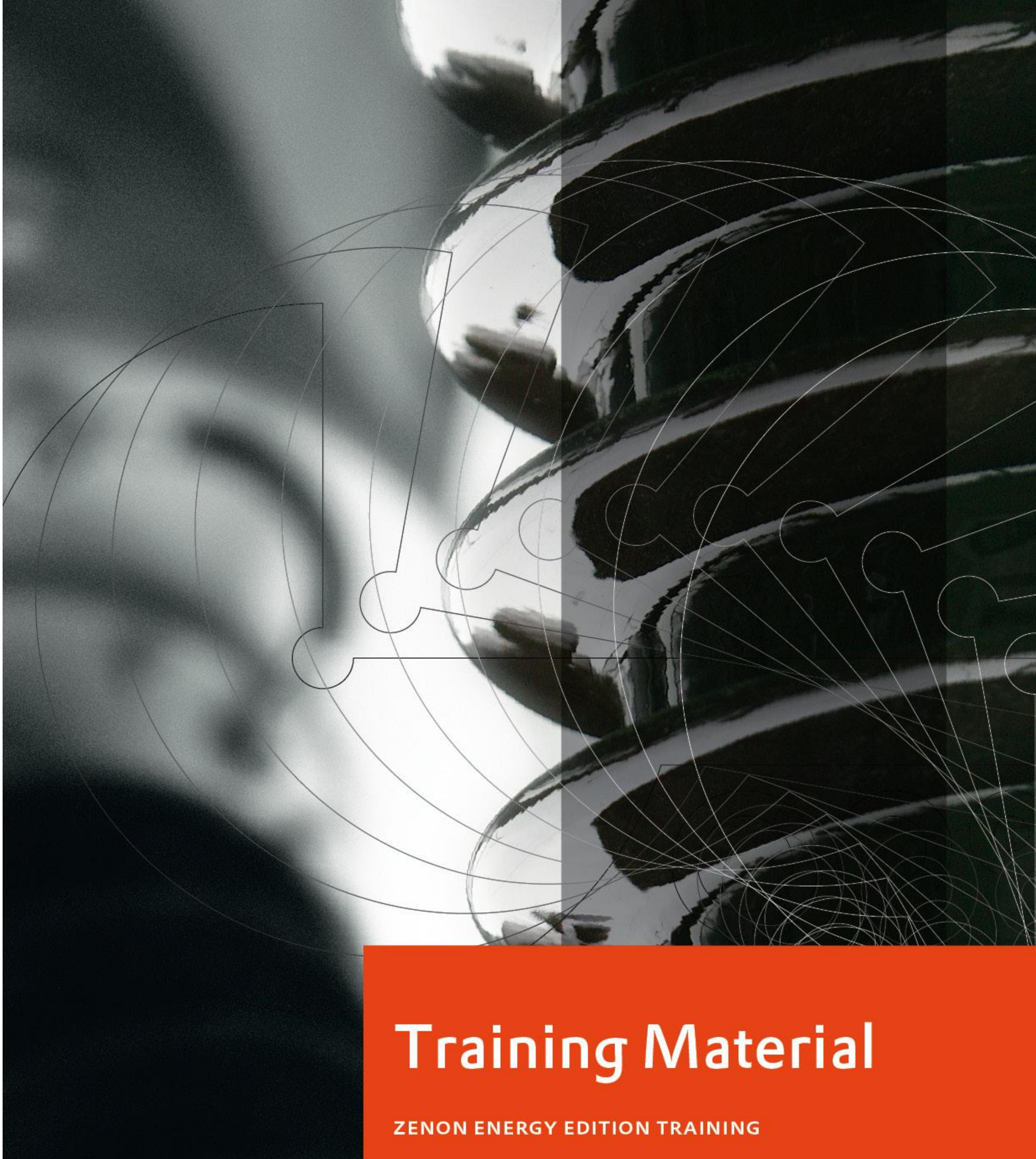


Training Material

ZENON ENERGY EDITION TRAINING

zenon
do it your way

Contents

Introduction

Welcome and thank you for choosing zenon Energy Edition. In this Training, we will realize a single line diagram for a virtual substation.

The single line diagram shows the operator fast and clear the current state of the substation.

Technically we will use the Modules ALC (Automatic Line Coloring), Command processing and the Topology.

As our database, we use variables based on IEC61850 Client driver connected to a training server application.

Preparatory work

As a basis for this training, we shall first create a new project with the *Project wizard*.

- Create a new project named 'SingleLineDiagram'.
- In the wizards dialog that opens now, execute the 'Project wizard'.
- Keep the default settings in the wizard dialog by confirming them with *Finish*.

This Training material is based on the use of a simulation IEC61850 Server. To set up this simulation server please read back the project backup provided (IEC850_FEEDER_vv_zz.zip) into your workspace.

Compile the project and start the Runtime. Exit the runtime. The simulation Project is using zenon Logic for the IEC61850 server, it is configured to keep running even if you stop the zenon runtime.

Now activate your before created project ‘SingleLineDiagram’.

CREATING VARIABLES

Learning objectives:

- You can configure the IEC61850 Client driver to connect to an IED.
- You can create variables (database) based on the IEC61850 Client driver.
- You know the possibilities for importing variables (database) from the IEC61850 Server or SCL file.

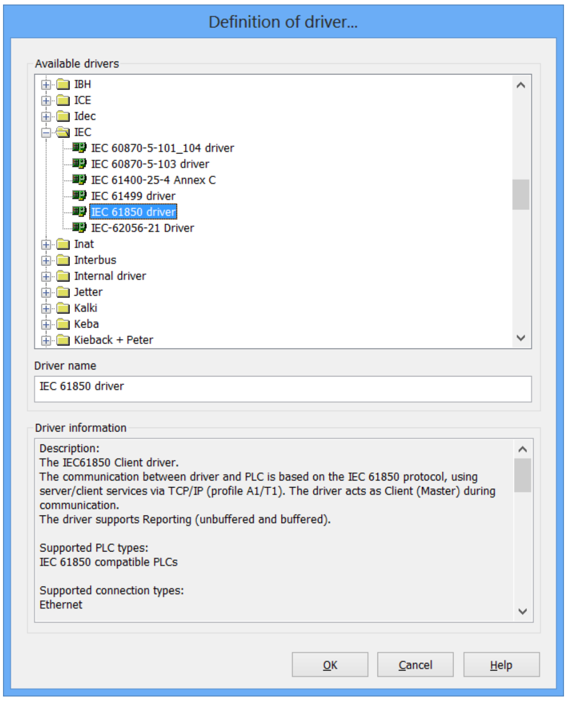
In order to test the special energy functionality, we will need some variables. We will now create some.

NAME	DRIVERS	TYPE
Source_A	Intern	Internal Variable/BOOL
Source_B	Intern	Internal Variable/BOOL
Grounded	Intern	Internal Variable/BOOL
Consumer1	Intern	Internal Variable/BOOL
Consumer2	Intern	Internal Variable/BOOL
Trafo	Intern	Internal Variable/BOOL

For the communication with our Simulation IEC61850 Server we will use the IEC61850 Client driver in our Project.

Create the IEC61850 driver:
Right click on the branch driver, Choose from context menu ‘new driver’:

Choose IEC61850 Client driver from the IEC folder.



Driver name

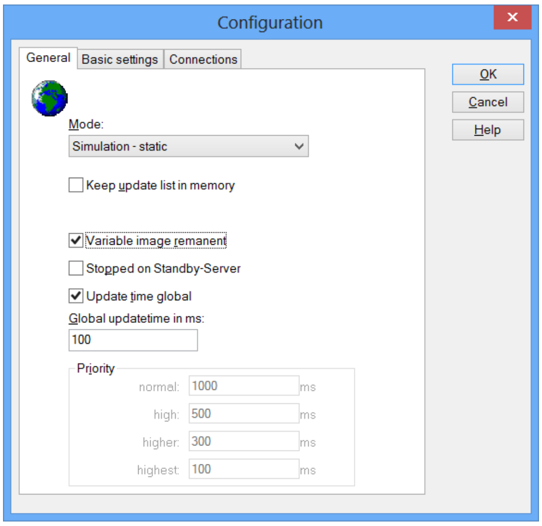
IEC 61850 driver

Driver information

Description:
The IEC61850 Client driver.
The communication between driver and PLC is based on the IEC 61850 protocol, using server/client services via TCP/IP (profile A1/T1). The driver acts as Client (Master) during communication.
The driver supports Reporting (unbuffered and buffered).

Supported PLC types:
IEC 61850 compatible PLCs

Supported connection types:
Ethernet



Mode: Simulation - static

☐ Keep update list in memory

☑ Variable image remanent

☐ Stopped on Standby-Server

☑ Update time global

Global updatetime in ms: 100

Priority

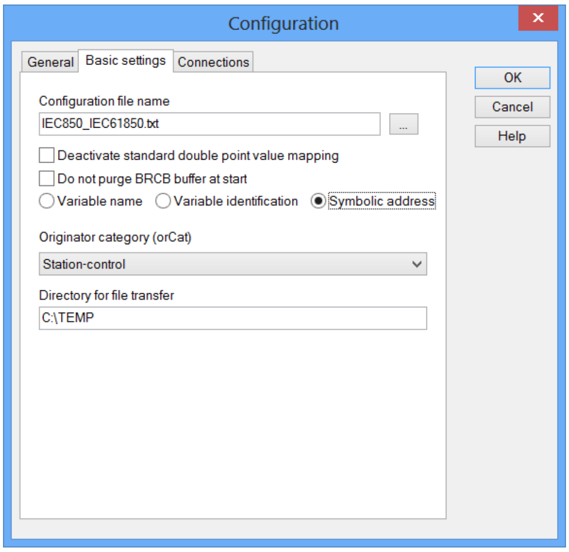
normal: 1000 ms

high: 500 ms

higher: 300 ms

highest: 100 ms

On the page General set the Mode to Simulation Static.
Activate the Property Variable image remanent.



Configuration file name
IEC850_IEC61850.txt

□ Deactivate standard double point value mapping

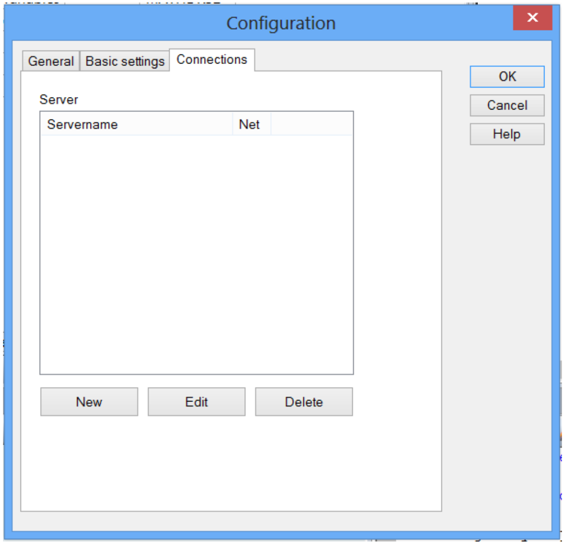
□ Do not purge BRCB buffer at start

○ Variable name ○ Variable identification ● Symbolic address

Originator category (orCat)
Station-control

Directory for file transfer
C:\TEMP

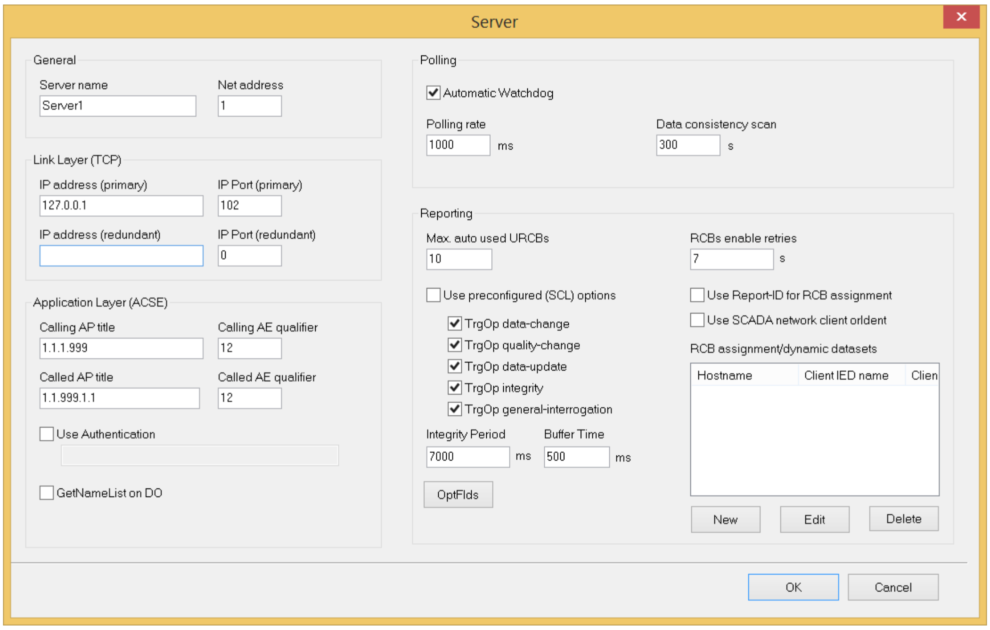
In Basic settings select Symbolic address. This means that the Symbolic address property will be used to configure the address in the variable.



Server

Servername	Net

With ‘New’ create a new connection.



Server

General

Server name: Server1

Net address: 1

Link Layer (TCP)

IP address (primary): 127.0.0.1

IP Port (primary): 102

IP address (redundant):

IP Port (redundant): 0

Application Layer (ACSE)

Calling AP title: 1.1.1.999

Calling AE qualifier: 12

Called AP title: 1.1.999.1.1

Called AE qualifier: 12

□ Use Authentication

□ GetNameList on DO

Polling

☑ Automatic Watchdog

Polling rate: 1000 ms

Data consistency scan: 300 s

Reporting

Max. auto used URCBs: 10

RCBs enable retries: 7 s

□ Use preconfigured (SCL) options

☑ TrgOp data-change

☑ TrgOp quality-change

☑ TrgOp data-update

☑ TrgOp integrity

☑ TrgOp general-interrogation

□ Use Report-ID for RCB assignment

□ Use SCADA network client orIdent

RCB assignment/dynamic datasets

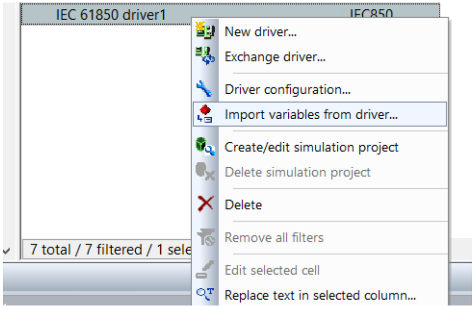
Hostname	Client IED name	Clien

Integrity Period: 7000 ms

Buffer Time: 500 ms

Only set IP address (primary) to the IP address where the simulation server is running (your local IP address). Alternative if you run the simulator locally set the loopback address: 127.0.0.1.
Close the dialogs with OK

Right click on the IEC61850 driver

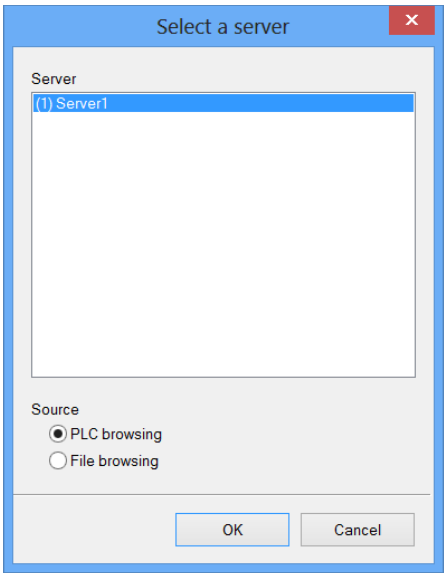


IEC 61850 driver1 | IEC850

7 total / 7 filtered / 1 sele

New driver...
Exchange driver...
Driver configuration...
Import variables from driver...
Create/edit simulation project
Delete simulation project
Delete
Remove all filters
Edit selected cell
Replace text in selected column...

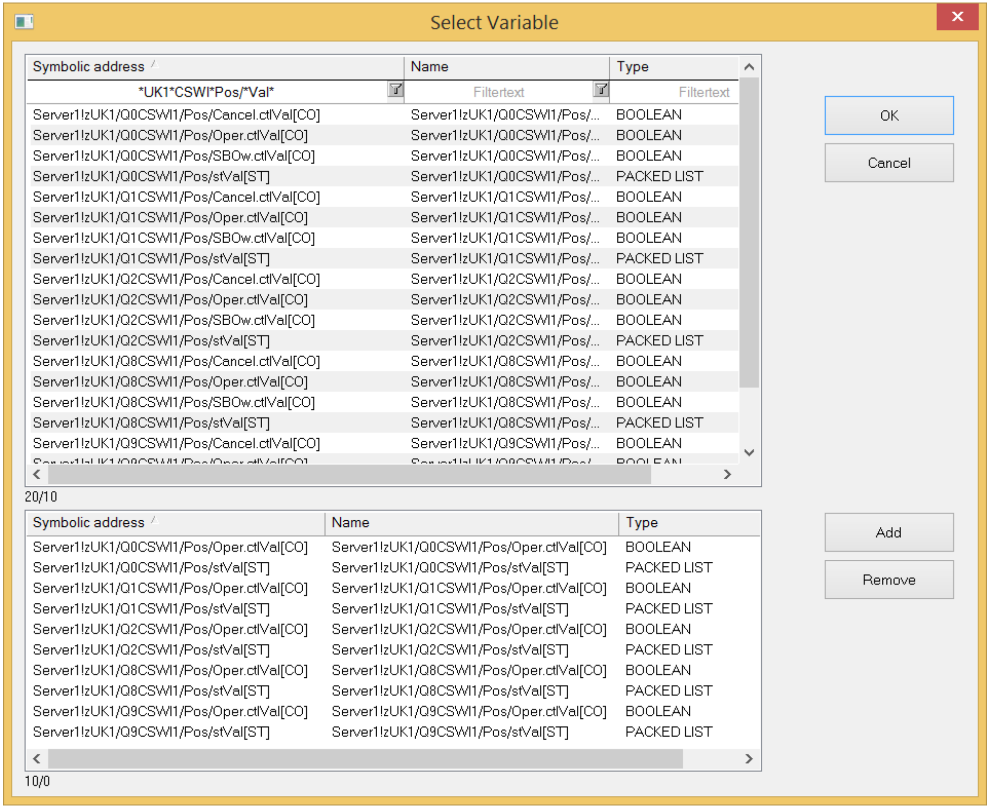
The import dialog opens



Server: (1) Server1

Source:
● PLC browsing
○ File browsing

Select Server1 and choose PLC Browsing



Symbolic address	Name	Type
*UK1*CSWI*Pos/*Val*	Filtertext	Filtertext
Server1!zUK1/Q0CSWI1/Pos/Cancel.ctlVal[CO]	Server1!zUK1/Q0CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q0CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q0CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q0CSWI1/Pos/SBOw.ctlVal[CO]	Server1!zUK1/Q0CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q0CSWI1/Pos/stVal[ST]	Server1!zUK1/Q0CSWI1/Pos/...	PACKED LIST
Server1!zUK1/Q1CSWI1/Pos/Cancel.ctlVal[CO]	Server1!zUK1/Q1CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q1CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q1CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q1CSWI1/Pos/SBOw.ctlVal[CO]	Server1!zUK1/Q1CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q1CSWI1/Pos/stVal[ST]	Server1!zUK1/Q1CSWI1/Pos/...	PACKED LIST
Server1!zUK1/Q2CSWI1/Pos/Cancel.ctlVal[CO]	Server1!zUK1/Q2CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q2CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q2CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q2CSWI1/Pos/SBOw.ctlVal[CO]	Server1!zUK1/Q2CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q2CSWI1/Pos/stVal[ST]	Server1!zUK1/Q2CSWI1/Pos/...	PACKED LIST
Server1!zUK1/Q8CSWI1/Pos/Cancel.ctlVal[CO]	Server1!zUK1/Q8CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q8CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q8CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q8CSWI1/Pos/SBOw.ctlVal[CO]	Server1!zUK1/Q8CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q8CSWI1/Pos/stVal[ST]	Server1!zUK1/Q8CSWI1/Pos/...	PACKED LIST
Server1!zUK1/Q9CSWI1/Pos/Cancel.ctlVal[CO]	Server1!zUK1/Q9CSWI1/Pos/...	BOOLEAN
Server1!zUK1/Q9CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q9CSWI1/Pos/	BOOLEAN

20/10

Symbolic address	Name	Type
Server1!zUK1/Q0CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q0CSWI1/Pos/Oper.ctlVal[CO]	BOOLEAN
Server1!zUK1/Q0CSWI1/Pos/stVal[ST]	Server1!zUK1/Q0CSWI1/Pos/stVal[ST]	PACKED LIST
Server1!zUK1/Q1CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q1CSWI1/Pos/Oper.ctlVal[CO]	BOOLEAN
Server1!zUK1/Q1CSWI1/Pos/stVal[ST]	Server1!zUK1/Q1CSWI1/Pos/stVal[ST]	PACKED LIST
Server1!zUK1/Q2CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q2CSWI1/Pos/Oper.ctlVal[CO]	BOOLEAN
Server1!zUK1/Q2CSWI1/Pos/stVal[ST]	Server1!zUK1/Q2CSWI1/Pos/stVal[ST]	PACKED LIST
Server1!zUK1/Q8CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q8CSWI1/Pos/Oper.ctlVal[CO]	BOOLEAN
Server1!zUK1/Q8CSWI1/Pos/stVal[ST]	Server1!zUK1/Q8CSWI1/Pos/stVal[ST]	PACKED LIST
Server1!zUK1/Q9CSWI1/Pos/Oper.ctlVal[CO]	Server1!zUK1/Q9CSWI1/Pos/Oper.ctlVal[CO]	BOOLEAN
Server1!zUK1/Q9CSWI1/Pos/stVal[ST]	Server1!zUK1/Q9CSWI1/Pos/stVal[ST]	PACKED LIST

10/0

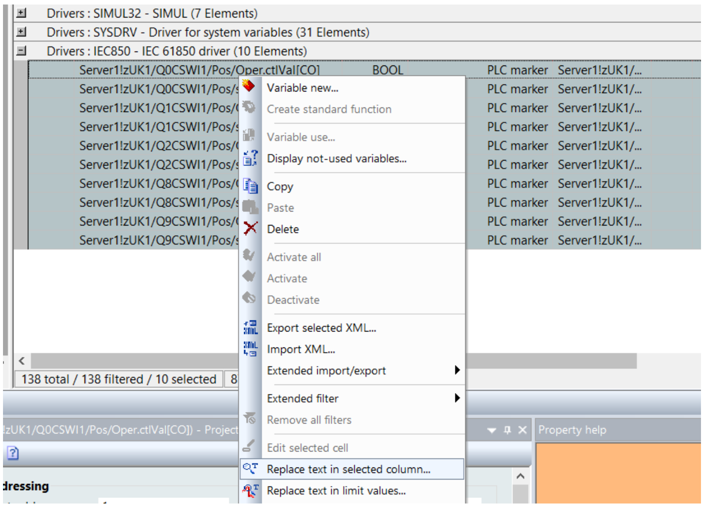
The Import dialog let you filter in the headline for better finding
You could set a filter string: *UK1*CSWI*Pos/*Val* to filter down to the position objects we are interested in.

Import from IEC61850 driver:
Server1!zUK1/Q0CSWI1/Pos/Oper.ctlVal[CO]
Server1!zUK1/Q0CSWI1/Pos/stVal[ST]

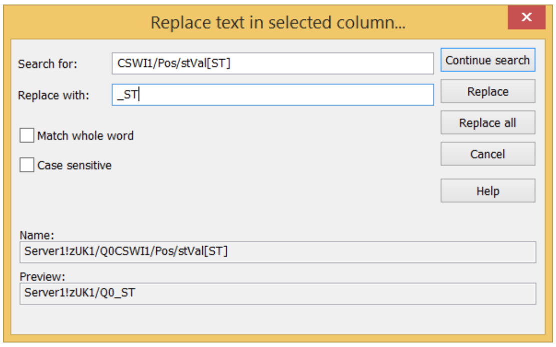
Server1!zUK1/Q1CSWI1/Pos/Oper.ctlVal[CO]
Server1!zUK1/Q1CSWI1/Pos/stVal[ST]
Server1!zUK1/Q2CSWI1/Pos/Oper.ctlVal[CO]
Server1!zUK1/Q2CSWI1/Pos/stVal[ST]
Server1!zUK1/Q8CSWI1/Pos/Oper.ctlVal[CO]
Server1!zUK1/Q8CSWI1/Pos/stVal[ST]
Server1!zUK1/Q9CSWI1/Pos/Oper.ctlVal[CO]
Server1!zUK1/Q9CSWI1/Pos/stVal[ST]

RENAME IMPORTED VARIABLES

Select all imported variables in the list, rightclick to open the contextual menu.


Drivers : SIMUL32 - SIMUL (7 Elements)
Drivers : SYSDRV - Driver for system variables (31 Elements)
Drivers : IEC850 - IEC 61850 driver (10 Elements)
Server1!zUK1/Q0CSWI1/Pos/Oper.ctlVal[CO]
BOOL
PLC marker
Server1!zUK1/...
Variable new...
Create standard function
Variable use...
Display not-used variables...
Copy
Paste
Delete
Activate all
Activate
Deactivate
Export selected XML...
Import XML...
Extended import/export
Extended filter
Remove all filters
Edit selected cell
Replace text in selected column...
Replace text in limit values...
138 total / 138 filtered / 10 selected
Property help

Select 'Replace text in selected column'



Search for: CSWI1/Pos/stVal[ST]

Replace with: _ST

☐ Match whole word

☐ Case sensitive

Name:
Server1!zUK1/Q0CSWI1/Pos/stVal[ST]

Preview:
Server1!zUK1/Q0_ST

Search for 'CSWI1/Pos/stVal[ST]' replace with '_ST'

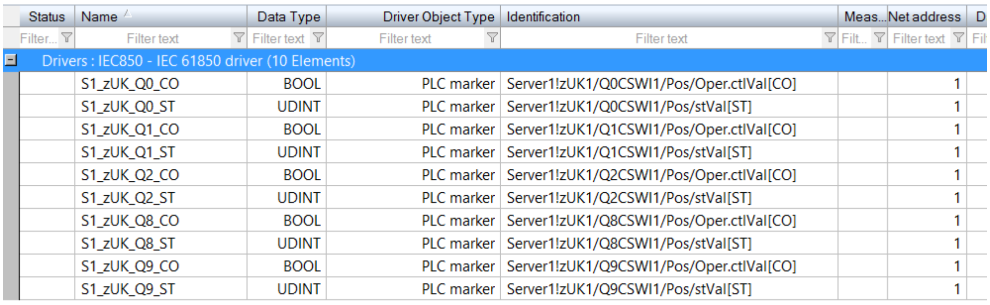
Do the same with Command variables

Search for 'CSWI1/Pos/Oper.ctlVal[CO]' replace with '_CO'

Again select all variables and

Search for ‘Server1!zUK1/’ replace with ‘S1_zUK_’

We should end up with a variable list S1_zUK_Q0_CO, S1_zUK_Q0_ST…



Drivers : IEC850 - IEC 61850 driver (10 Elements)

Status	Name	Data Type	Driver Object Type	Identification	Meas...	Net address	D
Filter...	Filter text	Filter text	Filter text	Filter text	Filt...	Filter text	Fil
	S1_zUK_Q0_CO	BOOL	PLC marker	Server1!zUK1/Q0CSWI1/Pos/Oper.ctlVal[CO]		1	
	S1_zUK_Q0_ST	UDINT	PLC marker	Server1!zUK1/Q0CSWI1/Pos/stVal[ST]		1	
	S1_zUK_Q1_CO	BOOL	PLC marker	Server1!zUK1/Q1CSWI1/Pos/Oper.ctlVal[CO]		1	
	S1_zUK_Q1_ST	UDINT	PLC marker	Server1!zUK1/Q1CSWI1/Pos/stVal[ST]		1	
	S1_zUK_Q2_CO	BOOL	PLC marker	Server1!zUK1/Q2CSWI1/Pos/Oper.ctlVal[CO]		1	
	S1_zUK_Q2_ST	UDINT	PLC marker	Server1!zUK1/Q2CSWI1/Pos/stVal[ST]		1	
	S1_zUK_Q8_CO	BOOL	PLC marker	Server1!zUK1/Q8CSWI1/Pos/Oper.ctlVal[CO]		1	
	S1_zUK_Q8_ST	UDINT	PLC marker	Server1!zUK1/Q8CSWI1/Pos/stVal[ST]		1	
	S1_zUK_Q9_CO	BOOL	PLC marker	Server1!zUK1/Q9CSWI1/Pos/Oper.ctlVal[CO]		1	
	S1_zUK_Q9_ST	UDINT	PLC marker	Server1!zUK1/Q9CSWI1/Pos/stVal[ST]		1	

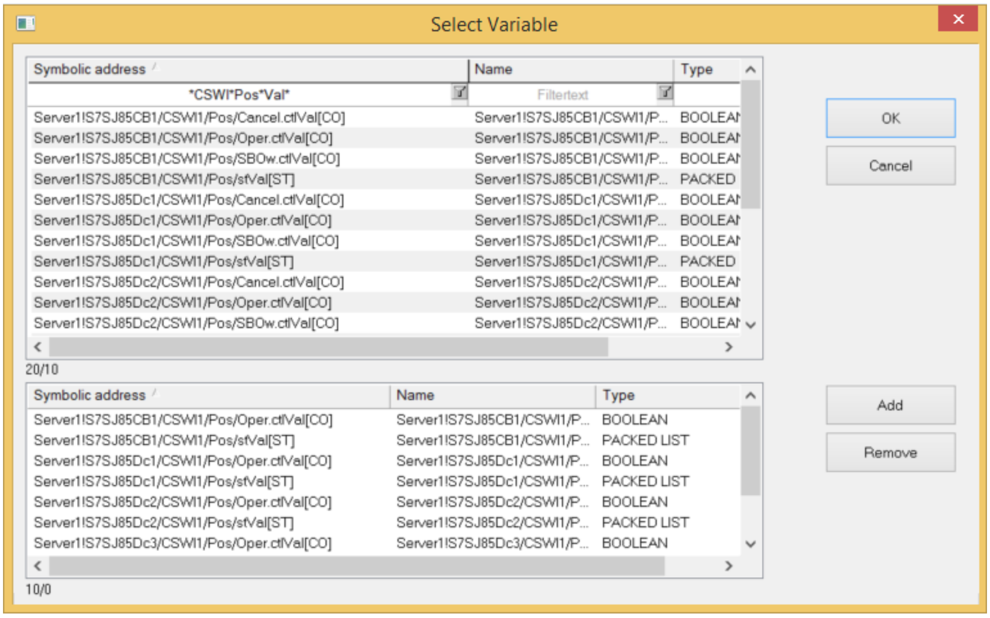
USING THE ZENIED FOR THE TRAINING:

Skip this topic if you are not using the zenIED. Skip this topic if you are not sure what to do and continue with ‘creating a screen’

You can follow these instructions if you want to you use your own IED for the training.

Set up a new server connection in the driver to connect to each zenIED in your network.

When browsing the variables from each zenIED use the Filter string: *CSWI*Pos*Val*



Select Variable

Symbolic address	Name	Type
*CSWI*Pos*Val*	Filtertext	
Server1!S7SJ85CB1/CSWI1/Pos/Cancel.ctlVal[CO]	Server1!S7SJ85CB1/CSWI1/P...	BOOLEAN
Server1!S7SJ85CB1/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85CB1/CSWI1/P...	BOOLEAN
Server1!S7SJ85CB1/CSWI1/Pos/SBOw.ctlVal[CO]	Server1!S7SJ85CB1/CSWI1/P...	BOOLEAN
Server1!S7SJ85CB1/CSWI1/Pos/stVal[ST]	Server1!S7SJ85CB1/CSWI1/P...	PACKED
Server1!S7SJ85Dc1/CSWI1/Pos/Cancel.ctlVal[CO]	Server1!S7SJ85Dc1/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc1/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85Dc1/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc1/CSWI1/Pos/SBOw.ctlVal[CO]	Server1!S7SJ85Dc1/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc1/CSWI1/Pos/stVal[ST]	Server1!S7SJ85Dc1/CSWI1/P...	PACKED
Server1!S7SJ85Dc2/CSWI1/Pos/Cancel.ctlVal[CO]	Server1!S7SJ85Dc2/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc2/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85Dc2/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc2/CSWI1/Pos/SBOw.ctlVal[CO]	Server1!S7SJ85Dc2/CSWI1/P...	BOOLEAN

20/10

Symbolic address	Name	Type
Server1!S7SJ85CB1/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85CB1/CSWI1/P...	BOOLEAN
Server1!S7SJ85CB1/CSWI1/Pos/stVal[ST]	Server1!S7SJ85CB1/CSWI1/P...	PACKED LIST
Server1!S7SJ85Dc1/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85Dc1/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc1/CSWI1/Pos/stVal[ST]	Server1!S7SJ85Dc1/CSWI1/P...	PACKED LIST
Server1!S7SJ85Dc2/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85Dc2/CSWI1/P...	BOOLEAN
Server1!S7SJ85Dc2/CSWI1/Pos/stVal[ST]	Server1!S7SJ85Dc2/CSWI1/P...	PACKED LIST
Server1!S7SJ85Dc3/CSWI1/Pos/Oper.ctlVal[CO]	Server1!S7SJ85Dc3/CSWI1/P...	BOOLEAN

10/0

After import of the variables, rename them following the above replace rules.

Take care of the different naming of the switches:

zenIED → Simulator

CB1 → Q0 CircuitBreaker

Dc1 → Q1 Bus bar disconnector

Dc2 → Q2 Bus bar disconnector

Dc3 → Q9 Line disconnector

Dc4 → Q8 Earthing disconnector

Example for renaming: Rename the Variables: Search for 'Server1!S7SJ85' replace with 'S1_zenIED1'

Search for 'CB1' replace with 'Q0'

Search for 'Dc1' replace with 'Q1'

…

Search for 'CSWI1/Pos/stVal[ST]' replace with '_ST'

Search for 'CSWI1/Pos/Oper.ctlVal[CO]' replace with '_CO'

CREATING A SCREEN

Now we need a screen to draw our network in.

- Create a new screen named 'SingleLineDiagram' (screen type *Standard*, template 'MAIN').
- Create a new function *Screen switch* for the screen 'SingleLineDiagram'.
- Open the screen 'Navigation'.
- Link the function to the first free button (create a new Button) and enter the caption 'SingleLineDiagram'.

Now we are done with the preparations and we can start working with the special functionality of the energy edition of the control system.

The Automatic Line Coloring (ALC)

Learning objectives:

- Understand the color forwarding of the Automatic Line Coloring Module (Single line diagram coloring).
- You understand the role of the combined element in the ALC when using it as Source, Switch, Disconnector, Trafo...
- You know the different visualization possibilities for lines depending on the grid state.
- You can design a fully functional single line diagram of a feeder bay.

In the single line diagram we will use Automatic Line Coloring to automatically show the current status of the grid (e.g.: under load, grounded, undefined). With topology package licensed the control system will automatically calculate the topologic interlocking based on our single line diagram.

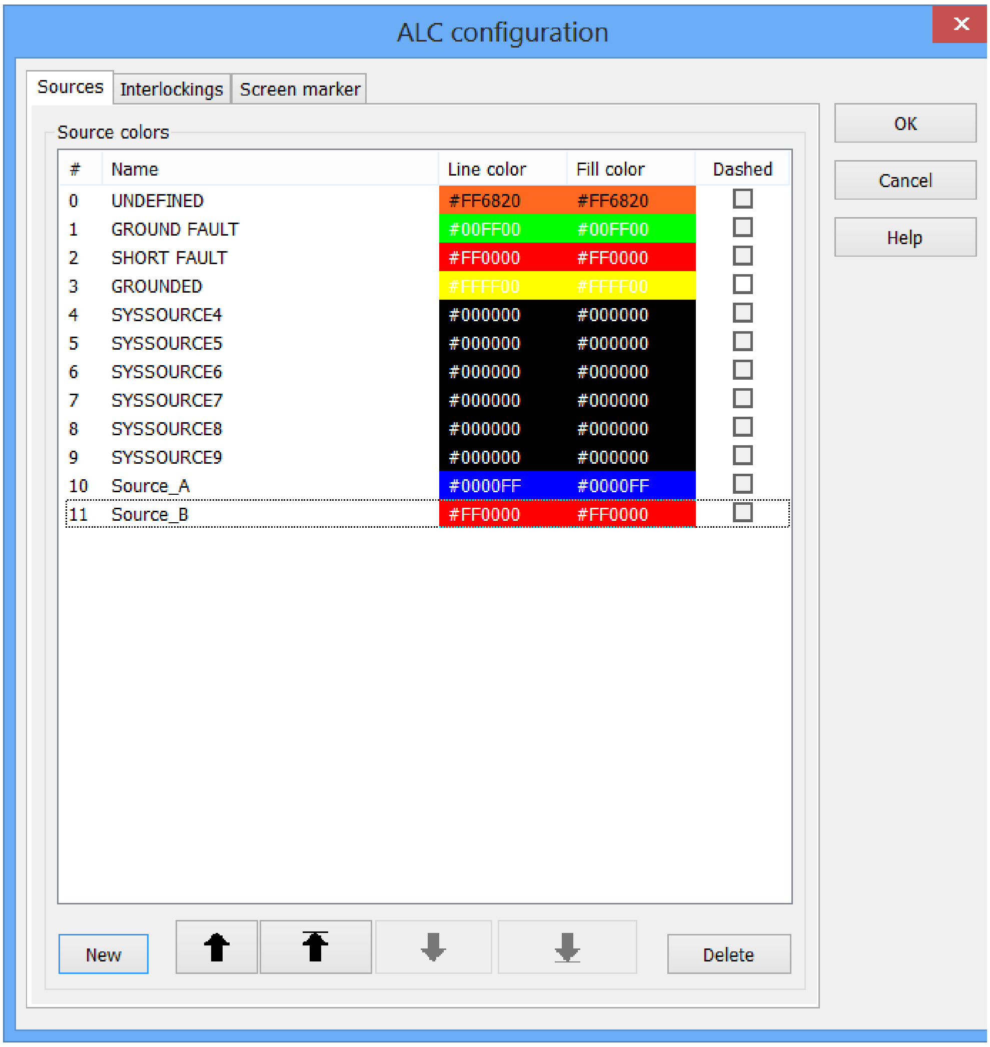
The Automatic Line Coloring distinguishes between procedural elements (sources, switches, disconnectors, drain ...) and lines or pipes. The procedural elements are all represented by the use of Combined-elements. For the lines, you can use lines, polylines or tubes.

The configuration of the sources

First, we have a look at the configuration of the sources.

- Select the project in the project manager.
- Open the section *Automatic Line Coloring* in the property window.
- Click on *ALC configuration*.

The following dialog for the configuration of the Automatic Line Coloring opens now:

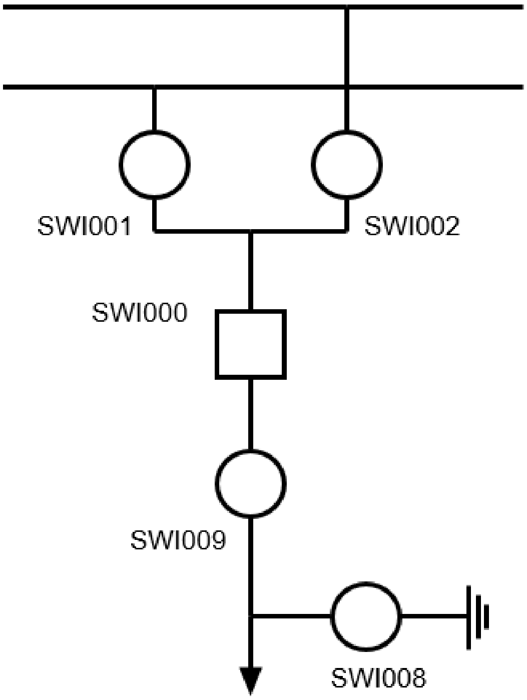


#	Name	Line color	Fill color	Dashed
0	UNDEFINED	#FF6820	#FF6820	□
1	GROUND FAULT	#00FF00	#00FF00	□
2	SHORT FAULT	#FF0000	#FF0000	□
3	GROUNDED	#FFFF00	#FFFF00	□
4	SYSSOURCE4	#000000	#000000	□
5	SYSSOURCE5	#000000	#000000	□
6	SYSSOURCE6	#000000	#000000	□
7	SYSSOURCE7	#000000	#000000	□
8	SYSSOURCE8	#000000	#000000	□
9	SYSSOURCE9	#000000	#000000	□
10	Source_A	#0000FF	#0000FF	□
11	Source_B	#FF0000	#FF0000	□

- Create a new Source, name it 'Source_A'.
- Create a new Source, name it 'Source_B'.
- Select the color ‘orange’or‘FF6820’ for UNDEFINED source.
- Select the desired colors for your sources.
- Close the dialog box with *OK*.

We will have a look on the other page of the dialog later.

In this Training, we will realize a feeder like this one:



SWI001	→ Q1	Busbar disconnector (Dc1)
SWI002	→ Q2	Busbar disconnector (Dc2)
SWI000	→ Q0	Circuit Breaker (CB1)
SWI009	→ Q9	Line disconnector (Dc3)
SWI008	→ Q8	Earthing disconnector (Dc4)

(switchgear tags from zenIED)

A source and the first lines

THE SOURCE

- Open the screen 'SingleLineDiagram'.
- Add a Combined-element to the bottom of the screen.
- Select the variable 'Source_A'.
- In the assistant, select the display type *Symbol from library*.

- On the next page click on the big button to select a symbol.

Now the symbol library opens.

- Select the *General symbol library*.
- Open the group '*Energy*'.
- Select the *'Diesel generator'*.
- Click on *New status*.

Keep the diesel generator also for the new status.

- Enter the value 1 for the status 1.
- Confirm the settings with *Finish*.

Now we still need to change some settings in the properties of the *Combined-element*.

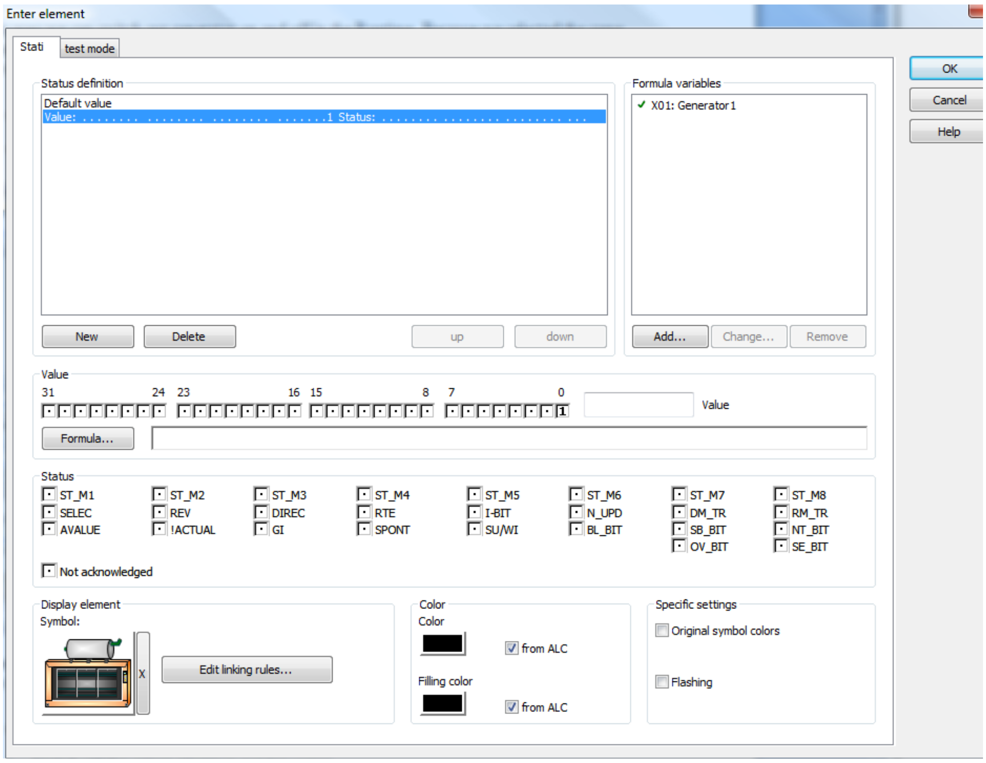
- Open the section *Automatic Line Coloring* in the property window.
- Under *Function type,* select *Source*.
- Under *Source,* you now can select our 'Source_A'.

Now the *Combined-element* is prepared for the *Automatic Line Coloring*. To be able to test the functionality in the Runtime we should however set some more properties.

- Open the section *Set value* in the property window.
- Under *Numeric value*, deactivate the property *SV active*.
- Under *Binary value*, activate the property *Switch*.

This way we can switch our source on and off in the Runtime. Because we selected the same symbol from the library for both statuses in the assistant, we will not see whether the source is on or off. We will change this in the next step.

- Open the section *Representation* in the property window.
- With *Configuration and test*, open the dialog for the definition of the status.



Enter element

Status definition:

Default value
Value: ....1 Status: ....

Formula variables:

✓ X01: Generator1

Not acknowledged

Color: ☑ from ALC

Filling color: ☑ from ALC

Specific settings:

☐ Original symbol colors

☐ Flashing

In addition to the default status, we created a second status with the value 1 in the assistant. For the default status (source is off), we activate the option *Original symbol colors*.

- Select the status with the value 1.
- Switch off the option *Original symbol colors*.
- For the *Color* and the *Filling color* of the symbol, you can now select the option *from ALC*.

Now the definition of our source is complete.

LINES

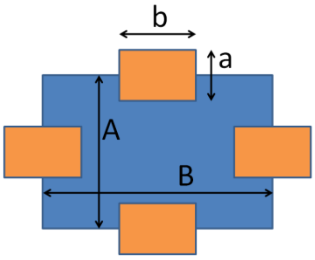
For the lines, we will use normal *Lines*. Instead, you can also use *Pipes or Rectangles*.

- Draw a vector element *Line* from the generator upward.
- In the section *Representation* of the properties, set the *Line width* to 3 pixels.
- Open the section *Automatic Line Coloring* in the property window.
- Activate the property *Color from ALC*.

Attention: Do not rotate lines. Lines with rotation are not supported by ALC.

Make sure that the *line* exactly touches the *Combined-element* of the source. Lines can only receive/forward the color information of the Automatic Line Coloring at their ends.

The connection points, where the color is forwarded, are:



The area where the color is forwarded is in the middle of each side of the Combined elements. It has a maximum size of 20 by 20 pixels.

A: High of Combined element.

B: Width of Combined element.

a: High of connection area max. 20 pixel or 1/3 of wide A.

b: Width of connection area max. 20 pixel or 1/3 of length B.

If the combined element is smaller than 30 by 30 pixels, it can happen that the connection points overlap.

- Now copy the line with CTRL-C and paste it with CTRL-V.
- Move the copied line so that it exactly touches the old one.

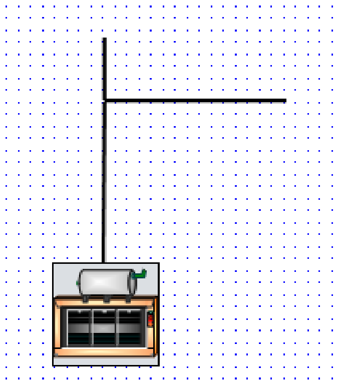
- From the point, where the first two lines touch, draw a new line to the right
- Open the section *Automatic Line Coloring* in the property window.
- Activate the property *Color from ALC*.

If you want to draw a branch connection in your network, the existing *line* has to be cut at the intersection, as the lines forward the color information only at their ends.

Now it is time to have a look at our lines in the Runtime.

- Start the Runtime.
- Open the screen 'SingleLineDiagram'.

Our network should look like this now.



Try out how the line react when you switch the source on and off.

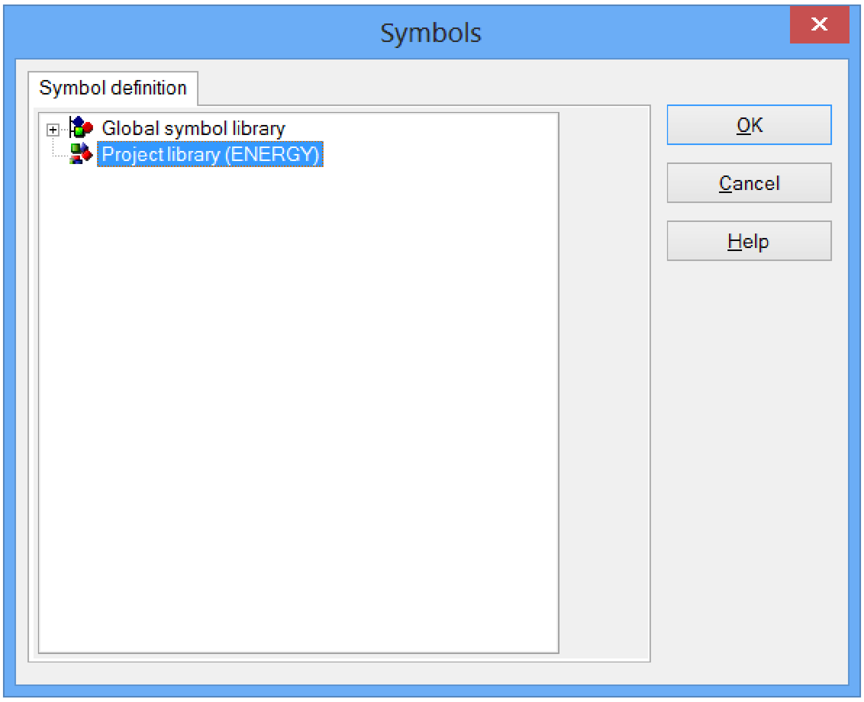
- Switch back to the Editor.

A SWITCH (DISCONNECTOR)

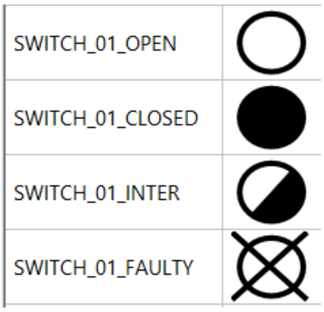
We will now add a switch to our network. For a switch, we also have to use the dynamic element Co*mbined-element*.

- Draw a vector element *circle* at the lower end of our branch.
- Create a symbol from our *circle*, either with the menu entry *Edit / Symbol / Create* or with the according icon.
- Enter the name 'Switch_01_CLOSED' for the *symbol*.

- Enter the symbol into the Project symbol library.'Rightclick on the symbol -> Symbol-> Insert into Symbol library



- Do the same for all this shapes:



SWITCH_01_OPEN	○
SWITCH_01_CLOSED	●
SWITCH_01_INTER	◐
SWITCH_01_FAULTY	⊗

- Draw a *Combined-element*.

Then take care the *Combined-element* exactly touches the end of the line.

In the variable selection, you will now see only bool variables. As we activated the property switch for the Combined-element of the source, this is a default for the new Combined-element. This is why only bit variables are available now.

- Close the variable selection with *No selection*.
- Open the section *Set value* in the property window.
- Under *Binary value*, deactivate the property *Switch*.
- Open the section *Variable / Function* in the property window.

- Under *Variable / Function*, link the variable

  ‘S1_zUK_Q9_ST'.
- Open the section *Set value* in the property window.
- Under *Numeric value*, activate the property *SV active*.

This way we can set different values for the state of our switch in the Runtime. In the next step, we assign the ALC function.

- Open the section *Automatic Line Coloring* in the property window.
- Under *Function type* select *Disconnector*

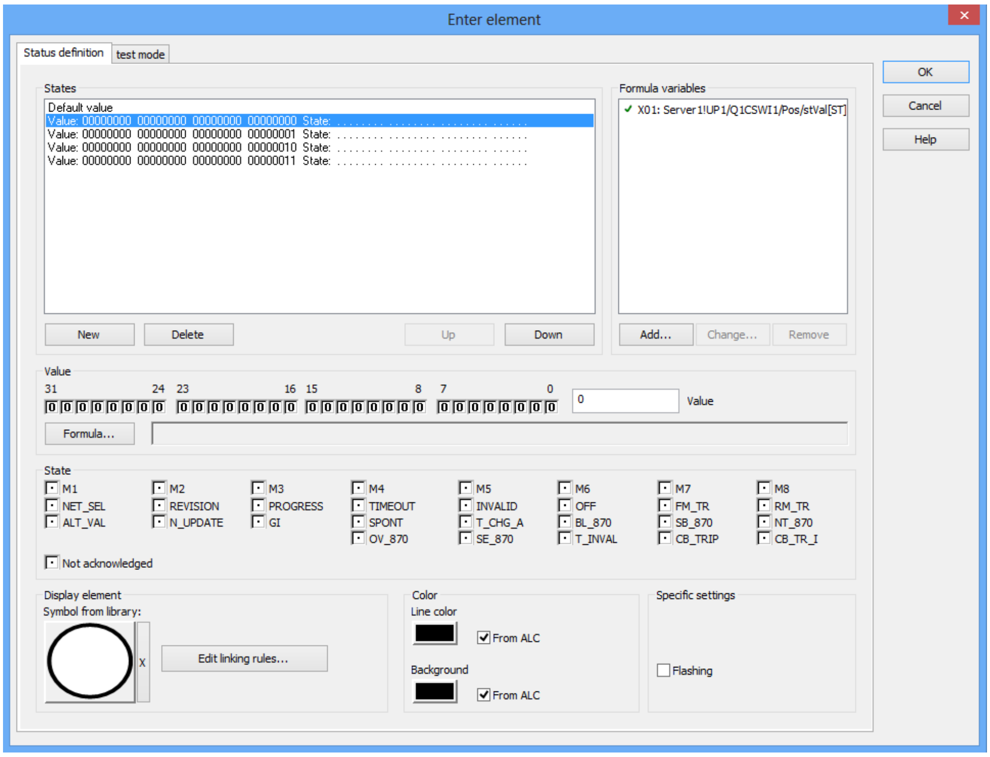
Now we can configure different representations for the Combined-element for the different states of our switch as we did for the source.

- Open the section *Representation* in the property window.
- With *Configuration and test*, open the dialog for the definition of the status.
- With *New status* add four more states for the variable values 0 to 3 and select the corresponding symbol form our library:

0	OPEN	○
1	CLOSED	●
2	INTERMEDIATE	◐
3	FAULTY	⊗

For the default status, keep the color black. For the other states:

- For the *Color* and the *Filling color* of the symbol, you can now select the option *from ALC*.



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: Server1!UP1/Q1CSWI1/Pos/stVal[ST]

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
```

Value: 0

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

- Confirm the settings with *OK*.

Now the functionality of our disconnector is complete and it is time to try it in the Runtime.

- Create new Runtime files.
- Switch to the Runtime.
- Reload the changes.

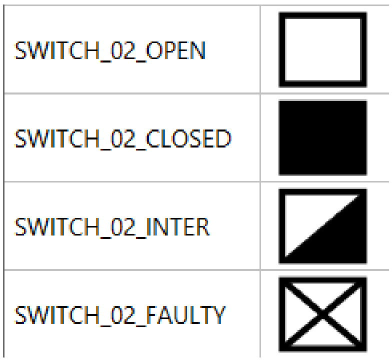
Depending on whether the switch has the value 0 or 1, the color information is forwarded to the following line. As we used a byte variable for the switch, it can also have other values. In this case, the line will be displayed as undefined if the source is activated.

- Try out the different values for the switch.

The Circuit Breaker

Now we will extend our electrical network. In the following step, we add a Circuit Breaker.

- Draw a *rectangle*.
- Create a *symbol* from the *rectangle* and name it 'SWITCH_02_OPEN'.
- Enter the symbol into the Project symbol library
- Do the same for all this shapes:



SWITCH_02_OPEN	□
SWITCH_02_CLOSED	■
SWITCH_02_INTER	◪
SWITCH_02_FAULTY	☒

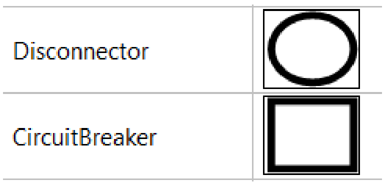
.

- Copy the *Combined-element* for the disconnector
- Link the variable ‘S1_zUK_Q0_ST' to the element.
- Open the Configuration and Test dialog by click the property in *Representation*
- Exchange all symbols for the different states by SWITCH_02* Symbols.
- Under *Automatic line coloring*, select *Switch* as the *Function type*.
- Continue the line to the bottom.

Now, the switch behaves like a disconnector in the Runtime. We will see the difference between a switch and a disconnector, when we will deal with the command input and the topological interlocking.

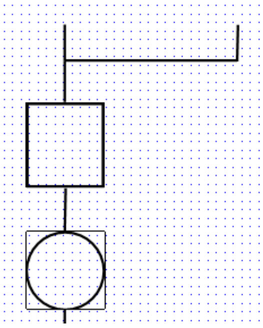
- Select the combined element of the disconnector.
- Create an Element Group, name it ‘disconnector’.
- Store the Element Group in the Project symbol library
- Do the same steps for the circuit breaker.

The Project symbol library should then have these two new symbols:



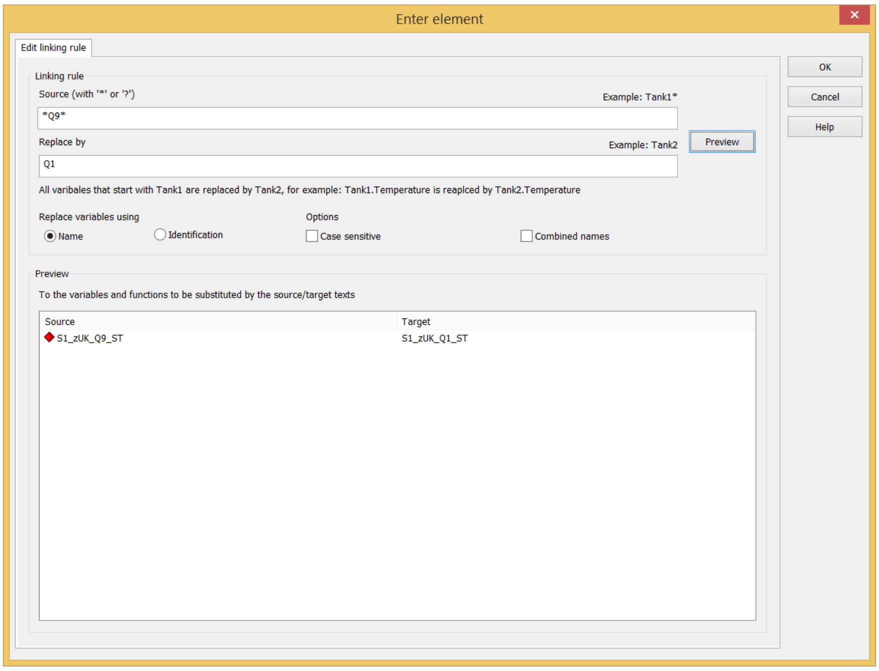
Disconnector	○
CircuitBreaker	□

- Now continue to draw the feeder with the lines like shown below:



- At each top end drag and drop a disconnector symbol from the Project library.

Now a substitute dialog opens:



Linking rule

Source (with '*' or '?') Example: Tank1*

Q9

Replace by Example: Tank2

Q1

All varibales that start with Tank1 are replaced by Tank2, for example: Tank1.Temperature is reaplced by Tank2.Temperature

Replace variables using: ● Name ○ Identification

Options: □ Case sensitive □ Combined names

Preview

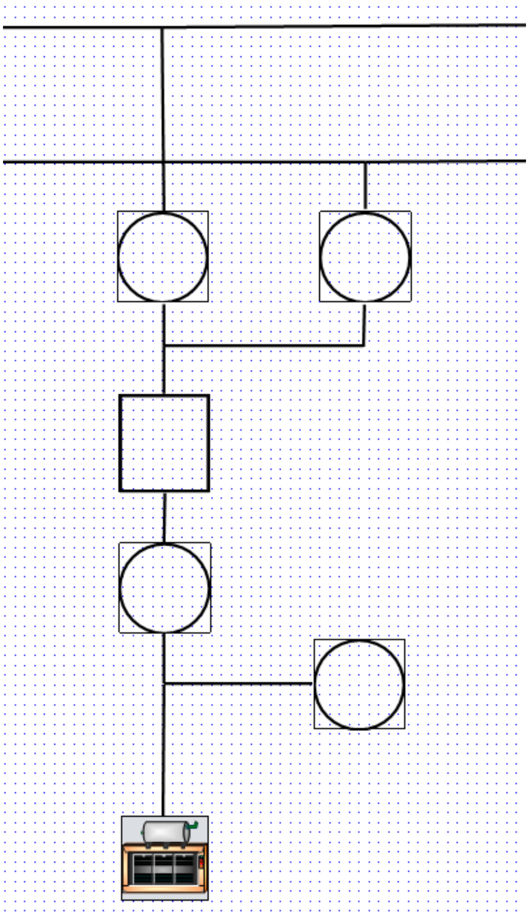
To the variables and functions to be substituted by the source/target texts

Source	Target
◆ S1_zUK_Q9_ST	S1_zUK_Q1_ST

- To link the variable ‘S1_zUK_Q1_ST’ write in the Source field *Q9* and in the Replace by field Q1. This will replace our Q9 with the Q1 variable. With Preview you can check first.
- Right click on the now inserted symbol and click on Symbol → convert linked to embedded symbol
- Do the same for the second busbar disconnector with variable ‘S1_zUK_Q2_ST’
- Again, continue the lines to the top.

- Draw a Line to the left and the right to visualize the bus bar.
- Add the Ground disconnector the same way as the busbar disconnectors.

Our network should look like this now:

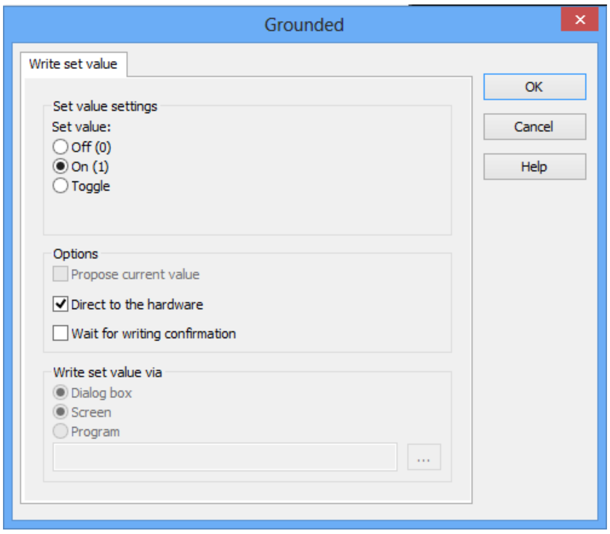


THE GROUNDING

A grounding is like a source that is always switched on. So the linked variable must always have the value 1.

- Create a function *Write set value* and select the variable 'Grounded'.
- Set the *Set value* to *On (1)*.

- Activate the option *Direct to hardware*.



Grounded

Write set value

Set value settings
Set value:
○ Off (0)
● On (1)
○ Toggle

Options
□ Propose current value
☑ Direct to the hardware
□ Wait for writing confirmation

Write set value via
● Dialog box
● Screen
○ Program

- Add this new function to the *script* 'AUTOSTART'.

In this case, reloading will not work. The AUTOSTART script is only executed at runtimestart. So we have to stop and restart the Runtime.

- Create a symbol for the grounding and name it 'Grounding'.
- Insert the Symbol to the project symbol library
- At the lower end of the left line add a *Combined-element* and select the variable 'Grounded'.
- In the assistant, select *symbol from library* as the *type of representation.*
- On the next page, select our symbol 'Grounding'.
- Close the assistant with *Finish*.

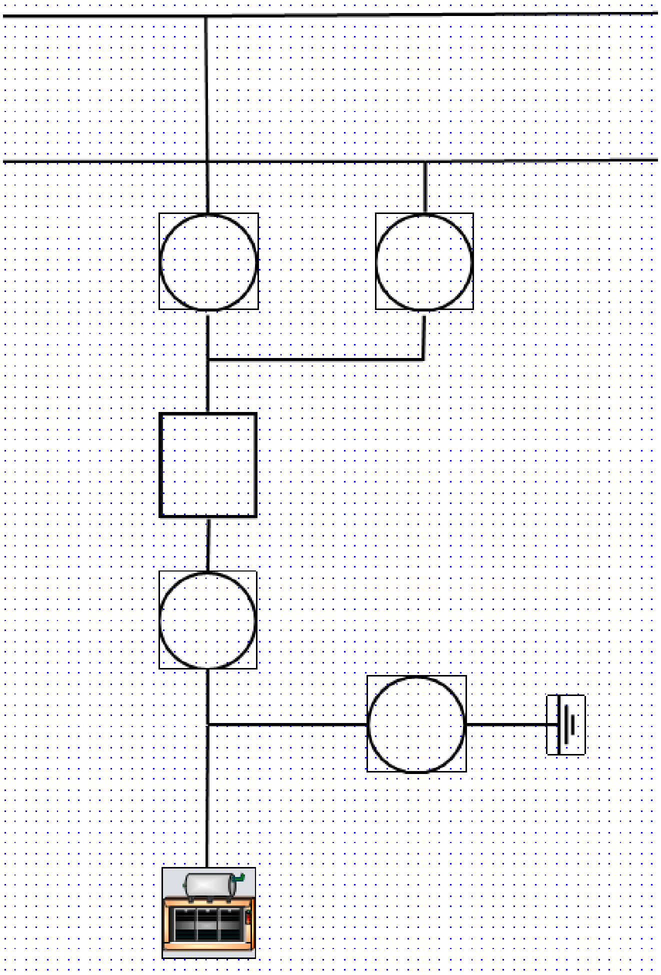
Now we still need to make some settings for our *Combined-element*.

- In the group *Automatic line coloring*, select a *Source* as the *Function type*.
- Select ‘GROUNDED [3]’ as the *Source*.
- Open the dialog box *Status and test* in the section *Representation*.
- Deactivate the *Original symbol colors* of the *Default status* and activate the option *from ALC* instead.

As the grounding is always on, we do not need to add another status here.

- Deactivate the properties *Binary value / Switch* and *Numeric value / SV active* in the section *Set value*.

This is what our line network should look like now:



The command processing

Learning objectives:

- You can create commands for secure switching.
- You know about the relation of the IEC61850 object model and the command processing.
- You can project commands that take into account external interlocking conditions.
- You know how to use command screen and the contextual menu to send commands.
- You know the topological interlocking and can configure it to fit your project needs.

Command input serves primarily for the secured switching of data points in energy switching stations, substations. 'Secured' means that there is a check whether the switching operation is allowed, according to the configured interlocking condition and with licensed topology pack the dynamically updated topology. The configuration of the topology is performed via the ALC.

These conditions determine whether the switching action is allowed now (no interlocking condition is active), whether it is forbidden (non-unlockable condition is active) or whether it can be performed after unlocking (unlockable interlocking is active).

A data point of the command input always consists of two data points: the response variable and the command variable. The action is performed on one of these data points.

User actions are performed via a context menu or with the screen type *Command*. Specific control elements are provided for this. This screen also includes a preprogrammed sequence that allows for unlocking, two-step execution and locking. The screen is activated via a screen element, a function or the context menu.

The synchronization of the execution of the command input is performed via an access object which is runtime-monitored and updated cyclically in the driver and which is associated to the response variable. The activation of this object is indicated by the status *Select* at the response variable.

The access objects are created on demand and associated with the creator on the RT side. Managing these objects in the driver makes sure that there is only one active access object per response variable.

The command groups can be exported and imported as well as exchanged and duplicated on the group level.

Screen for command processing

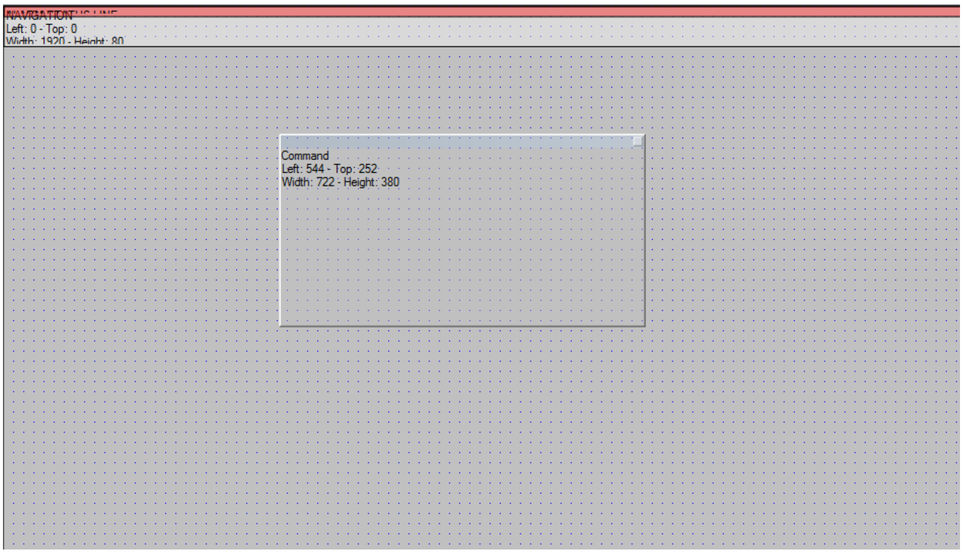
In this step by step example, we will configure and execute our first command input. During that, we will get to know a part of the functionality of the command input. After that, we will take a closer look at special functions.

A separate screen type *Command* was created for the user interaction of the command input during Runtime. In this screen, the operator performs the activities necessary for command execution. This can be e.g. the unlocking of an active interlocking or the confirmation of a two-step execution.

You can use specific control elements for this screen type, which allow all user actions necessary for command input and which visualize information about the status of the command input.

CREATE THE COMMAND SCREEN

- Create a new frame and name it 'Command input'.
- Adjust size and position according to the following illustration.



NAVIGATION
Left: 0 - Top: 0
Width: 1920 - Height: 80

Command
Left: 544 - Top: 252
Width: 722 - Height: 380

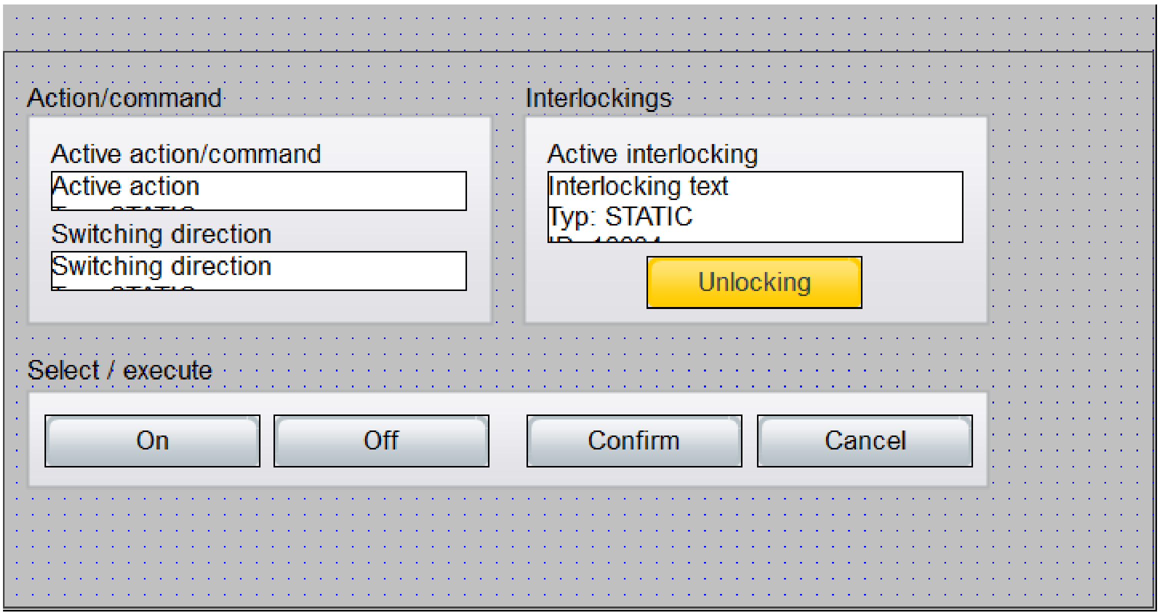
- In the group *General* select dark grey as the *Background color*.
- In the group *Border*, select *Size fixed* as the *Border type*.
- Activate the property *Title.*
- Activate the property *System Menu.*

This allows us to move the screen in the Runtime in case it overlaps important elements.

- Create a new screen and name it 'Command input 1'.
- Select *Commands* as the *Screen type*.
- Select our frame 'Command input' as the *frame*.

Next, we will add the standard control elements to the screen.

- Select the entry *Insert Template* in the menu *Control elements*.
- Choose *Standard* Template.



A number of control elements are added. We will look at the detailed functionality of the elements later. At the moment, we just use them.

REMA FOR THE SWITCHING DIRECTION TEXTS

The texts for the switching direction in the Runtime are determined via limits or REMA states. The texts are shown in the context menu and in the Control *Desired direction*.

- Create a new multi numeric reaction matrix and name it 'Switching direction'.
- Create a new status with the *Value* 0 and the *Limit text* 'open/off'.
- Create a new status with the *Value* 1 and the *Limit text* 'close/on'.
- Create a new status with the *Value* 2 and the *Limit text* 'diff'.
- Create a new status with the *Value* 3 and the *Limit text* 'failure'.
- For the *Default status*, enter the *Limit text* 'none'.

All other settings of the states are default.

- Select all variables with names ending with '_ST' or '_CO'.
- Open the section *Limits / Reaction matrix* in the property window.
- Activate the property *Reaction matrix active*.
- Under *Reaction matrix*, select the ReMa 'Switching direction'.

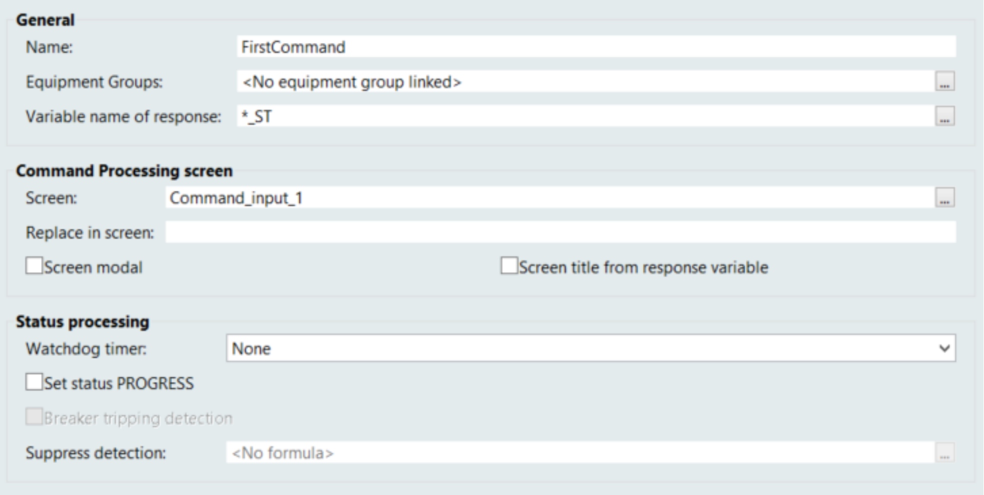
Creating a command group

Now we create a simple command group to get to know the configuration possibilities of the command group.

- Select the node *Command* in the *project manager*.
- Select the entry *New command group* in the context menu.

After creating a new command group, it is inserted in the tree with the standard name ‘Command group 1'.

Let us now have a look at the properties of the command group.



General

Name: FirstCommand

Equipment Groups: <No equipment group linked> ...

Variable name of response: *_ST ...

Command Processing screen

Screen: Command_input_1 ...

Replace in screen:

□ Screen modal □ Screen title from response variable

Status processing

Watchdog timer: None

□ Set status PROGRESS

□ Breaker tripping detection

Suppress detection: <No formula> ...

- Change the name to 'FirstCommand'.
- Under *Variable name of response*, enter '*_ST' .

The placeholder for the substitution text is the character '*' within a name. Only one placeholder can be used in a name. When entering text in the mask, it is important to take care that this name results in an existing variable name after substitution.

If the variable that is used here (substituted or absolute) does not exist during compiling, the command interlocking will not be available in the Runtime. A message in the output window will indicate this error when creating Runtime files.

- Under *Command screen* select our screen 'Command input 1'.

CREATING AN ACTION

Actions are the switching commands that are possible for the command interlocking, with the action-specific configuration.

- Select the node *Actions*.
- Open the context menu of the node *Actions*.
- Create a new switching command with *Command new*.
- Enter ' *_CO' as the *action variable* in the properties under *Action settings*.

Data point, on which writing actions are performed.

For some actions, this is the response variable. In this case, the field is locked.

The placeholder for the substitution text is the character '*' within a name. Only one placeholder can be used in a name.

If the variable that is used here (substituted or absolute) does not exist during compiling, this action will not be available in the Runtime. A message in the output window will indicate this error when creating Runtime files. Initialization: No Allocation

- Select *Switching command* as the *Action Type*.
- Activate the property *Nominal/current value comparison*.

If this property is activated, there will be a check whether the value of the response variable already matches the switching direction. If this is true, an unlockable interlocking variable is shown.

- Activate the property *Close automatically*.

This means that the command screen will be closed after the action has been executed.

- Select *Action 1 / Button_1* as the *Action button*.

This allocated the action to an action button in the screen. If such an allocation is missing, the action will not be available in the screen. Only the action buttons that were not allocated yet are provided in the selection list.

We will keep the other settings.

- Create another action with the same properties.
- Set the *Switching direction* to *OFF* and the *Command setting status* to 0.
- Select *Action 2 / Text Button_2* as the *Command button*.

ASSIGNING A COMMAND TO A VARIABLE

The command interlocking must be assigned to the variables, so that it is actually used.

- Select the variables. **_CO, *_ST.*
- Open the section *Write Set value* in the property window.
- Select *FirstCommand* under *Command group*.

ASSIGNING A COMMAND INPUT TO ELEMENTS

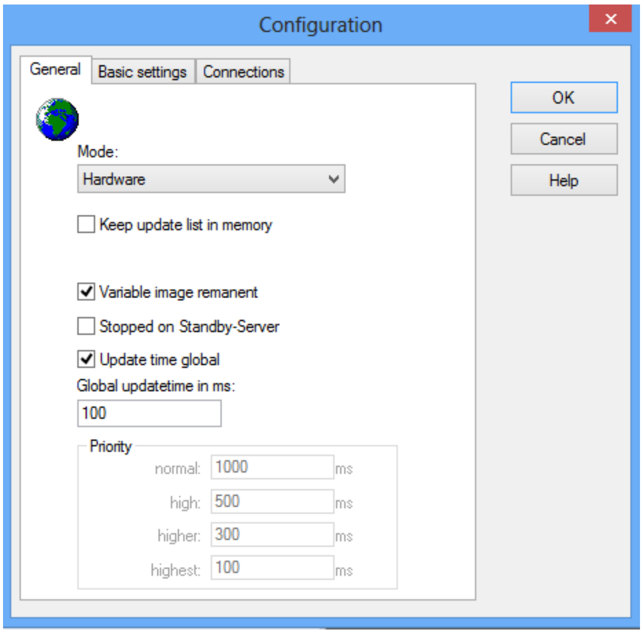
You can assign a command input to several elements.

- Select the Combined-elements of switches 1 and 2 in the 'SingleLineDiagram' screen.
- Open the section *Write set value / Numeric value* in the property window.
- Select *Command* for *Set value via*.

Now it is time to try out our command input in the Runtime.

THE COMMAND IN THE RUNTIME

- Change the driver mode to *Hardware* in the driver configuration dialog



Mode: Hardware

☐ Keep update list in memory

☑ Variable image remanent

☐ Stopped on Standby-Server

☑ Update time global

Global updatetime in ms: 100

Priority

normal: 1000 ms

high: 500 ms

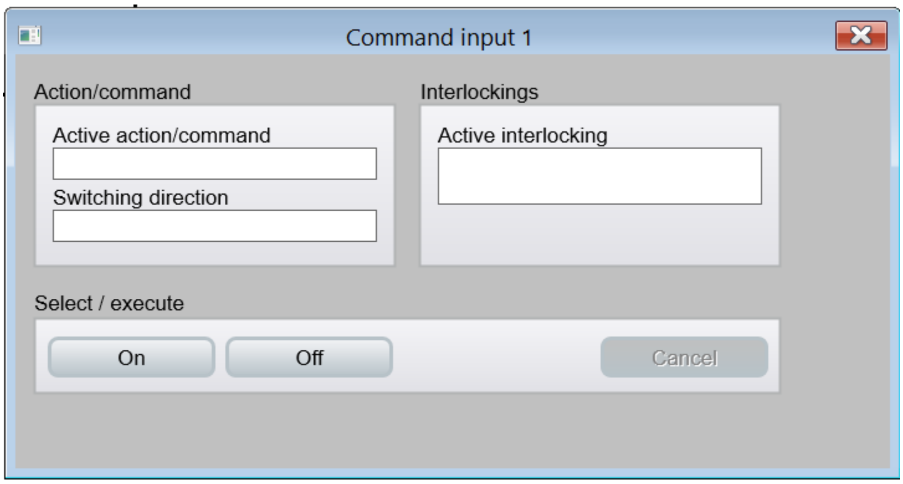
higher: 300 ms

highest: 100 ms

- Create new Runtime files in the Editor.

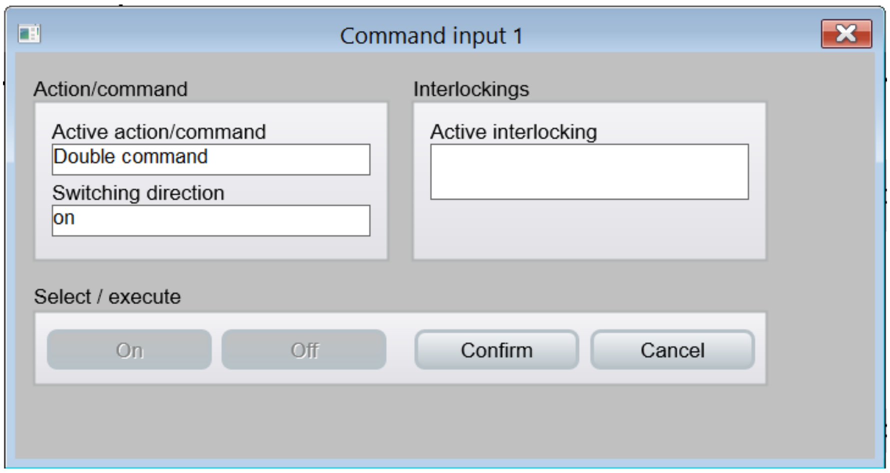
- Reload the changes in the Runtime.
- Click on switch 1 in the screen 'SingleLineDiagram'.

After allocating a command input to this switch, our command screen will now be opened instead of the standard dialog box for setting values.



You can execute the actions of the assigned command input via the action buttons.

- Click the button ‘On’.



Active action/command: Double command

Switching direction: on

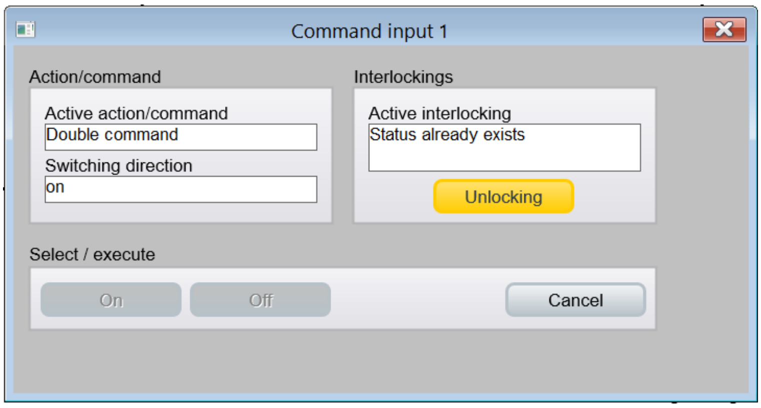
As the option two-stage is activated in the properties of this action by default, the action will not be executed immediately. First, you will be informed about the current status of the assigned command and response variables and the selected action.

- Click the button ‘Confirm’.

Only after this confirmation, the value will be written to the command variable. The command screen is closed automatically, as we activated the option *Close automatically* in the properties of our actions.

- Click on Switch 1 again.
- Click the button ‘On’.

This time, the button ‘Confirm’ does not appear at all. Instead, the interlocking *Status already exists* is displayed. This happens because we activated the property *Desired/Actual value comparison* when we created the action. However, you can unlock the interlocking with the button 'Unlock' and then execute the action anyway.



Command input 1

Action/command

Active action/command: Double command

Switching direction: on

Interlockings

Active interlocking: Status already exists

Command input in the context menu

Command input can also be activated via an element-specific context menu. But before we will extend our command input.

NEW ACTIONS

- Create a new *action* of type *Replace* in our command input.
- Set the *switching direction* to *ON*.
- Create another *action* of type *Replace* in our command input.
- Set the *switching direction* to *OFF*.
- Deactivate the property *Two-stage* for the *OFF* action
- Create another *action* of type *Release* in our command input. This action resets the ALT_VAL alternative value status bit.

Via the context menu, we will then be able to open and close the switches and also to substitute a value in alternative value mode via Replace..

Action Types:

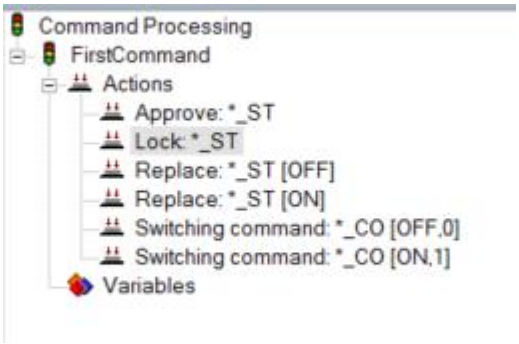
•	Command new	Switching command or pulse command. Uses the value of the command variable to write the configured command status to the controller. Note: the switching command is suitable for individual and dual commands with the Energy driver (IEC60870, IEC61850, DNP3).
•	New auto/remote command	The 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	The forced command action type allows the setting of a command, even if the response variable is empty, OFF, not top or INVALID . Note: the action is intended for emergency shutdowns and should only be used with caution.
•	New set value input	Writes a desired numerical value to the command variable.
•	New status input	Changes the status bits of the response variable. Only applicable for status bits in the modifiable status list.

• New replace	Changes the status of the response variable to substitute the value (alternative value.-.ALT_VAL) and writes value to the response variable.
• New revision	Sets the REVISION status bit of the response variable. Note: Alarm handling is suppressed in the revision.
• New manual correction	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	Switches off the response variable (OFF status bit). Note: the switched-off variables are no longer read by the connected hardware.
• New release	Sets substitute value replacement value (ALT_VAL) to 0. Note: as a consequence, the response variable has received the value from the controller again.
• Check response value	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	The response variable locks for further action upon entry of a valid locking code.

Lock and Tagging

THE ACTION LOCK

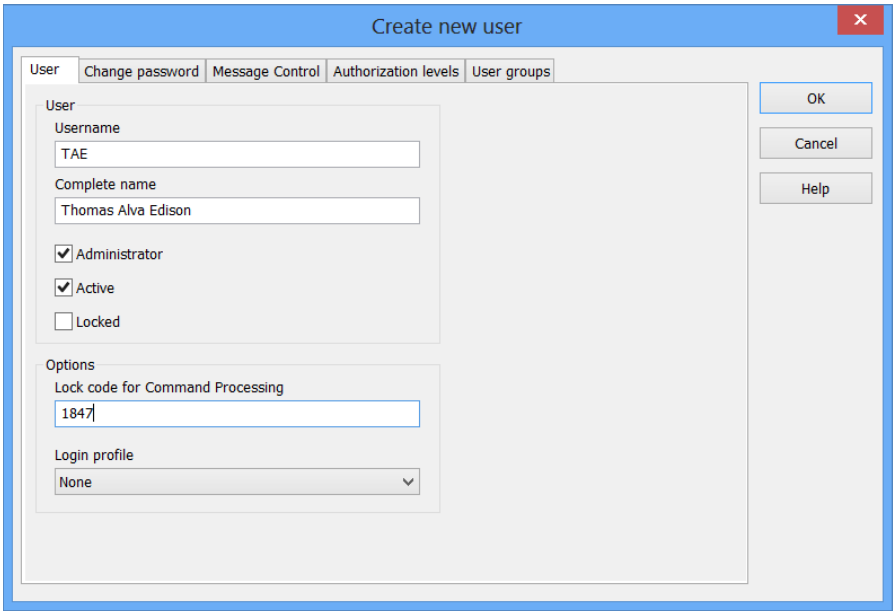
- Create a new *action* of type *New Lock* in our command input



Command Processing
FirstCommand
Actions
Approve: *_ST
Lock: *_ST
Replace: *_ST [OFF]
Replace: *_ST [ON]
Switching command: *_CO [OFF,0]
Switching command: *_CO [ON,1]
Variables

USER ADMINISTRATION

- Open the User administration branch from the Project Tree
- Select User, create a new User



User

Username: TAE

Complete name: Thomas Alva Edison

☑ Administrator

☑ Active

□ Locked

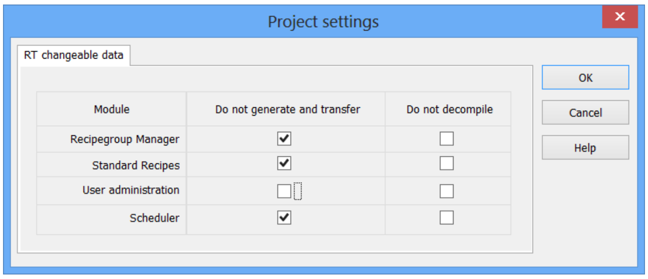
Options

Lock code for Command Processing: 1847

Login profile: None

- Select Administrator

- Enter a Lock code for Command Processing.
- Select the next tab *Change password* and set a password
- Close the dialog change back to the Project properties
- Take care of the RT changeable data settings in the project properties:

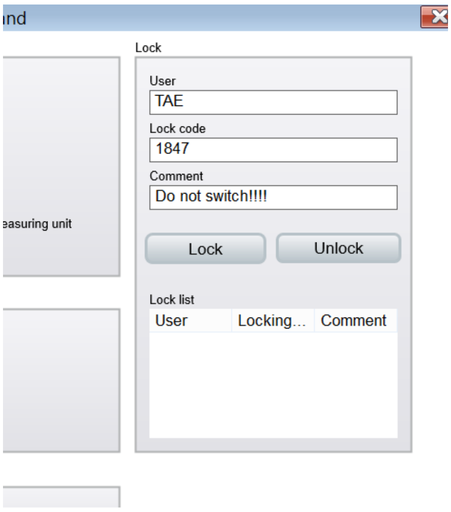


Module	Do not generate and transfer	Do not decompile
Recipegroup Manager	☑	□
Standard Recipes	☑	□
User administration	□	□
Scheduler	☑	□

- Change to the Screens branch select the frame *Command*
- Increase the Size to fit the complete template (~950 x 590)
- Edit the screen 'Command input 1'
- Insert new Template ‘Complete’ enable ‘delete existing screen elements’
- Save the command screen.

Now start / change to the Runtime

Click on a disconnector, the command screen opens.



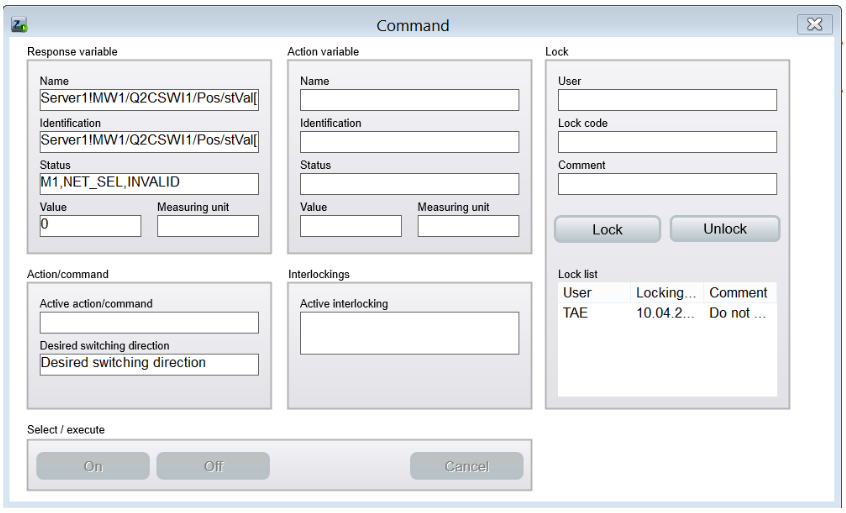
User: TAE

Lock code: 1847

Comment: Do not switch!!!!

- Type in the User TAE and the Lock code 1847.
- Type in a comment.
- Click Lock

The switch is locked now with user comment. No one can perform any action on it except lock. There can be a list of Lockings.



Name: Server1!MW1/Q2CSWI1/Pos/stVal[

Identification: Server1!MW1/Q2CSWI1/Pos/stVal[

Status: M1,NET_SEL,INVALID

Value: 0

Measuring unit:

Active action/command:

Desired switching direction: Desired switching direction

User	Locking...	Comment
TAE	10.04.2...	Do not ...

To unlock the switch, type in the USER ID and the Lock code and click Unlock.

The lock and release is reported in the chronologic event list.

DEFINING A MENU FOR COMMANDS

- Select the node *Menus* in the *project manager*.
- Select the *Context menus*.
- Create a new context menu named 'Commands'.

In the Energy Edition of the control system, we have an additional action type for switching commands. We will use it now.

- Open the section *Representation / Type* in the properties of the first menu entry.
- Select *Command* as the *Action type*.

‘Command input’ will be used as *Text automatically*. We do not have to worry about this, because this is a dynamic menu entry, which is automatically created from the linked command input. This single entry can also be replaced by multiple entries in the Runtime, which correspond to the actions of the linked command input.

Via the property *Menu ID*, we can restrict the type of actions to be displayed in the menu, if required. After selecting the *Command input* as the *Action type*, we can select the *Menu ID* from a list.

- Select *ID_CMD_AUTO* as the *Menu ID*.

This means that all actions of the linked command input will be displayed in the Runtime.

We have not assigned a command input to the menu entry yet. However, this is not necessary. The control system automatically recognizes the command input from the variable when we assign the context menu to a dynamic element. Therefore, this single context menu can be used for variables with different command inputs, always offering the right actions in the Runtime.

ASSIGNING THE MENU

Now we are going to assign the context menu to different dynamic elements.

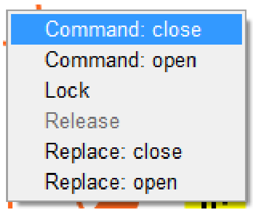
- Select the Combined-elements of switches 3 to 5 and the disconnector 1 in the ‘SingleLineDiagram’ screen.
- Open the section *Runtime* in the property window.
- Select our context menu ‘Commands’ under *Menu*.

From the property *Set value / Interlocking* of the linked variables, the control system can now tell which command inputs must be used for the context menu.

- Create new Runtime files in the Editor.
- Reload the changes in the Runtime.

THE CONTEXT MENU IN THE RUNTIME

- Activate the context menu of Switch 3 with the right mouse button.



- Select the entry *Command: close*

Now the switch is not turned on right away, because the property *General / Two-stage* is activated in the properties of the action. Therefore, our command input window opens, in which you can execute the second step of the action.

- Click the button ‘Execute 2.’.(Confirm)
- Open the context menu again and execute the action *Replace: close*.

Again, you must execute the second step of the action in the command input window.

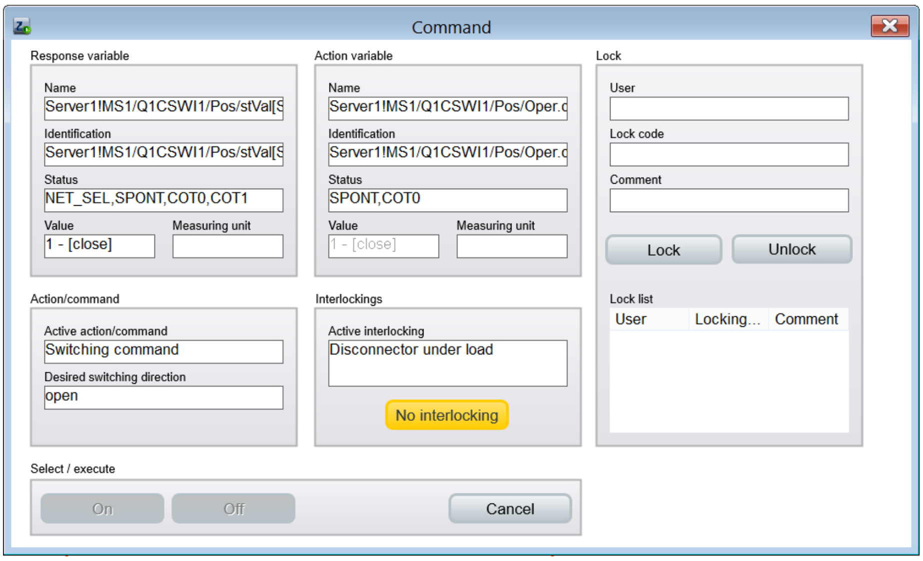
- Open the context menu again and execute the action *Replace: open*.

This time, the command input screen does not open. Instead, the action will be executed right away, as we deactivated the property *Two-stage* for this action.

Make sure that Switches Q1 and Q0 are turned on.

- Open the context menu on the disconnector Q9 and execute the action *Replace Close.*

The command screen opens again. This time, however, the button ‘Execute 2.’ is not available.



Response variable

Name: Server1!MS1/Q1CSWI1/Pos/stVal[S

Identification: Server1!MS1/Q1CSWI1/Pos/stVal[S

Status: NET_SEL,SPONT,COT0,COT1

Value: 1 - [close]

Measuring unit:

Action variable

Name: Server1!MS1/Q1CSWI1/Pos/Oper.c

Identification: Server1!MS1/Q1CSWI1/Pos/Oper.c

Status: SPONT,COT0

Value: 1 - [close]

Measuring unit:

Lock

User:

Lock code:

Comment:

Action/command

Active action/command: Switching command

Desired switching direction: open

Interlockings

Active interlocking: Disconnector under load

Lock list

User	Locking...	Comment

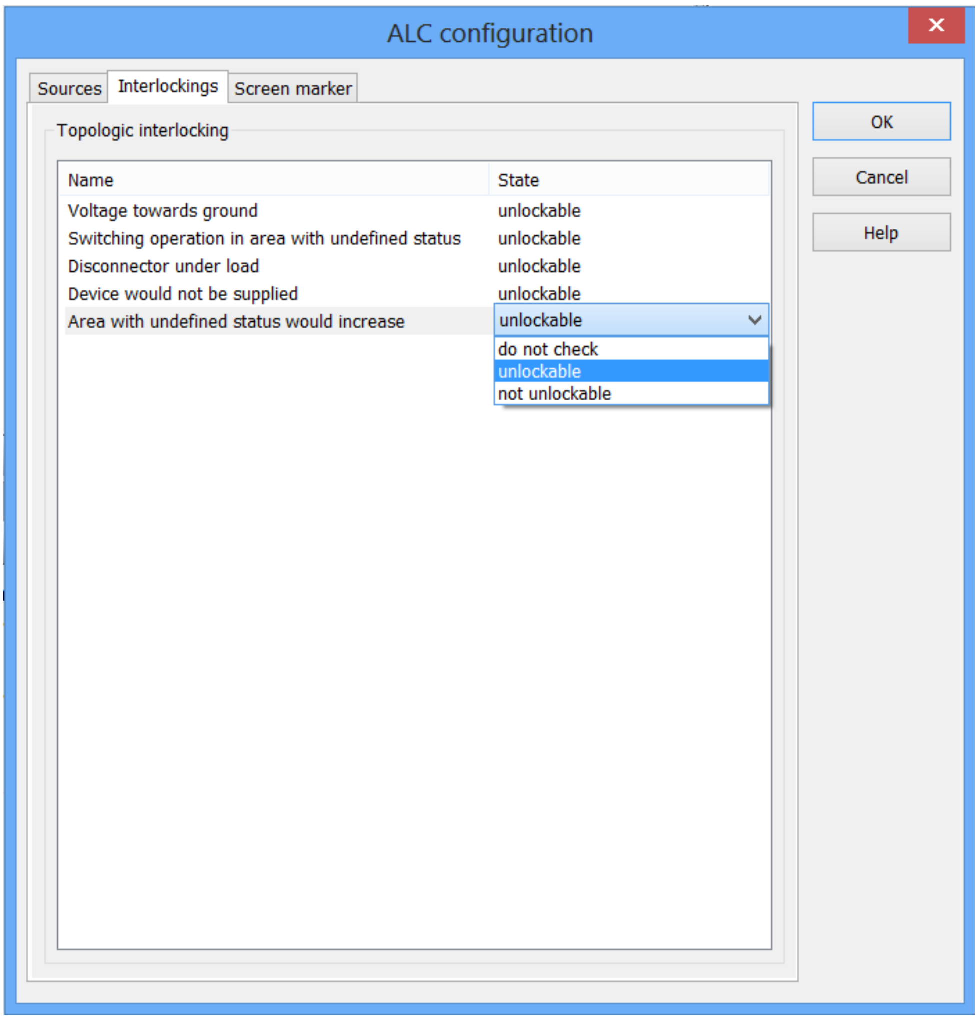
However, we can see an unlockable interlocking *Disconnector under load*. This a topological interlocking. We will take a closer look at that now.

- Switch back to the Editor.

Topological interlockings

Topologic interlockings are pre-defined in the energy edition of the control system.

- Select the project in the project manager.
- Open the section *Automatic Line Coloring* in the property window.
- Open the dialog *ALC configuration*.
- Switch to the second page *ALC interlockings*.



Topologic interlocking

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 topological interlockings are an integral part of the control system and will be checked automatically by the Automatic Line Coloring in the background. However, you can define for every type of interlocking whether it is unlockable or not. You can also define that an interlocking is not checked at all.

Projecting of a Substation

Learning objectives:

- You can create a library of substation related symbols for your future projects
- You can reuse your created symbols for efficient drawing single line diagrams of the whole substation.
- You know how to project a transformer for your single line diagram.
- You know how to project and use Worldview zoom and pan for your project.

The Worldview screen

Now we have one feeder in our SingleLineDiagram screen, if we want to expand this screen, we need a larger screen.

A very fast way to get more space for more elements is the Worldview picture.

To make a wordview picture we only need to enlarge the screen just by editing the properties for the screen size.

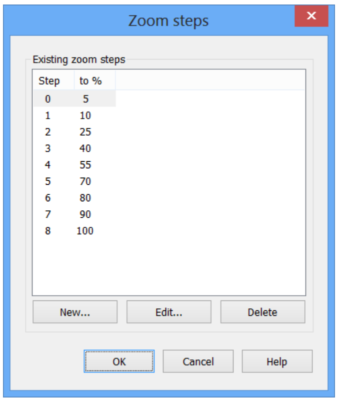
- Select the screen ‘SingleLineDiagram’ and *unselect* in the properties *Size: Size from frame [ ]*
- Type in the now enabled Boxes: Width [pixels]: 4000 and Height [pixels] 6000
- Leave the option do not adapt element to screen resolution unselected.

With these settings, we get now more space for drawing further feeder in our screen.

When you now open the SingleLineDiagram screen in the runtime, you can rightclick and hold the mouse and move the screen.

We also want to zoom the screen. For this, we need to adapt some properties in the Project settings.

- Select the Project in the Project manager tree.
- In the properties for *Graphical design* click on the property *Zoom steps for world view [Click here->]*
- Enter minimum 4 new steps



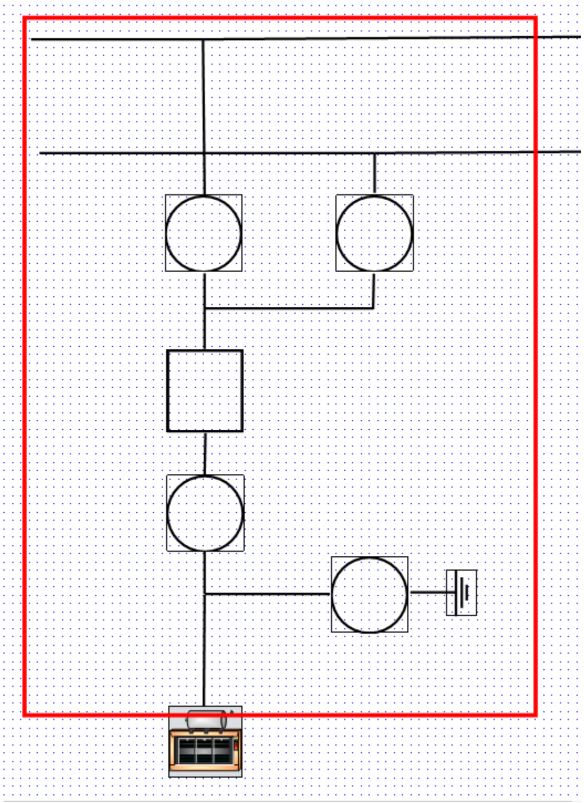
Step	to %
0	5
1	10
2	25
3	40
4	55
5	70
6	80
7	90
8	100

With those settings, we can now zoom the screen in the runtime. If you hold the [ctrl]-key and move the mouse-wheel you can zoom in and out the worldview screen.

Feeder Symbols

Creating symbols of feeder

Mark all elements of a feeder by leftclick and hold the mouse move it over the feeder.



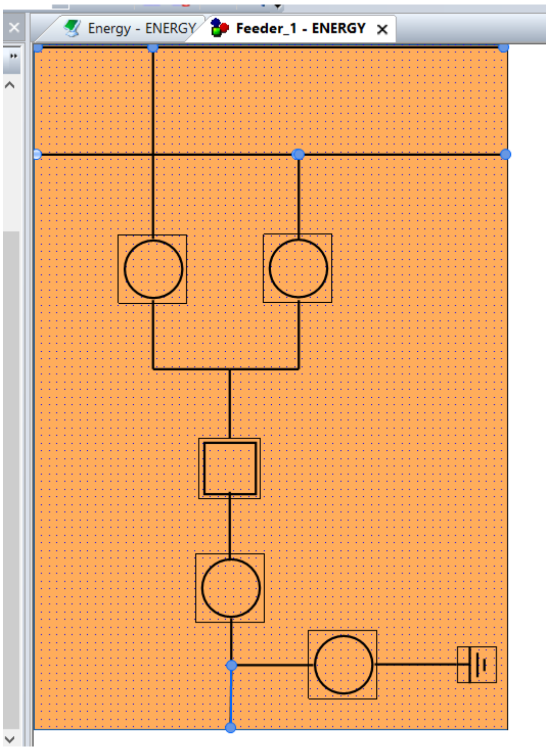
When all elements and lines are marked rightclick and select Symbol -> Create embedded Symbol.

Rename the Symbol to Feeder1.

Rightclick on the symbol and select Symbol → Insert into Symbol Library.

Insert it into the project library.

Now change to the project library and select the just created feeder symbol.



Extend all lines, that they are ending on the border of the symbol, this is needed that we can arrange them to a substation.

Building the Substation

Preparatory Work:

For our new feeders, we need to create the database first.

The simulated IEC61850 server contains the IEDs ‘UK1’,’MW1’,’MS1’ and ‘FS1’

Change back to the driver. Select from the contextual menu ‘import variables from driver’ import the additional variables as we did at the beginning of our project.

You could set a filter string to filter down to the position objects we are interested in:

Filter: *MW1*CSWI*Pos/*Val*

```
Server1!zMW1/Q0CSWI1/Pos/Oper.ctlVal[CO]
Server1!zMW1/Q0CSWI1/Pos/stVal[ST]
Server1!zMW1/Q1CSWI1/Pos/Oper.ctlVal[CO]
```

```
  Server1!zMW1/Q1CSWI1/Pos/stVal[ST]
  Server1!zMW1/Q2CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMW1/Q2CSWI1/Pos/stVal[ST]
  Server1!zMW1/Q8CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMW1/Q8CSWI1/Pos/stVal[ST]
  Server1!zMW1/Q9CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMW1/Q9CSWI1/Pos/stVal[ST]
```

Filter: *MS1*CSWI*Pos/*Val*

```
  Server1!zMS1/Q0CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMS1/Q0CSWI1/Pos/stVal[ST]
  Server1!zMS1/Q1CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMS1/Q1CSWI1/Pos/stVal[ST]
  Server1!zMS1/Q2CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMS1/Q2CSWI1/Pos/stVal[ST]
  Server1!zMS1/Q8CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMS1/Q8CSWI1/Pos/stVal[ST]
  Server1!zMS1/Q9CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zMS1/Q9CSWI1/Pos/stVal[ST]
```

Filter: *FS1*CSWI*Pos/*Val*

```
  Server1!zFS1/Q0CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zFS1/Q0CSWI1/Pos/stVal[ST]
  Server1!zFS1/Q1CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zFS1/Q1CSWI1/Pos/stVal[ST]
  Server1!zFS1/Q2CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zFS1/Q2CSWI1/Pos/stVal[ST]
  Server1!zFS1/Q8CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zFS1/Q8CSWI1/Pos/stVal[ST]
  Server1!zFS1/Q9CSWI1/Pos/Oper.ctlVal[CO]
  Server1!zFS1/Q9CSWI1/Pos/stVal[ST]
```

- When you have imported all variables, rename the variables the same way as done in section ‘rename imported variables’.

You should end up with variable names:

```
S1_zMW_Q0_ST
S1_zMW_Q0_CO
…
S1_zMS_Q0_ST
S1_zMS_Q0_CO
…
```

S1_zFS_Q0_ST
S1_zFS_Q0_CO
…

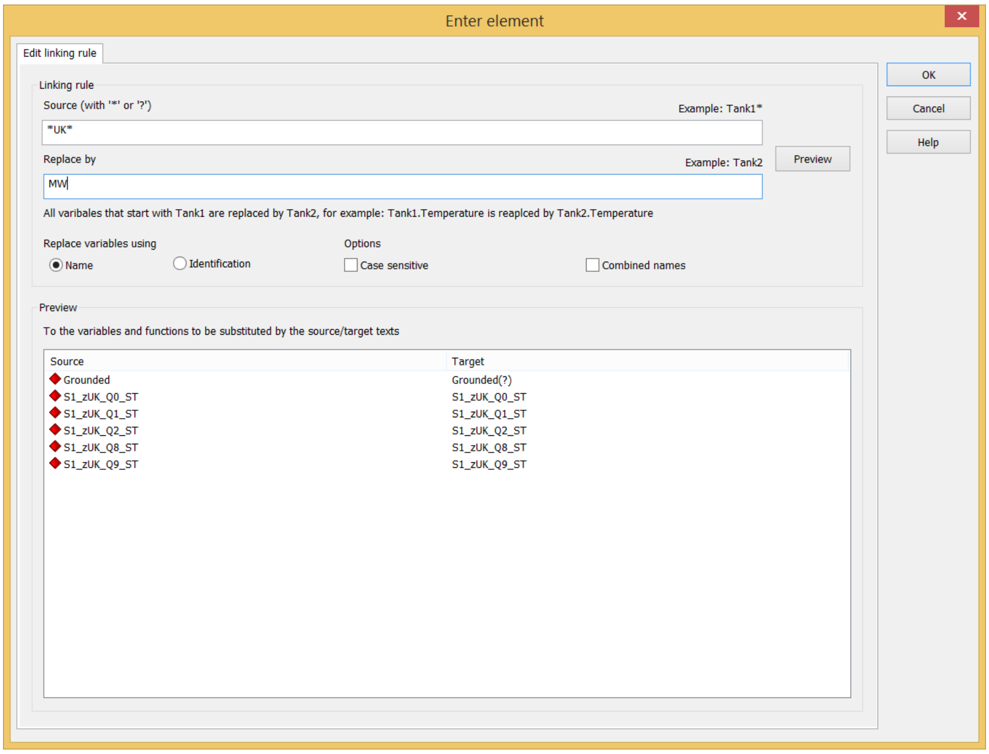
- Select all of them and link the reaction matrix 'Switching direction' in the limit properties.

The command interlocking must be assigned to the variables, so that it is actually used.

- Select the variables. **_CO, *_ST.*
- Open the section *Write Set value* in the property window.
- Select *FirstCommand* under *Command group*.

Creating the Substation with our symbols of the feeder

When you now drag and drop the feeder symbol to our SingleLineDiagram screen a substitute dialog opens:



Enter element

Edit linking rule

Linking rule

Source (with '*' or '?') Example: Tank1*

UK

Replace by Example: Tank2

MW

All varibales that start with Tank1 are replaced by Tank2, for example: Tank1.Temperature is reaplced by Tank2.Temperature

Replace variables using: ● Name ○ Identification

Options: □ Case sensitive □ Combined names

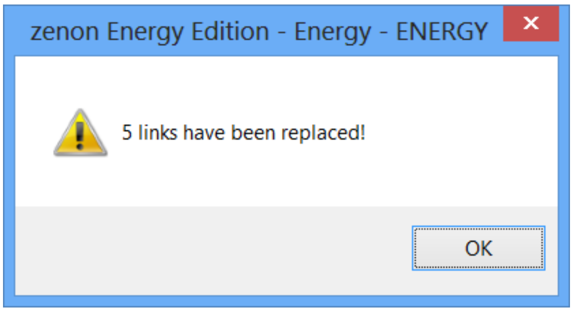
Preview

To the variables and functions to be substituted by the source/target texts

Source	Target
◆ Grounded	Grounded(?)
◆ S1_zUK_Q0_ST	S1_zUK_Q0_ST
◆ S1_zUK_Q1_ST	S1_zUK_Q1_ST
◆ S1_zUK_Q2_ST	S1_zUK_Q2_ST
◆ S1_zUK_Q8_ST	S1_zUK_Q8_ST
◆ S1_zUK_Q9_ST	S1_zUK_Q9_ST

Enter '*UK*' for Source and 'MW' (MS,FS) for the Replace by parameter.

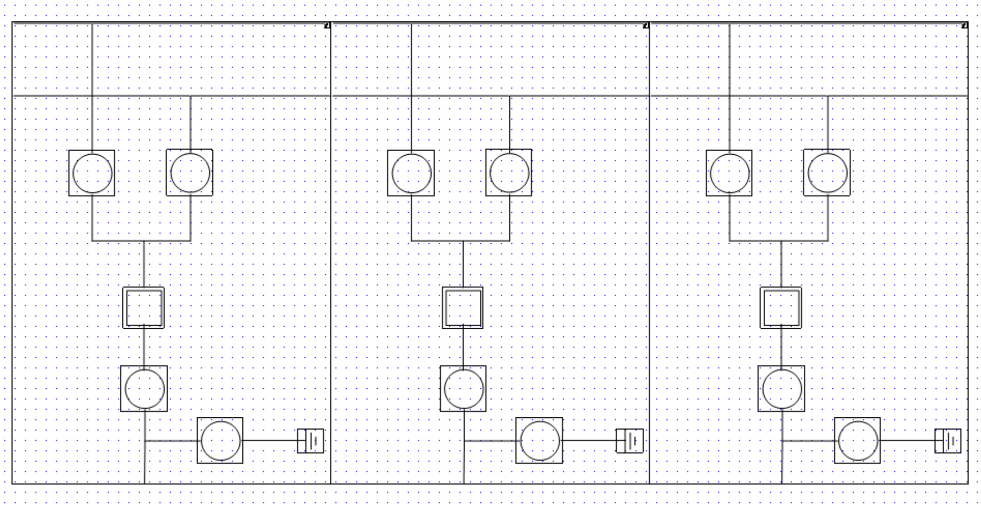
When you click on Preview you should get the Info message of replaced links



zenon Energy Edition - Energy - ENERGY

5 links have been replaced!

Place now a second Feeder right next to the just placed one. Take care that the lines connect. Symbol by symbol the substation is build. For our Training Project we only use the UK, MW and MS variables. Of course, in a real Project you would then substitute the correlated IED variables to each symbol.



You can now create a feeder in shape ‘upside down’

Follow the steps from above to create also a symbol out of it.

Now we have all symbols to complete our substation.

A second source

In the next step, we will add another source. We will find out how easily existing objects can be reused in the control system at any time.

THE SECOND SOURCE

- Switch back to the Editor.
- Copy the source from our first feeder.
- Assign the variable 'Source_B' to this source.
- In the group *Automatic line coloring*, select 'Source_B' as the *Source*.

All other settings remain unchanged.

- Place the source combined element at the end of the second feeder

DISPLAYING MULTIPLE SUPPLY

Now we will lead our sources together. Our bus bar can now be multiple supplied

- Open the feeder symbol in the project symbol library.
- Select the lines from the bus bar in the feeder symbol.
- Open the section *Automatic Line Coloring* in the property window.
- In property Priority for Display select 'Multiple supply'.
- In property *Display multiple supplies*, select '*two highest priority sources'*.

The last line can now be supplied from two different *sources* at the same time. We can define this in the properties of the *Automatic line coloring*.

In the Runtime, you can now see that the line is displayed as dashed - in the colors of the two sources, if both sources and switches are switched on.

THE CONSUMER

Now, we are going to put a consumer at the end of one feeder.

- Create a symbol for the consumer and name it 'Consumer1' and save it in the project library.
- At the lower end of the line, add a *combi-element* and select the variable 'Consumer1'.
- In the assistant, select *symbol from library* as the *type of representation.*
- On the next page, select the symbol 'Consumer1'.
- Close the assistant with *Finish*.

Now we still need to make some settings for our *combi-element*.

- In the group *Automatic line coloring*, select a *Drain* as the *Function type*.
- Open the dialog box *Configuration and test* in the section *Representation*.
- Deactivate the *Original symbol colors* of the *Default status* and activate the option *from ALC* instead.
- Deactivate the properties *Binary value / Switch* and *Numeric value / SV active* in the section *Set value*.

The Transformer

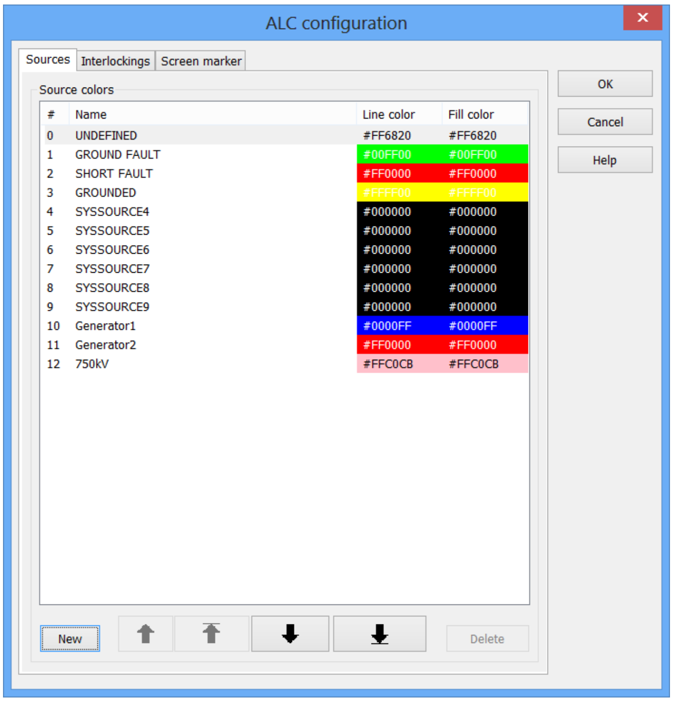
The transformer is the heart of the substation. From ALC point of view a transformer is like a source with different voltage levels on both ends. It is a drain and a source the same time.

- Draw a new combined element, select the variable ‘*Trafo’* for it.
- Set in the properties for the Automated Line Coloring the Function Type Transformer.

We need to select also a Source and a Source for reverse feed.

The source for reverse feed can be one of our already existing ones. To see better the functionality of ALC we will now introduce a new source.

- Open the ALC configuration dialog in the project properties
- Add a new Source to the ALC Configuration, name it 750kV



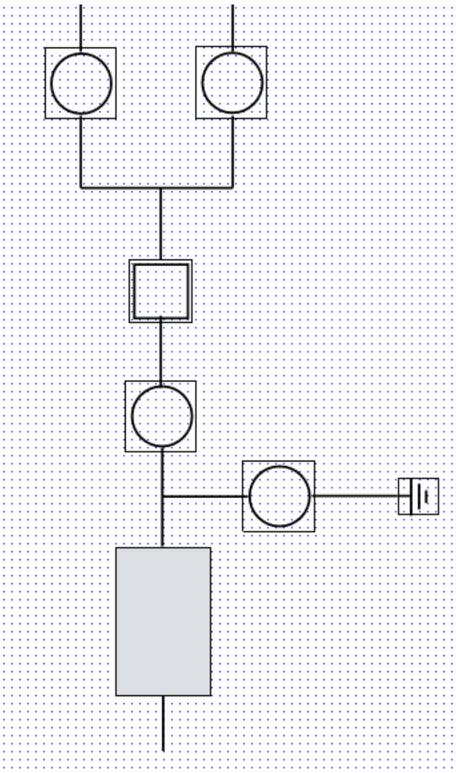
#	Name	Line color	Fill color
0	UNDEFINED	#FF6820	#FF6820
1	GROUND FAULT	#00FF00	#00FF00
2	SHORT FAULT	#FF0000	#FF0000
3	GROUNDED	#FFFF00	#FFFF00
4	SYSSOURCE4	#000000	#000000
5	SYSSOURCE5	#000000	#000000
6	SYSSOURCE6	#000000	#000000
7	SYSSOURCE7	#000000	#000000
8	SYSSOURCE8	#000000	#000000
9	SYSSOURCE9	#000000	#000000
10	Generator1	#0000FF	#0000FF
11	Generator2	#FF0000	#FF0000
12	750kV	#FFC0CB	#FFC0CB

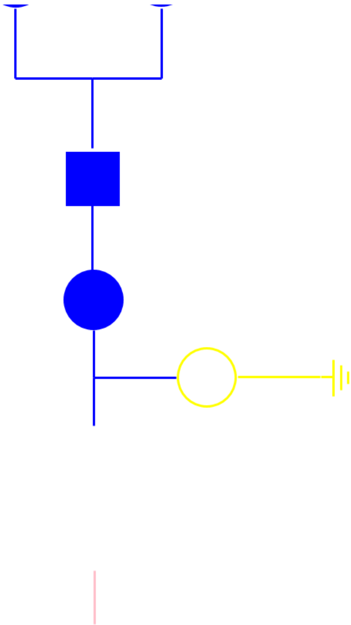
We can now define the source for our transformer

- Select the transformer Combined element.
- In the properties for the ALC set as Source 750kV
- As Source in reverse feed select Source_A

Place the transformer that it is feed by the first feeder, remove the motor we defined first. Add a line below the transformer.

The Feeder should look like this now:



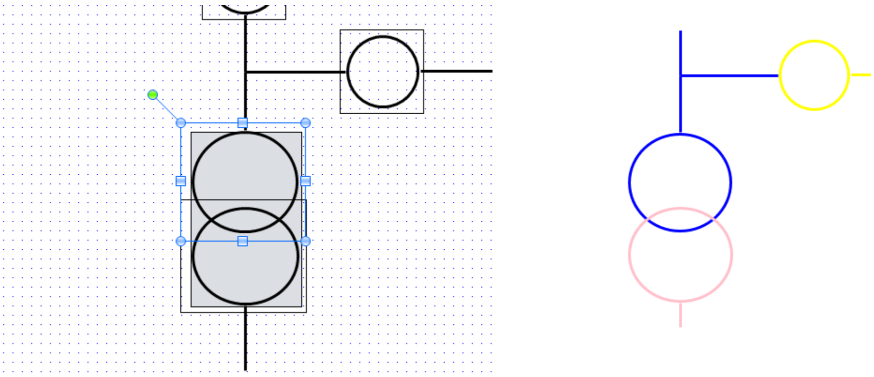


Start the Runtime and test how the lines are colored.

The transformer itself is only a ‘virtual’ device for ALC. To get a Visualisation of a Transformer, we use our switch (disconnector) open symbols. To have a variable for the switch we use the internal boolean variable ‘Trafo’.

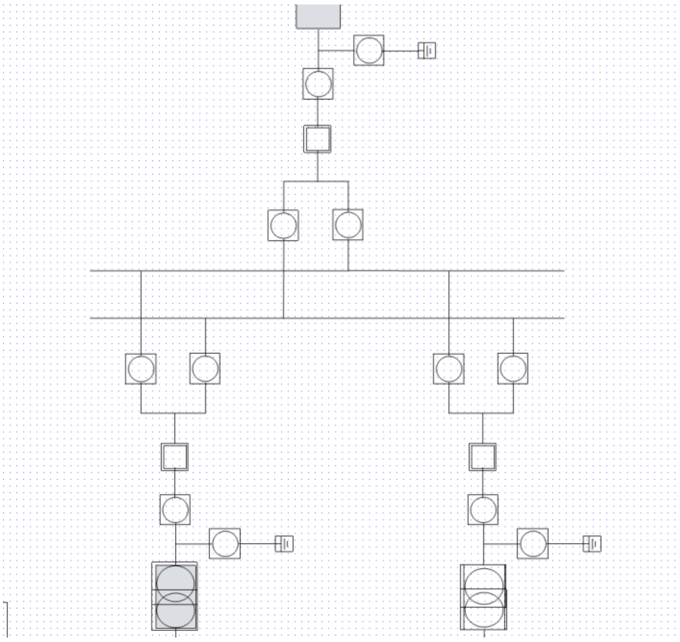
- Place two combined elements with the ALC function type ‘switch’ and a Circle as Symbol (e.g. diconnector open) linked over the transformer combined element. Link the upper Element to the upper line, the lower to the lower line.

The transformer should look like this now in Editor and Runtime:



Add a source to the in feeding lines, place a drain or/and a transformer to the out feeding lines.

It can now look like this below, but create your very own substation.



Check in the runtime if your projecting is correct.

Set up defined Interlocking condition

DEFINING A CONDITION VARIABLE

User defined interlocking conditions can be defined for every action. These conditions allow an additional restriction to the executability of the action.

One situation where an external condition would block the command is the local/remote operation status information. The next chapter will show a short example to lock the command if the status information variable is true.

These conditions are defined with formulas, in which you can use the variables from the active projects. The variables available for the command interlocking must be defined.

We will use internal variables for our project.

- Create variables based on the Internal driver, type BOOL:

S1_zUK_Q0_IV

S1_zMW_Q0_IV

S1_zMS_Q0_IV

S1_zFS_Q0_IV

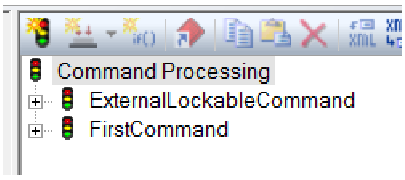
- Set in the internal Variable settings*:* *Network* for calculation type

The formula addresses these variables via the Index. The condition variable is automatically substituted if you use a ’*’ in the definition.

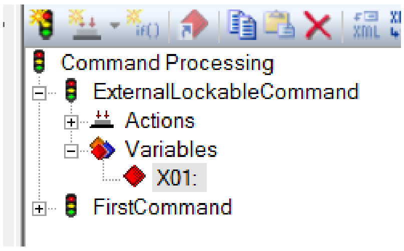
We will now create a new command group where we will use this interlock conditions.

- Copy our 'First command' with the contextual menu entry copy or Ctrl+C
- Paste a new command group Ctrl+V

Rename the new created command group ‘ExternalLockableCommand’



In the context menu of the node *Variable*, you can add a new condition variable with *AddVariable*. Close the variable selection dialog with OK. This results in the creation of an empty definition.



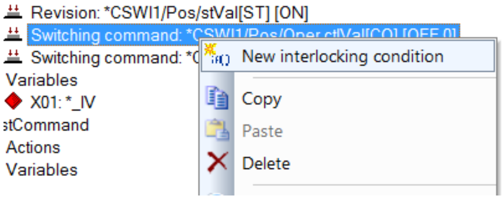
- Enter '*_IV' in the properties under *General / Variable*.

DEFINING A NEW INTERLOCKING CONDITION

Often it is necessary to lock a command depending on an external state. E.g., enable/disable a command because it is temporary switched to local operation.

For the switching command action in the command group 'ExternalLockableCommand' create a new interlocking condition:

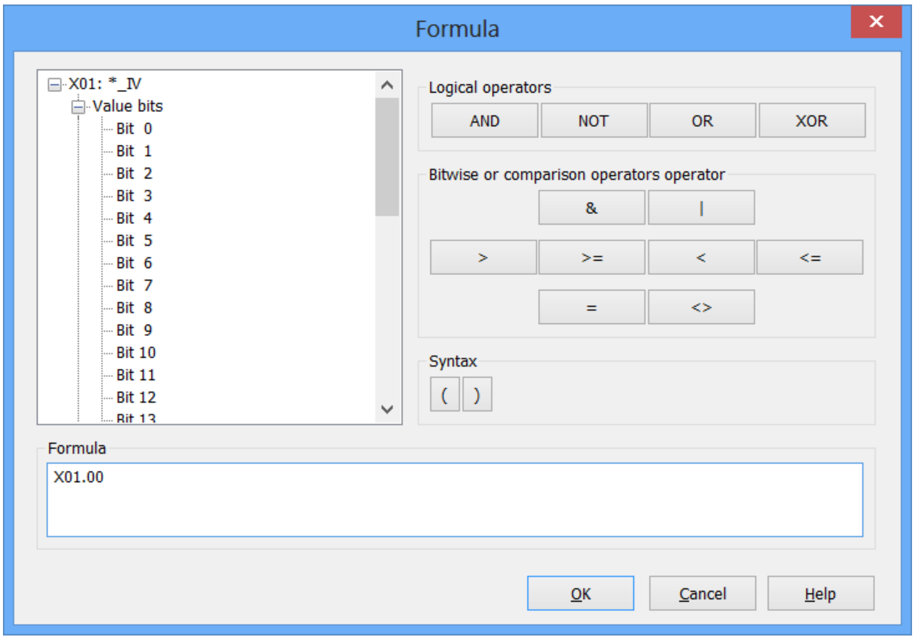
- Rightclick on action *Switching command: *_CO [OFF,0].*
- Create *new interlocking condition*


Revision: *CSWI1/Pos/stVal[ST] [ON]
Switching command: *CSWI1/Pos/Oper.ctlVal[CO] [OFF,0]
Switching command: *C
Variables
X01: *_IV
stCommand
Actions
Variables
New interlocking condition
Copy
Paste
Delete

Now we can define the condition in the formula editor. This logical condition must be true in order to activate the interlocking of this action.

- Click on *Logical link*.

Now the formula editor opens.



X01.00

- Enter the formula as displayed above and close the formula editor with *OK*.
- Enter 'Interlocked by external condition' as the *Interlocking text*.

This text will be shown in the command screen if the condition is met.

- Activate the property *Unlockable*.

This allows you to unlock the interlocking in the Runtime, in case you want to execute the action anyway.

- Copy this interlocking condition and paste it to the action: *Switching command: *_CO [ON,1].*

ASSIGNING THE COMMAND TO A VARIABLE

The new created command group must be assigned to the circuit breaker variables, so that it is actually used.

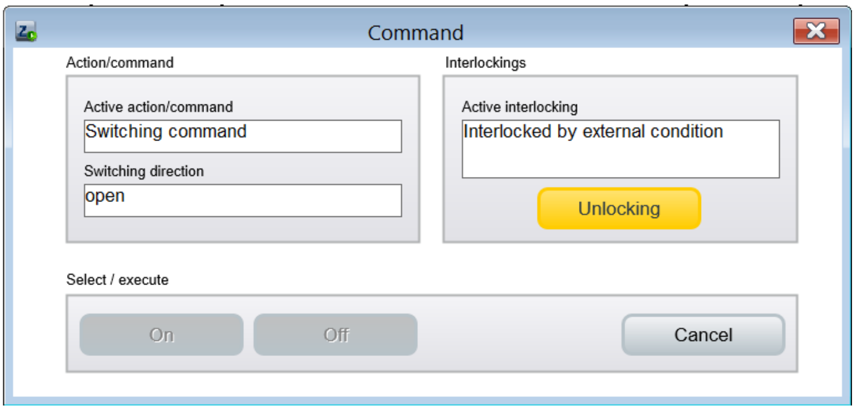
- Select the variables **Q0_CO, *Q0_ST.*

- Open the section *Write Set value* in the property window.
- Select *ExternalLockableCommand* under *Command group*.

Create for each interlocking Variable *_IV one numerical Value element in the screen and place them according to their name to the switches Q0 of each feeder.

Start the Runtime.

If you now set the interlocking variable to 1 (true) and perform a command, the external interlocking will be displayed in the Active interlocking field.



Active action/command: Switching command

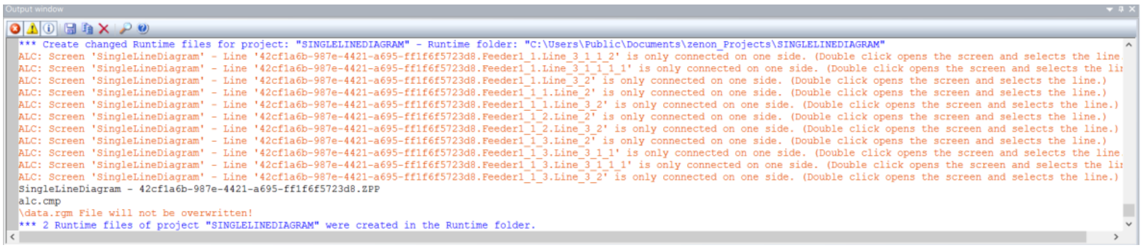
Switching direction: open

Active interlocking: Interlocked by external condition

Finalizing the Grid

TERMINATOR FOR LINES

Now when compiling the project many error messages appear in the Output window:



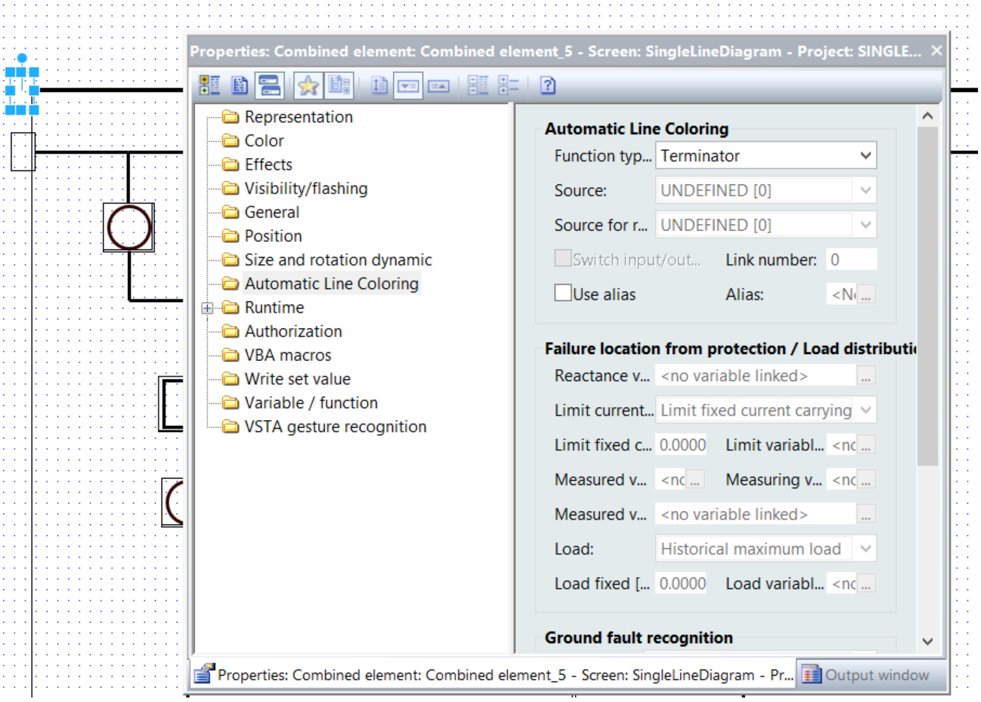
```
*** Create changed Runtime files for project: "SINGLELINEDIAGRAM" - Runtime folder: "C:\Users\Public\Documents\zenon_Projects\SINGLELINEDIAGRAM"
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1.Line_3_1_1_2' is only connected on one side. (Double click opens the screen and selects the line
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1.Line_3_1_1_1_1' is only connected on one side. (Double click opens the screen and selects the li
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1.Line_3_2' is only connected on one side. (Double click opens the screen and selects the line.)
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_1.Line_2' is only connected on one side. (Double click opens the screen and selects the line.)
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_1.Line_3_2' is only connected on one side. (Double click opens the screen and selects the line.)
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_2.Line_2' is only connected on one side. (Double click opens the screen and selects the line.)
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_2.Line_3_2' is only connected on one side. (Double click opens the screen and selects the line.)
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_3.Line_2' is only connected on one side. (Double click opens the screen and selects the line.)
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_3.Line_3_1_1' is only connected on one side. (Double click opens the screen and selects the line
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_3.Line_3_1_1_1' is only connected on one side. (Double click opens the screen and selects the li
ALC: Screen 'SingleLineDiagram' - Line '42cf1a6b-987e-4421-a695-ff1f6f5723d8.Feeder1_1_3.Line_3_2' is only connected on one side. (Double click opens the screen and selects the line.)
SingleLineDiagram - 42cf1a6b-987e-4421-a695-ff1f6f5723d8.ZPP
alc.cmp
\data.rgm File will not be overwritten!
*** 2 Runtime files of project "SINGLELINEDIAGRAM" were created in the Runtime folder.
```

This error messages are caused by open line ends in the ALC grid.

This open ends can be connected to ‘Terminators’.

- Draw a combined element

- Close the variable selection dialog with No selection
- In the Automatic Line Coloring properties choose as function type '*Terminator*'
- In the Color properties for the combined element enable *'Transparent'*
- Place the combined element to each open end of your grid.



Properties: Combined element: Combined element_5 - Screen: SingleLineDiagram - Project: SINGLE...

Automatic Line Coloring

Function typ... Terminator

Source: UNDEFINED [0]

Source for r... UNDEFINED [0]

□ Switch input/out... Link number: 0

□ Use alias Alias: <N...

Failure location from protection / Load distributi

Reactance v... <no variable linked>

Limit current... Limit fixed current carrying

Limit fixed c... 0.0000 Limit variabl... <nc...

Measured v... <nc... Measuring v... <nc...

Measured v... <no variable linked>

Load: Historical maximum load

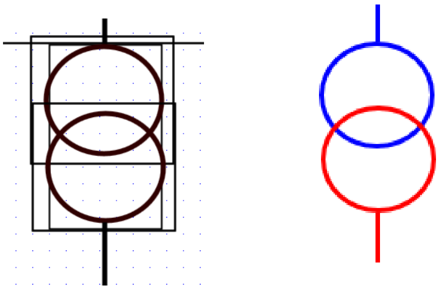
Load fixed [... 0.0000 Load variabl... <nc...

Ground fault recognition

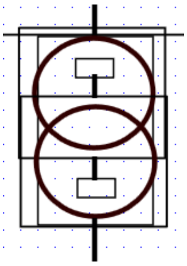
Properties: Combined element: Combined element_5 - Screen: SingleLineDiagram - Pr... Output window

TERMINATOR IN TRANSFORMER

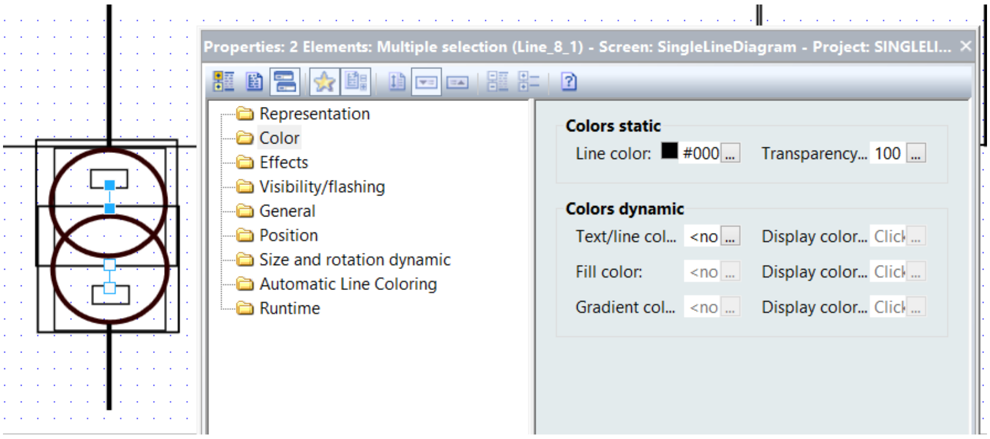
For the transformer we used the switch function type of the ALC to visualize the color from the connected source. These switch has now one open end. That's why the output window still has two error messages.



- Draw a line from each switch and place a terminator on the open end of this line



- Finally set the property color Tranparency to 100% for the two helping lines in the transformer



Properties: 2 Elements: Multiple selection (Line_8_1) - Screen: SingleLineDiagram - Project: SINGLELI...

Colors static

Line color: ■ #000 ... Transparency... 100 ...

Colors dynamic

Text/line col... <no ... Display color... Click ...

Fill color: <no ... Display color... Click ...

Gradient col... <no ... Display color... Click ...

The project should now be compiled without ALC errors.

Our substation automation project is now ready.

You can use this project as a base for your future projects. Also take a look on our provided demoprojects (ask your trainer for a copy of our special energy edition demo project.

The project based on this training material is also ready to get the command sequencer implemented.

We offer this as an enhanced energy training. Ask your trainer for details or write an email to training@copadata.com.