent via the driver (using a command).

zenon Energy drivers update the following status bits in the process (by means of command variables):

- `SE_870, COTx`
- With the corresponding `N_CONF` configuration.

These status bits and NET-SEL cannot be configured by the person configuring the project. The status bits are set by the corresponding driver of zenon modules.

The **Watchdog timer** property only has an influence on the Execute command. The Select command is not influenced by **Watchdog timer**. After sending a Select, the command action waits for the time period configured in the **Timeout** property **Options**properties group in the command action).

In doing so, this can lead to the following dependencies:

- SPS does not react (on page 149)

- SPS reacts negatively (on page 150)
- SPS reacts positively (on page 150)
    - User does not send an Execute or Cancel (on page 151)

CANCELING A SELECT

For canceling (Cancel) a Select - regardless of the type of **Watchdog timer** engineering - the following is applicable:

1. If the **Select Before Operate** property is activated for the action variable, a **Select** for **two-stage** commands (**two-stage** command action property activated) can be canceled by a user. After a successful Select (COT=7, SE), the Command Processing can send a Deactivation (COT=8, SE) if the **Cancel** button is clicked.
2. There is then a wait for a response from the PLC or a **Timeout**.
   The action ends:
    - if the configured **timeout time** has expired
    - if the PLC confirms the cancellation (COT=9, any SE or PN).
      The receipt of a COT=9 discards the PROGRESS status bit.
3. The measurement time starts when the user clicks the **Cancel** button. The time period that has expired - whilst the action waits for a Select - does not influence the time period in which the action waits for a confirmation of the cancellation.
4. The response variable does not receive a TIMEOUT status bit.
5. The same action always applies, regardless of the configuration of the **Watchdog timer** property.

Note: the `PN` bit - Positive(0)/Negative(1) - is reflected on the status bit `N_CONF` of the action variable.

SBO - no reaction from the PLC

The command processing first sends a Select to the driver. All buttons in the `command processing` screen are grayed out while the action waits for a reaction from the PLC.

Information is shown in the command processing screen once the **Timeout** action has expired. This message is displayed in the field of the interlocking text. `Internal interlocking condition "Timeout for SBO activation".`

In this case, only the **Cancel** button is available. All other buttons of the Command Processing screen are still grayed out.

Entries in the context menu are not available.

Note: The response variable contains neither a `TIMEOUT` nor a `PROGRESS` status bit.
The same action applies for Select , regardless of which value is configured for **Watchdog timer** . The process is the same for two-step actions and one-step actions.

SBO - negative reaction from the PLC

The command processing triggers the driver to send a Select (`COT_act(6) + SE`) . If a negative response is received by the PLC (`COT_actcon(7) + SE + PN=`1), waiting is no longer carried out.

- The `PROGRESS` bit is removed by clicking on the **Cancel** button. The response variable does not receive a `TIMEOUT` status bit in the process.
- If the action receives a negative response to Select, this information is displayed in the field of the interlocking text: `internal interlocking condition "SBO rejected"`.
  In this case, only the **Cancel** button is available. All other buttons of the Command Processing screen are grayed out.

Entries in the context menu are also not available.

The same action applies to Select , regardless of which value is configured for **Watchdog timer**. The process is the same for two-step actions and one-step actions.

Note: the `PN` bit - Positive(0)/Negative(1) - is reflected on the status bit `N_CONF` of the action variable.

SBO - positive reaction from the PLC

If the PLC reacts positively to a Select (`COT_actcon(7) + SE`) (thus a confirmed Select), the action goes to the next step => execution

- The command processing ends the waiting for a Select.
- The response variable has its `PROGRESS` status bit set. However this is only if the current value of the response variable is different to the value of the command.

The following applies once the Select has been confirmed:

a) One-step commands:
   The command processing automatically sends an Operate/Execute (`COT_act(6)`, no SE) to the PLC.

b) Two-stage actions:
   The **Execute 2nd Stage** and **Cancel** become active in the command screen. If a user confirms the **Execute 2nd step** button, the command processing sends an Operate/Execute to the PLC.

The `PROGRESS` bit is reset if the PLC confirms the Execute (`COT_actcon(7)`) or after expiry of the following **Watchdog timer**.

SBO - positive reaction from the PLC but the user does not send an Execute or Cancel

If the user, after a successful Select , triggers neither an Execute nor a Cancel , there is a wait for user interaction with no time limitation. It can thus happen - provided the PLC supports **Select Timeout** - that this time expires. If this happens, the PLC sends a Select-Termination (`COT=10, SE, PN=1`). The command processing reacts to the Select-Termination received so that - if the user does in fact trigger an Execute - the command automatically sends another Select first.
Once the **Cancel** button in a command processing screen has been clicked on, a Select or Cancel is not sent to the PLC again.

Note: Is only relevant for two-step actions.

Watchdog timer

The simplest runtime monitoring, not envisaged for Energy drivers, is carried out if:

- The **Watchdog timer** property in the command group is `none` or `only via response variable (RV)`;
- Neither the **Select Before Operate** property (SBO) nor **Cancel Operate** has been activated.

For this configuration, the runtime monitoring checks the interlockings and writes a command to the controller. After this, the runtime monitoring waits to see whether the response variable changes the value according to the command.

In conjunction with Energy drivers (for **Select Before Operate** or `COT` process), the cause of transmission (COT) is used to exchange information between zenon and the controller about whether a command is to be written or whether writing was successful. With Energy drivers, the action variable gets a `COT` according to the stage of the command. In the background, the command processing then checks to see if the response variable then changes its value and if the `COT` action variable changes according to the command.

Note: Value changes of the response variable will only be taken into account after `COT_act(6)`.

Information

`COTx` Status bits result in a value. This value can be evaluated in Runtime - just like all other status bits - using multi numeric or multi binary reaction matrices.

Note: *`COT` is supported not only by IEC870, but also by some other Energy drivers - different versions thereof. Some drivers support COT although the protocol itself does not contain COT (e.g. IEC850, DNP3). You can find details in the corresponding driver documentation.*

RUNTIME MONITORING AND INTERLOCKING CONDITIONS OF THE ACTION:

Interlockings are checked during runtime monitoring:

1. For direct execution (= without **Select Before Operate**).

2. When the Select is activated (= before Select).
3. After the Select OK - before Execute.

- The current command action shows information about active interlockings in the `interlocking text` control element. The user must unlock these. To do this, the user must have the corresponding user permissions. In addition, the interlocking must also be configured as unlockable. Otherwise this action can only be canceled.
- If this action is canceled by a configured interlocking, no command is sent to the PLC.
- If a Select has already been sent, cancellation is automatic. In this case, a deactivation (Cancel) is sent.
- The response variable does not receive a `TIMEOUT` bit in the process. If a `PROGRESS` bit is already set, this is reset.

WITH ONE-STEP CONFIGURATION OF THE ACTION:

If there is an interlocking in the first stage, the `interlocking text` is displayed and there is a wait for a reaction from the user. If the user selected **On** or **Off**, the **Confirm** button will be active. The value is then sent to the PLC. If the user clicks on **Cancel** and a Select has already been carried out, the action sends a deactivation (Cancel).

WITH TWO-STEP CONFIGURATION OF THE ACTION:

The two-step action checks the interlocking and provides a message during the first step:

- Direct execution - no **Select Before Operate** :
  Applicable for the moment when the first button is clicked
- Active **Select Before Operate**:
  Applicable for the moment if the confirmation for Select is received or the **Timeout** for Select has expired.

By clicking on the button to unlock the interlocking, the ***Execute 2nd step*** *button becomes available. If the button is clicked on, an Execute is sent to the PLC. If* ***Cancel*** *has been clicked on and a Select has already been sent, the action sends a deactivation (`COT_deact(8)+SE`) to the PLC.*

If there was no outstanding interlocking for the first stage before the conditions have changed and before the user has carried out a ***Confirm*** *by clicking, the action will check the interlocking conditions again. If there are then still interlockings pending, a message is displayed and there is a wait for a cancellation from the user.*

Note: *In this case, the interlocking cannot be unlocked by a user.*

Information

Single-step actions have the same action in all scenarios.

RUNTIME MONITORING AND NEGATIVE RESPONSES FROM THE PLC:

- `Execute` negative
  During watchdog timer, the PLC can react negatively to an Execute/Operate `(COT_actcon(7) + PN)`. If the negative ration to an Execute was received, runtime monitoring is ended.
- `Execute Termination` negative
  If Execution Termination `(COT_actterm(10) + PN)` is reported as negative by the PLC, runtime monitoring will no longer wait for a value change to the response variable and ends immediately.

The `PROGRESS` status bit is reset. The `TIMEOUT` status bit is not set for the response variable.

Note: the `PN` bit - Positive(0)/Negative(1) - is reflected on the status bit `N_CONF` of the action variable.

Cancellation of runtime monitoring

If the **Timeout can be canceled** property has been activated for a command processing action, the user can cancel runtime monitoring - the second stage of execution (Execute) - with the "**Cancel**" button.

In doing so, the command processing sends the cancellation to the driver. The driver (the **IEC850 driver** for example) only forwards a Cancel Request to the PLC if the **Cancel Operate** property has been activated during configuration. You can find this property in the **Write set value** variable property group.

If the **Cancel Operate** property is activated for the command variable, the user can also cancel the runtime monitoring with the **Cancel** button in the **command input** screen. It is automatically considered an activated **Timeout can be canceled**.

Note: Not all drivers - that support the cancellation of a Select - also support the cancellation of an Execute. You can find further information in the respective driver documentation.

Runtime monitoring via response variable only (RV)

The **Watchdog timer** via the response variable is the most-used type of runtime checking. This reacts to a change of the value of the response variable. The value which is expected from the response variable as a result of the command defined in the property **Return state/switching direction** (`on/off`).

Negative responses from the PLC `(COT_actcon(7) + PN)` end runtime monitoring.

Note: Changes to the value of the response variable are only taken into account after `COT_act(6)` of the action variable. Early value changes of the response variable are ignored. This can occur, for example, if there are already value changes for the response variable after a Select without an Operate/Execute being sent.

Runtime monitoring via cause of transmission only (COT)

With **Watchdog timer** `via COT only`, runtime monitoring does not react to the value of the response variable (RV) but only to the cause of transmission - the status bits `COTx` status bits of the action variable.

EXAMPLE WITHOUT SBO:

The process in detail:

1. The value and `COT_act(6)` are sent to the action variable.
2. The PROGRESS status bit is sent to the response variable.
3. If the controller receives the value `COT_act`, there is a wait for the subsequent values `COT_actcon(7)` and `COT_actterm(10)`.
4. End of the process:
   a) The process is ended if `COT_actterm` has been received.
   b) If, in the configured timeout time, no `COT_actterm` has been received, then:
   - the process is ended and
   - the TIMEOUT status bit of the response variable is activated.
   Note: You configure this time in the **Timeout** property for the command action.
5. The PROGRESS status bit is reset.

Runtime monitoring via COT (cause of transmission) and RV (response variable)

With **Watchdog timer** `via COT and RV`, the runtime monitoring reacts to the value of the response variable (RV) and to the cause of transfer - the `COTx` status bits of the action variable.

EXAMPLE WITHOUT SBO:

The process in detail:

1. The value and `COT_act(6)` are sent to the action variable.
2. The PROGRESS status bit is sent to the response variable.
3. If the controller receives the value `COT_act`, there is a wait for the subsequent values `COT_actcon(7)` or `COT_actterm (10)`.
4. End of the process:
   a) The process is ended if both conditions have been met:

- `COT_actterm` was received

  and
- The value of the response variables corresponds to the **switching direction** (**Return state/switching direction** property).
  It does not matter which of the two conditions is met first. As soon as both of them are fulfilled, the procedure will be terminated.

a) If only one or none of the above conditions from item 4a is met within the configured **Timeout**, then:
- the process is ended and will be terminated and
- the TIMEOUT status bit of the response variable is activated.

5. The PROGRESS status bit is reset.

4.4.2 Screen type Command Processing

A **command processing** screen allows control in Runtime and an overview of the command processing. The command processing can be controlled via buttons. Templates with different appearances are provided.

ENGINEERING

Steps to create the screen:

1. Create a new screen:

   In the tool bar or the context menu of the **Screens**node, select the **New screen** command.
   An empty `Standard` screen is created.
2. Change the properties of the screen:
   a) Name the screen in the **Name** property.
   b) Select `Command Processing` in the **Screen type** property.
   c) Select the desired frame in the **Frame** property.
3. Configure the content of the screen:
   a) select menu item **Control elements** from the menu bar
   b) Select `Insert template` in the drop-down list.
   The dialog to select pre-defined layouts is opened. Certain control elements are inserted into the screen at predefined positions.
   c) Remove elements that are not required from the screen.

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

4. Create a screen switch function.

Process in the Command Processing screen

The `command input` screen has a multi-stage process, which consists of the following steps:

1. Initialization
2. Step 1
   Specifying the action
3. Unlocking
   Checking the interlockings and requesting a Select
4. Step 2
   Executing the command
5. Waiting until execution is complete

And an independent step - "lock"

Depending on the step in which the process is currently in, the control elements are updated, opened or hidden in the screen.

Note: These steps are shown in the `[command input] screen step` **system driver** variable.

INITIALIZATION

This step installs the internal process administration. There is then an immediate switch to "Step 1", without user interaction.

Initialization is executed regardless of the reason for switching when the screen is opened:

- The response variable and the command variable (if this is the switch variable) is requested by the driver. In doing so, there is an evaluation to see whether the data points exist. A LOG entry is generated in the event of an error.
- The `NET_SEL` status bit for the response variable is activated.
- Settings for the display are set:
  - The **Screen modal** properties and the title of the screen - from the **Screen title from response variable** property - are set accordingly.
  - The buttons without an assigned action are hidden.
    You can read more about this in the Blocked or locked elements (on page 159).

STEP 1

The action to be executed can already be specified in this step. This can then be the case if the user has selected the action from a context menu. Even if the command screen is still open due to a previous (already-executed) action, it remains specified. If no action has been specified yet, the action to be executed is determined automatically. The result of this determination of the action to be executed is shown in the `Active action` control element.

The switch to the next step, "unlocking" is carried out:

- By clicking on an action button:
  The user thus specifies the action to execute manually.
  Example: **On/Off** (**Screen type specific action**: `Action N`).
- By clicking on the **Execute** button (**Screen type specific action**: `Execute`);
  The button relates to the action that is currently specified.

The action to be executed is determined by calling up the command input screen (by clicking on a screen element, for example). The action to be executed is then not yet specified, because the command input screen can include several actions. The result of the determination depends on the control elements configured in the screen and the actions that are linked to the switch variable.

The determination is carried out in accordance with the following criteria:

1. `Setpoint input`
   If a control element has been configured for the entry of `action variable value` or `action variable value (slider)` in the screen and the `setpoint input [set value]` action exists for the switch variable.
2. `Status input`
   If a `set status` control element has been configured in the screen and there is the `status input [none]` action for the switching variable.
3. `Lock`
   If the screen has been opened using the **Screen switch** function with the **initial step** `lock` property and the command group contains the `lock` action.
4. If no action could be determined from any of the tests, the screen is opened without an active action. The control elements for entry and also the **Execute** button are then deactivated.

Note: As a result, you can call up certain `command input` screens you have configured yourself for certain actions and the focus is placed on the respective relevant screen.

UNLOCKING

The step is activated directly when operated via the **context menu**.

In this step, a check is carried out to see whether there are interlockings active. Only if no active interlocking has been detected is a switch to "Step 2" possible.

At the same time, there is a Select request to the controller if the **Select Before Operate** property is active for the command variable. There is a wait until the PLC confirms the Select or the **Timeout** defined for the action has expired. In the event of a timeout or rejection of the Select, a corresponding, non-unlockable interlocking message is shown.

If there are interlocking conditions, the step is only left once all interlockings have been unlocked. If logging in the CEL has not been deactivated for the action (**Suppress CEL entry** property), unlocking is recorded in the CEL.

When the **Cancel** button is clicked on. (**Screen type specific action**: `Cancel`) a switch back to "Step 1" is made. This can be necessary if there is a non-unlockable interlocking active. A Select that is already active is then deactivated (Cancel).

STEP 2

In this step, the fundamental command (Operate) is sent to the controller.

If the **Confirm** (**Screen type specific action**: `Execute 2nd stage` button is clicked on and

- there is a valid Select ,
- no interlocking condition active,

execution of the action is started. If there is a signing configured with this button, execution is started after successful signing.

Status bits of the response variable:

- The `PROGRESS` bit is activated
- The `TIMEOUT` bit is deactivated if a previous action had activated this status bit.

STEP 2 FOR TWO-STAGE ACTIONS

With two-stage execution, there is always a wait for confirmation from a user. Only once the user has clicked in the **Confirm** button is another check for interlockings carried out. If interlocking conditions have become active in the meantime, execution is not started. In this case, there is a return to "Step 1" by clicking on the **Cancel** button and the command input is carried out again.

If the `NET_SEL` bit is no longer active (on the network client after redundancy switching for example), an error message is logged. The action is not executed.

If the Select in the controller has expired (command variable has received the status bits `SE_870` + `COT_actterm`(10) + `N_CONF` in this case) a Select is requested from the PLC again. The action is carried out after a positive confirmation. If, in the process, the repeated SBO activation fails, a message is shown accordingly.

If, instead of the **Confirm** button, the **Cancel** button is clicked on, there is a switch back to "Step 1". A Select that is already active is then deactivated (Cancel).

WAITING UNTIL EXECUTION IS COMPLETE

In this step, there is a wait until the action has been completed in full. The duration of the execution depends on the driver (or rather the controller) and the configuration. Negative responses from the controller cancel execution; a negative response to an Operate, for example.

Execution waits until all of the following points have been fulfilled:

- Configured **Select Before Operate** for the command variable:
  If Select has been confirmed positively by the PLC, there is a wait until Termination.
- Conclusion of the configured **Watchdog timer** (if not all configured with `none`).
- Conclusion of edge generation for the `pulse command` action.
  Note: The **Edge delay** is not executed if **Select Before Operate** has been activated for the command variable.

If execution has been completed and the **Close automatically** setting has been configured for the action, the screen is closed. Otherwise there is a switch to the "initialization" stage.

If runtime monitoring has ended, the following applies for the status bits of the response variable:

- The `PROGRESS` bit is deactivated
- The `TIMEOUT` is activated if runtime monitoring has been ended with an error.

LOCK

This step is activated if the screen has been called up with the **Screen switch** function and with the **initial step** `lock` in the process. What is special about this step is that some control elements are not displayed. Only the control elements from the response variable and lock groups are visible.

Note: the locking/unlocking functionality is also available in other steps if corresponding control elements have been placed in the screen.

If, for the `lock` action, the **Close automatically** has been configured, the screen is closed once the **Lock** or **Unlock** buttons are clicked on.

This step cannot be switched to another step.

Blocked or locked elements

Some requirements must be met in order for the entries in the **context menu** and the control elements in the screen to become active. Because these mostly concern several elements, a summary of these is documented here.

the entries in the context menu and the control elements in the opened screen are locked up to **Close** if:

- Another screen is already the current owner of the active `NET_SEL` status bit of the response variable (through a network client, for example)
- No command processing was configured for the add-on variable
- the response variable does not exist
- The response variable has not received a value yet
- The `INVALID` or `OFF` status is active for the selected variable
  Exception: `forced command`, `replace`
- The response variable was locked for command processing:
  - the status bit `M1`, i.e. the command lock, of the response variable was set

  Exception: the control elements for the `Lock` action
- An action is running for the action variable and runtime monitoring has not yet been completed
  Exception: The **Cancel** button can be active here - regardless of the configuration.
- There is a wait for the SBO confirmation from the Select (SBO)
- the data of the lock are being transmitted
- the data of the lock are invalid
- the currently-registered user does not have the necessary authorization levels.

UNLOCK

Unlocking is only possible if:

- The authorization level of the user who is logged in allows execution
- In accordance with the configuration of the interlocking:
  - For command condition:
    if property **Unlockable** was activated
  - for topological interlocking
    If, in the **ALC configuration**, the `unlockable` status has been selected.

LOCK CONTEXT MENU ACTION

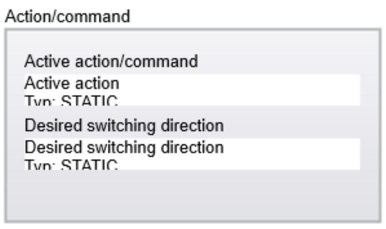
If the screen is called up with the `Lock` command action, only the following control elements are visible in the screen:

- Response variable (name)
- Response variable (identification)
- Response variable (value)
- Response variable (status)
- Response variable (measuring unit)

- Lock
- User
- Apply lock code

Other control elements in the screen are hidden.

Control elements - group: Action/command


Action/command

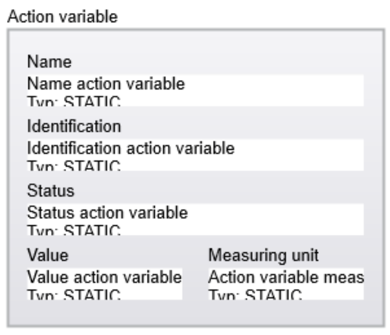
Active action/command
Active action
Typ: STATIC

Desired switching direction
Desired switching direction
Typ: STATIC

Control element	Description
Active action/command	Type of activated command action (**Action type**). Examples: `Switching command`, `Check response value`).
Switching direction	Value for the **Return state/switching direction** property configured for the active action. Depending on the active action, the following text is shown: - Command - Revision - Manual correction - Replace Text from limit value, depending on switching direction. - Status: On or Off - Other: empty

Note: The information is also displayed in the `[command] identification of the action` and `[command] parameter of the action` **system driver** variables.

Control elements - group: Action variable



Action variable

Name

Name action variable
Tvn: STATIC

Identification

Identification action variable
Tvn: STATIC

Status

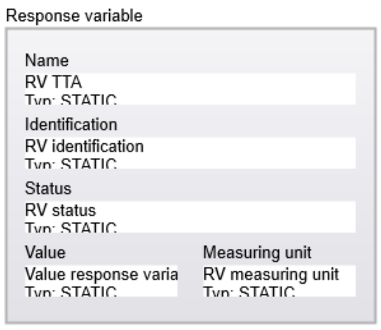
Status action variable
Tvn: STATIC

Value	Measuring unit
Value action variable Tvn: STATIC	Action variable meas Tvn: STATIC

Control element	Description
Name	Name of the action variable.
Identification	Identification of the action variable.
Status	Contains the short description of the status bits for the action variable. Example: ▶ Bits for COT ▶ Status SE_870 during Select ▶ Status N_CONF (PN-Bit) in the event of a negative response from the PLC **Note:** Includes the planned status bits of the response variable of the action `Status input`.
Value	Current value of the action variable or input field for `setpoint input` command processing action. **Note:** With drivers that also allow the reading of command variables from the PLC, this value changes during the process of the action, from an existing value to the current value. The display of the value is then only refreshed with COT=7 (COT_actcon) or WR-SUC. ▸ The following is applicable for a configured setpoint input: The value to be set for the `Set value` action is stipulated by this control element. By clicking the control element, it is switched to edit mode and the setpoint input is possible. Edit mode is left again by pressing **Enter**. However the new value is only set once the **Execute** control element has been clicked on. The control element is locked when: ▸ The status REVISION(9) of the response variable is set. ▸ No action is active. ▸ The screen is not in 'Step 1'.
Measuring unit	Measuring unit of the action variable

Note: The name is also displayed in the `[command] name of the action variable` **system driver** variable.

Control elements - group: Response variable


Response variable

Name
RV TTA
Typ: STATIC

Identification
RV identification
Typ: STATIC

Status
RV status
Typ: STATIC

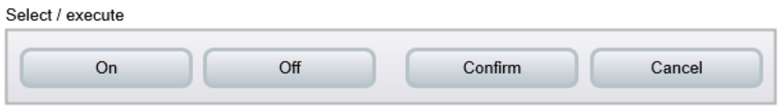
Value
Value response varia
Typ: STATIC

Measuring unit
RV measuring unit
Typ: STATIC

Control element	Description
Name	Name of the response variable
Identification	Name of the response variable
Status	Contains the short description of the status bits for the response variable.
Value	Current value of the response variable
Measuring unit	Measuring unit of the response variable

Note: The name is also in the `[command] name of the response variable` **system driver** variable.

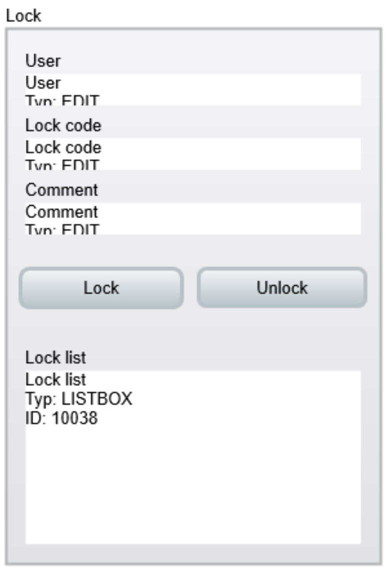
Control elements - group: Select / execute



Control element	Description
On	First-step command button, to close a switch for example. If **SBO** is activated for the command variable - sends a Select to the controller. Note: Only visible in Step 1.
Off	First-step command button, to open a switch for example. If **SBO** is activated for the command variable - sends a Select to the controller. Note: Only visible in Step 1.
Confirm	Second-stage command button - Execute/Operate. Confirms the execution of the command after successful checking of the interlocking and positive confirmation of the Select from the PLC. Note: Only visible in Step 2.
Cancel	Second-stage command button Cancel. Aborts the execution of the command processing and returns to 'Step 1'. Note: Only visible in Step 2.
Close	Closes the command window in zenon Runtime

Note: The steps are also displayed in the `[command] screen step` **system driver** variable.

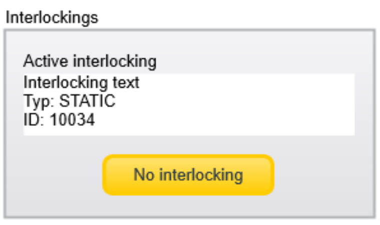
Control elements - group: Lock



Control elements from the **Lock** group are locked if no `Lock` action is configured in the command group.

Control element	Description
User	For entering the user identification for the lock.
Lock code	For entering the user-specific lock code.
Comment	Optional text that can be entered by the user for the lock.
Lock	Activates a lock by the user entered in the **User** control element. **Note:** This user action is logged in the CEL, if not suppressed by the engineering.
Unlock	Removes the lock that has been set up by the user entered in the **User** control element. This guarantees that only people's own locks can be removed. **Note:** This user action is logged in the CEL, if not suppressed by the engineering.
Lock list	List of active locks: ▶ `User` Name of the user who has activated the lock. ▶ `Locking time` Time stamp of the interlocking ▶ `Note` Text for the interlocking.

Control elements - group: Interlockings


Interlockings
Active interlocking
Interlocking text
Typ: STATIC
ID: 10034

Control element	Description
Interlocking text	The active interlocking (on page 125) according to the configuration or texts from ALC - topological interlocking (on page 33).
Unlock	This button unlocks an active, unlockable interlocking. Note: This control element is shown only when the screen is in the step 'Unlock' (Chapter: **Process in the Command Processing screen**). The Control element is locked when the upcoming interlocking is not unlockable.

Note: The interlocking texts are also displayed in the `[command] interlocking message` **system driver** variable.

4.4.3 Reload project online

The following is to be taken into account when reloading a command configuration:

- If Watchdog timer, edge generation or SBO is active, the reloading is delayed until this has ended.
- An opened `Command Processing` screen is closed and the process is started again after reloading depending on the current step:

Step before reloading	Action after reloading
Step 1	Screen is called up again for Step 1.
Interlocking or Step 2	Unlocking step is activated. The interlocking is then executed again.
Lock	Lock is activated again.

- Before it is called up again, the response variable, command variable, condition variables, command group and action are determined again. The control elements are locked if one of the objects is no longer present in the Runtime.
- If the command group is removed or replaced for the variable that was for calling up command input, the screen is called up with locked control elements. The screen must be called up again or the command processing must be executed again.
- If the command group was removed or replaced for the response variable, all locks triggered by the command processing are removed by the variables.
- If a user who has activated a command lock no longer exists, the lock is removed. The `M1` status bit and the `LOCK.BIN` file are updated accordingly.

4.4.4 Logging in the CEL

In the CEL, the following user actions are logged in addition to the switching actions.

Parameters	Description
Unlock	The unlocking of an active interlocking is noted in the CEL.
Unlock all	A corresponding CEL entry is created for each unlocked interlocking.
Execute action	If the "Suppress CEL entry" action setting is not active, the execution of the action is logged in the CEL.

4.4.5 Server change in redundant operation

If the standby server takes on the role of the Primary Server, the drivers take on the writing to the controller for the current Primary Server. An ongoing **Select Before Operate** is canceled as a result.

The handling of the **Select in the network** - `NET_SEL` status bit - of the response variable cannot be taken over from the previous Primary Server and the command must be executed again.

Please note the behavior or Runtime (on page 169) if redundancy switching is not triggered by a failure of the Primary Server but by the user.

4.4.6 Exit Runtime

As long as there are still active actions in the system, the proper exiting of the runtime (e.g. over a function call) is delayed.

An active **Select Before Operate** process also delays the ending of zenon Runtime. If **Select Before Operate** is activated, a deactivation (Cancel) is carried out.

Information

This situation occurs most of all with single-stage execution of the `pulse command` *action with runtime monitoring or edge generation. Runtime is only ended once the action has been completed.*

4.4.7 Lock return variable

A response variable is locked if the `M1` user status bit has been set.

In the `Lock` action, the lock can be activated or deactivated by entering the user name and a **Lock code** (configured during user definition). A user can activate a lock for each response variable. Several users can lock the same response variable.

The active locks are saved remanently in the `LOCK.BIN` file and are also taken into account after the system is restarted.

Information

There is automatic synchronization of locking in the zenon network. Locking can thus also be used in redundant operation.