

TeslaSCADA2 IDE User manual

Version of TeslaScada IDE: 2.61

**© 2024 LLC Tesla
LLC Tesla**

1.	About TeslaSCADA IDE	11
1.1	Direct architecture	12
1.2	Client-Server architecture	12
2.	System requirements	13
2.1	Windows	13
2.2	MacOS	14
2.3	Linux	14
2.4	Raspberry PI	14
3.	Installation	14
3.1	Windows	14
3.2	MacOS	19
3.3	Linux	23
3.4	Raspberry PI	26
4.	Databases	28
4.1	SQLite	29
4.2	MySQL	31
4.2.1	MySQL Workbench	47
4.3	MSSQL	55
4.4	PostgreSQL	58
5.	Start TeslaSCADA IDE	59
5.1	Main menu	61
5.1.1	File	62
5.1.2	Edit	63
5.1.3	Arrange	64
5.1.3.1	Align	66
5.1.4	Project	67
5.1.4.1	New server	68
5.1.4.2	New Database	69
5.2	Toolbar	69
5.3	Project window	72
5.3.1	Screens	73
5.3.2	Scripts	76
5.3.3	Servers	79
5.3.4	Tags	81
5.3.5	Users	86

5.3.6	Databases	88
5.3.7	Reports	89
5.4	Canvas	91
5.5	Property Sheet	91
5.6	Screen window	92
5.7	Status bar	94
5.8	Hot keys and tips	94
6.	Project	98
6.1	Project properties	100
6.1.1	General tab	101
6.1.2	Screens tab	106
6.1.3	Events/History tab	107
6.1.3.1	E-mail client	111
6.1.3.2	Telegram bot	113
6.1.3.2.1	Create Telegram Bot	115
6.1.3.3	Push notifications	118
6.1.3.4	GSM-modem	120
6.1.4	OPC UA tab	122
6.1.5	MQTT Publisher tab	125
6.1.6	Web-server tab	127
6.1.7	HTTP-server	129
6.1.8	Redundant server	131
6.1.9	Cloud	132
6.2	Screens	133
6.2.1	Screen properties	135
6.2.2	Designing screen	136
6.2.3	Graphical objects	143
6.2.3.1	Simple objects library	149
6.2.3.1.1	Line	150
6.2.3.1.2	Rectangle	151
6.2.3.1.3	Ellipse	152
6.2.3.1.4	Polyline	153
6.2.3.1.5	Polygon	156
6.2.3.1.6	Sector	158
6.2.3.1.7	Text/EditField	159
6.2.3.1.8	Border	161
6.2.3.1.9	Image	162
6.2.3.1.10	Scale	163
6.2.3.1.11	Text Area	165
6.2.3.2	3D Objects library	166
6.2.3.2.1	Sphere	167

6.2.3.2.2	Cylinder	168
6.2.3.2.3	Cone	169
6.2.3.2.4	Sector 3D	170
6.2.3.2.5	Polygon 3D	171
6.2.3.2.6	Tank	173
6.2.3.2.7	Border 3D	174
6.2.3.2.8	Text/EditField 3D	175
6.2.3.2.9	Value with history and event	177
6.2.3.2.9.1	Grid	179
6.2.3.3	Buttons and Switches library	180
6.2.3.3.1	Button	181
6.2.3.3.2	Image button	183
6.2.3.3.3	Switch	184
6.2.3.3.4	Apple switch	185
6.2.3.3.5	Three position switch	186
6.2.3.4	Lights/Indicators library	187
6.2.3.4.1	Light	188
6.2.3.4.2	Indicator	189
6.2.3.5	Pipes library	190
6.2.3.5.1	Pipe	191
6.2.3.5.2	Pipeline	192
6.2.3.6	Valves library	194
6.2.3.6.1	Valve	194
6.2.3.6.2	Ball valve	196
6.2.3.6.3	Position valve	197
6.2.3.7	Pumps and Motors library	198
6.2.3.7.1	Pump	198
6.2.3.8	Fans library	200
6.2.3.8.1	Fan	200
6.2.3.9	Tanks library	202
6.2.3.9.1	Vertical tank	202
6.2.3.10	Conveyers library	204
6.2.3.10.1	Belt conveyer	204
6.2.3.10.2	Screw motion conveyer	205
6.2.3.11	Analog meters library	207
6.2.3.11.1	Analog meter	208
6.2.3.11.2	Range indicator	209
6.2.3.11.3	Other analog meters	210
6.2.3.12	Digital meters library	211
6.2.3.12.1	Digital meter	211
6.2.3.13	Controls library	213
6.2.3.13.1	Slider and Apple slider	214
6.2.3.13.2	Slider vertical and horizontal	215
6.2.3.13.3	Counter and Counter rectangle	216
6.2.3.13.4	Selector and Combo box	217
6.2.3.13.5	Menu box	218

6.2.3.13.6	Check box and Check list	219
6.2.3.13.7	Menu check list	221
6.2.3.13.8	Parameter list	223
6.2.3.14	Electrical library	225
6.2.3.14.1	Electrical switch	225
6.2.3.14.2	Transformer	227
6.2.3.15	Trends and Charts library	228
6.2.3.15.1	Trend	228
6.2.3.15.1.1	Grid	231
6.2.3.15.2	Pie chart	232
6.2.3.15.3	Bar chart	234
6.2.3.15.3.1	Grid	235
6.2.3.15.4	Wind chart	236
6.2.3.16	Events library	238
6.2.3.16.1	Events log	238
6.2.3.16.1.1	Columns	242
6.2.3.16.2	Events ticker	243
6.2.3.17	Recipes library	245
6.2.3.17.1	Recipe selector	246
6.2.3.17.2	Recipe table	247
6.2.3.17.3	Parameter table	248
6.2.3.17.4	Schedule table	250
6.2.3.18	History DB library	252
6.2.3.18.1	History DB table	252
6.2.3.18.2	History DB trend	255
6.2.3.18.2.1	Grid	257
6.2.3.18.3	XY Trend	258
6.2.3.18.3.1	Grid	261
6.2.3.18.4	History Excel report and History Max and Min report	263
6.2.3.19	Odoo ERP	264
6.2.3.19.1	Odoo ERP table	265
6.2.3.20	Databases library	268
6.2.3.20.1	Database table	269
6.2.3.21	Widgets library	271
6.2.3.21.1	WebView	272
6.2.3.21.2	Video	273
6.2.3.21.3	Analog clock	275
6.2.3.21.4	Digital clock	276
6.2.3.21.5	Date and time	277
6.2.3.21.6	Color picker and Color rectangle	278
6.2.3.21.7	Date and time selector	279
6.2.3.22	Tiles	280
6.2.3.22.1	Percentage	281
6.2.3.22.2	DigitalClockTile	283
6.2.3.22.3	GaugeTile	285
6.2.3.22.4	Sparkline	287

6.2.3.22.5	Smoothed chart	290
6.2.3.22.6	Analog clock	294
6.2.3.22.7	Number	295
6.2.3.22.8	HighLow	297
6.2.3.22.9	Counter	299
6.2.3.22.10	Slider	301
6.2.3.22.11	Switch	303
6.2.3.22.12	Button	305
6.2.3.22.13	Time	307
6.2.3.22.14	Bar chart	309
6.2.3.22.15	Radial chart	312
6.2.3.22.16	Donut chart	315
6.2.3.22.17	Circular progress	318
6.2.3.22.18	Stock	320
6.2.3.22.19	Gauge spark line	322
6.2.3.22.20	Bar gauge	324
6.2.3.22.21	Led	326
6.2.3.22.22	Color	328
6.2.3.22.23	Fluid	330
6.2.3.23	SVG objects library	332
6.2.4	User-defined library	333
6.2.4.1	Example: How to create new graphical object	335
6.2.5	Properties	344
6.2.5.1	Flash	345
6.2.5.2	Rotation	347
6.2.5.3	Motion	348
6.2.5.4	Visibility	349
6.2.5.5	Line color	350
6.2.5.6	Fill color	352
6.2.5.7	Filling	354
6.2.5.8	Text color	355
6.2.5.9	Control (for buttons)	357
6.2.5.10	Text input	359
6.2.5.11	Output value	362
6.2.5.12	Indicator	362
6.2.5.13	Rotation indicator	363
6.2.5.14	Image	364
6.2.5.15	Color	366
6.2.5.16	Control (for sliders)	367
6.2.5.17	Control (for counters)	368
6.2.5.18	Value (for meters)	369
6.2.5.19	Value (for range indicators and gauges)	370
6.2.5.20	Switch control	372
6.2.5.21	Switch control (for 3 position switch)	373
6.2.5.22	Selector	374
6.2.5.23	Row number	376
6.3	Servers	376

6.3.1	Modbus RTU	377
6.3.2	Modbus TCP	379
6.3.3	Siemens	381
6.3.4	Allen Bradley	382
6.3.5	OPC UA	383
6.3.6	MQTT	385
6.3.7	Omron	386
6.3.8	BACnet/IP	388
6.3.9	Common RTU Server	389
6.3.10	Common TCP Server	390
6.3.11	Raspberry GPIO	391
6.3.12	HTTP-server	391
6.3.13	Cloud	392
6.4	Scripts	393
6.4.1	Script properties	395
6.4.2	FBD language	396
6.4.2.1	Script objects of FBD language	400
6.4.2.1.1	Input/Output library	401
6.4.2.1.2	Logical library	402
6.4.2.1.3	Bitmap operations library	403
6.4.2.1.4	Arithmetic library	404
6.4.2.1.5	Compare library	405
6.4.2.1.6	Select library	406
6.4.2.1.7	Arrays library	407
6.4.2.1.8	Triggers/Counters library	408
6.4.2.1.9	Trigonometric library	409
6.4.2.1.10	Hex operations library	410
6.4.2.1.11	Call screen library	411
6.4.2.1.12	Strings library	412
6.4.2.1.13	Date and time library	413
6.4.2.1.14	Servers library	414
6.4.2.1.15	Recipes library	415
6.4.2.1.16	Base64 library	415
6.4.3	ST language	416
6.4.3.1	What is Structured Text Programming?	416
6.4.3.2	Starting with the Syntax of Structured Text	417
6.4.3.3	Making Statements with Structured Text	418
6.4.3.4	Types in Structured Text	418
6.4.3.5	Operators and Expressions in STL	419
6.4.3.5.1	Operators	419
6.4.3.5.1.1	Arithmetic Operators	420
6.4.3.5.1.2	Relational Operators	420
6.4.3.5.1.3	Logical Operators	421
6.4.3.5.1.4	Bitwise Operators	421

6.4.3.5.2	Operators and Statements	422
6.4.3.5.2.1	Assignment Statement and Operator	422
6.4.3.5.2.2	Conditional Statements	422
6.4.3.5.3	Boolean and Numeric Expressions	423
6.4.3.5.4	Iteration with Repeating Loops	424
6.4.3.5.4.1	FOR Loops	425
6.4.3.5.4.2	While Loops	425
6.4.3.6	User-defined functions	426
6.4.3.7	Using Tags in Structured Text	426
6.4.3.8	Using Object property fields in Structured Text	427
6.4.3.9	Using Server parameter fields in Structured Text	428
6.4.3.10	Using User parameter fields in Structured Text	429
6.4.3.11	Embedded functions	429
6.4.3.11.1	Print library	430
6.4.3.11.2	Arithmetic library	430
6.4.3.11.3	Bitmap operations library	430
6.4.3.11.4	Select library	432
6.4.3.11.5	Trigonometric library	432
6.4.3.11.6	Strings library	432
6.4.3.11.7	Hex operations library	433
6.4.3.11.8	Base64 library	433
6.4.3.11.9	Date and time library	434
6.4.3.11.10	Server library	435
6.4.3.11.11	Recipes library	436
6.4.3.11.12	E-mail library	437
6.4.3.11.13	Odoo ERP library	438
6.4.3.11.14	Excel and screenshot library	439
6.4.3.11.15	Database library	441
6.4.3.11.16	HTTP library	446
6.4.3.11.17	Global arguments library	447
6.4.3.11.18	Tag properties library	448
6.4.3.11.19	Dialog box library	450
6.4.3.11.20	Trend's curve library	451
6.4.3.11.21	Screen library	451
6.4.3.11.22	Files library	452
6.4.3.11.23	Report library	457
6.4.3.11.24	Common RTU and TCP library	458
6.4.3.11.25	Call external software	459
6.4.3.11.26	User library	460
6.4.3.11.27	Push library	460
6.5	Tags	461
6.5.1	General tab	462
6.5.1.1	Modbus tag settings	465
6.5.1.2	Siemens tag settings	466
6.5.1.3	Allen Bradley tag settings	467

6.5.1.3.1	Micrologix tag settings	467
6.5.1.4	OPC UA tag settings	468
6.5.1.5	MQTT tag settings	469
6.5.1.6	Omron tag settings	470
6.5.1.7	BACnet tag settings	470
6.5.1.8	Raspberry GPIO tag settings	471
6.5.2	Scaling tab	472
6.5.3	Alarms tab	473
6.5.4	History tab	474
6.5.5	Script tab	477
6.5.6	Cloud	478
6.6	Users	480
6.7	Databases	483
6.7.1	Recipe	483
6.7.2	History DB	485
6.7.3	Odoo ERP	488
6.8	Reports	489
6.8.1	Report properties	491
6.8.2	Design report	493
6.8.3	Other report objects	496
6.8.3.1	Common report library	498
6.8.3.1.1	Label	499
6.8.3.1.2	Tag.PV	499
6.8.3.1.3	Two Tag.PV values	500
6.8.3.1.4	Date and time	501
6.8.3.1.5	Two DateTime values	502
6.8.3.1.6	Gap	503
6.8.3.1.7	Variable	503
6.8.3.1.8	Page number	504
6.8.3.1.9	Image	505
6.8.3.1.10	Object image	505
6.8.3.2	Container library	506
6.8.3.3	Chart library	506
6.8.3.3.1	Time chart	507
6.8.3.3.2	XY chart	508
6.8.4	Table report objects	509
6.8.4.1	General history table	511
6.8.4.2	General events table	515
6.8.4.2.1	Columns	517
6.8.4.3	History database table	518
6.8.5	Reports from trend's and event's dialog boxes	521
6.9	Simulation	523

7. Load on Device

525

8.	Import for iOS	527
9.	Examples	530
9.1	Change the color of an object	530
9.1.1	Simple color change	530
9.1.2	Simple multiple color change	531
9.1.3	Simple multiple color change with scripts	533
9.1.4	Complex color change	535
9.1.5	Complex color change with scripts	539
9.2	Object flashing	544
9.2.1	Simple flashing	544
9.2.2	Simple multiple flashing	546
9.2.3	Complex flashing with scripts	547
9.3	Object visibility	552
9.3.1	Simple visibility	553
9.3.2	Complex visibility with scripts	554
9.4	Change the text of an object	560
9.4.1	Simple text change	561
9.4.2	Simple multiple text change	562
9.4.3	Display tag's value	564
9.4.4	Enter tag's value	566
9.4.5	Complex text change with scripts	568
9.5	Call popup	574
9.5.1	Complex call popup with scripts	574
9.6	HTTP requests	581
9.6.1	Weather from weatherstack.com	581
9.7	Trends	584
9.7.1	Simple trend example	585
9.7.2	Trend example with Y axis change	587
9.7.3	Add and remove curve to/from trend dynamically	590
9.8	Change tag's value	593
9.8.1	Change values of 2 tags by one click	594
9.8.2	Write value when screen is opened and closed	595
9.9	IOT clouds	597
9.9.1	IBM Watson IOT	598
9.9.2	Yandex cloud	615

1 About TeslaSCADA IDE

TeslaSCADA IDE is an integrated development environment used for configuring, developing and managing HMI/SCADA applications. In this manual you will find everything you need to create a full-featured SCADA (Supervisory Control and Data Acquisition) project visualization. With this tool you can create and manage TeslaSCADA projects, configure connections with devices, enter tags, alarms, and trends.

A simple to use interface allows for easy manipulation of the project's configuration and data processing. The project data are stored in a single file (based on xml) for easy backup and restoration.

TeslaSCADA IDE has an integrated GUI (Graphical User Interface) visualization editor for easy creation of professionally looking graphics.

Main features of TeslaSCADA projects

- Use on MacOS, Windows, Linux, Android and iOS.
- Supports many industrial protocols - Modbus RTU and TCP(UDP), Siemens ISO/TCP, Ethernet/IP, Omron FINS/TCP(UDP) devices, OPC UA and MQTT servers.
- Lots of graphical objects for developing screens.
- Supports user-defined images in *.png, *.jpg and *.gif format.
- Supports creating group objects.
- Supports scripts based on FBD and ST language.
- Supports events. Use SQL Lite or MySQL databases to store tag's event information.
- Supports event notifications by E-mail, Telegram messenger and third part HTTP services.
- Supports history. Use SQL Lite or MySQL databases to store tag's history information.
- Configure user permissions.
- Web-server.
- Report system in Excel.
- Direct printing reports.
- Supports Import/Export screens, tags (including excel format), scripts.
- Supports touch panel.
- Supports sound notification on alerts.

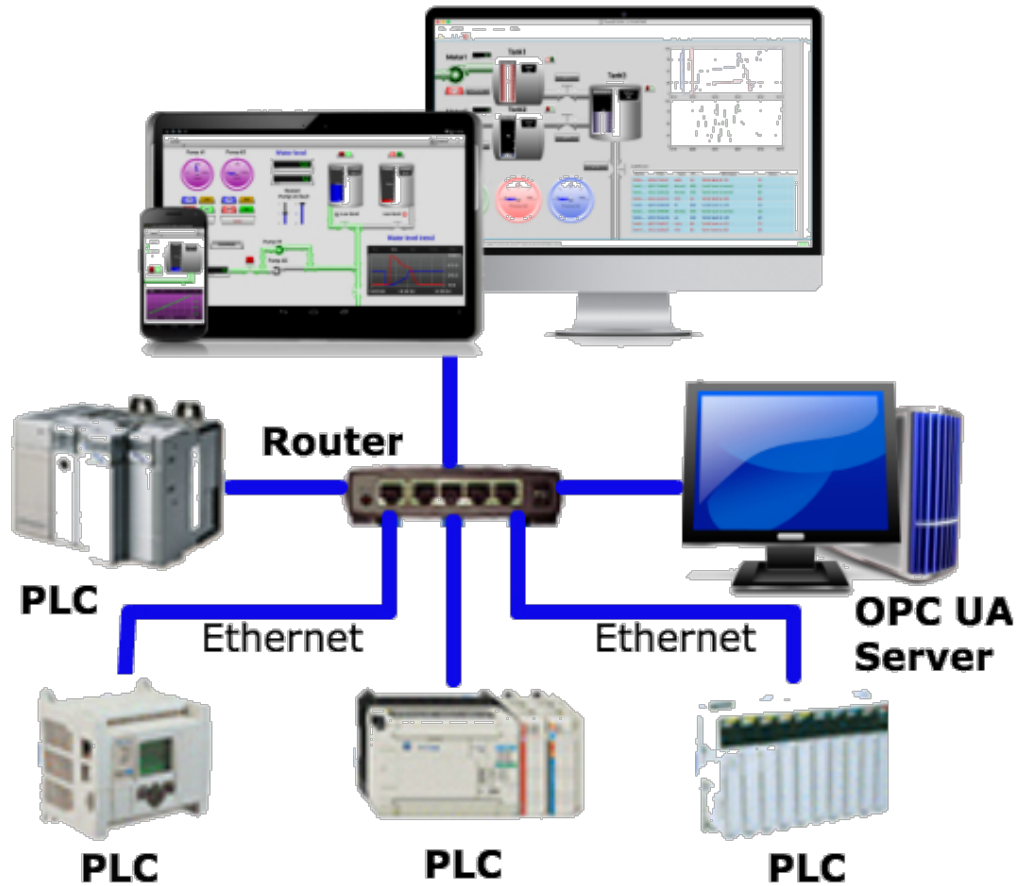
Also information about TeslaSCADA2 you can find on our site:
<https://teslascada.com/products/teslascada2>

There are 2 possible ways to use TeslaSCADA:

- [Direct architecture](#)^[12]
- [Client-server architecture](#)^[12]

1.1 Direct architecture

In the simplest process control system based on TeslaSCADA, every device (computer or mobile) is, in fact, a server, since it reads and writes tag values from/to devices, works with a database, etc.

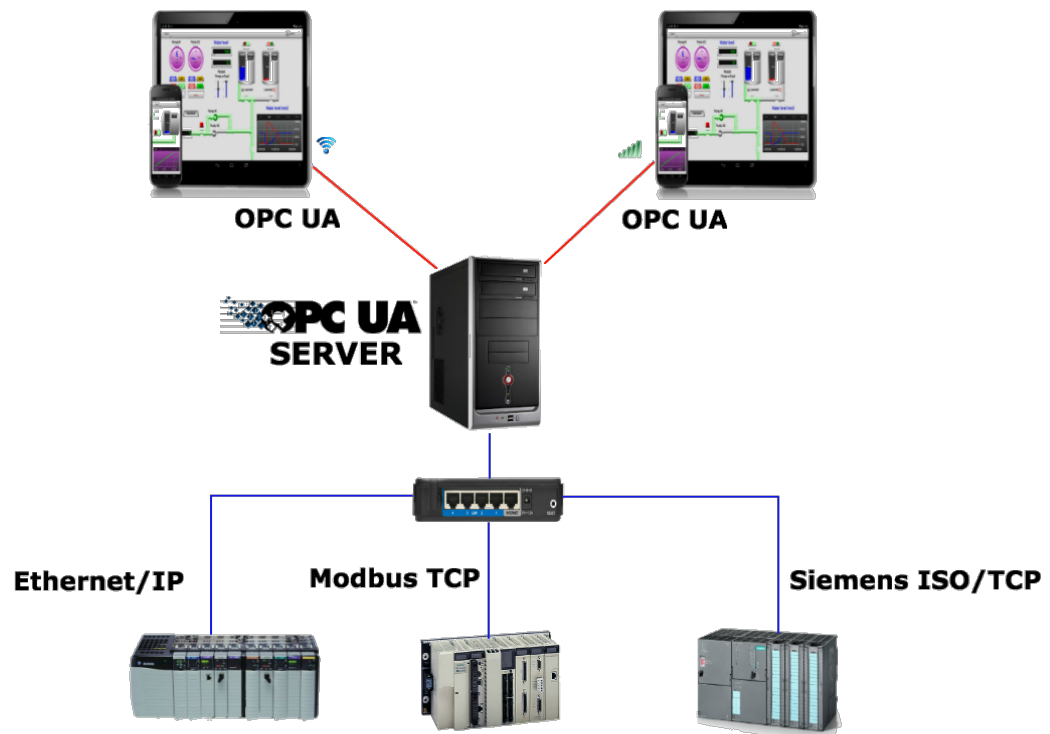


The advantage of this architecture is that there is no need to use some intermediate device for the server. All connections are made directly to industrial devices and servers. A PC or mobile device can be used as an HMI.

The disadvantages of such a system are that with a large number of devices (PCs and mobiles) with TeslaSCADA, there is a load on the controller, the exchange of data when communicating with the controllers is via an unencrypted channel, and all the necessary calculations in scripts are carried out on each device.

1.2 Client-Server architecture

If your process control system is large and you want to use many devices (PC or mobile) with TeslaSCADA for display and control You can use Client-Server architecture:



A built-in (or third-party) OPC UA server or built-in HTTP-server or a third-party MQTT broker can be used as a server.

The advantage of this architecture is to reduce the load on the controller when using a large number of devices with TeslaSCADA, encryption when exchanging data with the server (especially important for mobile devices used remotely) and the ability to perform all calculations on the server.

The disadvantage of this architecture is the need to use an intermediate device with an installed OPC UA server, HTTP-server or MQTT broker.

2 System requirements

TeslaSCADA IDE requires Windows, Mac OS or Linux operating systems.

2.1 Windows

Processors: Intel Pentium 4, Intel Centrino, Intel Xeon, or Intel Core Duo (or compatible) 1.8 GHz minimum.

Operating systems: Windows 10, Windows 8 (Modern UI (i.e. Metro Mode) is not supported), Windows 7, Windows Vista, Windows XP (not recommended but supported).

Memory: 1 GB (2 GB recommended).

Disk Space: 2 GB of free disk space (4 GB of free disk space).

2.2 MacOS

Processors: Dual-Core Intel, PowerPC G5
Operating systems: 10.7.3 or greater
Memory: 1 GB (2 GB recommended).
Disk Space: 2 GB of free disk space (4 GB of free disk space).

2.3 Linux

Processors: Intel Pentium 4, Intel Centrino, Intel Xeon, or Intel Core Duo (or compatible) 1.8 GHz minimum.
Operating systems: Ubuntu 10.4 + gtk2 2.18+ Memory: 1 GB (2 GB recommended).
Disk Space: 2 GB of free disk space (4 GB of free disk space).
Media: You must install the following in order to support AAC audio, MP3 audio, H.264 video, and HTTP Live Streaming:
libavcodec52 and libavformat52 on Ubuntu Linux 10.04, 10.10, 11.04 or equivalent.
libavcodec53 and libavformat53 on Ubuntu Linux 11.10, 12.04 or equivalent.

Important! We've tested Linux version only on Ubuntu 14, Ubuntu 20, PEД OC and Astra Linux (Orel) OS. Unfortunately we didn't test it on other Linux OS.

2.4 Raspberry PI

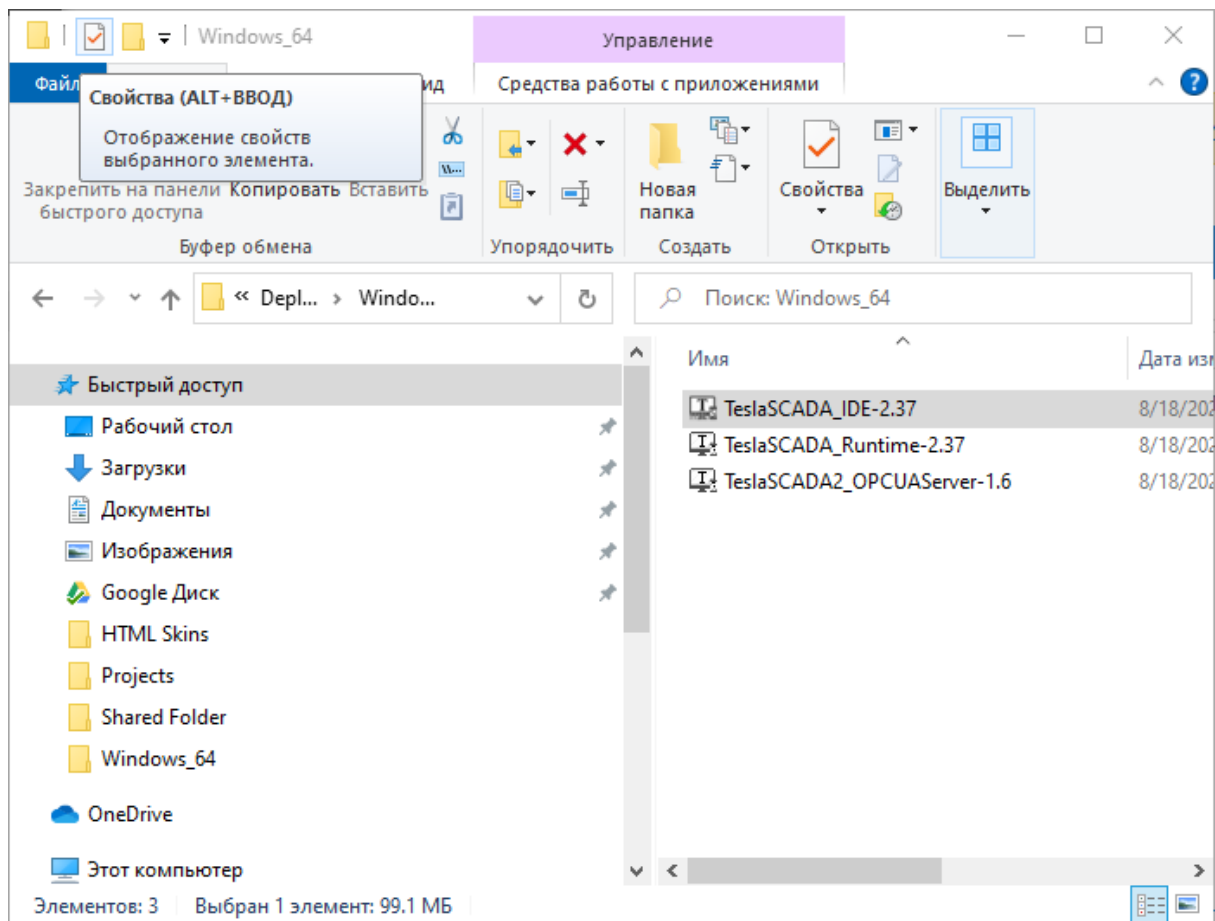
Processors: processors on Raspberry PI 3,4
Operating systems: Raspbian
Memory: 1 GB (2 GB recommended).
Disc Space: 2 GB of free disc space (4 GB of free disc space).

3 Installation

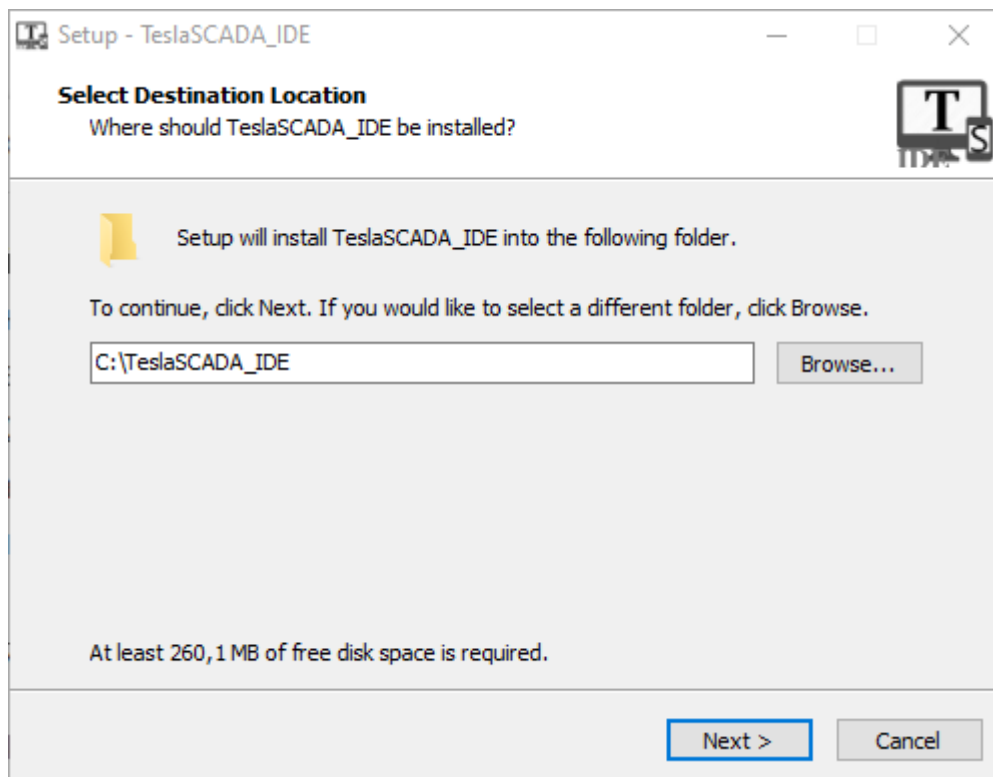
Installation depends on operating system.

3.1 Windows

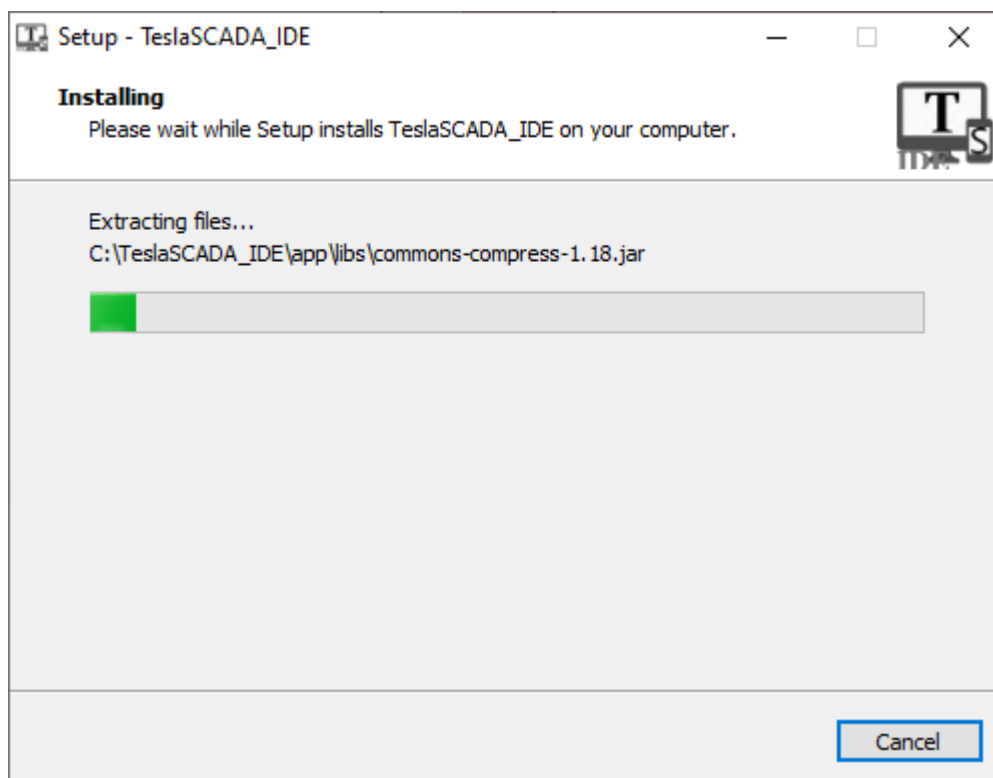
To install TeslaSCADA IDE download EXE package for your operating system, then you need to run the installation file:



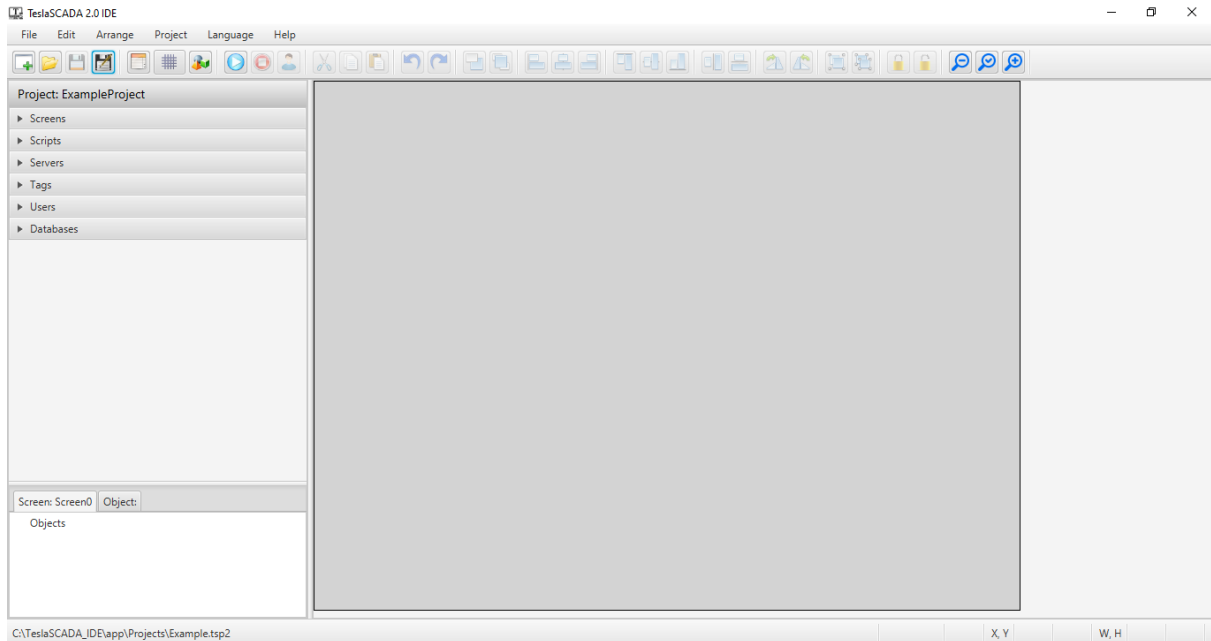
Then the window for selecting directories for the main program files and user data will be displayed. It is recommended to install the "system" part of TeslaSCADA IDE to the system drive in the "C:\TeslaSCADA_IDE\" folder, and the folder with user files can be selected at the user's discretion. The main thing is that the OS allows the creation, modification and deletion of files in this folder without requiring administrator rights. Also if you want to use Web Server the path shouldn't contain white spaces. After selecting the directories, click "Next":



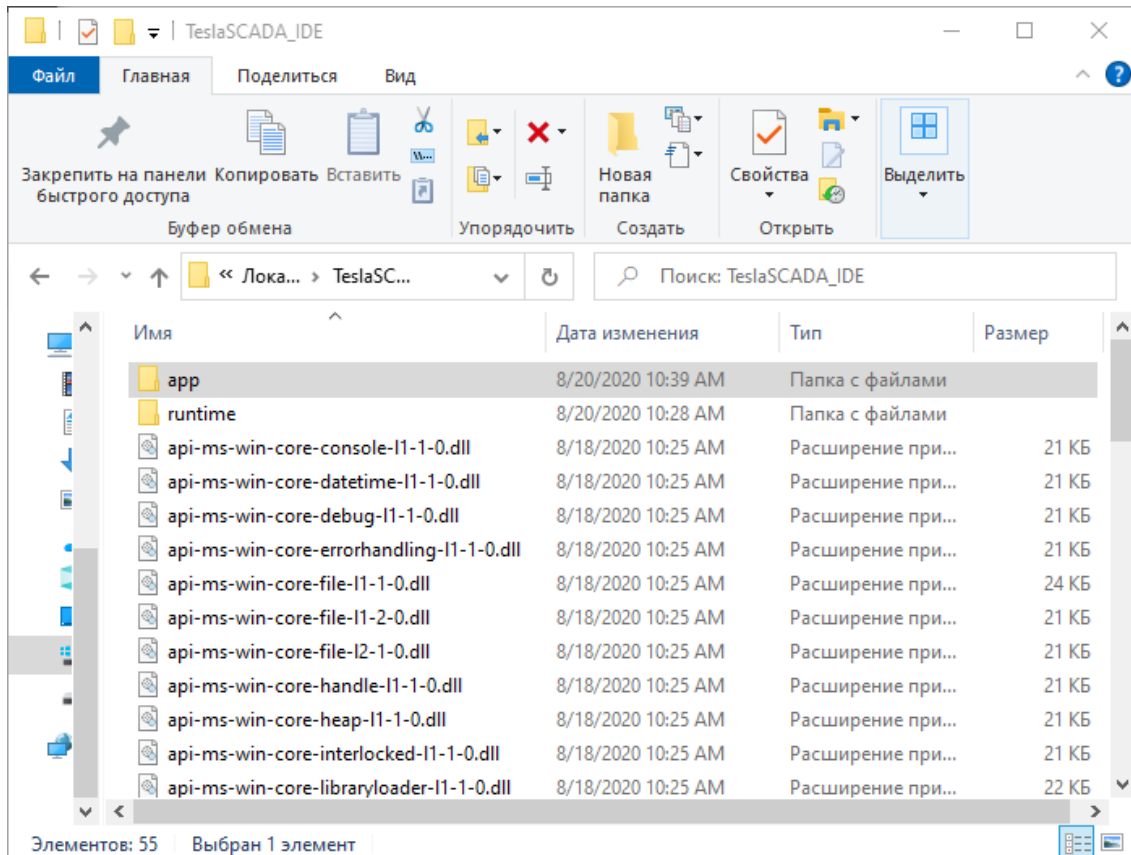
After clicking "Next" application will be installed:



After the installation is complete, TeslaSCADA IDE will be started automatically:



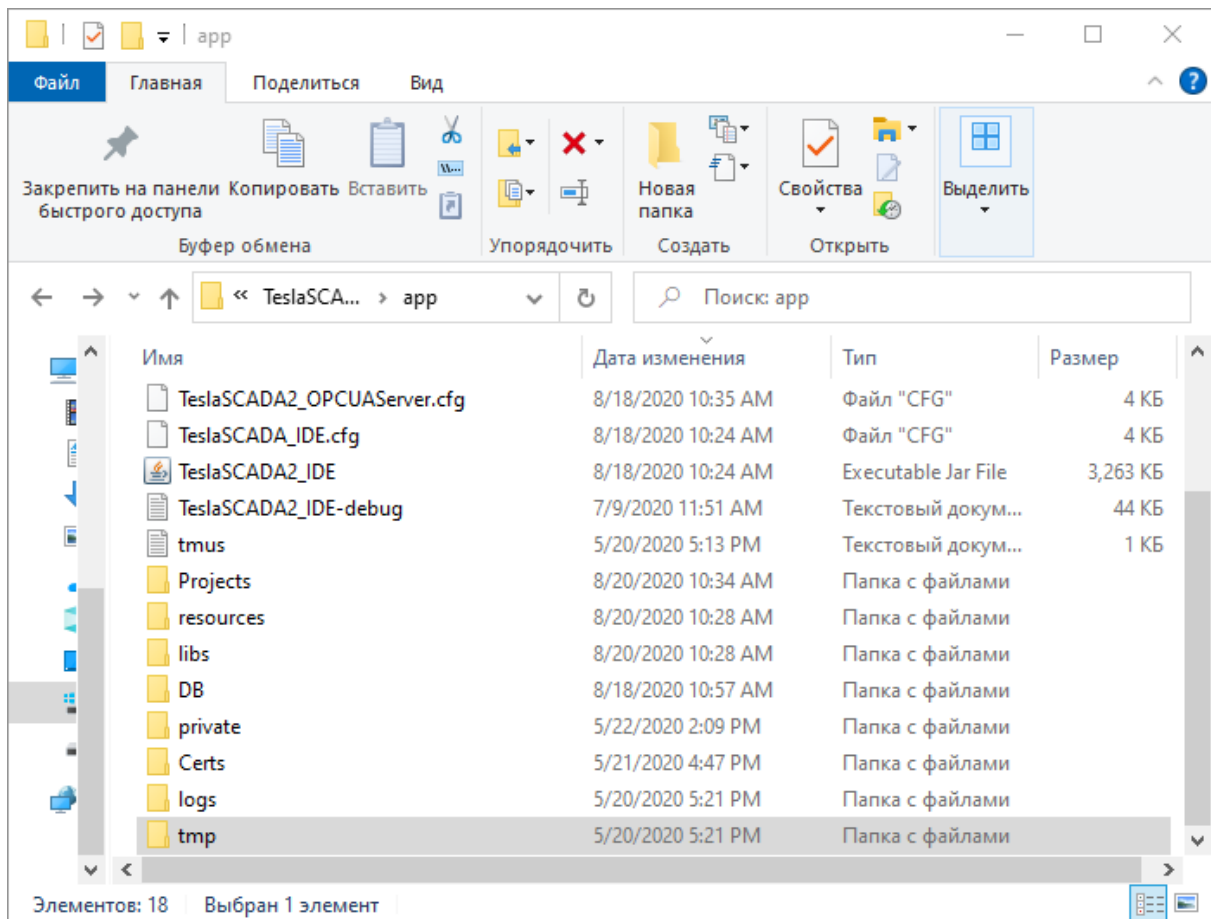
To study folder where you install TeslaSCADA IDE open it:



Consider its contents:

- **app** - contains application information.
- **runtime** - contains JRE. TeslaSCADA based on Java language. The folder runtime contains JRE for Windows environment. If you don't use Web server in your project you don't need to install Java separately. TeslaSCADA IDE will work any way. If you want to use Web server in your project you have to install Java on your PC.

Let's study app folder:



Consider its contents:

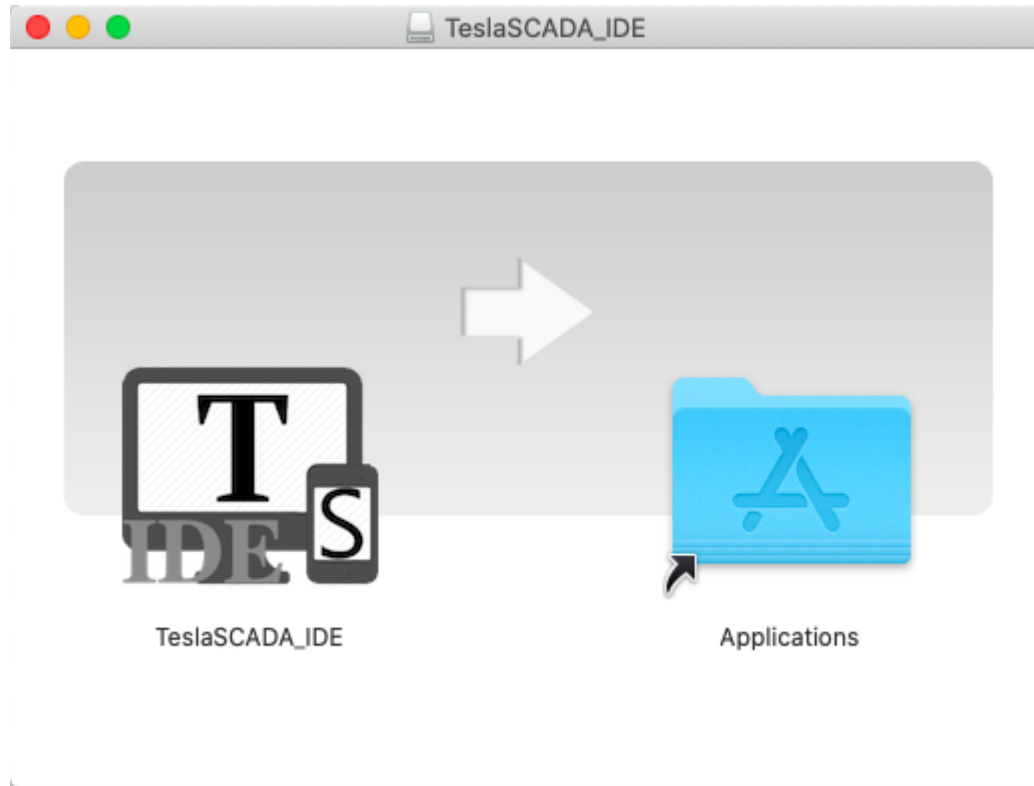
- **Projects** - default project folder of TeslaSCADA IDE. You can save projects in other folders.
- **DB** - project contains SQL Lite databases. If you use SQL Lite databases for history, events and recipes they will be stored in this folder.
- **private** - contains certificates and keys for OPC UA and MQTT protocols if you use OPC UA or MQTT clients in your project.
- **Certs** - contains certificates and keys for OPC UA server if you use it.
- **TeslaSCADA_IDE-debug** - contains Log information about application working.
- **Other folders and files** - related to working of application and Web server.



Install TeslaSCADA2 on Windows

3.2 MacOS

To install TeslaSCADA IDE download DMG package for your operating system. DMG package provides a simple possibility to install application by double clicking on it:



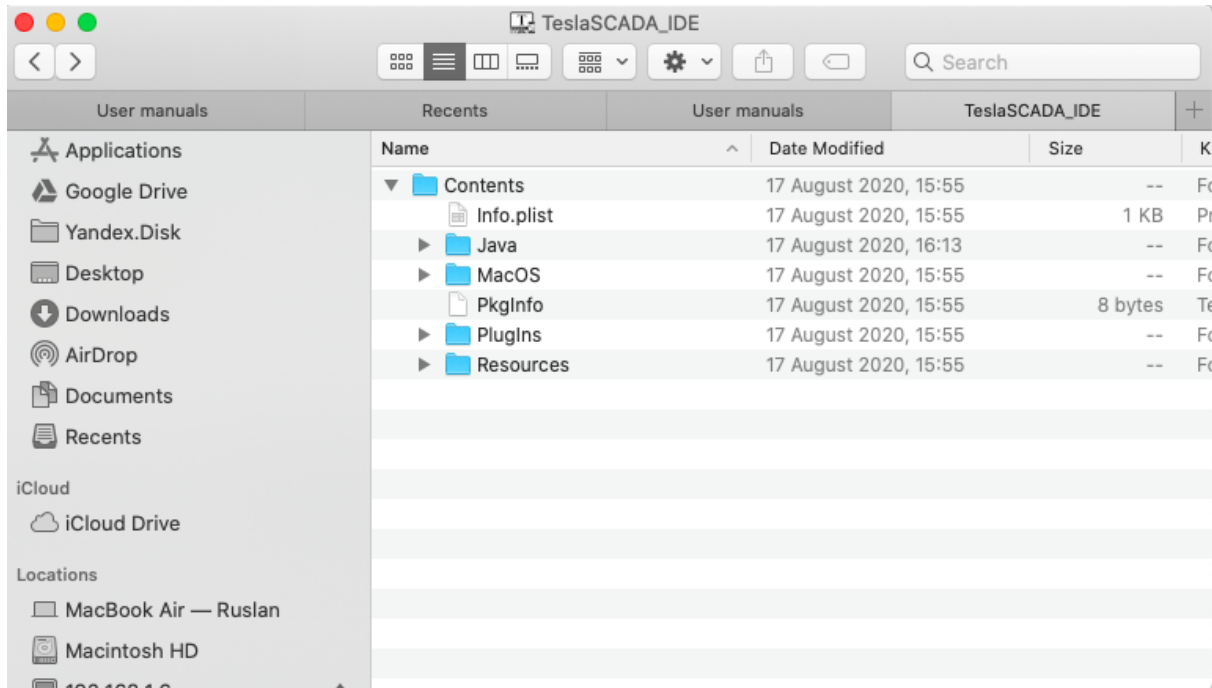
By using left mouse button of the mouse drag and drop TeslaSCADA IDE in Applications. Now you can open it in Applications.

Important! Sometimes you've got error message: "TeslaSCADA_IDE.app is damaged and can't be opened. You should move it to the Trash." Like in the picture below:



To solve this problem you should open Terminal and execute the command below:
sudo xattr -rd com.apple.quarantine /Applications/TeslaSCADA_IDE.app

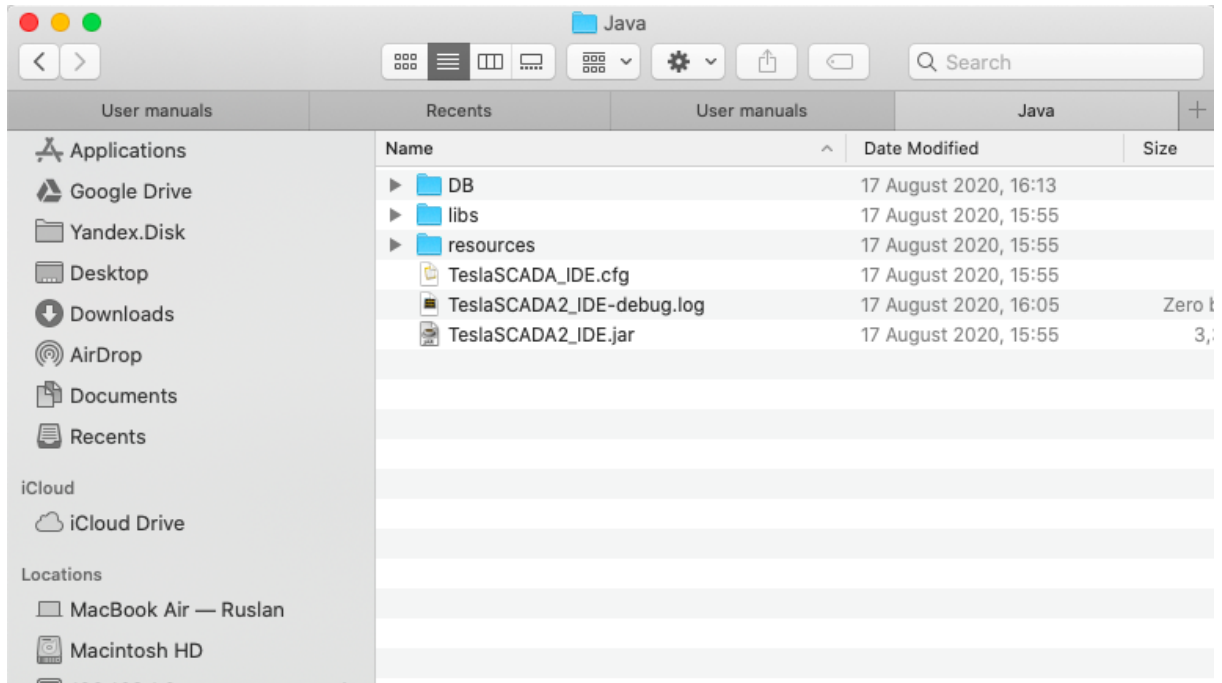
To study folder in Applications click by right mouse button on TeslaSCADA_IDE and choose Show Package Contents. You'll see:



Consider its contents:

- **Java** - contains application information.
- **Plugins** - contains JRE. TeslaSCADA based on Java language. The folder runtime contains JRE for MacOS environment. If you don't use Web server in your project you no need to install Java separately. TeslaSCADA IDE will work any way. If you want to use Web server in your project you have to install Java on your PC.
- **MacOS and Resources** - related to working application.

Let's study Java folder:



Consider its contents:

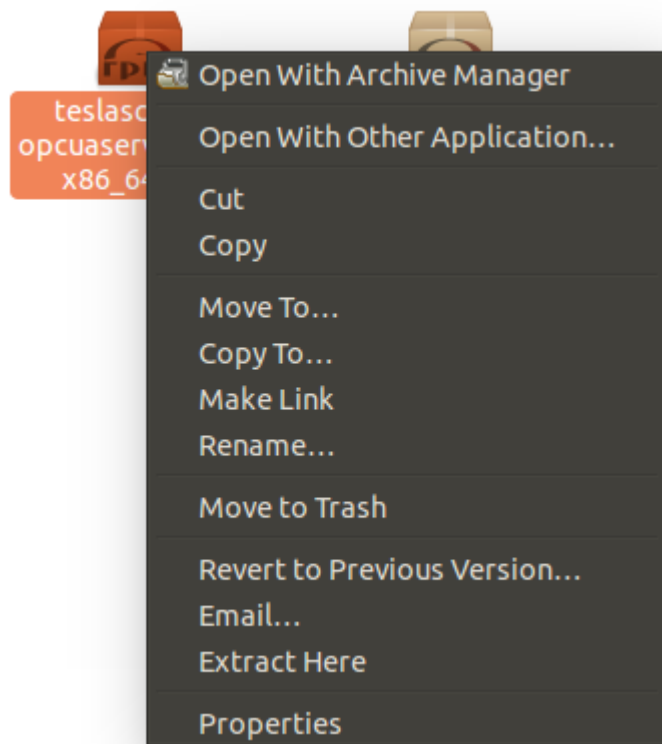
- **Projects** - default project folder of TeslaSCADA IDE. You can save projects in other folders (it's not shown in this picture).
- **DB** - project contains SQL Lite databases. If you use SQL Lite databases for history, events and recipes they will be stored in this folder.
- **private** - contains certificates and keys for OPC UA and MQTT protocols if you use OPC UA or MQTT clients in your project (it's not shown in this picture).
- **Certs** - contains certificates and keys for OPC UA server if you use it (it's not shown in this picture).
- **TeslaSCADA_IDE-debug** - contains Log information about application working.
- **Other folders and files** - related to working of application and Web server.



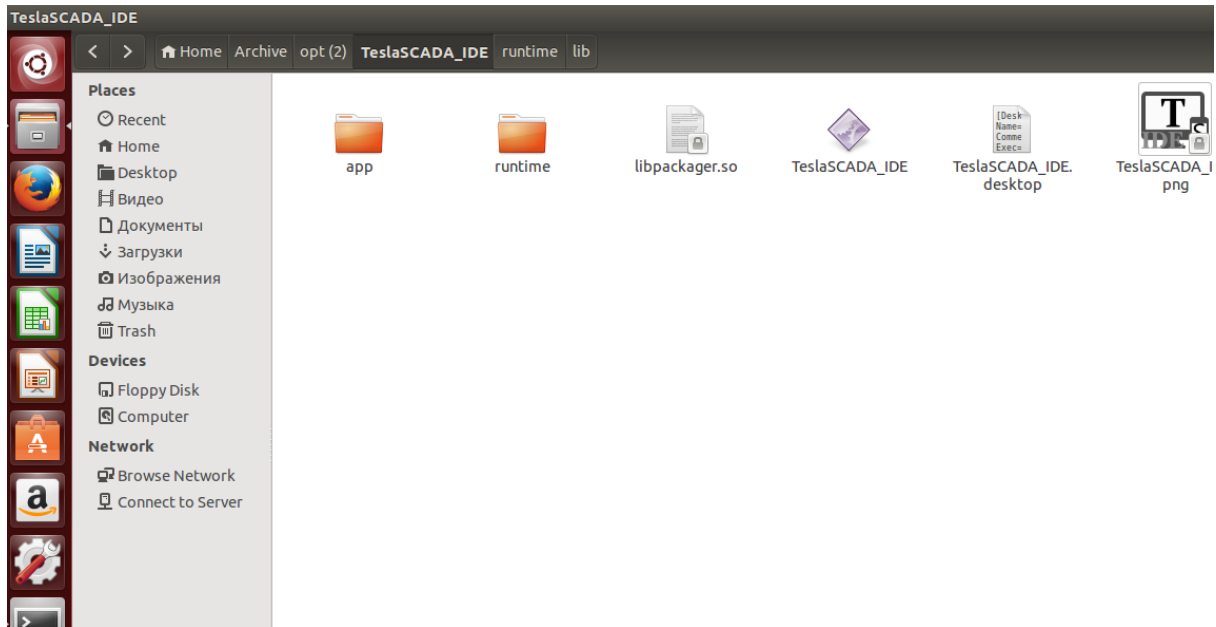
Installation TeslaSCADA2 on MacOS

3.3 Linux

To install TeslaSCADA IDE download RPM package for your operating system. Right click mouse button on RPM package and choose Extract Here:



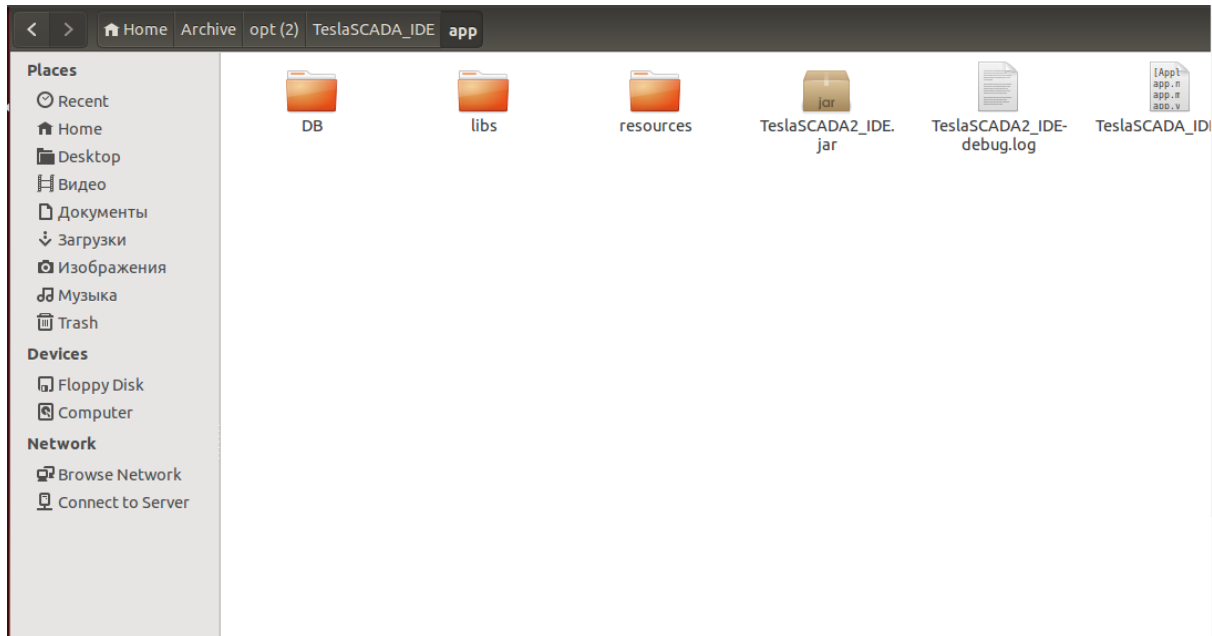
By default RPM package will install the application to /opt, add a shortcut to the application menu. RPM package does not have any UI for installation (normal behavior for Linux). Let's study opt folder. In this folder you can find TeslaSCADA_IDE folder. When you open it:



Consider its contents:

- **app** - contains application information.
- **runtime** - contains JRE. TeslaSCADA based on Java language. The folder runtime contains JRE for Linux environment. If you don't use Web server in your project you don't need to install Java separately. TeslaSCADA IDE will work any way. If you want to use Web server in your project you have to install Java on your PC.

Let's study app folder:



Consider its contents:

- **Projects** - default project folder of TeslaSCADA IDE. You can save projects in other folders (it's not shown in this picture).
- **DB** - project contains SQL Lite databases. If you use SQL Lite databases for history, events and recipes there will be stored in this folder.
- **private** - contains certificates and keys for OPC UA and MQTT protocols if you use OPC UA or MQTT clients in your project (it's not shown in this picture).
- **Certs** - contains certificates and keys for OPC UA server if you use it (it's not shown in this picture).
- **TeslaSCADA2_IDE-debug** - contains Log information about application working.
- **Other folders and files** - related to working of application and Web server.

Important: We've tested Linux version only on Ubuntu 14, Ubuntu 20, РЕД ОС and Astra Linux (Orel) OS. Unfortunately we didn't test it on other Linux OS.



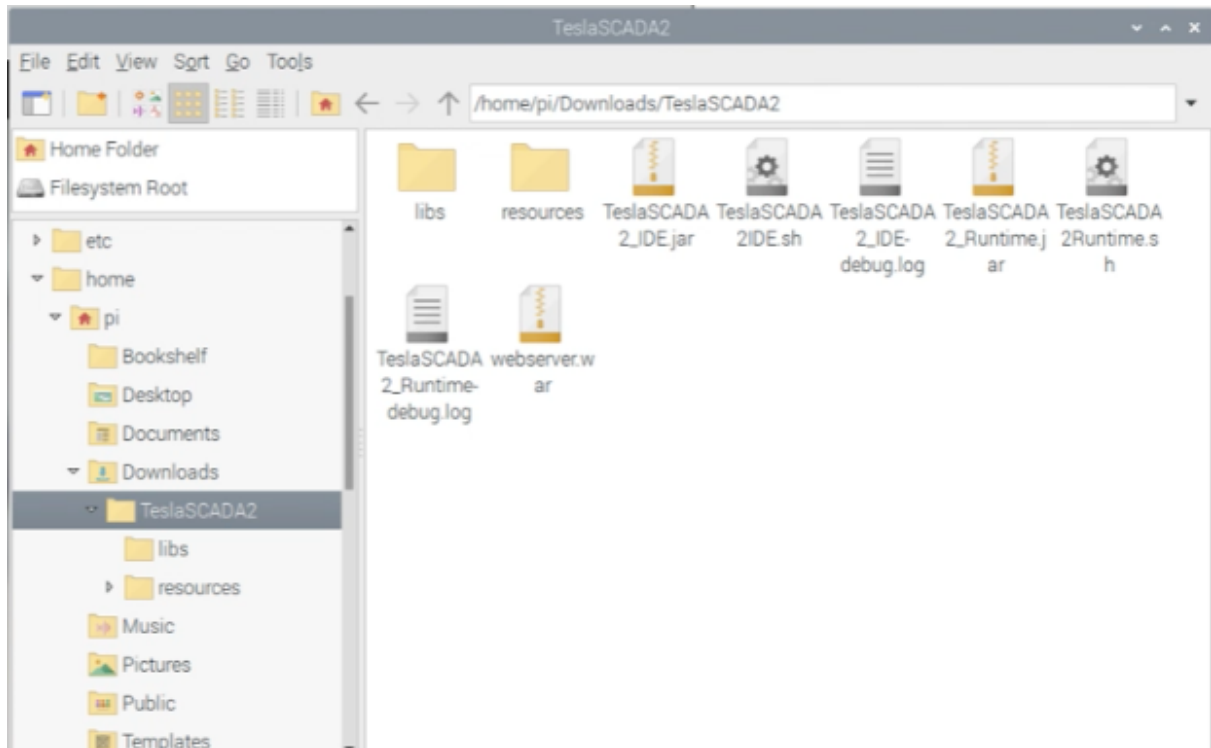
Install TeslaSCADA2 on Linux

3.4 Raspberry PI

Package for Raspberry PI doesn't contain JVM. First you have to install Java 11 with Java FX. We recommend to install Java 11 from [Bell Soft for ARM 32 bit](#). You can do it by downloading installation from the [link](#). Then you can install Java 11 with Java FX libraries by using command:

```
sudo apt-get install ./bellsoft-jdk11.0.11+9-linux-arm32-vfp-hflt-full.deb
```

After that you can download archive from our site and unpack it:



You can start TeslaSCADA2 IDE by double clicking on the script TeslaSCADA2IDE.sh.
You can start TeslaSCADA2 Runtime by double clicking on the script TeslaSCADA2Runtime.sh.



Install TeslaSCADA2 on Raspberry PI

4 Databases

The databases in TeslaSCADA2 are necessary for archiving alarms, operator actions, trends and recipes. When archiving into a database, the subsequent retrieval of data (viewing trends, messages) is much faster, especially over long time intervals. TeslaSCADA2 supports 2 types of databases:

- [SQLite](#)^[29]
- [MySQL](#)^[31]
- [MSSQL](#)^[55]
- [PostgreSQL](#)^[58]

Event database

The database for collecting events you can setup in **Project properties->Events/History tab**^[107] in Events DB name field. There are several types of events saved in database:

- Tag's events. You can setup them in **Tag properties->Alarms tab**^[473].
- Server events. This information about connection, disconnection and lost connection [servers](#)^[376] in the project.
- [User](#)^[480] login/logout information.
- If you setup in [User settings](#)^[482] it's possible to save user operation.

You can show all events by [Events log](#)^[238] graphical object from Events library.

General history database

The database for collecting history information you can setup in **Project properties->Events/History tab**^[107] in History DB name field. If you want that tag's history information is saved in this database you have to Enable history in **Tag properties->History tab**^[474], setup Storage period and check Store in DB.

The history values will be saved every storage period during execution if the value of the tag is changed (if Use deadband is enabled the delta between current value and value last saved should be greater Deadband).

You can show history information collected in General history database by using [Trend DB](#)^[228] graphical object from [Trends and charts](#)^[228] library.

History database

It's another way for collecting history information. The differences between General history database and [History database](#)^[485] is in History database you save only selected tag's values and values are saved in two ways:

- Time interval. Tags values are saved every time interval independently values are changed or not.
- Tag. Tags values are saved when set tag's value become TRUE independently saved values are changed or not.

You can create history database in [Project Window](#)^[72] -> [Databases](#)^[88] or in the menu item [Project](#)^[67]->[New Database](#)^[69] of the [Main menu](#)^[61].

Tag's value will be saved in the History database if you check Enable history in Tag properties (you no need to check Store in DB in this case) and include this tag in History database properties collection of tags.

Like for General history database you can use as [SQLite](#)^[29] as [MySQL](#)^[31] databases.

You can show history information collected in History database by using [History DB table](#)^[252] and [History DB trend](#)^[255] from [History DB](#)^[252] library. [History Excel Report](#)^[263] and History Max and Min Report also work with this database.

Recipe database

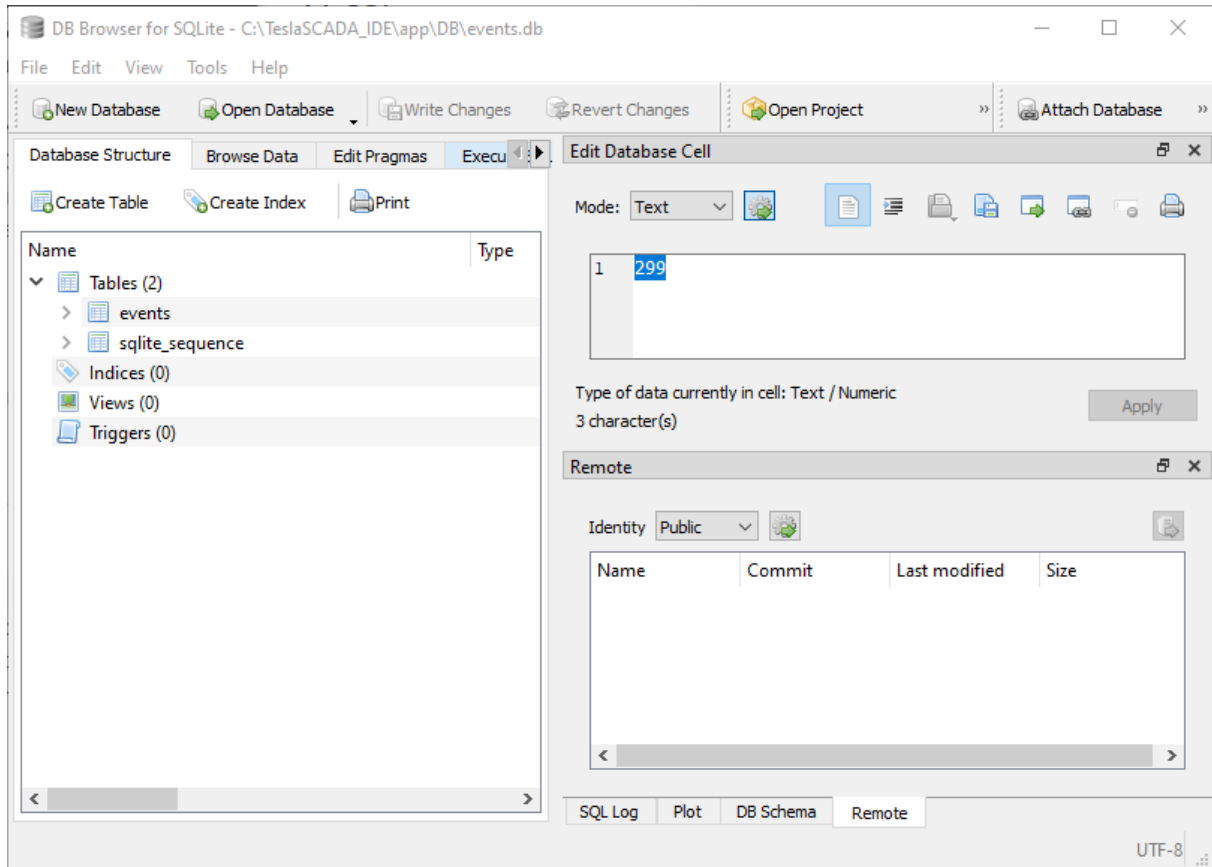
The database for working with recipes and parameters. You can create recipe database in [Project Window](#)^[72] -> [Databases](#)^[88] or in the menu item [Project](#)^[67]->[New Database](#)^[69] of the [Main menu](#)^[61].

Like for events and history databases you can use as [SQLite](#)^[29] as [MySQL](#)^[31] recipe databases.

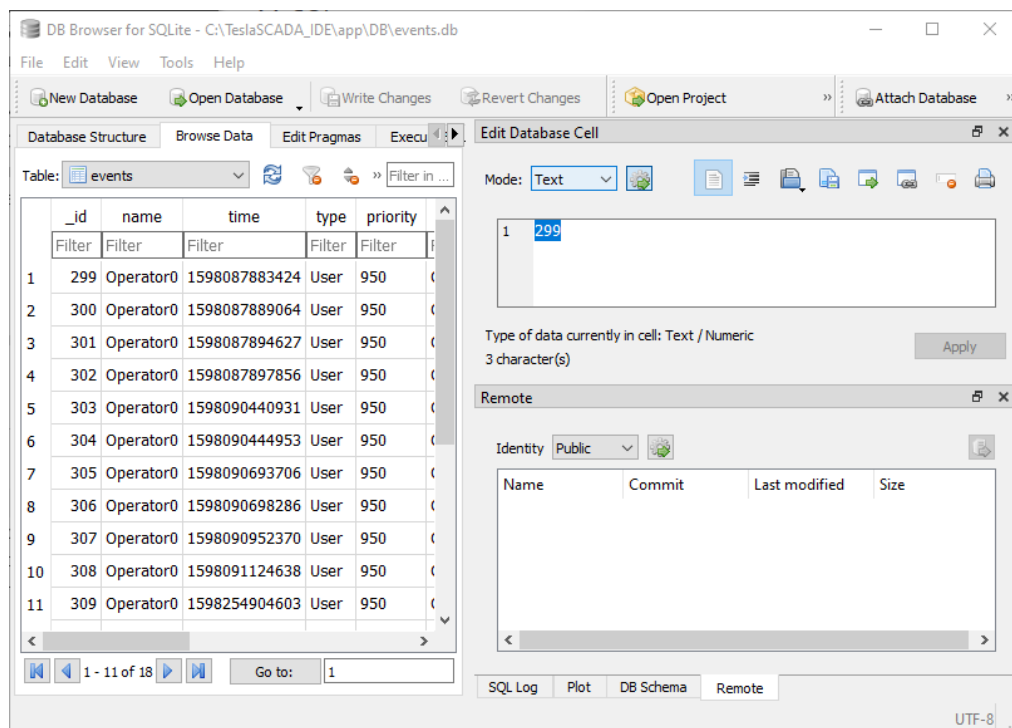
You can show recipe information collected in Recipe database by using graphical objects from Recipes library.

4.1 SQLite

If you want to use SQLite database in your project you no need to install any additional software on your PC. All databases are created automatically during application running. Databases are stored in the folder [DB](#)^[18] in the place where TeslaSCADA2 was installed. If you want to open database use some SQLite DB browsers. For example, for Windows you can use this one: <https://sqlitebrowser.org/dl/>. How looks SQLite database in this browser you can see here:



DB data looks like here:

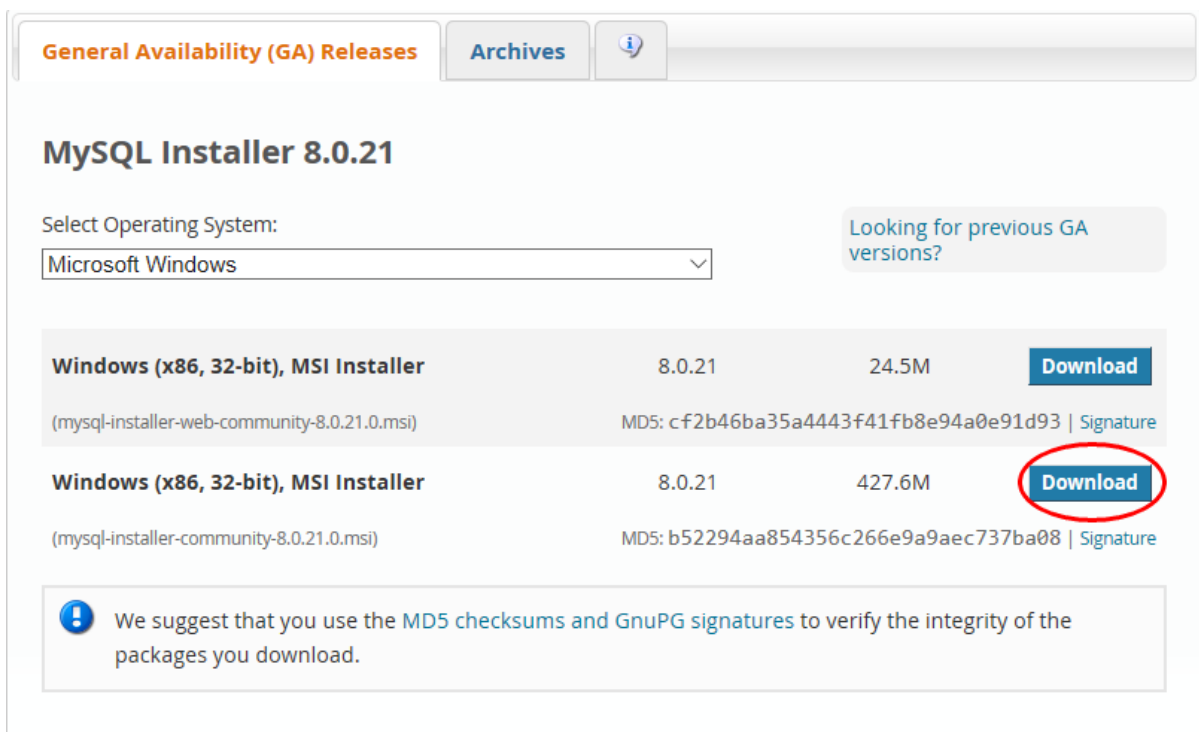


4.2 MySQL

To work with MySQL databases you have to install it on your PC. This chapter provides a step-by-step overview of the MySQL database installation process on Windows PC - this does not require special skills and knowledge, everything is quite simple. TeslaSCADA2 works with MySQL versions 5.6.2 and higher. The current MySQL version can be found on the official download page: <https://dev.mysql.com/downloads/windows/installer/>

Important! On Windows 7 x32 only [MySQL 5.7.25](#) can be installed.

After going to the download page at the bottom you can see the "MySQL Installer" block - click "Download":



General Availability (GA) Releases Archives ⓘ

MySQL Installer 8.0.21

Select Operating System:
Microsoft Windows

[Looking for previous GA versions?](#)

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.21.0.msi)	8.0.21	24.5M	Download
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.21.0.msi)	8.0.21	427.6M	Download

MD5: c-f2b46ba35a4443f41fb8e94a0e91d93 | [Signature](#)

MD5: b52294aa854356c266e9a9aec737ba08 | [Signature](#)

! We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

To download MySQL without registration, click on the link "No thanks, just start my download":

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »
using my Oracle Web account

Sign Up »
for an Oracle Web account

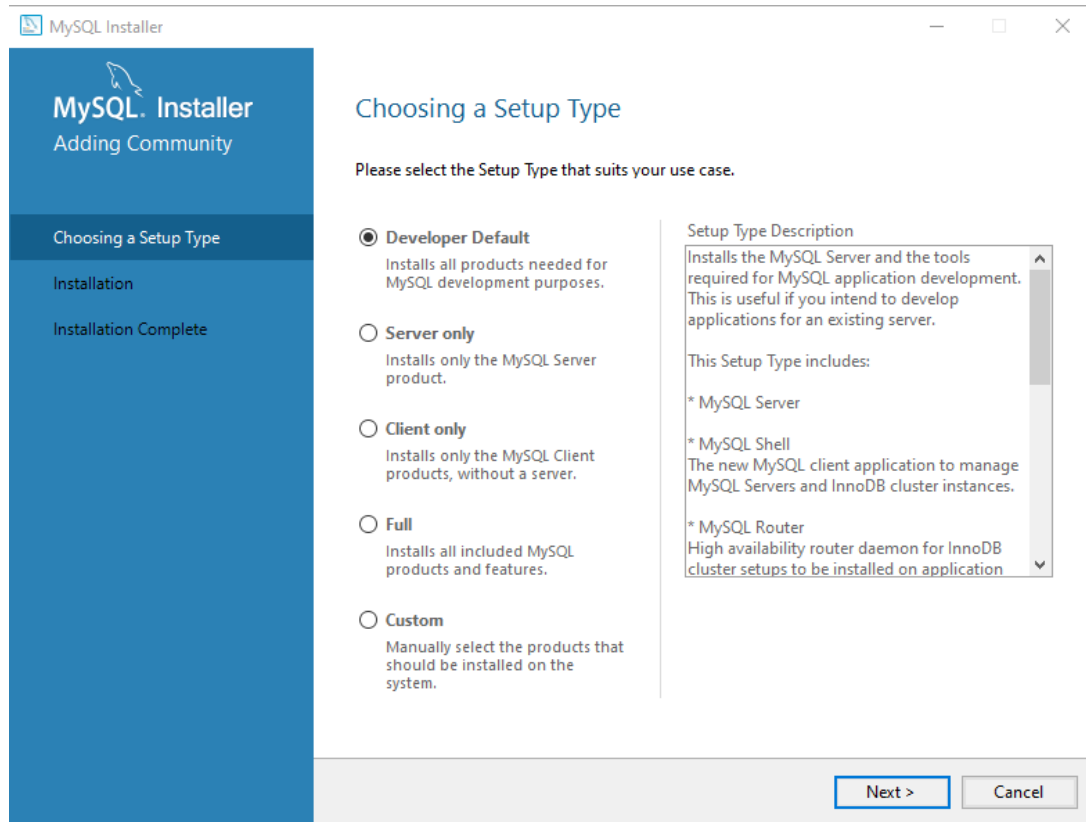
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

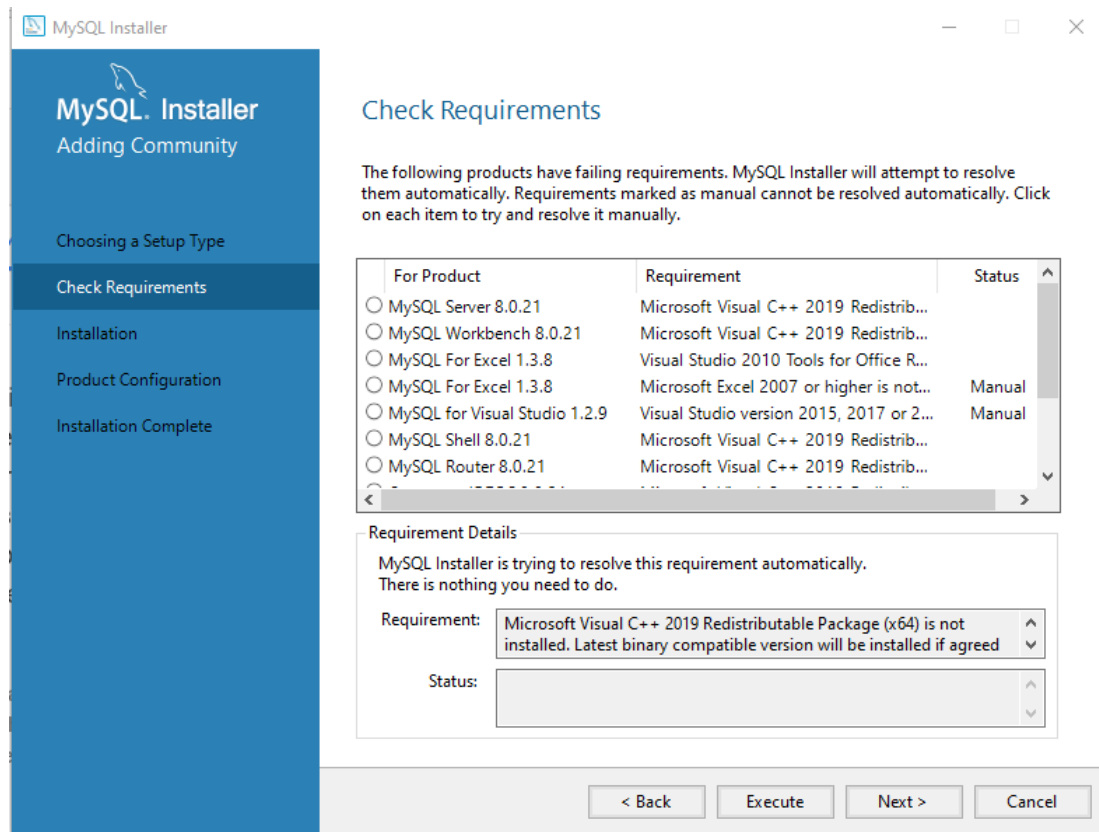
After the download is complete, you should make sure that the components necessary for installing MySQL are installed on the system:

- [Microsoft .NET Framework 4.5.2](#)
- [Microsoft Visual C ++ Redistributable for Visual Studio 2015](#)

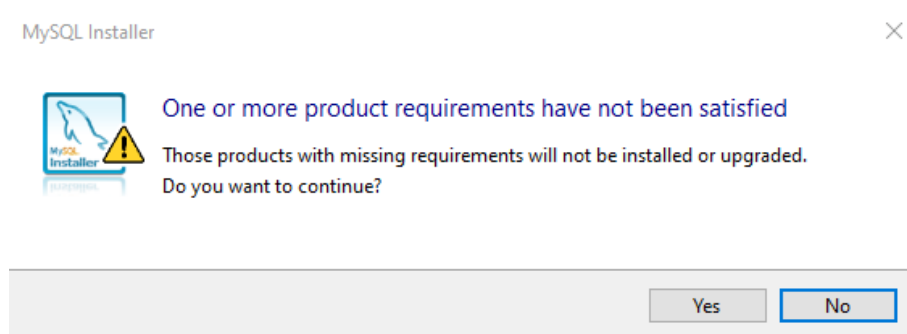
We select the default installation type "Developer Default" and click "Next":



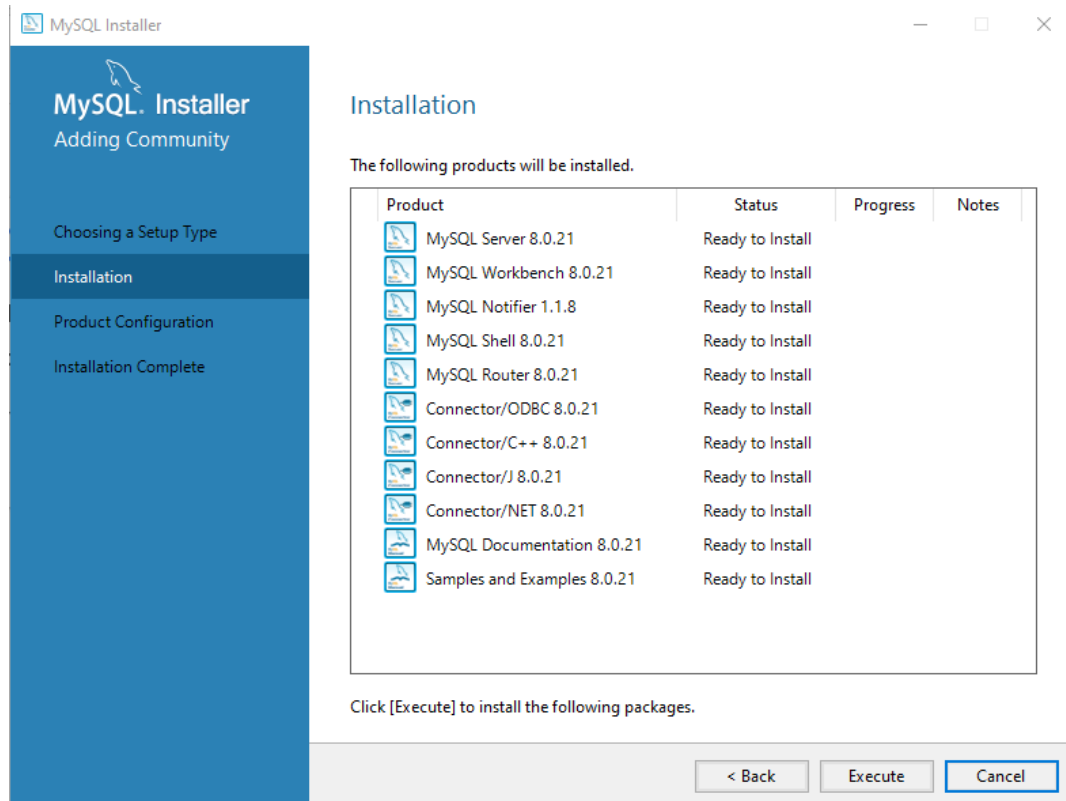
Next, the installer will show a list of components in the "For Product" column and a list of software required to install them in the "Requirement" column. For example, this list might include MySQL Workbench if Microsoft Visual C++ Redistributable for Visual Studio 2015 is not installed on the system. If you ignore the warning and continue with the installation, MySQL Workbench will not be installed. To install Microsoft Visual C++ Redistributable for Visual Studio 2015 click "Execute". Installation of other components is not required - you can continue the installation by clicking "Next":



A warning will appear - press "YES":

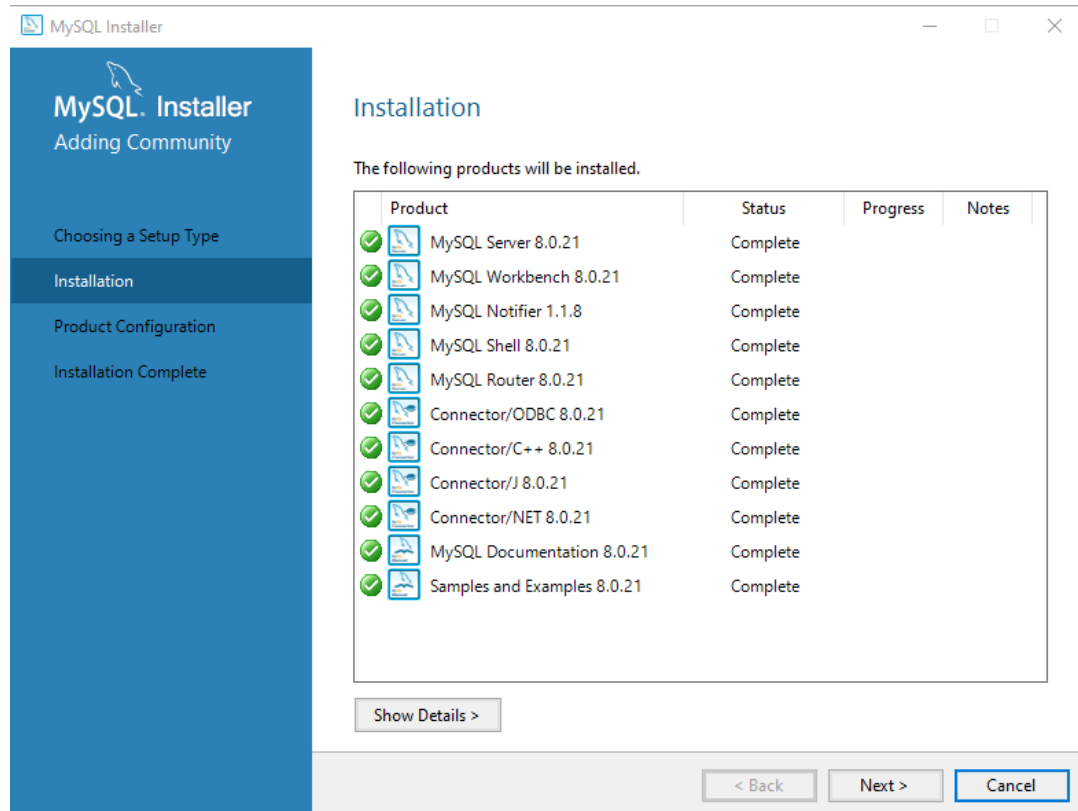


Then the installer will show you what exactly it will install, click "Execute":

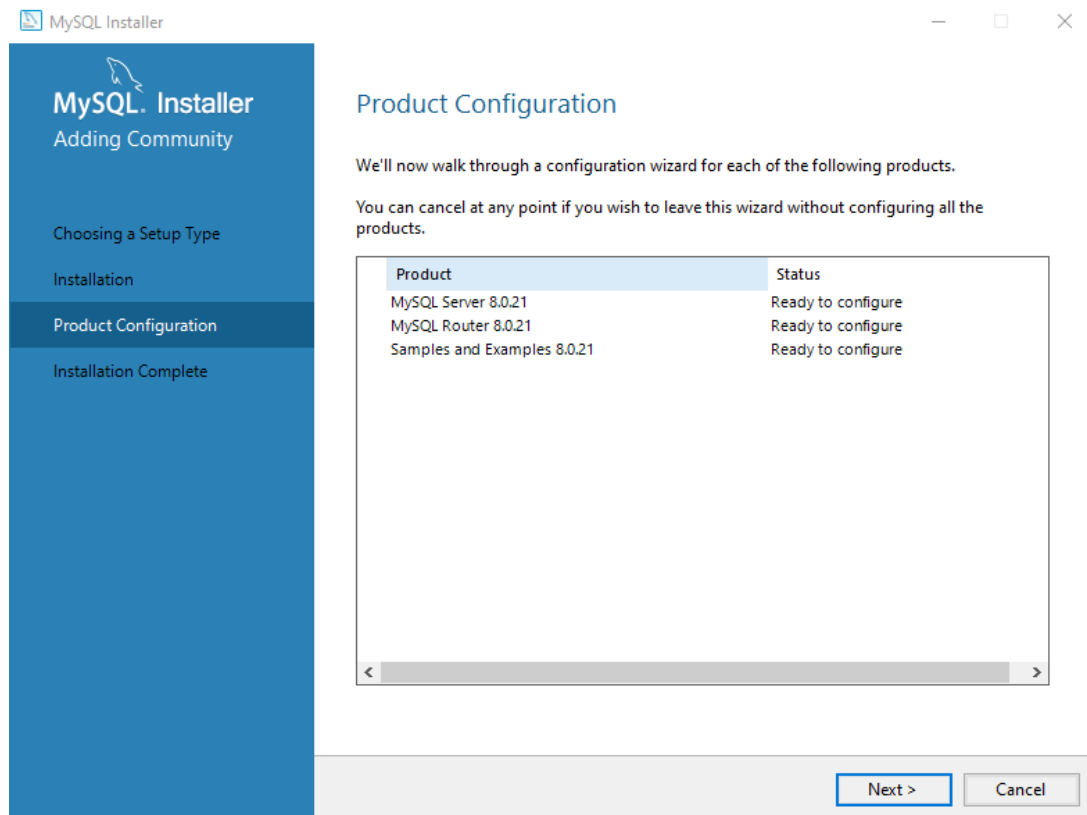


Important! If during the installation MySQL Server does not install with the error "This application requires Visual Studio 2015 Redistributable. Please install the Redistributable then run this installer again", then you need to install the 32-bit package Microsoft Visual C++ Redistributable for Visual Studio 2015 (vc_redist_x86) even if you are using a 64-bit operating system.

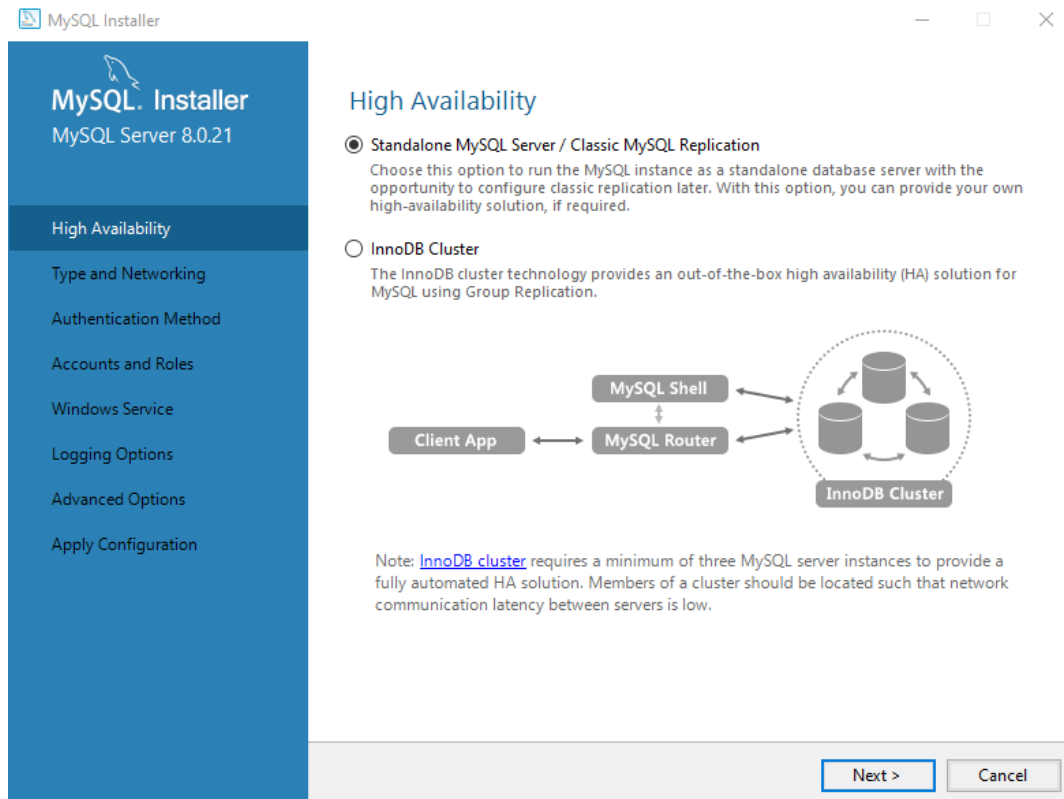
The installation process begins. After installing all the components, the "Next" button will appear, click it:



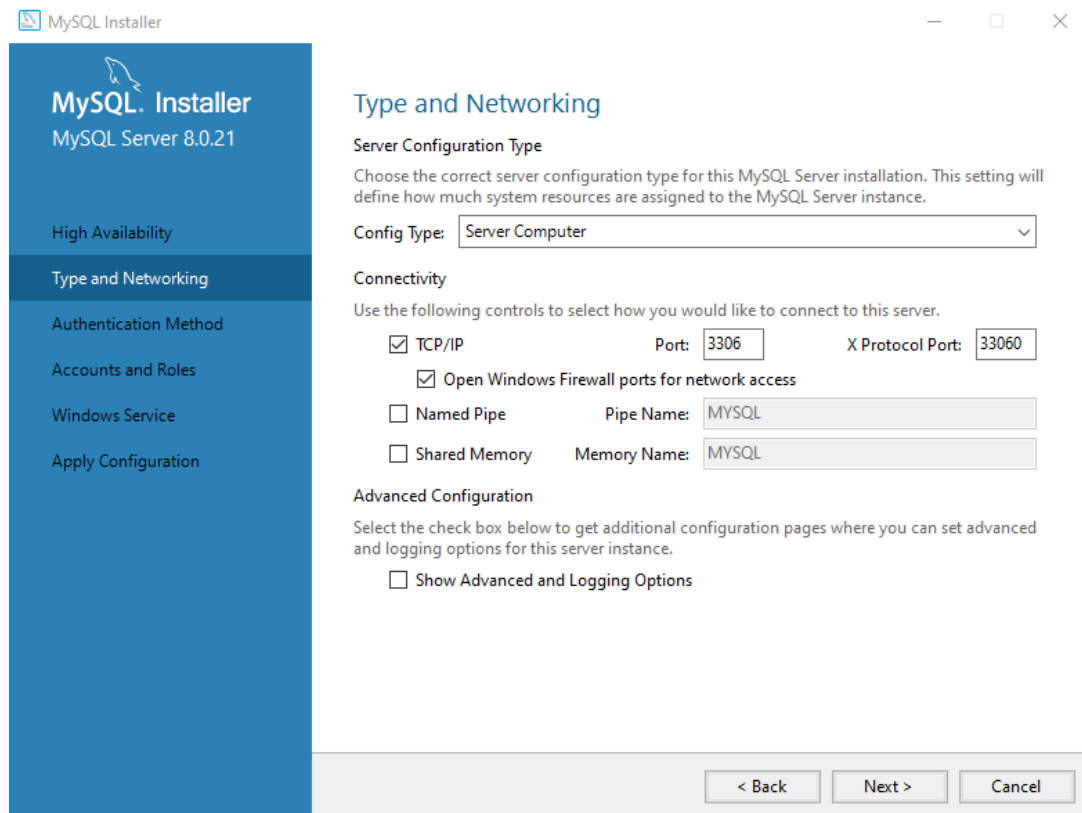
Next, you need to configure the MySQL server, click "Next":



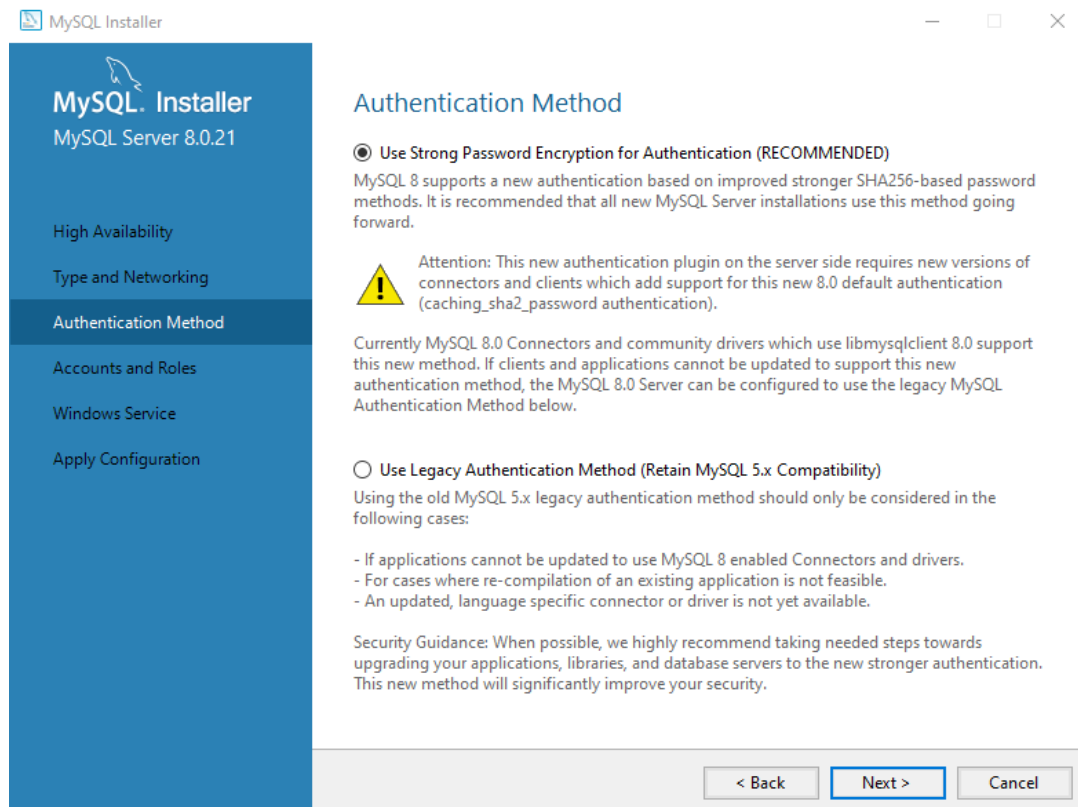
Select the "Standalone MySQL Server / Classic MySQL Replication" item and click "Next":



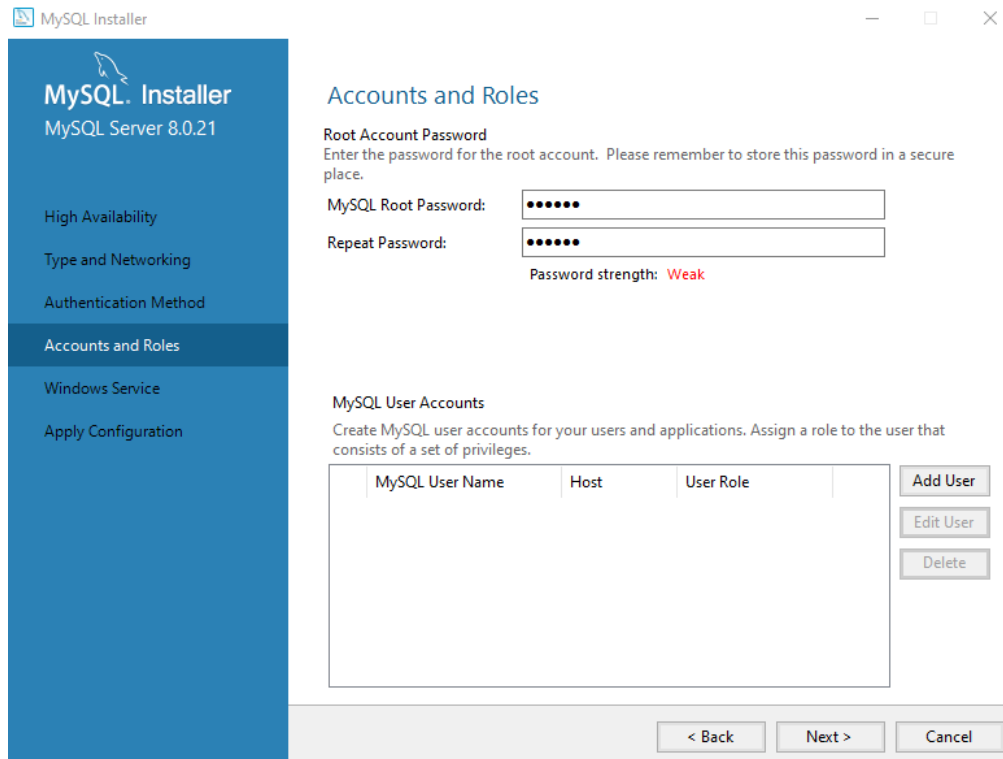
Next, in the "Config Type" parameter, select "Server Computer" and click "Next":



Select "Use Strong Password Encryption for Authentication" and click "Next":



In the next window, you need to set a password for the root user (administrator). Also, here you can add other users (by clicking the "Add User" button), if necessary. After entering the password, click "Next":



The screenshot shows the 'Accounts and Roles' step of the MySQL Installer for MySQL Server 8.0.21. The left sidebar lists the installation steps: High Availability, Type and Networking, Authentication Method, Accounts and Roles (selected), Windows Service, and Apply Configuration. The main area is titled 'Accounts and Roles' and contains the 'Root Account Password' section. It prompts the user to enter a password for the root account, with a 'MySQL Root Password' field and a 'Repeat Password' field. The password strength is indicated as 'Weak'. Below this is the 'MySQL User Accounts' section, which instructs the user to create MySQL user accounts. It features a table with columns 'MySQL User Name', 'Host', and 'User Role', and buttons for 'Add User', 'Edit User', and 'Delete'. At the bottom, there are navigation buttons: '< Back', 'Next >', and 'Cancel'.

MySQL Installer

MySQL Server 8.0.21

High Availability

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

Accounts and Roles

Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password strength: **Weak**

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role
-----------------	------	-----------

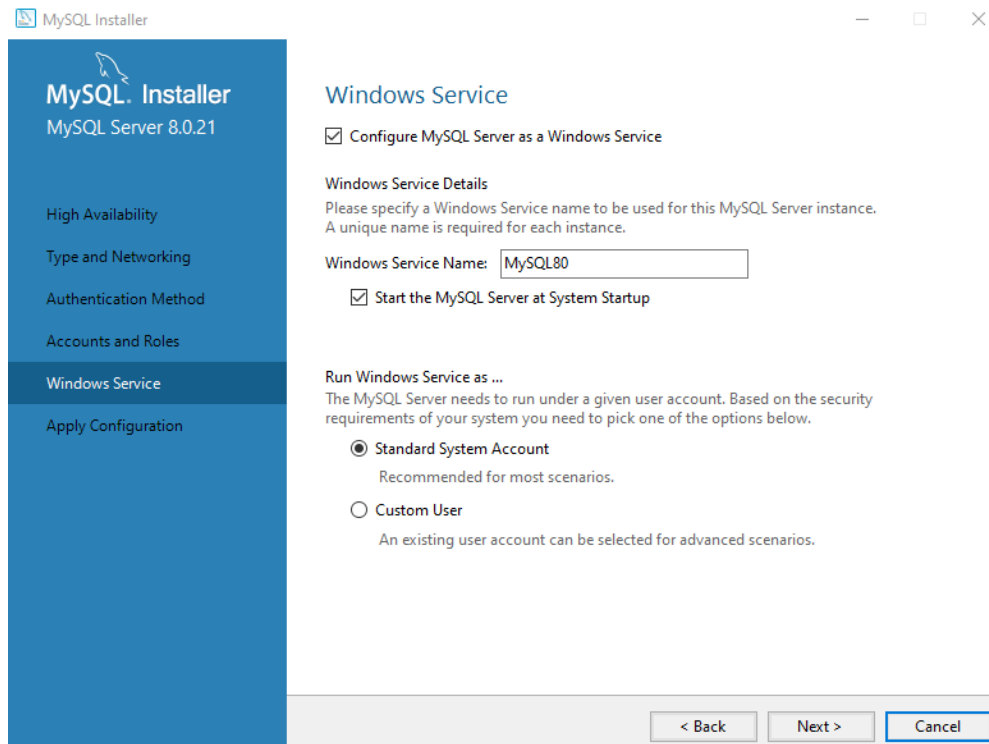
Add User

Edit User

Delete

< Back Next > Cancel

At the next step, we leave all the default settings, click "Next":



The screenshot shows the 'Windows Service' step of the MySQL Installer for MySQL Server 8.0.21. The left sidebar lists the installation steps: High Availability, Type and Networking, Authentication Method, Accounts and Roles, Windows Service (selected), and Apply Configuration. The main area is titled 'Windows Service' and contains the 'Configure MySQL Server as a Windows Service' checkbox, which is checked. Below this is the 'Windows Service Details' section, which prompts the user to specify a Windows Service name to be used for this MySQL Server instance. A unique name is required for each instance. The 'Windows Service Name' field contains 'MySQL80'. The 'Start the MySQL Server at System Startup' checkbox is also checked. Below this is the 'Run Windows Service as ...' section, which prompts the user to select a user account. The 'Standard System Account' radio button is selected, and the 'Custom User' radio button is unselected. At the bottom, there are navigation buttons: '< Back', 'Next >', and 'Cancel'.

MySQL Installer

MySQL Server 8.0.21

High Availability

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

Windows Service

☒ Configure MySQL Server as a Windows Service

Windows Service Details
Please specify a Windows Service name to be used for this MySQL Server instance. A unique name is required for each instance.

Windows Service Name:

☒ Start the MySQL Server at System Startup

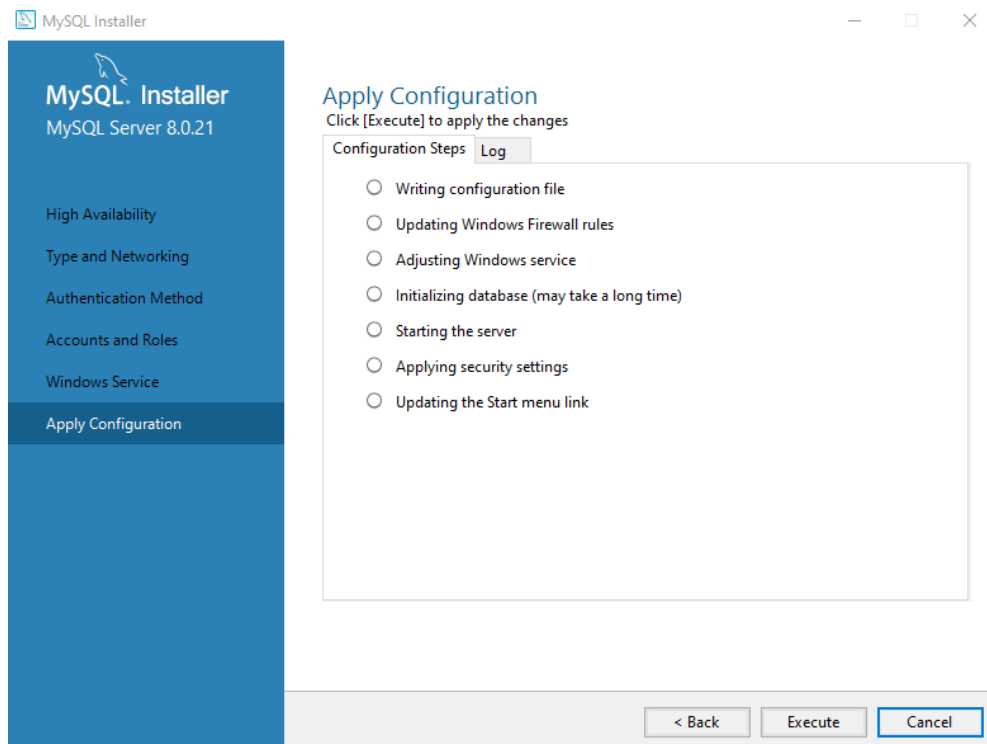
Run Windows Service as ...
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

☒ Standard System Account
Recommended for most scenarios.

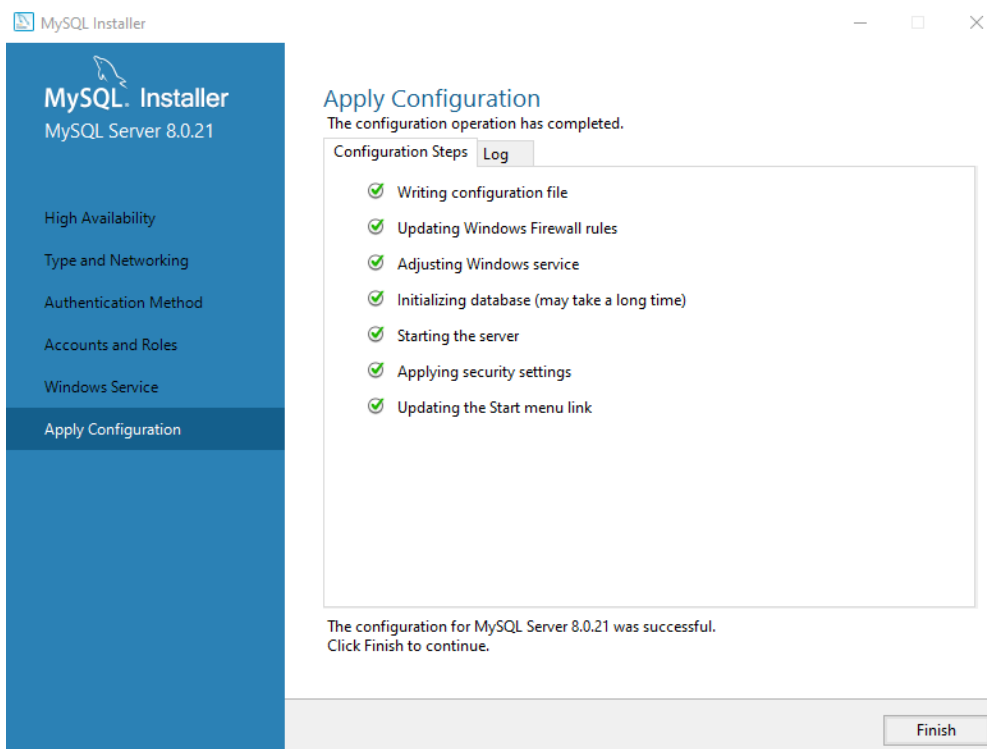
☐ Custom User
An existing user account can be selected for advanced scenarios.

< Back Next > Cancel

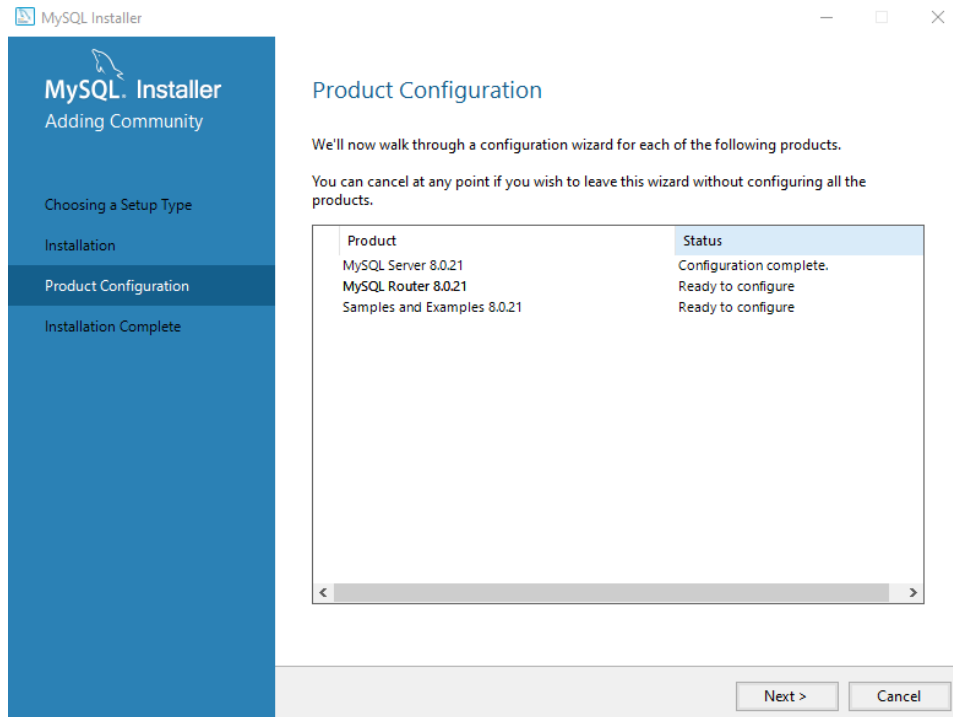
Next, you need to apply the MySQL server settings by clicking "Execute":



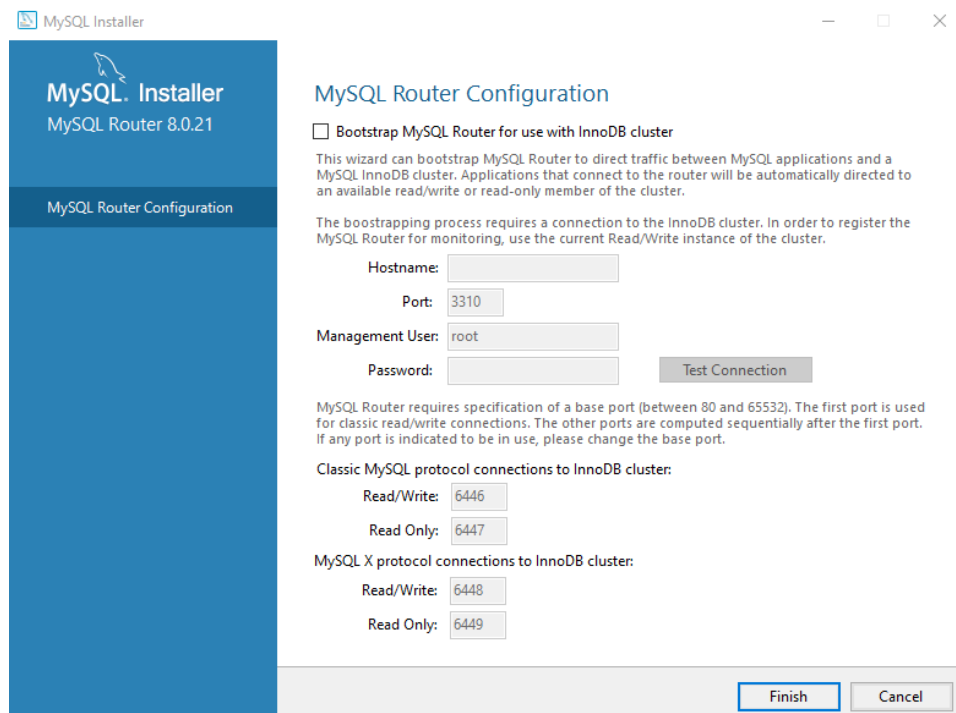
MySQL server is configured, click "Finish":



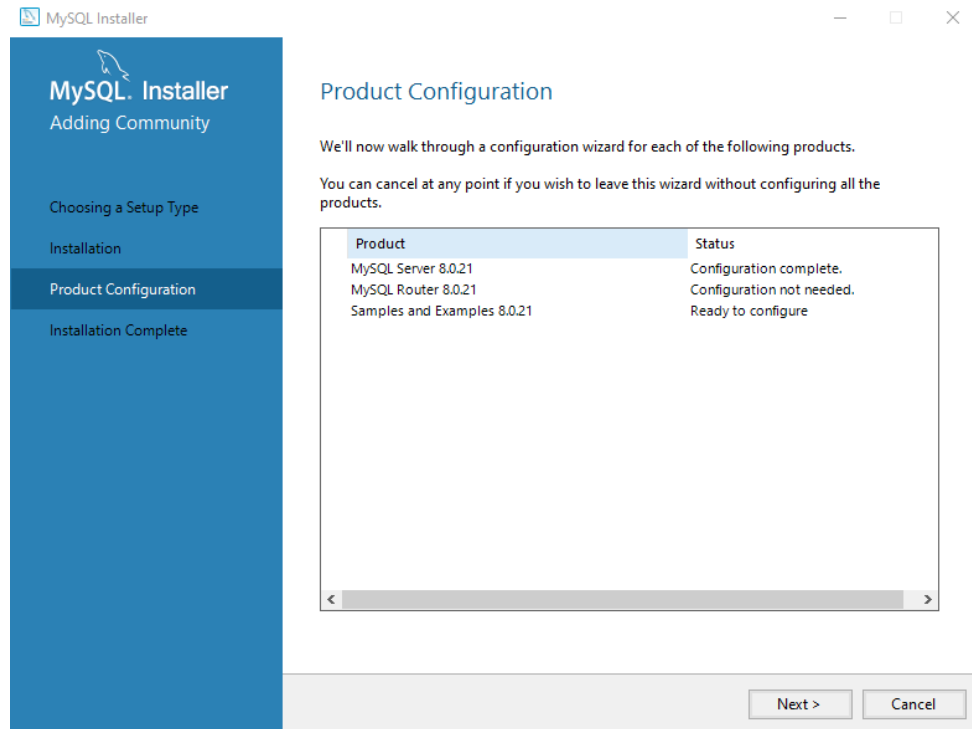
Next, let's move on to configuring MySQL Router. Click "Next":



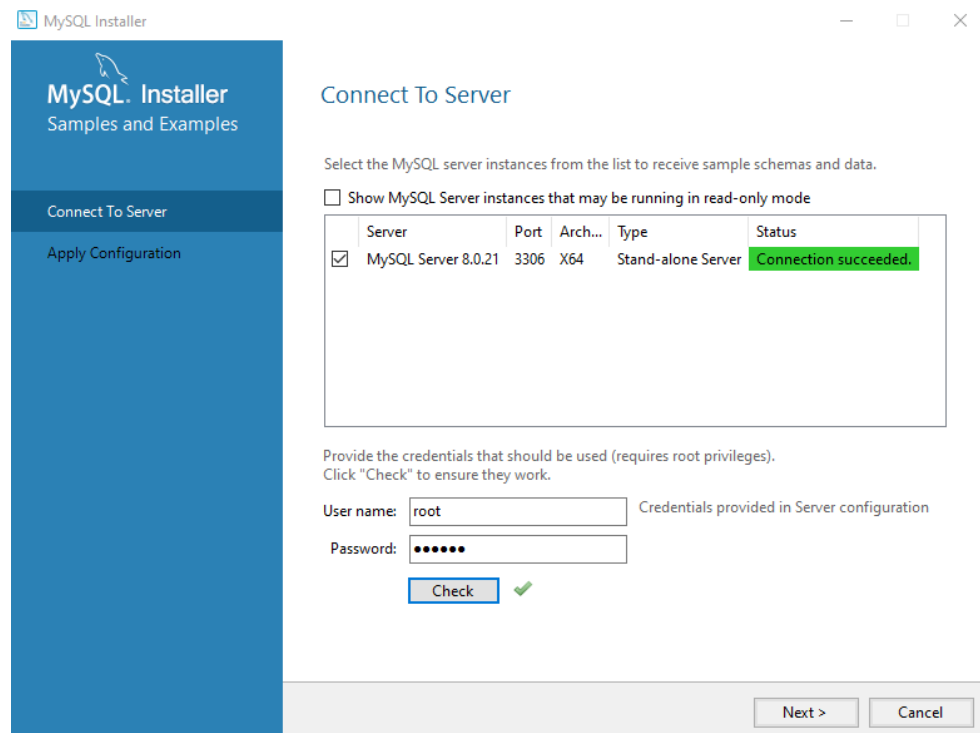
Leave all the default settings and click "Finish":



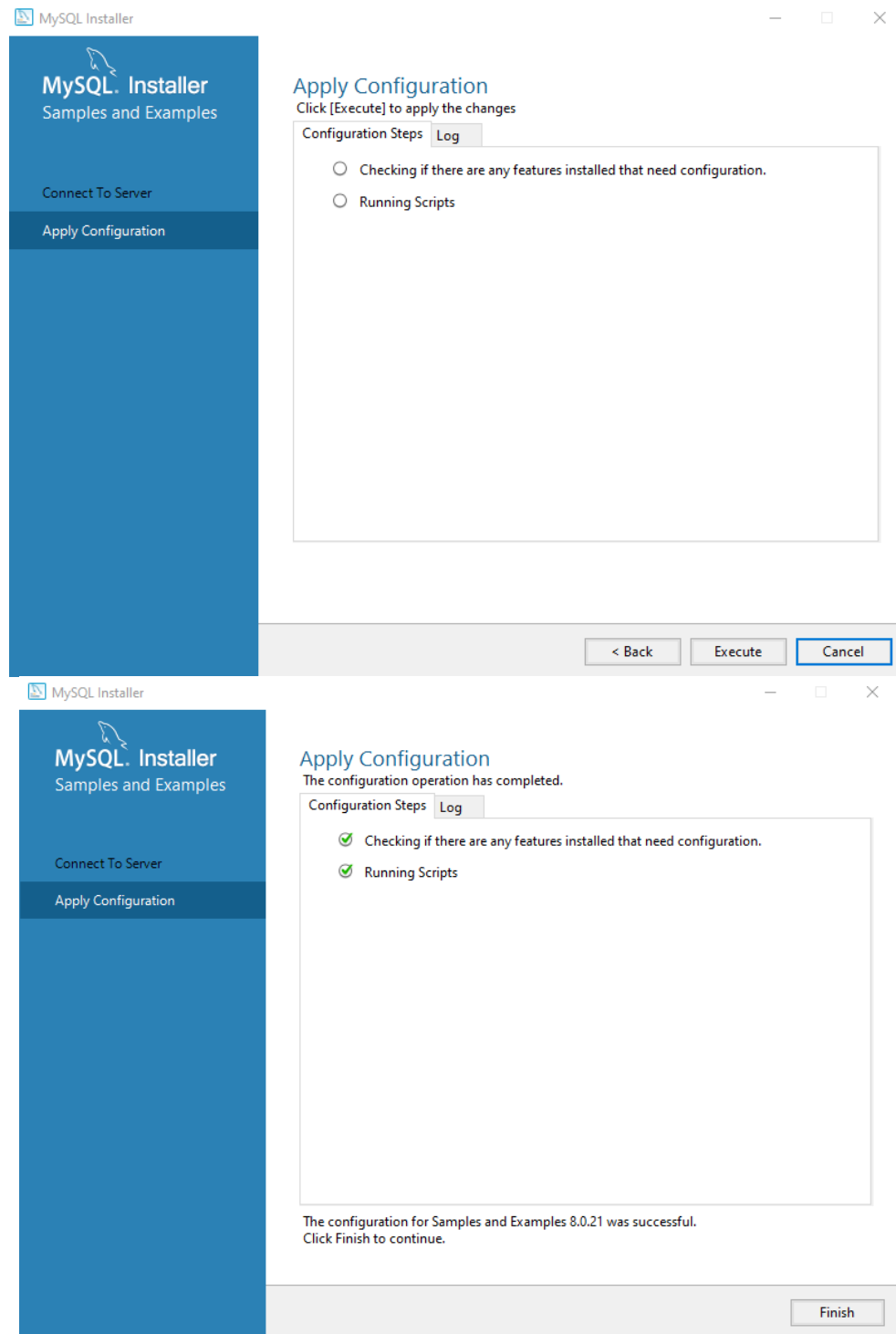
Now you need to check the created database, click "Next":



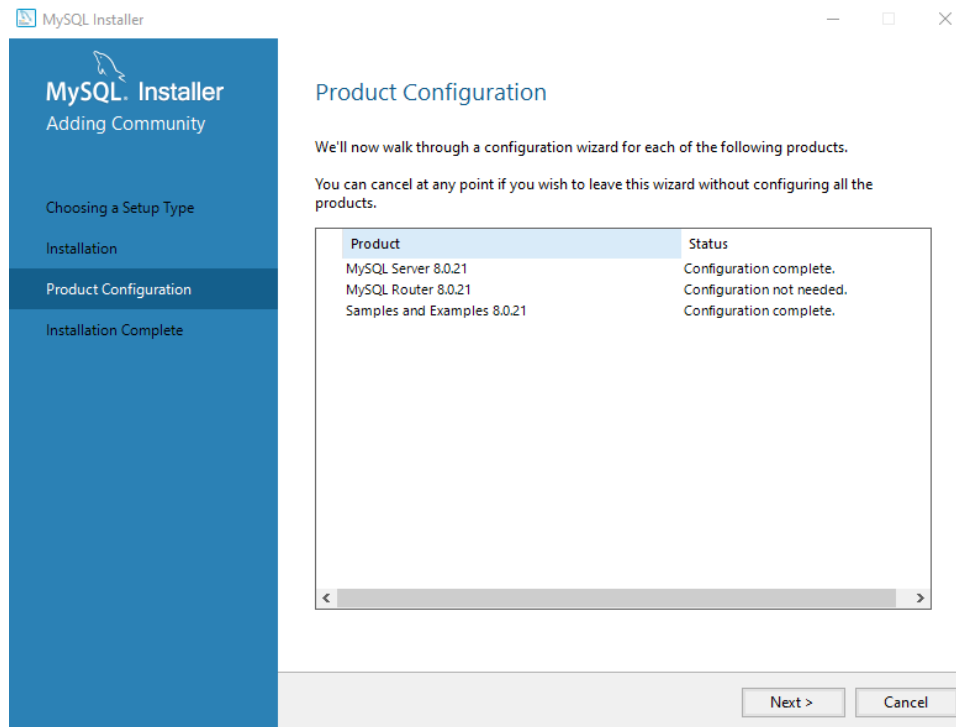
Check the connection. Enter the password, click "Check", then click "Next":



Next, click "Execute" and then "Finish":



Installation is almost complete, click "Next":



Installation is complete - click "Finish". If you check the "Start MySQL Workbench after Setup" box, the [MySQL Workbench](#)^[47] program will start, in which you can make additional database settings.

You have to [create a database](#)^[49] in MySQL server by using MySQL Workbench manually. After creating database you can use it for collecting event and history information. To do this open Project properties and in [Event/History tab](#)^[107] setup My SQL database by using jdbc:mysql: at the beginning of the Events DB name and History DB name:

In our case it's: jdbc:mysql://192.168.1.6:3306/test
where:

- **jdbc:mysql:** - beginning for MySQL.
- **192.168.1.6:3306** - IP address and port.
- **test** - name of the database (created in [MySQL Workbench](#)^[49]).

Also you can use My SQL database in Databases - Recipes and History DB. To do this in Db name of the database use jdbc:mysql: at the beginning also.

Important! If you get during the first running TeslaSCADA2 IDE or TeslaSCADA2 Runtime the Error message like this: "[java.sql.SQLException: The server time zone...](#)" , you have to setup time zone for your My SQL server, to do this open [MySQL Workbench](#)^[51].

4.2.1 MySQL Workbench

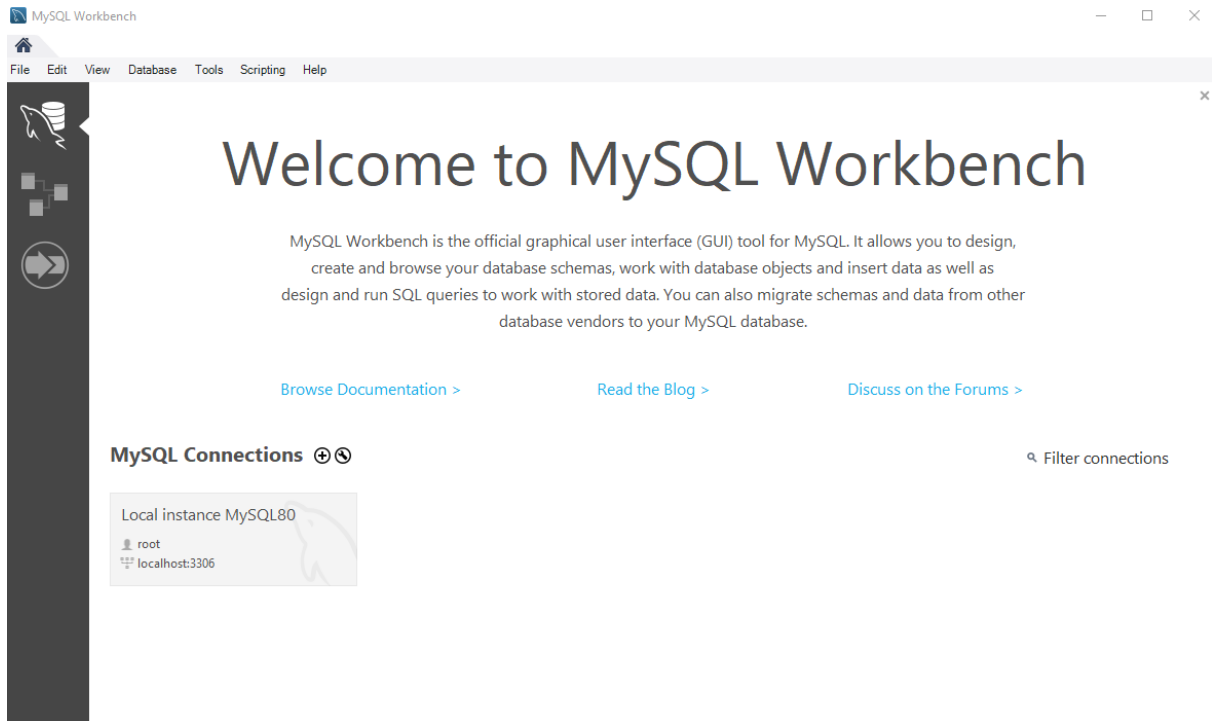
MySQL Workbench is a visual database design tool that integrates database design, modeling, creation and operation. Its capabilities will be useful to us for:

- backing up and restoring the database (also useful for transferring the database to another PC).
- settings for connecting to a remote database.

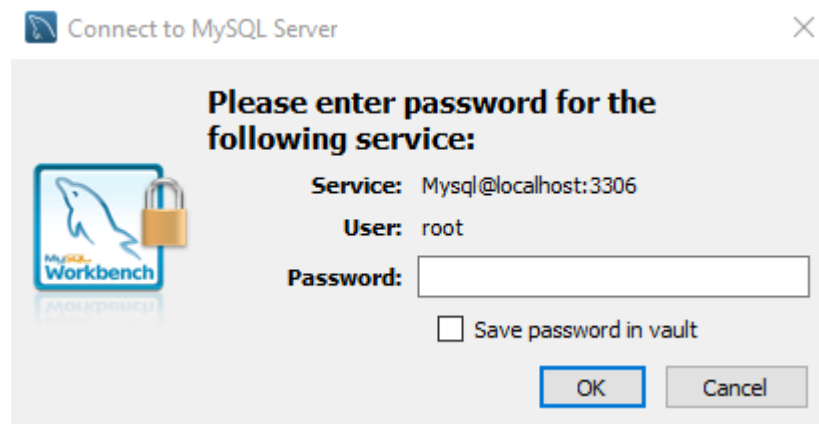
- changing the path of saving the database - "by default" is saved to disk "C".
- viewing database statistics.

If the database was installed according to the instructions in the previous section, then MySQL Workbench was installed along with MySQL, otherwise it can be downloaded from this link: <http://dev.mysql.com/downloads/workbench/>

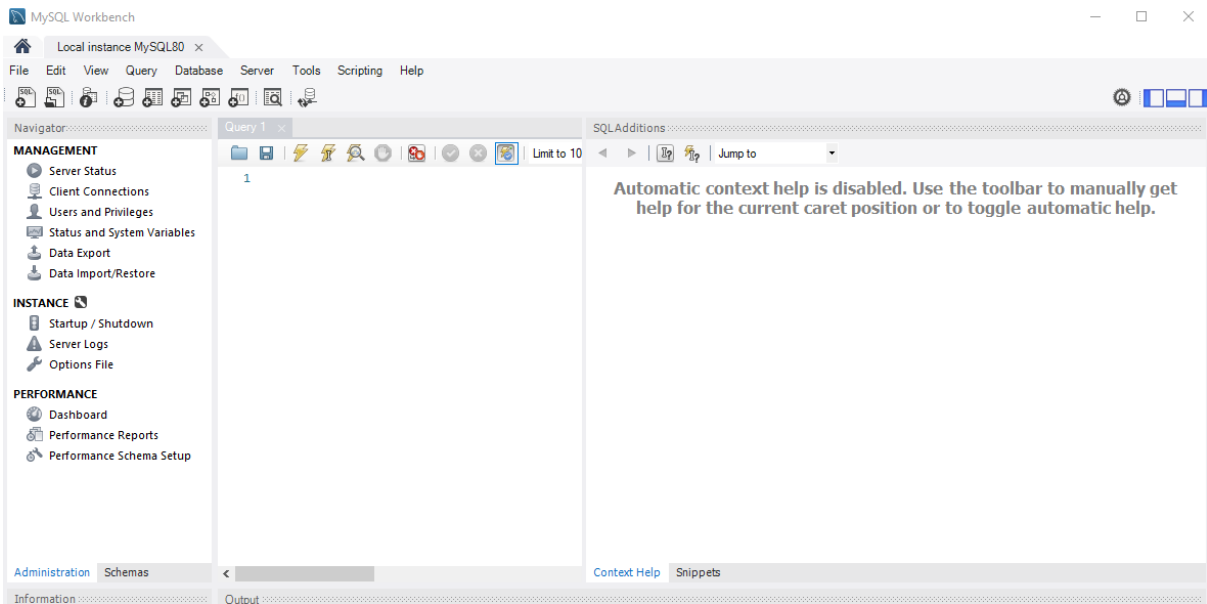
After starting MySQL Workbench, you need to select an instance of the MySQL server in order to connect to it. In our case, it is only one (local) - click on it:



Enter the root user password (which was invented when setting up MySQL):



After connecting to the MySQL server, we will see the start page:

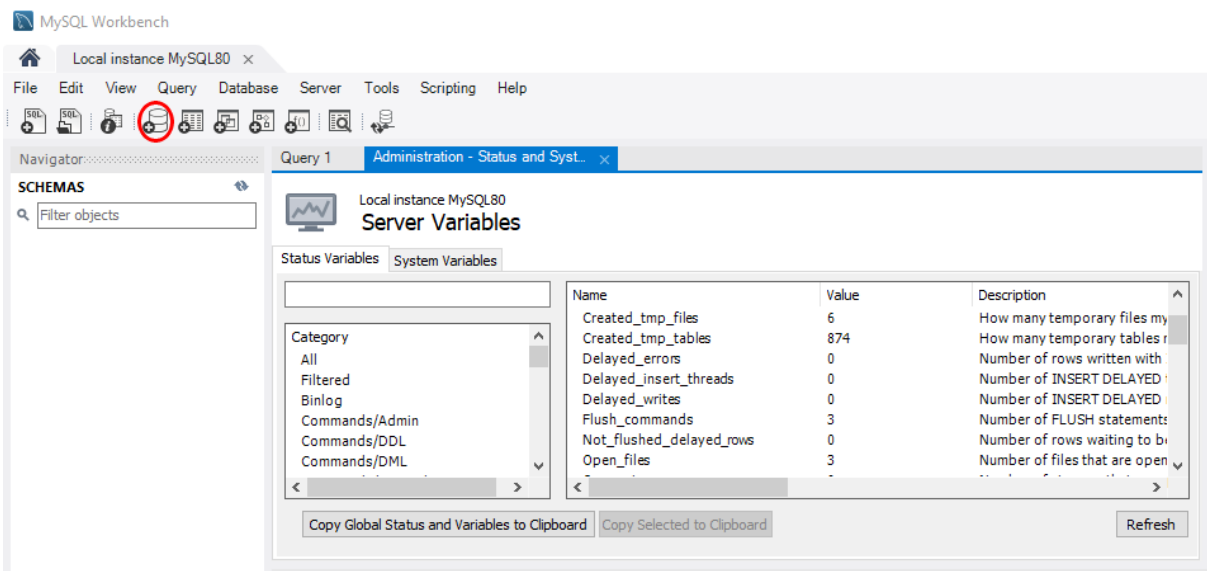


1. Administration - Settings of the MySQL server.
2. Schemas (Database area) - the list of created databases is displayed here. Also, when you first turn it on, test databases can be displayed here - they can be deleted by clicking on the name of the database with RMB - a pop-up menu will appear in which you need to select "Drop Schema".

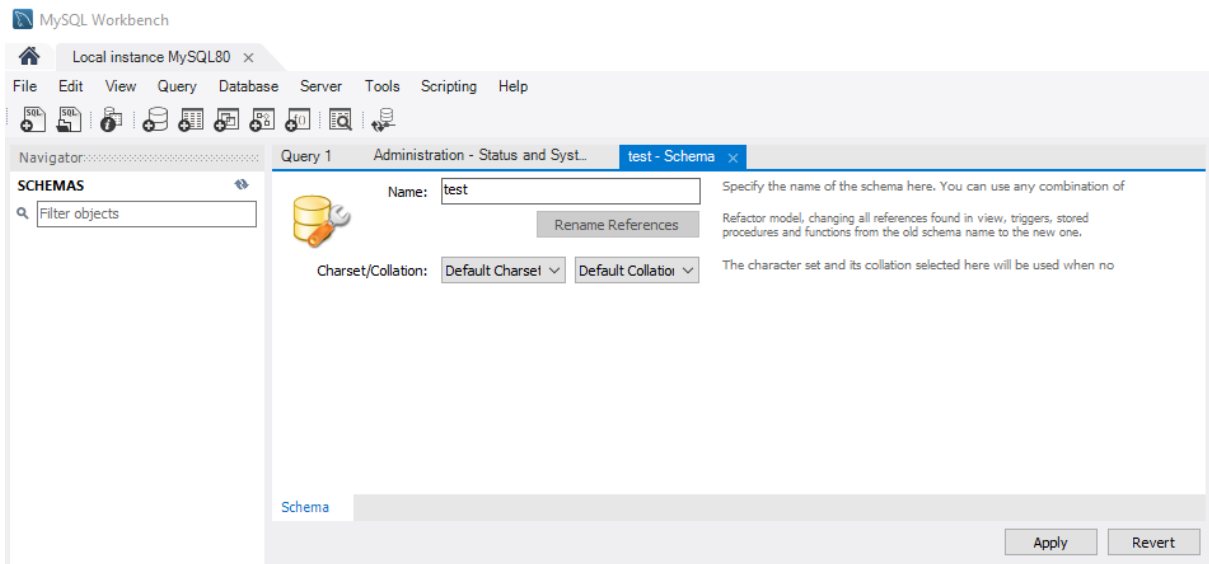
Create database

You have to create database manually:

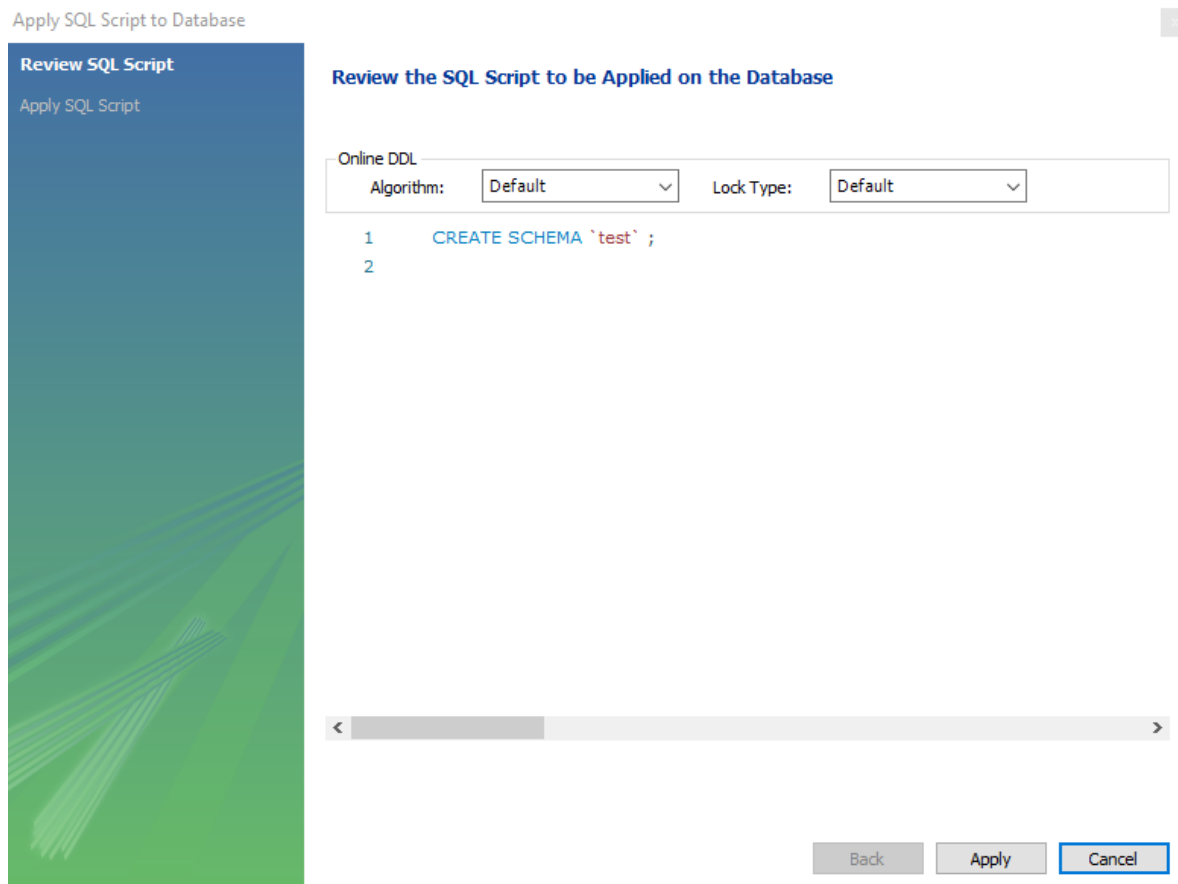
Open "Schemas" tab. Click "Create a new schema in the connected server" icon:



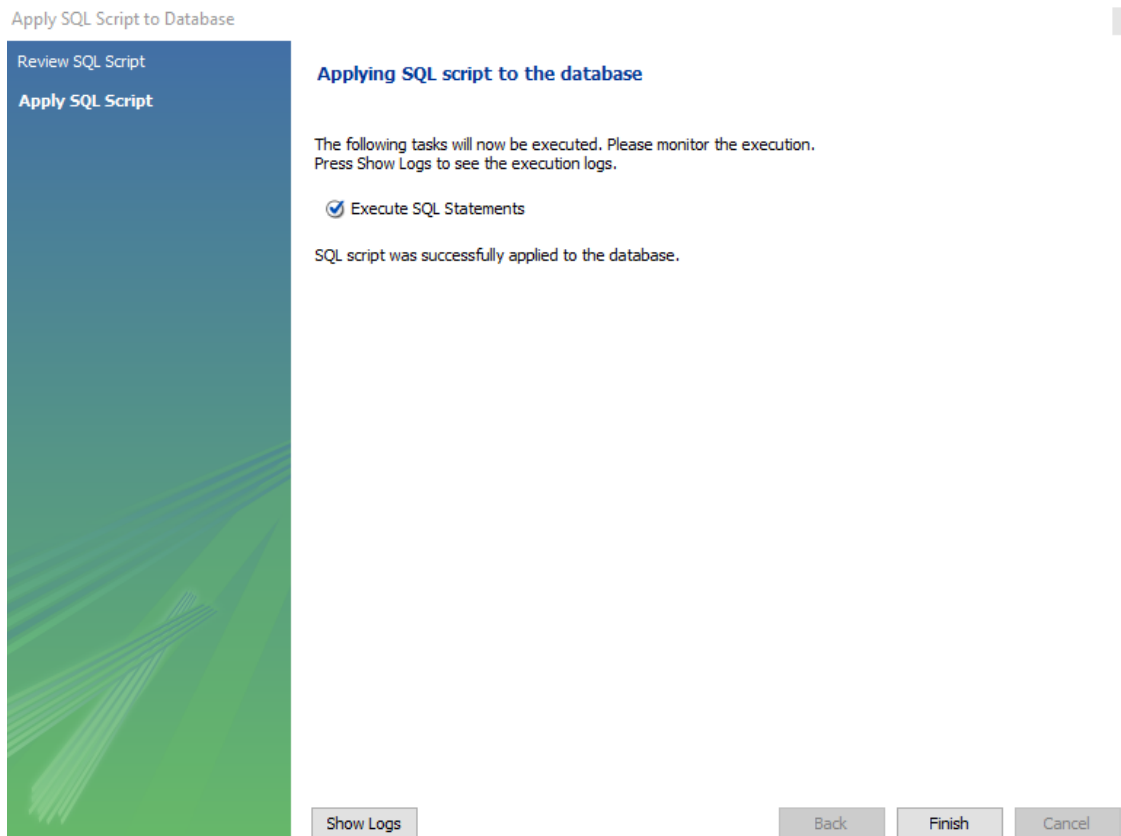
Enter "Name" of the schema and click Apply:



"Apply SQL script to Database" window will be appeared. Click Apply:

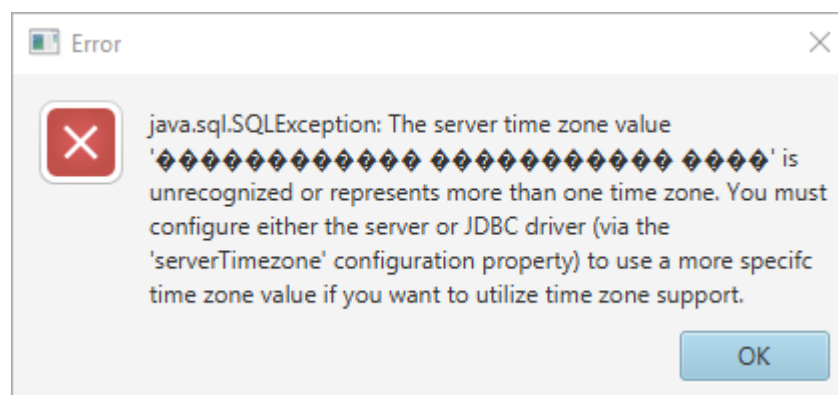


Then click "Finish":



Change server time zone

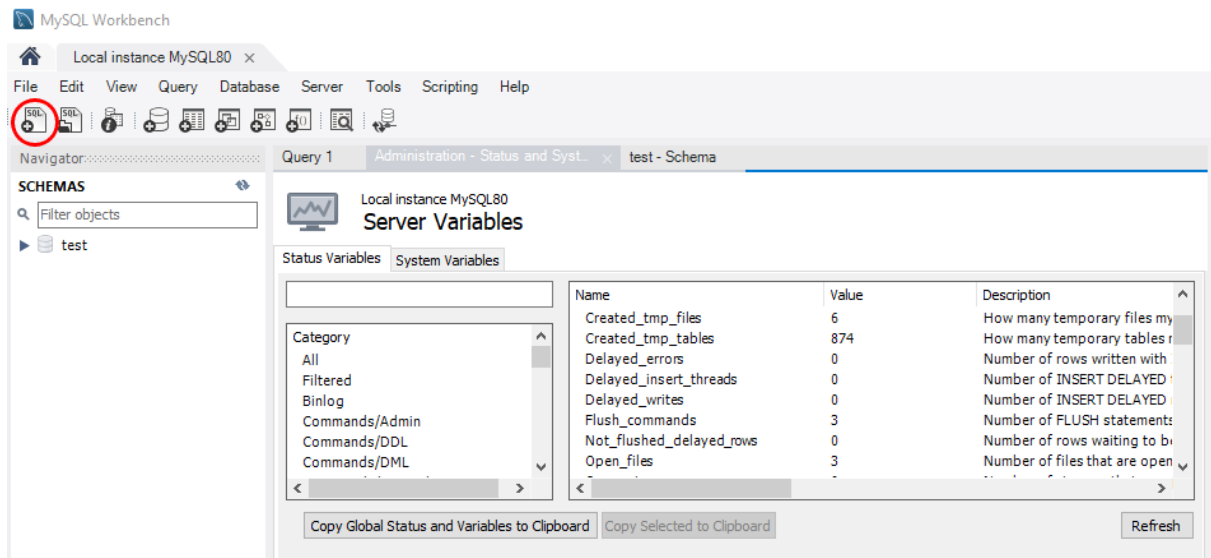
If you get during the first running TeslaSCADA2 IDE or TeslaSCADA2 Runtime the Error message like this:



You can fix it in 2 ways:

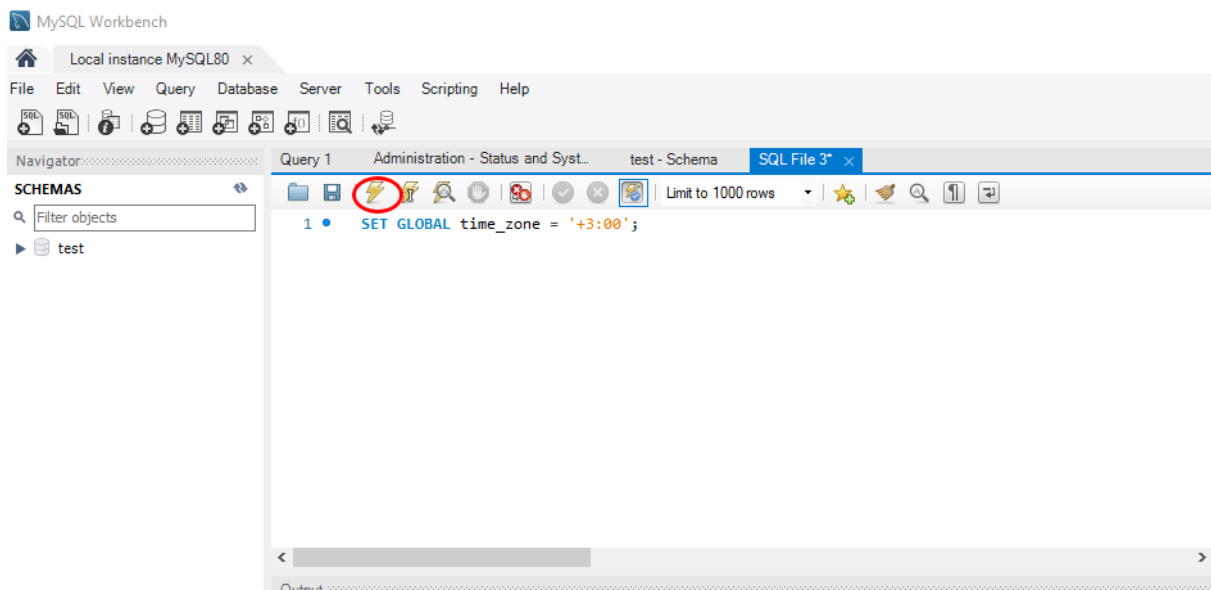
1. Set global by SQL query (it's a temporary solution, after restart your MySQL server the problem will return).

To do this you have to setup time zone for your My SQL server to do this open MySQL Workbench and click icon "Create a new SQL tab for executing queries":



Enter: `SET GLOBAL time_zone = '+3:00';`

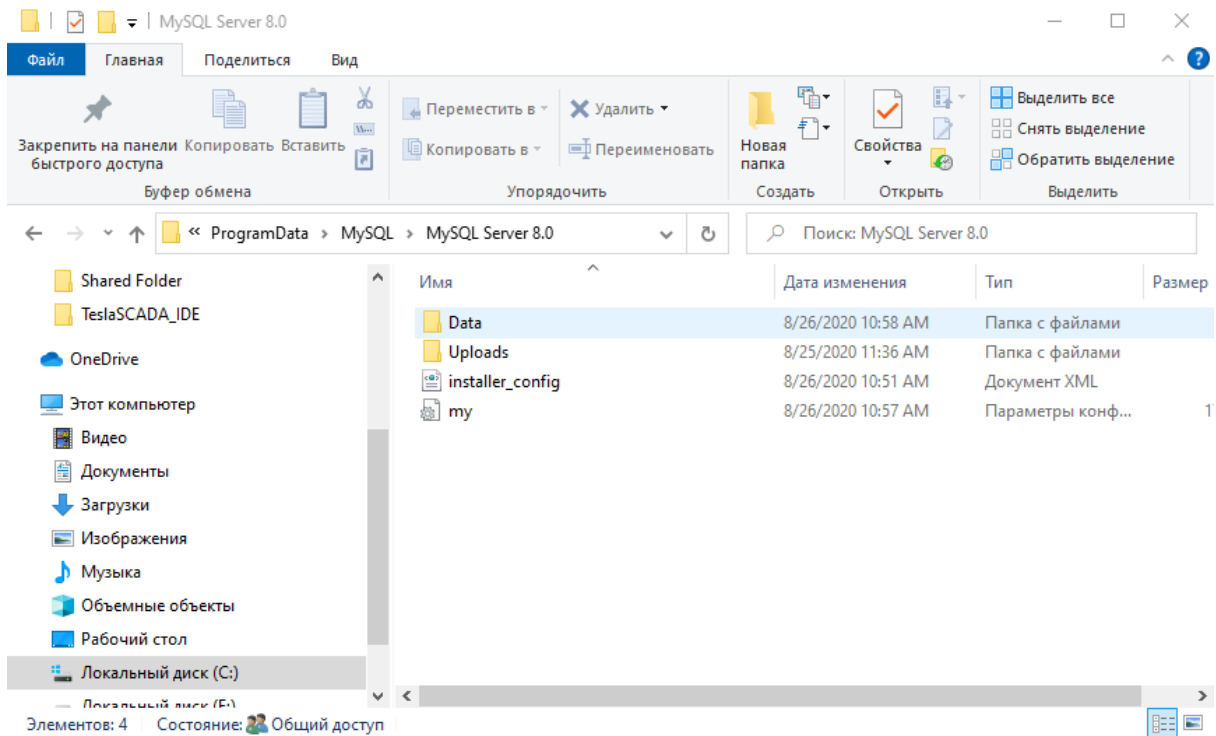
Where instead of '+3:00' you have to enter your time zone. And then click "Execute..." icon:



Now you can try Run TeslaSCADA2 project again in TeslaSCADA2 IDE or in TeslaSCADA2 Runtime.

2. Change my.ini (Windows) or my.inf (Linux) file.

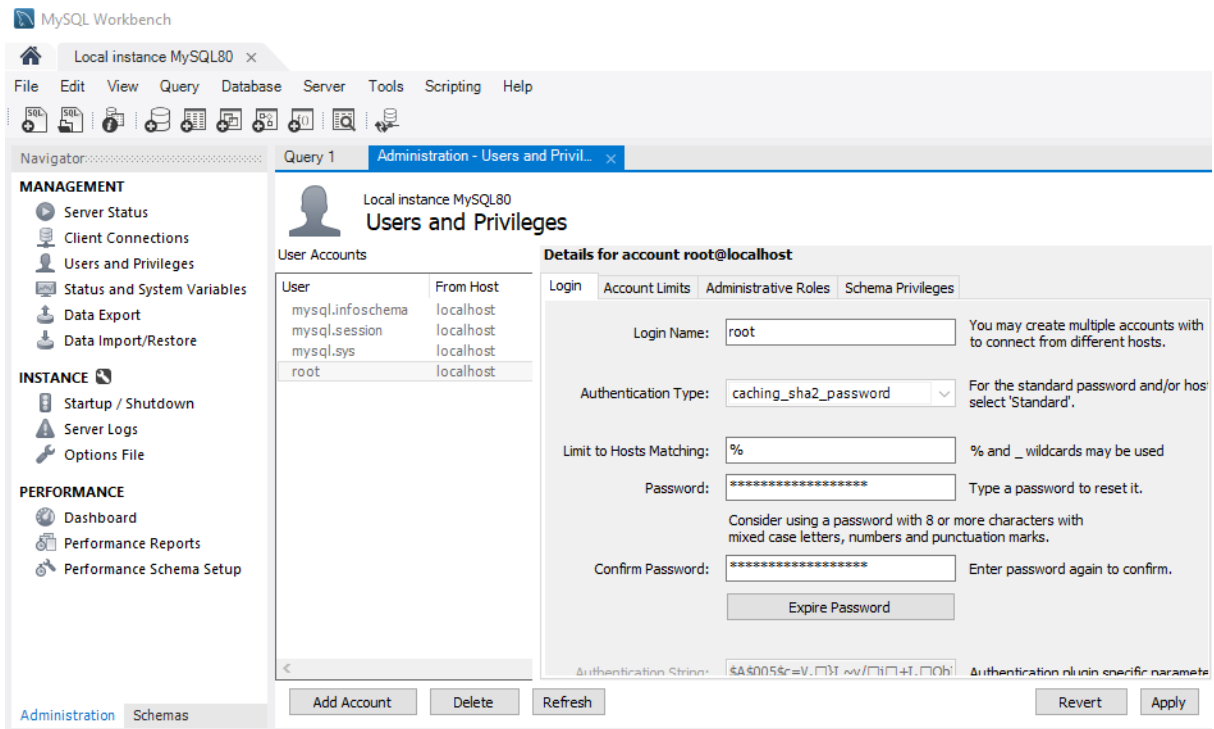
2.1 Find my.ini or my.inf file. It's an initialization file for MySQL server. Usually it's placed in C:/ProgramData/MySQL/MySQL Server 8.0/:



- 2.2. Open file my.ini and add this line: default-time-zone='+03:00' Where instead of '+3:00' you have to enter your time zone.
- 2.3. Save file (your current user should have access to this folder).
- 2.4. Restart your MySQL server.

Now you can try Run TeslaSCADA2 project again in TeslaSCADA2 IDE or in TeslaSCADA2 Runtime.

Settings required for connecting to a remote database



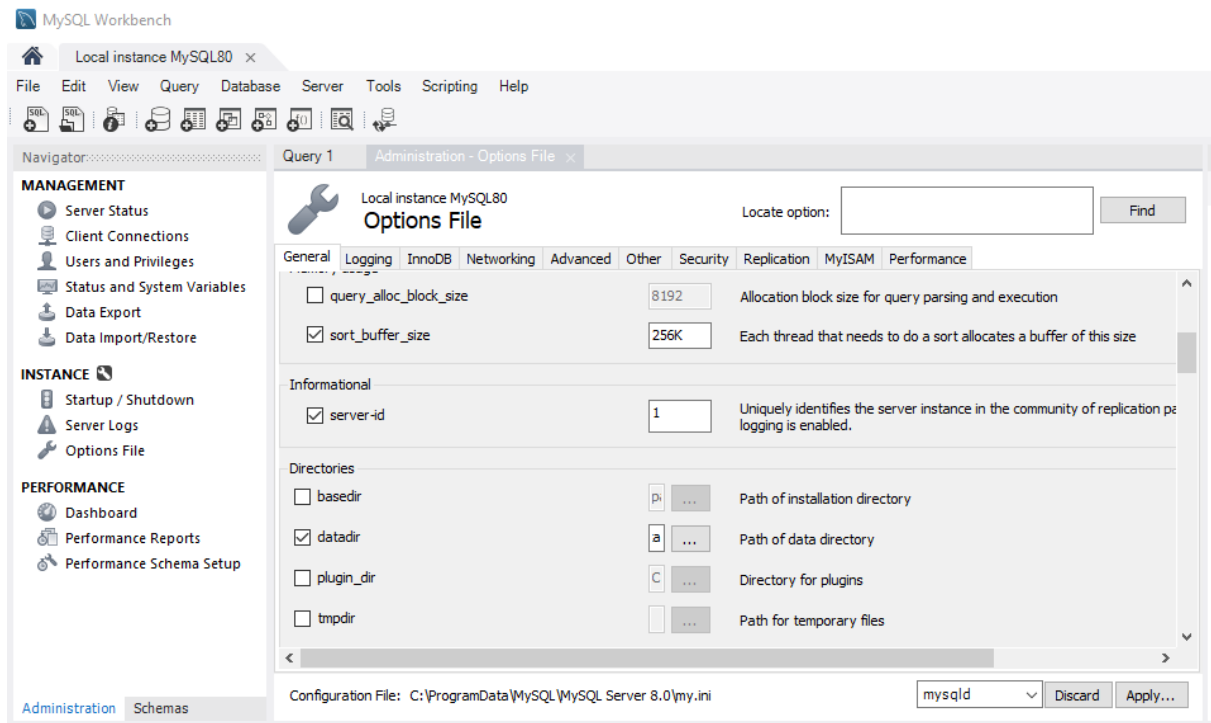
1. Select the item "Users and Privileges".
 2. Select the user "root".
 3. In the "Limit to Hosts Matching" field, enter "%".
 4. Save the changes by clicking the "Apply" button.
- Now you can connect to the database from a remote PC.

Changing the database save path

To change the path for saving the database, you should do the following:

- stop the MySQL service via Windows services.
- move the entire "data" directory from the current location (by default "C: \ ProgramData \ MySQL \ MySQL Server 5.x \ data") to a new one (cut - paste).

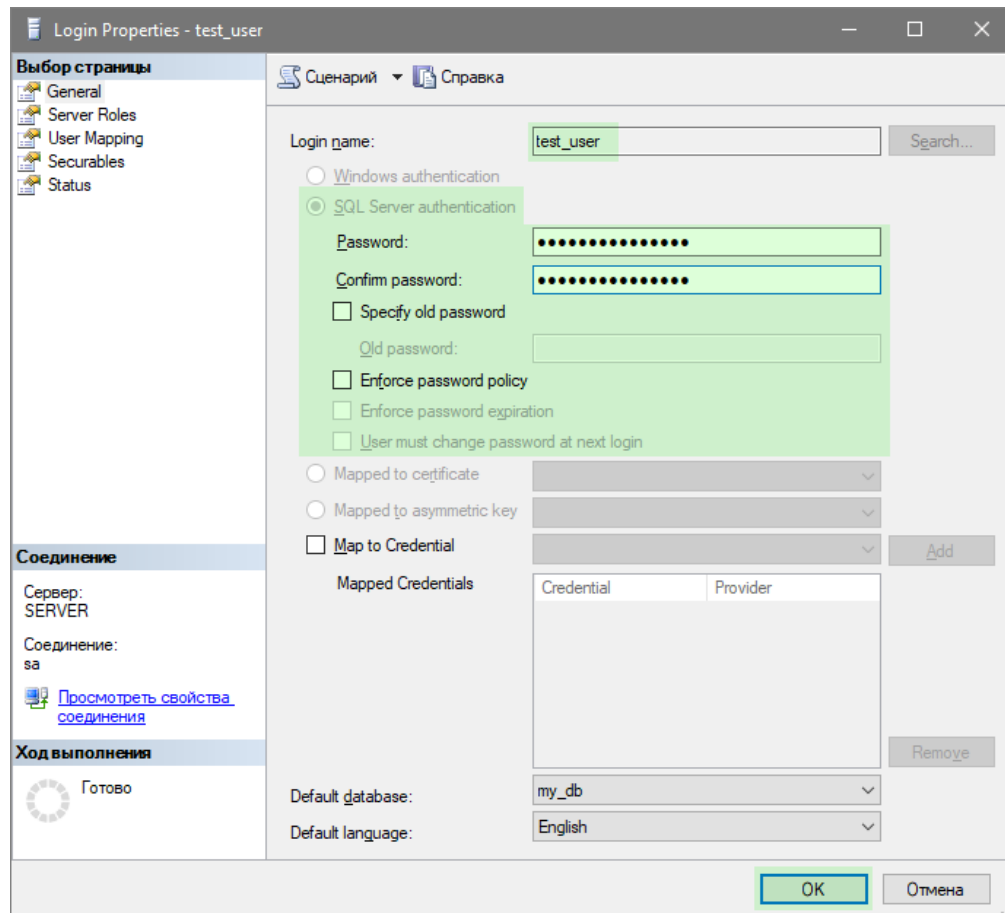
Next, you need to run MySQL Workbench "as administrator" and change the "datadir" parameter to a new location for the data directory:



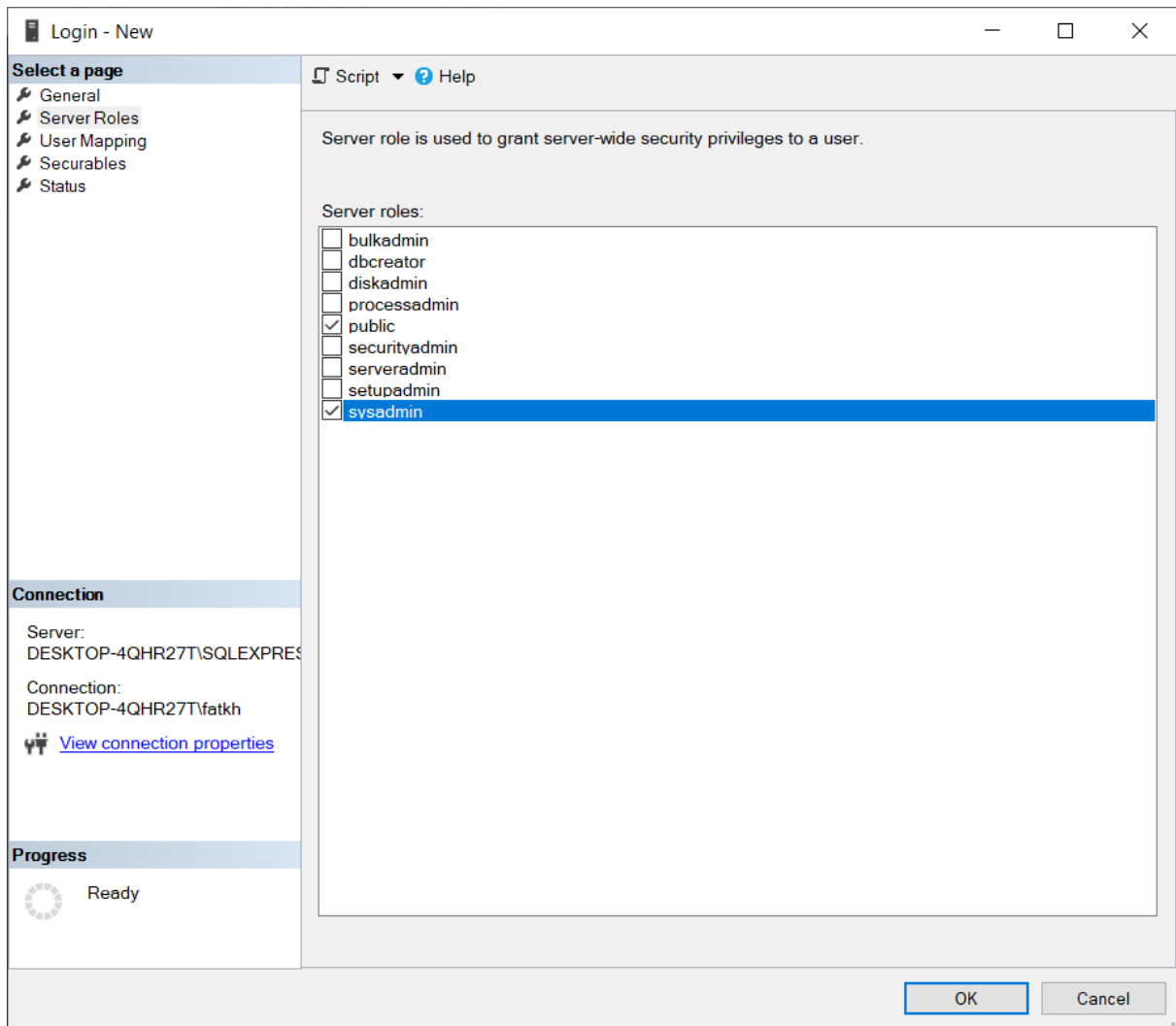
4.3 MSSQL

To work with MSSQL databases you have to install MSSQL server on your PC. How to install MSSQL server you can find in the Internet. For example, how to install Express version you can find [here](#).

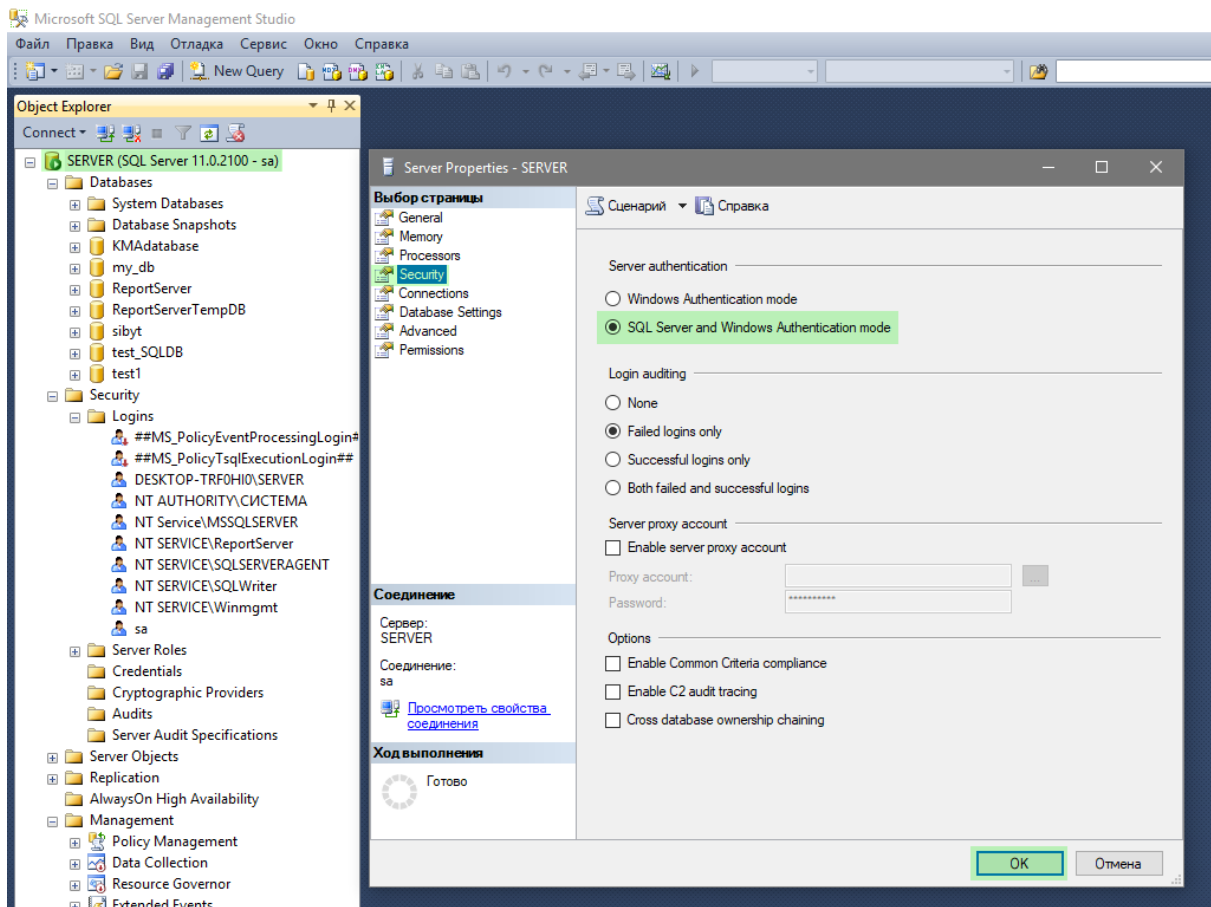
Important! For TCP / IP connection the user should be created in SQL Server with the ability to authorize through the SQL server (not through Windows!):



And you have to provide user possibility to create tables, read, write into database and other function. The most easy way to is to provide sysadmin server role:



And SQL Server itself should allow more than just Windows authentication:



Also don't forget to check your firewall. Port used by MSSQL server (default 1433) should be opened.

4.4 PostgreSQL

To work with PostgreSQL databases you have to install PostgreSQL server on your PC. How to install server you can find in the Internet. For example [here](#). To setup firewall use this command in command line (should be started under Administrator rights):

```
netsh advfirewall firewall add rule name="Postgre Port" dir=in action=allow protocol=TCP localport=5432
```

Settings required for getting exception in English

If you are getting exceptions with not readable symbols (actual for users who doesn't use english language) you have to find file postgresql.conf in the directory where install PostgreSQL server. Find property lc_messages and change it into 'en-En.utf-8'. Save the file and restart postgresql service (you can find it task manager -> Services tab).

Settings required for connecting to a remote database

In order to remotely access a PostgreSQL database, you should set the two main PostgreSQL configuration files:

postgresql.conf

pg_hba.conf

Here is a brief description how you can set them (note that the following description is purely indicative: To configure a machine safely, you should be familiar with all the parameters and their meanings). First of all, configure PostgreSQL service to listen on port 5432 on all network interfaces in Windows machine:

open the file postgresql.conf (usually located in C:\Program Files\PostgreSQL\{your version}\data) and sets the parameter

listen_addresses = '*' (if it didn't setup)

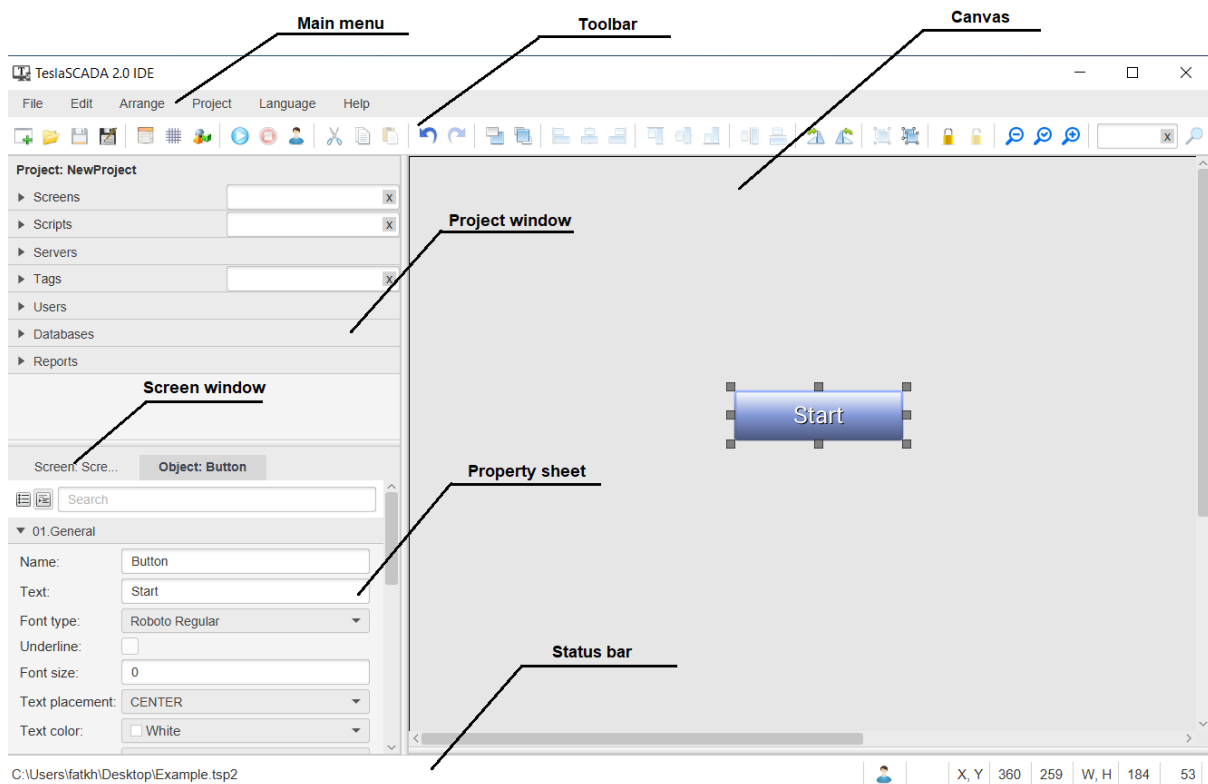
Open and add in the pg_hba.conf file:

host all all 0.0.0.0/0 md5

Save the files and restart postgresql service (you can find it in task manager ->Services tab).

5 Start TeslaSCADA IDE

After opening the application you will see the start screen. Look at the picture below to brief?y get to know the TeslaSCADA IDE interface:



There are several zones:

- [Main menu](#) ^[61]
- [Toolbar](#) ^[69]
- [Project window](#) ^[72]
- [Canvas](#) ^[91]
- [Property sheet](#) ^[91]
- [Screen window](#) ^[92]
- [Status bar](#) ^[94]
- **Debug window** - you can slide up debug window between **Status bar** and **Canvas** to monitor ST scripts messages by print function.



Lesson 3. SCADA for beginners. Start Tesla...

Смотреть Поделись

Start TeslaSCADA IDE




Посмотреть на  YouTube
<https://teslascada.com>


Start TeslaSCADA2 IDE

Lesson 2. SCADA for beginners. Quick Start.

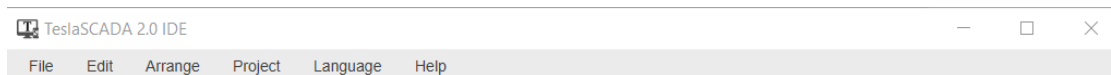
Quick Start tutorial
Create simple project
in TeslaSCADA

Посмотреть на  <https://teslascada.com>

Quick Start TeslaSCADA2



5.1 Main menu



File^[62] - manipulation with project files.

Edit^[63] - manipulation with objects (cut, copy, paste and etc.).

Arrange^[64] - arrange manipulation with objects (align, rotate and etc.).

Project^[67] - possibility to create new objects of the project, change its properties and run/stop simulation. Also in this menu you can login (change operator) and make screenshots.

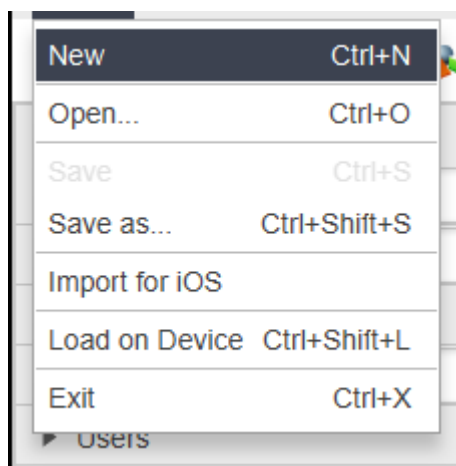
Language - possibility to change language of the interface.

Help - opens the help menu.



Main menu

5.1.1 File



New - [create a new project](#)^[98].

Open - open existing project.

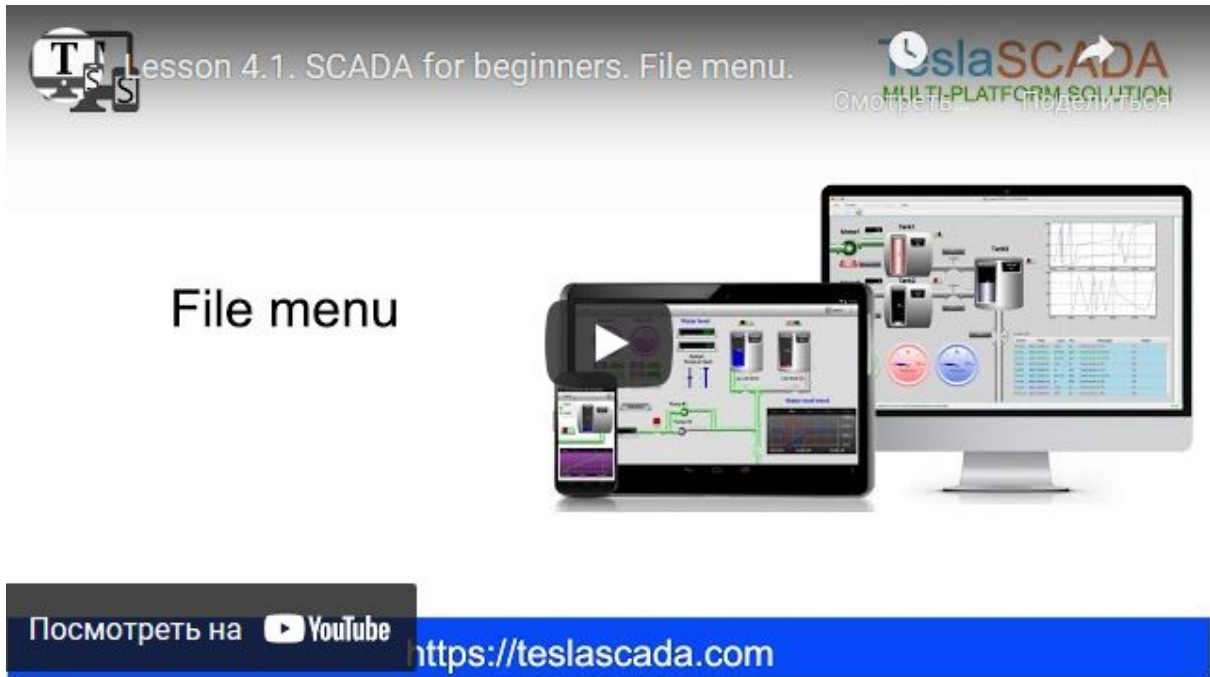
Save - save project under the current name.

Save as... - save project under a new name.

Import for iOS^[527] - import project in iOS format. For iOS devices, a different format is used than the format used for desktop and Android devices.

Load on Device^[525] - call dialog box for uploading current project on remote desktop or Android device.

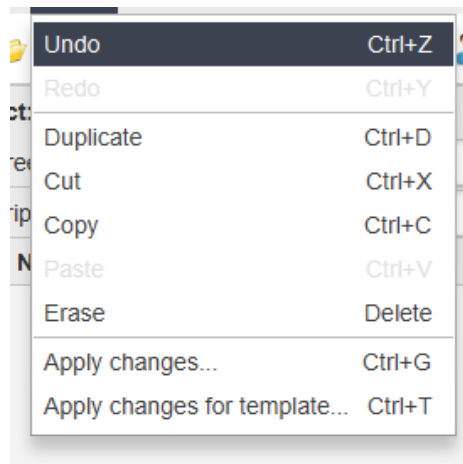
Exit - exit application.



File menu

File menu

5.1.2 Edit



Undo - undo the last action.

Redo - redo the last action.

Duplicate - duplicate selected graphical object(s).

Cut - cut selected graphical object(s).

Copy - copy selected graphical object(s).

Paste - paste selected graphical object(s).

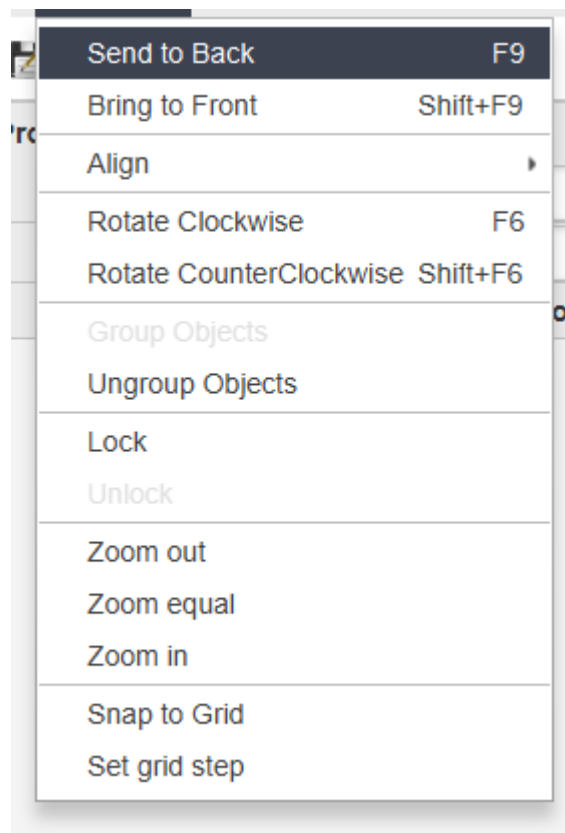
Erase - erase selected graphical object(s).

Apply changes... - apply changes of the selected object for all objects with the same name. In the window that appears, you must choose to replace tags or not.

Apply changes for template... - apply changes of the selected group object for all objects with the same template name. In the window that appears, you must choose to replace tags or not.


*Edit menu*

5.1.3 Arrange



Send to Back - send to back of the screen selected object.

Bring to Front - bring to front of the screen selected object.

Align  - align selected objects.

Rotate Clockwise - rotate clockwise selected object(s). To current rotation angle 90 degrees will be added.

Rotate CounterClockwise - rotate counter clockwise selected object(s). From current rotation angle 90 degrees will be subtracted.

Group Objects - group selected graphical objects.

Ungroup Objects - ungroup selected group of graphical objects.

Lock - lock selected object. You'll not be able to move this object after lock it.

Unlock - unlock selected object. You'll be able to move this object after unlock it.

Zoom out - zoom out screen.

Zoom equal - return to initial screen scale.

Zoom in - zoom in screen.

Snap to Grid - enable/disable the display of the grid on the drawing area.

Set grid step - setup the size of grid cells.

Lesson 4.3. SCADA for beginners. Arrange...

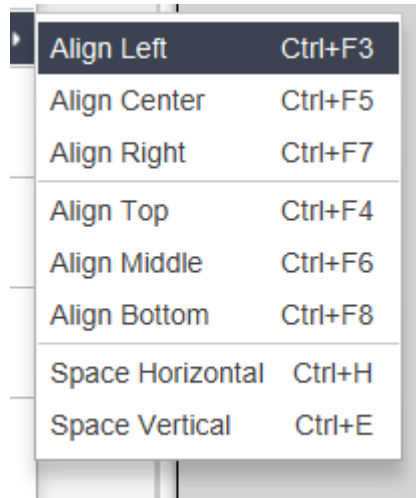
Arrange menu



Посмотреть на  YouTube <https://teslascada.com>

Arrange menu

5.1.3.1 Align



Align Left - align the selected graphical objects to the left.

Align Center - center selected graphical objects horizontally.

Align Right - align the selected graphical objects to the right.

Align Top - align the selected graphical objects to the top.

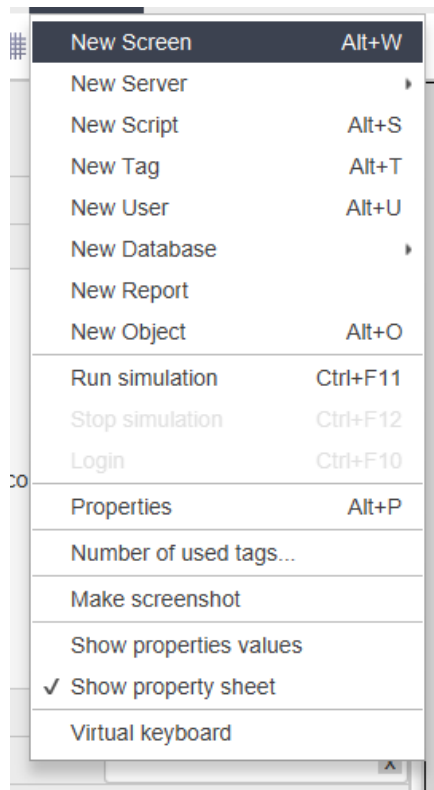
Align Middle - center selected graphical objects vertically.

Align Bottom - align the selected graphical objects to the bottom.

Space Horizontal - distribute the selected objects evenly horizontally.

Space Vertical - distribute the selected graphical objects evenly vertically.

5.1.4 Project



[New Screen](#) ¹³³ - create new screen in the project.

[New Server](#) ⁶⁸ - create new server in the project.

[New Script](#) ³⁹³ - create new script in the project.

[New Tag](#) ⁴⁶¹ - create new tag in the project.

[New User](#) ⁴⁸⁰ - create new user in the project.

[New Database](#) ⁶⁹ - create new database in the project.

New Report - create new report in the project.

[New Object](#) ¹³⁶ - add new graphical object in the project.

Run simulation - run simulation of the project.

Stop simulation - stop simulation of the project.

Login - logout and login new user.

[Properties](#) ⁹⁸ - open project properties window.

Number of used tags... - call dialog box with information about number of tags used in the project. It's useful if you want to check license you need if you want to buy tags dependent license.

Make screenshot - make screen shot of the project.

Show properties values - check this menu item if you want to monitor properties values by placing the mouse cursor over the graphical object during run simulation.

Show property sheet - check if you to edit properties of the graphical object in property sheet or uncheck if you want edit properties only in graphical object dialog boxes.

[Virtual keyboard](#) ⁹⁸ - check if you want to use virtual keyboard. It's useful if you want to use your project on sensor panel.

Lesson 4.4. SCADA for beginners. Project m... **TeslaSCADA**
MULTI-PLATFORM SOLUTION

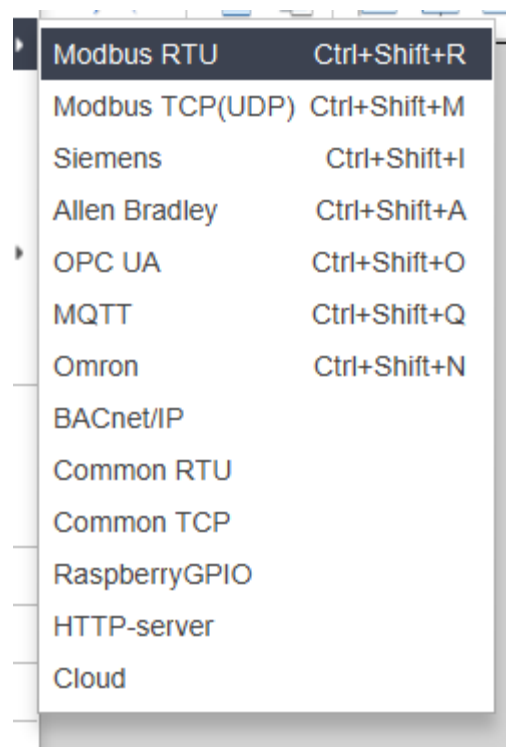
Project menu

Посмотреть на YouTube <https://teslascada.com>



Project menu

5.1.4.1 New server



Modbus RTU ³⁷⁷ - create new Modbus RTU server and open window to edit its properties.

[Modbus TCP\(UDP\)](#)^[379] - create new Modbus TCP(UDP) server and open window to edit its properties.

[Siemens](#)^[381] - create new Siemens server and open window to edit its properties.

[Allen Bradley](#)^[382] - create new Allen Bradley server and open window to edit its properties.

[OPC UA](#)^[383] - create new OPC UA server and open window to edit its properties.

[MQTT](#)^[385] - create new MQTT server and open window to edit its properties.

[Omron](#)^[386] - create new Omron server and open window to edit its properties.

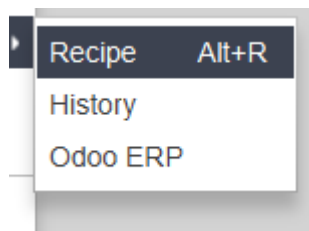
[BACnet/IP](#)^[388] - create new BACnet/IP server and open window to edit its properties.

[Raspberry GPIO](#)^[391] - create new Raspberry GPIO server and open window to edit its properties.

[HTTP-server](#)^[391] - create new HTTP server and open window to edit its properties.

[Cloud](#)^[392] - create new Cloud client and open window to edit its properties.

5.1.4.2 New Database



[Recipe](#)^[483] - create database for recipe and open window to edit its properties.

[History](#)^[485] - create database for history and open window to edit its properties.


















[Odoo ERP](#)^[488] - create object to work Odoo ERP and open window to edit its properties.















5.2 Toolbar



The toolbar consists of the following functions:

	New project	Creates a new project.
	Open project	Opens an existing project.
	Save	Saves your project.
	Save as	Saves your project with a new name.

	Properties	Properties of your project.
	Snap to Grid	ON/OFF snap to grid.
	New object	Creates a new graphical object.
	Run simulation	Start simulation of your project.
	Stop simulation	Stop simulation of your project.
	Login	Change (logout/login) user.
	Cut	Cut selected object(s).
	Copy	Copy selected object(s).
	Paste	Paste selected object(s).
	Undo	Undo the last operation.
	Redo	Redo the last operation.
	Send to Back	Send to back selected object.
	Bring to Front	Bring to front selected object.
	Align Left	Align to the left side the selected objects.
	Align Center	Align the vertical center of the selected objects.
	Align Right	Align to the right side the selected objects.
	Align Top	Align on top of the selected objects.

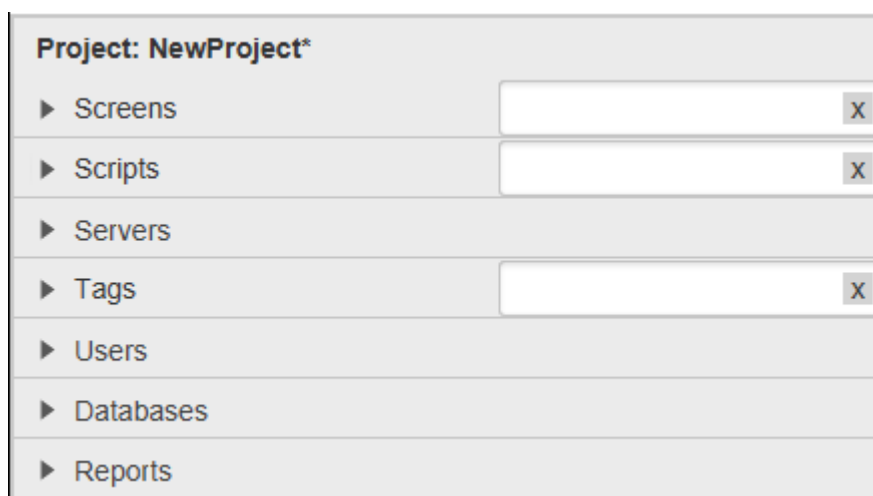
	Align Middle	Align the horizontal center of the selected objects.
	Align Bottom	Align to the bottom of the selected objects.
	Space Horizontal	Align the horizontal spacing between the selected objects.
	Space Vertical	Align the vertical spacing between the selected objects.
	Rotate Clockwise	Rotate clockwise selected object(s).
	Rotate Counter Clockwise	Rotate counterclockwise selected object(s).
	Group Objects	Group selected objects.
	Ungroup Objects	Ungroup selected objects.
	Lock Object	Lock object to the position
	Unlock Object	Unlock object from the position.
	Zoom Out	Zoom out screen with all objects.
	Zoom Equal	Zoom screen with all objects to original sizes.
	Zoom In	Zoom in screen with all objects.
	Find	Find graphical object. Name you enter in the field.



Toolbar

Toolbar

5.3 Project window



Project window contains:

- **Project name.** You can change project name in the Project properties. If a "*" is displayed next to the project name, then changes have been made to the project since the last save.
- Tab **Screens**¹⁰⁶. This tab contains all screens used in the project.
- Tab **Scripts**⁷⁶. This tab contains all scripts used in the project.
- Tab **Servers**⁷⁹. This tab contains all servers used in the project. Server refers to all devices and servers to which you are connecting.
- Tab **Tags**⁸¹. This tab contains all tags used in the project.
- Tab **Users**⁸⁶. This tab contains all users used in the project.

- Tab **Databases**^[88]. This tab contains all databases used in the project.

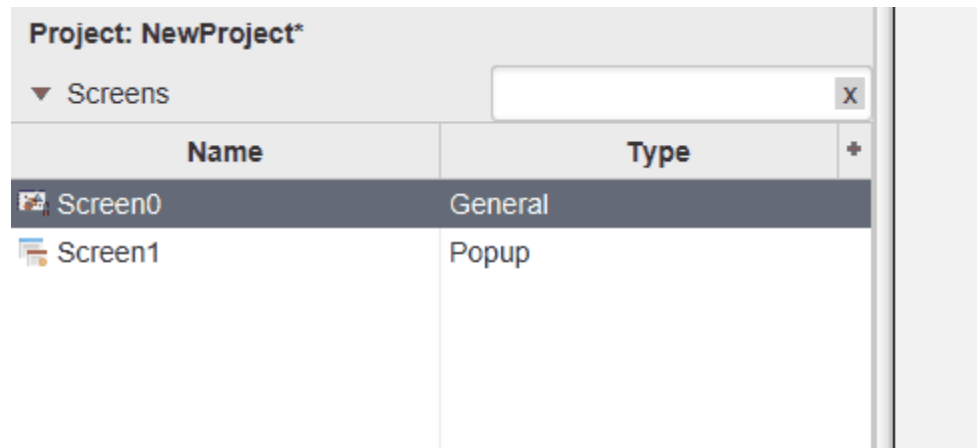


5.3.1 Screens

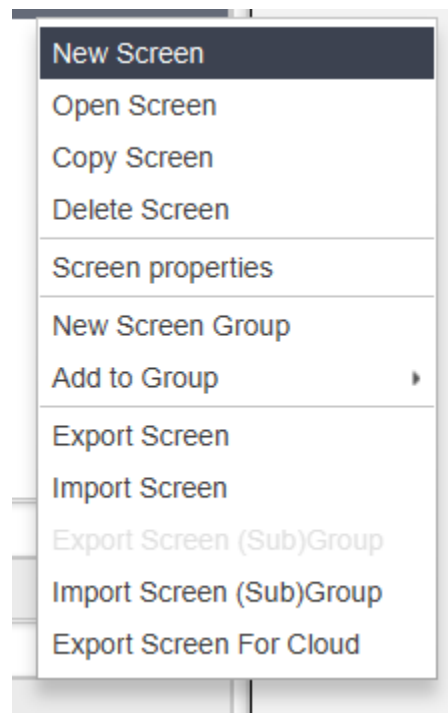
The screens are shown in the table. The first column contains the screen name, the second column contains the screen type - General or Popup:

Screens		X
Name	Type	+
LivingRoom	General	^
Events	General	
Setup	Popup	
Sensors	General	
Trends	General	
Modbus	General	
Siemens	General	v

You can hide or show columns by clicking "+" button:



By clicking right button on the screen you can call context menu:

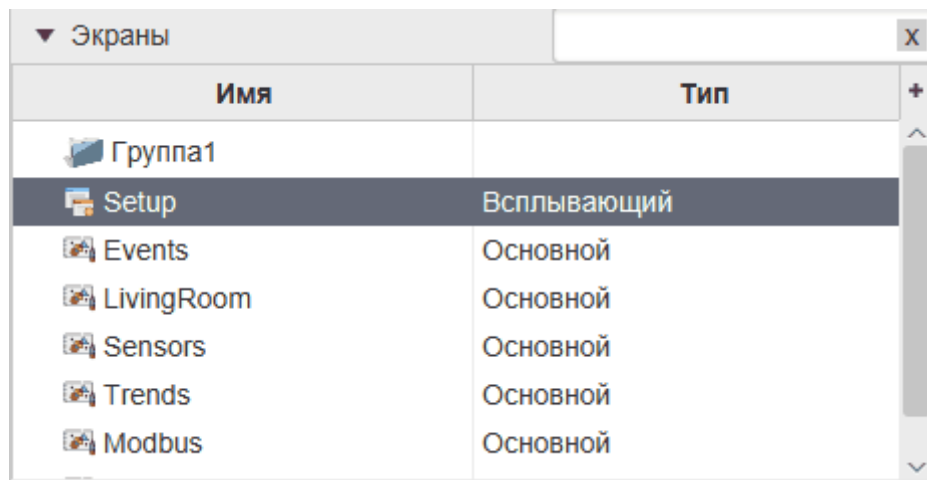


List of menu items with their functions:

- **New Screen** - create a new screen. You can also create a new screen in the main menu [Project](#)->New Screen. Then dialog window will be opened for editing screen properties.
- **Open Screen** - open the screen. It is opened for design purposes. You can also open the screen by double clicking on it.
- **Copy Screen** - copy the screen. It is copied with all graphical objects on it.
- **Delete Screen** - delete the screen. It is deleted from the project.
- **Screen properties** - open window for editing screen properties.

- **New Screen Group** - create a new screen group. It's useful to simplify the structure of the project.
- **New Screen Subgroup** - this menu item is shown when you right click on the screen group. It creates new screen's sub group. It's also useful to simplify of the project structure.
- **Add to Group** - add this screen to the screen group or subgroup from the list.
- **Export Screen** - export this screen for another project. File is saved with .tsp2screen extension.
- **Import Screen** - import the screen from the file with .tsp2screen extension.
- **Export Screen (Sub)Group** - export all screens of the group or(and) subgroups including global images of these screens. File is saved with .tsp2groupscreen extension.
- **Import Screen (Sub)Group** - import screens of the group or(and) subgroups, including global images of these screens, from the file with .tsp2groupscreen extension.
- **Export Screen for Cloud** - export screen for cloud. File is saved with .tsp2json extension. You can upload this file on ESP device and use it for WEB interface if you want.

You can also manage screens in screen groups and subgroups by using drag and drop technology:



Имя	Тип
Группа1	
Setup	Всплывающий
Events	Основной
LivingRoom	Основной
Sensors	Основной
Trends	Основной
Modbus	Основной

Lesson 6.1. SCADA for beginners. Screens...

Смотреть

Поделиться

Screens window



Посмотреть на YouTube

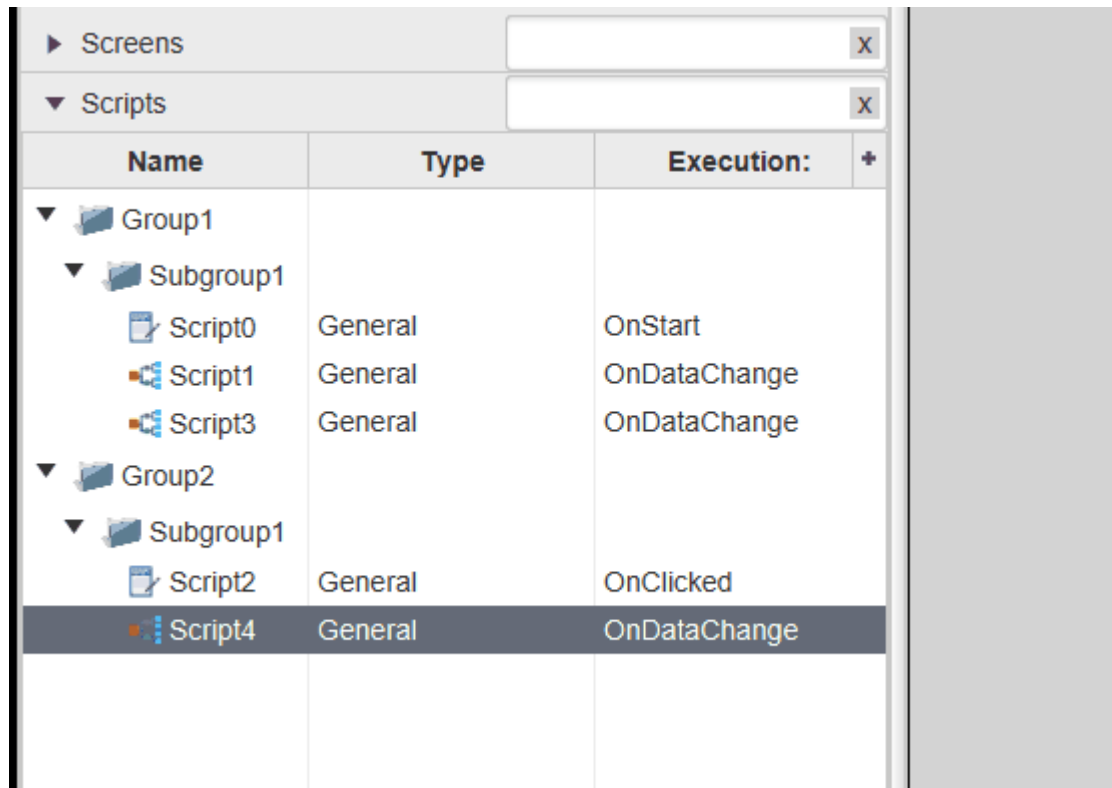
<https://teslascada.com>

5.3.2 Scripts

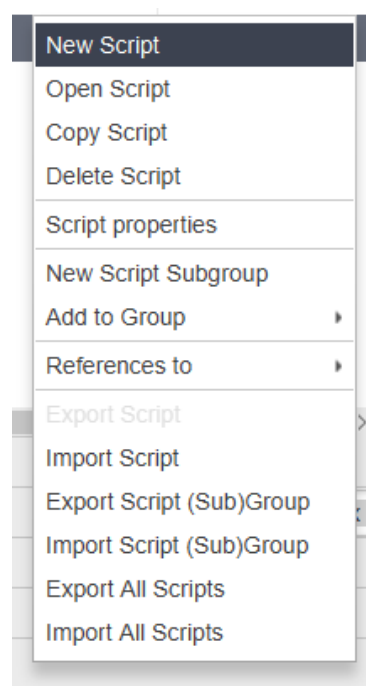
The scripts are shown in the table. The first column contains the script name, the second column contains the script type - General, Screen, Tag or Object, the third column contains the execution type - onDataChange, onStart, onClick and others:

▼ Scripts			X
Name	Type	Execution:	+
▼ Group1			
▼ Subgroup1			
Script0	General	OnStart	
Script1	General	OnDataChange	
▼ Group2			
▼ Subgroup1			
Script2	General	OnClicked	



You can hide or show columns by clicking "+" button:



By clicking right button on the script you can call context menu:



List of menu items with their functions:

- **New Script**  - create a new script. You can also create a new script in the main menu **Project**  -> New script. Then dialog window will be opened for editing script properties.
- **Open Script** - open the script. It is opened for develop script command purposes. If script use FBD language design screen for FBD objects will be opened. If script use ST language code area will be opened. You can also open the script by double clicking on it.
- **Copy Script** - copy the script. It is copied with all FBD objects if you use FBD language or with all code if you use ST language.
- **Delete Script** - delete the script. It is deleted from the project.
- **Script properties** - open dialog window for editing script properties.
- **New Script Group** - create a new script group. It's useful to simplify the structure of the project.
- **New Script Subgroup** - this menu item is shown when your right click on the script group. It creates new script's sub group. It's also useful to simplify of the project structure.
- **Add to Group** - add this script to the script group or subgroup from the list.
- **Reference to** - help to find the script in the project. General and Screen types of the script will be searched in the screen, Tag type in tags and Object type in objects.
- **Export Script** - export this script for another project. File is saved with .tsp2script extension.
- **Import Script** - import the script from the file with .tsp2script extension.
- **Export Script (Sub)Group** - export scripts from the group or(and) subgroup. File is saved with .tsp2groupscripts extension.
- **Import Script (Sub)Group** - import scripts with the group or(and) subgroup from the file with .tsp2groupscripts extension.
- **Export All Scripts** - export all scripts for another project. File is saved with .tsp2allscripts extension.
- **Import All Scripts** - import all scripts from the file with .tsp2allscripts extension.

You can also manage scripts in script groups and subgroups by using drag and drop technology:

► Screens		X
▼ Scripts		X
Name	Type	Execution:
▼ Group1		
▼ Subgroup1		
Script0	General	OnStart
Script1	General	OnDataChange
▼ Group2		
▼ Subgroup1		
Script2	General	OnClicked
Script3	General	OnDataChange
Script4	General	OnDataChange

Lesson 6.2. SCADA for beginners. Scripts wi... **TeslaSCADA**
MULTI-PLATFORM SOLUTION

Смотреть на YouTube

Scripts window



<https://teslascada.com>

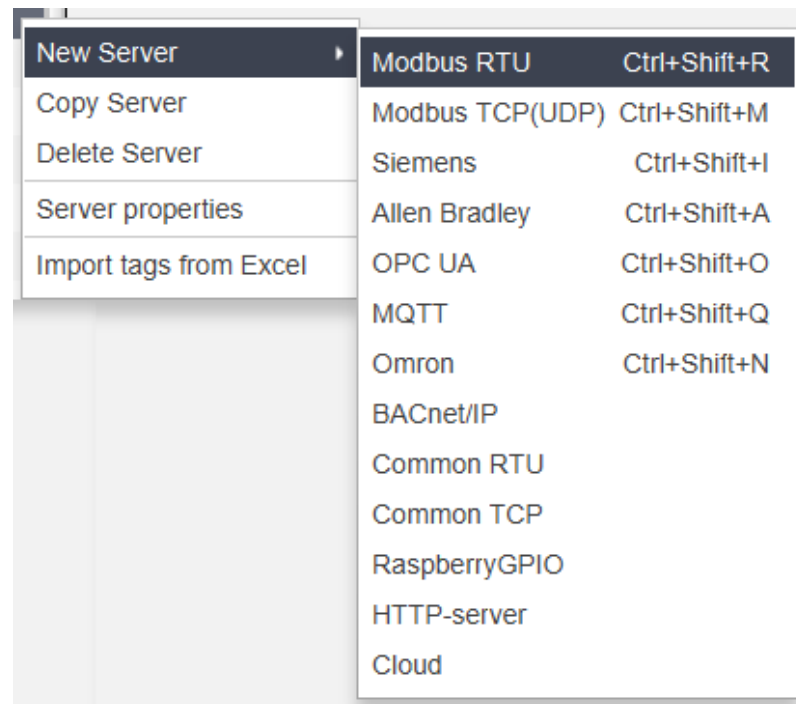
Scripts window

5.3.3 Servers

The servers are shown in the list. The list contains names of the servers used in the project:



By clicking right button on the server you can call context menu:



List of menu items with their functions:

- **New Server** - create a new server. You can also create a new server in the main menu [Project](#)^[67] -> New Server. Choose [server](#)^[68] you want to add. Then dialog window will be opened for editing server properties.
- **Copy Server** - copy the server.
- **Delete Server** - delete the server. It is deleted from the project.
- **Server properties** - open window for editing server properties. You can also do it by double clicking on the server you want to edit.
- **Import tags from Excel** - possibility to import tags from PLC through Excel files.

Lesson 6.3. SCADA for beginners. Servers w...
TeslaSCADA
MULTI-PLATFORM SOLUTION
Смотреть на YouTube

Servers window



Посмотреть на YouTube <https://teslascada.com>

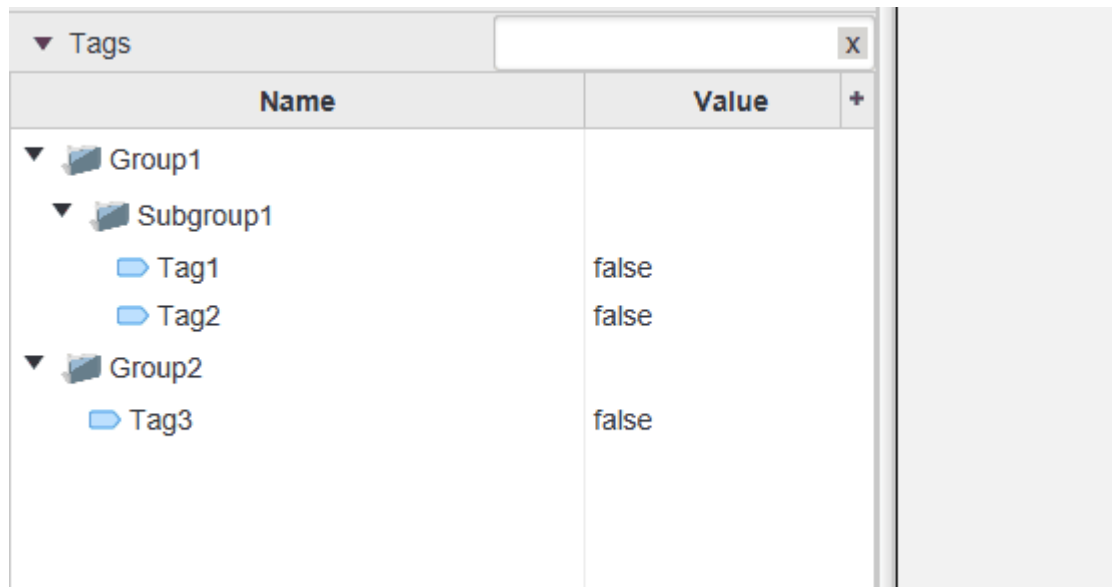
Servers window

5.3.4 Tags

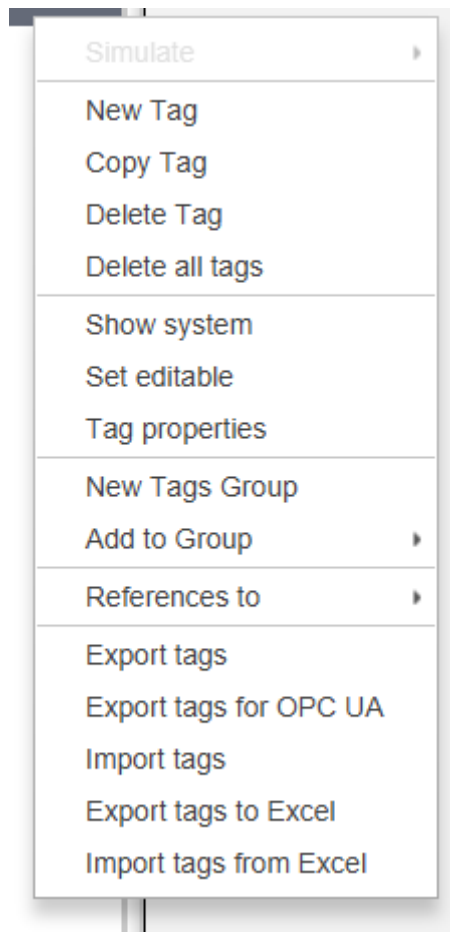
The tags are shown in the table. The first column contains the tag name, the second column contains the tag's value, the third column contains the tag's input source (pointer in string format and it depends on server). You can use filter field to find tag you want by entering its name:

Tags		
Name	Value	+
▼ Group1		
▼ Subgroup1		
Tag1	false	
Tag2	false	
▼ Group2		
Tag3	false	



You can hide or show columns by clicking "+" button:



By clicking right button on the tag you can call context menu:



List of menu items with their functions:

- **Simulate** - this menu is enable only when you run simulation. By using sub menu items you can **Set value** of the tag, for some tag's types you can use *Random value* for simulation tag's random value, *Ramp value* for simulation value from 0 to 100. By using sub menu *Cancel* you can reset Random value and Ramp value simulation.
- **New Tag**  - create a new tag. You can also create a new tag in the main menu **Project**  -> New Tag. Then dialog window will be opened for editing tag properties.
- **Copy Tag** - copy the tag.
- **Delete Tag** - delete the tag. It is deleted from the project.
- **Delete all tags** - delete all tags from the project.
- **Show system** - check this menu item if you want to show system tags in this window. You can't edit values of this tags, but you can use its in the project.

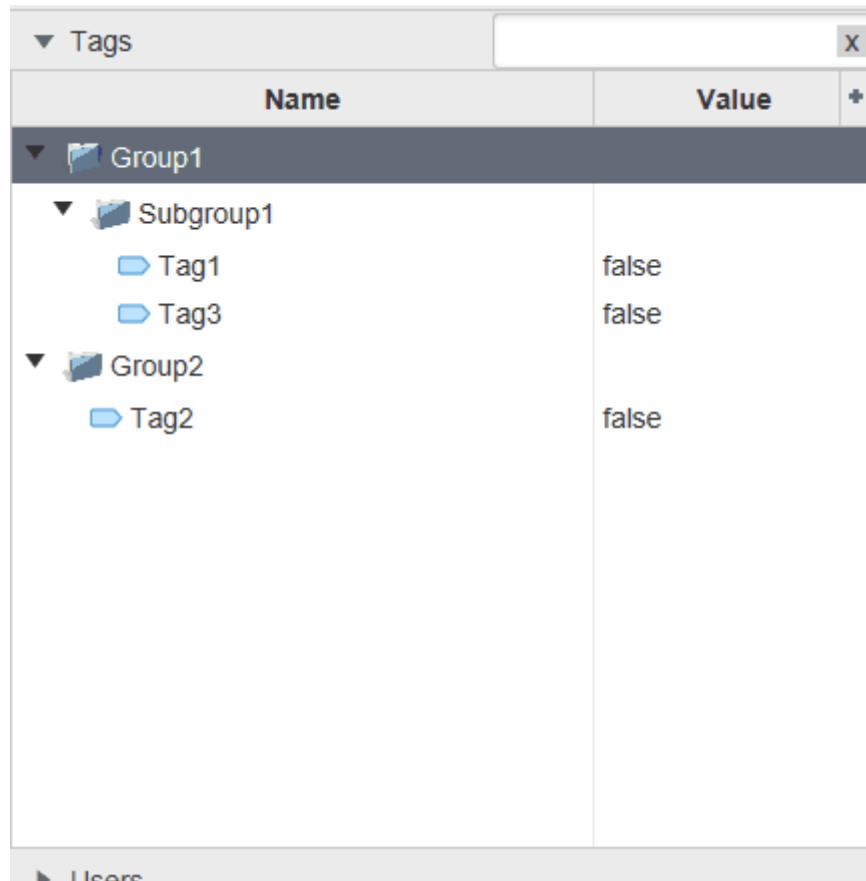
Name	Value	Description
▼ System		
SystemCurrentDateTime		Current date and time
SystemCurrentDateTimeDay		Current day
SystemCurrentDateTimeHour		Current hour
SystemCurrentDateTimeMinute		Current minute
SystemCurrentDateTimeMonth		Current month
SystemCurrentDateTimeNewDay		New day occur
SystemCurrentDateTimeNewHour		New hour occur
SystemCurrentDateTimeNewMinute		New minute occur
SystemCurrentDateTimeSecond		Current second
SystemCurrentDateTimeYear		Current year
SystemCurrentScreen	Screen Main	Current screen name
SystemCurrentUserAccessLevel		Current user access level
SystemCurrentUserAccessLevelBelow500		Current user access level below 500
SystemCurrentUserAccessLevelGreater500		Current user access level greater 500
SystemCurrentUserName		Current user name
SystemPreviousScreen	Screen Contacts	Previous screen name

Table of system tags:

Name	Data type	Description
SystemCurrentDateTime	String	Current date and time in string format (YYYY.MM.DD HH:mm:ss).
SystemCurrentDateTimeDay	Integer	Current day of the month.
SystemCurrentDateTimeNewDay	Boolean	Becomes TRUE from FALSE every day.

Name	Data type	Description
SystemCurrentDateTimeHour	Integer	Current hour in 24 format.
SystemCurrentDateTimeNewHour	Boolean	Becomes TRUE from FALSE every hour.
SystemCurrentDateTimeMinute	Integer	Current minute.
SystemCurrentDateTimeNewMinute	Boolean	Becomes TRUE from FALSE every minute.
SystemCurrentDateTimeMonth	Integer	Current month (01-January, 02-February...).
SystemCurrentDateTimeSecond	Integer	Current second.
SystemCurrentDateTimeYear	Integer	Current year.
SystemCurrentScreen	String	Name of the current opened screen.
SystemCurrentUserAccessLevel	Integer	Current user access level.
SystemCurrentUserAccessLevelBelow500	Boolean	TRUE if current user's access level below 500.
SystemCurrentUserAccessLevelGreater500	Boolean	TRUE if current user's access level greater or equal 500.
SystemCurrentUserName	String	Current user's name.
SystemPreviousScreen	String	Previous opened screen.


- **Set editable** - check this menu item if you want to change name of the tag or its input directly in the table.



- **Tag properties** - open dialog window for editing tag properties. You can also do it by double clicking on the tag you want to edit.
- **New Tags Group** - create a new tag group. It's useful to simplify the structure of the project.
- **New Tags Subgroup** - this menu item is shown when you right click on the tag group. It creates new tag's sub group. It's also useful to simplify the project structure.
- **Add to Group** - add this tag to the tag group or subgroup from the list.
- **Reference to** - help to *find the tag in* the project. You can find in which scripts this tag is used and to which objects this tag is bound.
- **Export tags** - export all tags of the project. File is saved with .tsp2tags extension.
- **Export tags for OPC UA** - export all tags of the project for OPC UA client if you want to use current project in the Client-Server architecture.
- **Import tags** - import all tags from the file with .tsp2tags extension.
- **Export tags to Excel** - export all tags to Excel file. File is saved with .xls extension.
- **Import tags from Excel** - import all tags from the Excel file with .xls extension.


You can also manage tags in tag groups and subgroups by using drag and drop technology:

Tags	
Name	Value
▼ Group1	
▼ Subgroup1	
▼ Group2	
Tag1	false
Tag2	false
Tag3	false

Lesson 6.4. SCADA for beginners. Tags win...  Смотреть Поделись

Tags window



Посмотреть на  <https://teslascada.com>

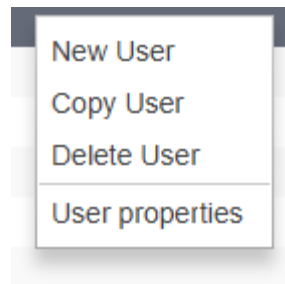
Tags window

5.3.5 Users

The users are shown in the list. The list contains names of the users used in the project:



By clicking right button on the user you can call context menu:



List of menu items with their functions:

- **New User** ⁴⁸⁰ - create a new user. You can also create a new user in the **main menu Project** ⁶⁷ -> **New User**. Then dialog window will be opened for editing user properties.
- **Copy User** - copy the user.
- **Delete User** - delete the user. It is deleted from the project.
- **User properties** - open window for editing user properties. You can also do it by double clicking on the user you want to edit.



Users window

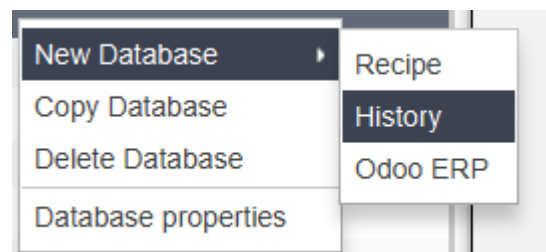
Users window

5.3.6 Databases

The databases are shown in the list. The list contains names of the databases used in the project:



By clicking right button on the database you can call context menu:



List of menu items with their functions:

- **New Database**^[483] - create a new database. You can also create a new database in the main **menu Project**^[67] -> **New Database**. Choose **database**^[69] you want to add. Then dialog window will be opened for editing database properties.

- **Copy Database** - copy the database.
- **Delete Database** - delete the database. It is deleted from the project.
- **Database properties** - open window for editing database properties. You can also do it by double clicking on the database you want to edit.

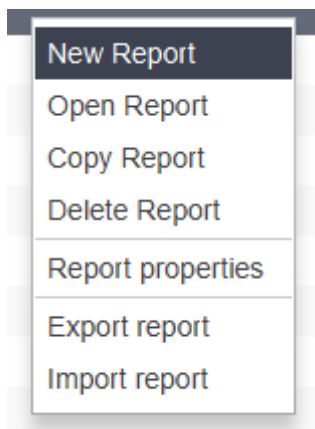


5.3.7 Reports

The reports are shown in the list. The list contains names of the reports used in the project:



By clicking right button on the report you can call context menu:



List of menu items with their functions:

- **New report** - create a new report. You can also create a new report in the **main menu** [Project](#)^[67] -> **New Report**. Then dialog window will be opened for editing report properties.
- **Open Report** - open the report for design properties. You can open the report by double clicking on it also.
- **Copy Report** - copy the report.
- **Delete Report** - delete the report. It is deleted from the project.
- **Report properties** - open window for editing report properties.
- **Export report** - export report.
- **Import report** - import report.

Lesson 6.7. SCADA for beginners. Reports w...  [Смотреть](#) [Поделиться](#)

Reports window



Посмотреть на  <https://teslascada.com>

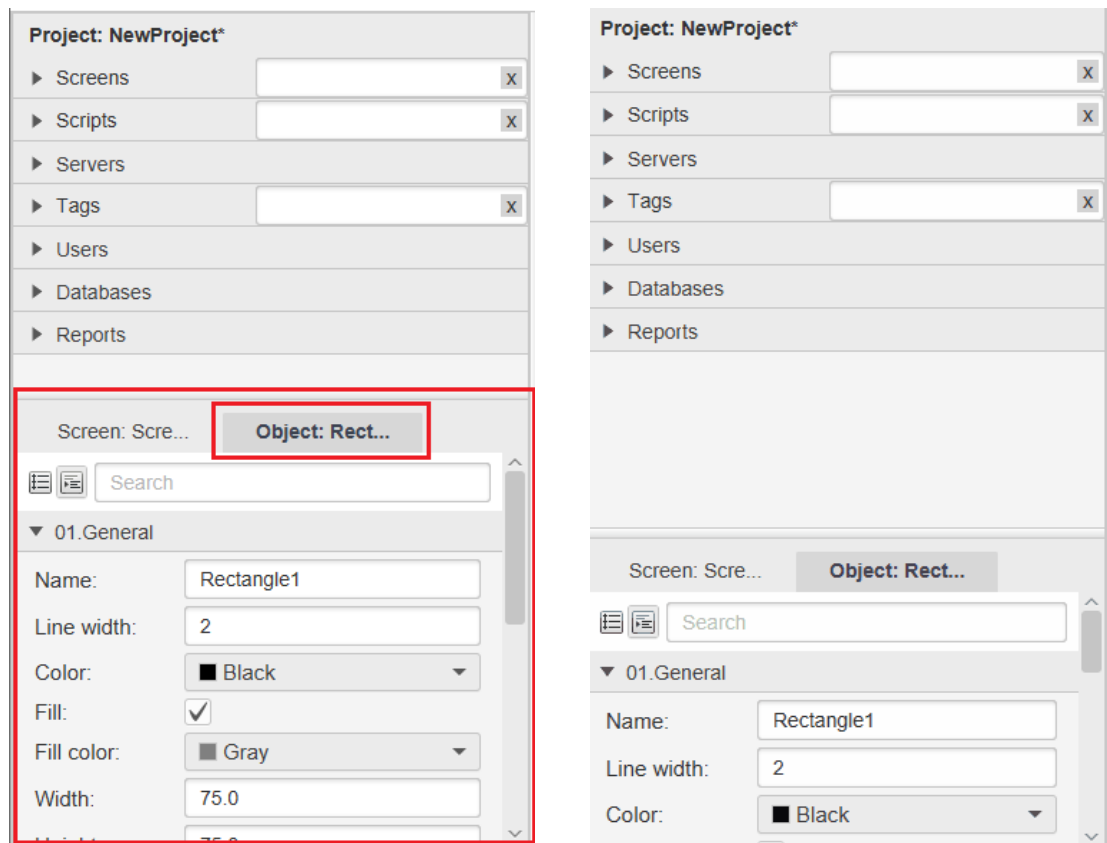
Reports window

5.4 Canvas

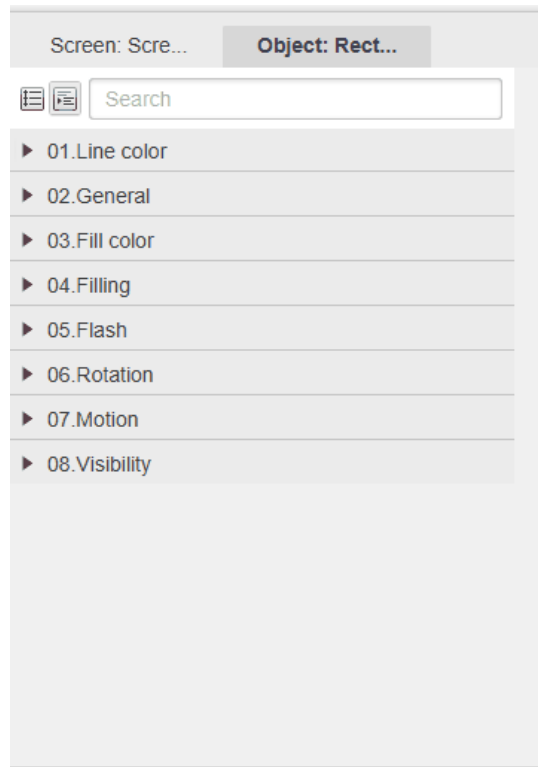
In the canvas the project is created using the graphical objects. For convenient control and navigation in the drawing area, you can use the information in the "[Hot keys and Tips](#)"^[94] section.

5.5 Property Sheet

When you select any object, the property sheet display the properties available for this object. "**Show property sheet**" menu item should be checked in [Project](#)^[67] main menu. You can expand property sheet if you want:



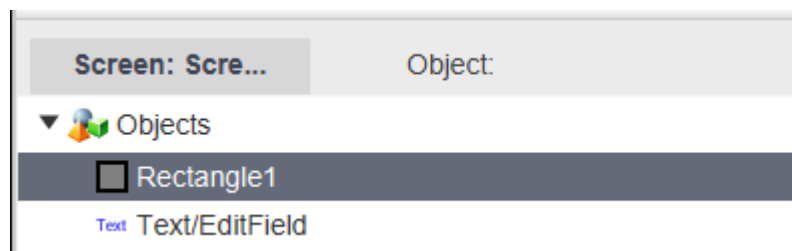
By default only General properties are enabled for new created object. To enable other property groups you have to check enable for them:



It's possible to edit Object's properties not only in Property sheet, but also in Object properties window. To call this window you have to double click by left mouse button on the object you want to edit or click by right button on the Object and choose Object properties menu item.

5.6 Screen window

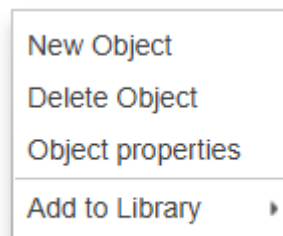
In the same place where the property sheet is located you can find the screen window:



The screen window is useful to find graphical objects that placed on the screen behind other graphical objects and to find and edit properties of the object inside the group object:



By clicking right button on the screen object you can call context menu:



List of menu items with their functions:

- [New object](#)^[136] - create a new graphical object and add it in the project and on [canvas](#)^[91] and screen window.
- **Delete object** - delete selected graphical object from the project.
- **Object properties** - call object properties window for selected object.

- **Add to Library** - add selected object to the library (preliminary you have to create user-defined library in [Add graphical object](#) ¹³⁶ window).

5.7 Status bar

Status bar in all modes contains information about location of the project file in the left. In design mode contains information about coordinate and dimension of the selected object in the right.



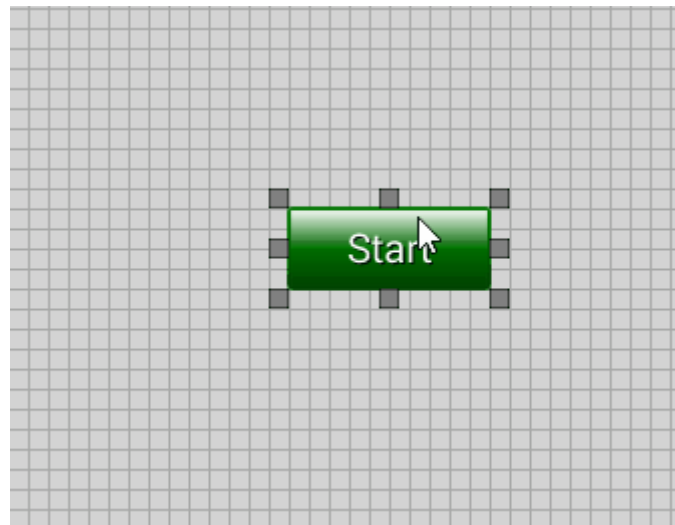
In simulation mode it contains Run label information about simulation mode and information about current user.



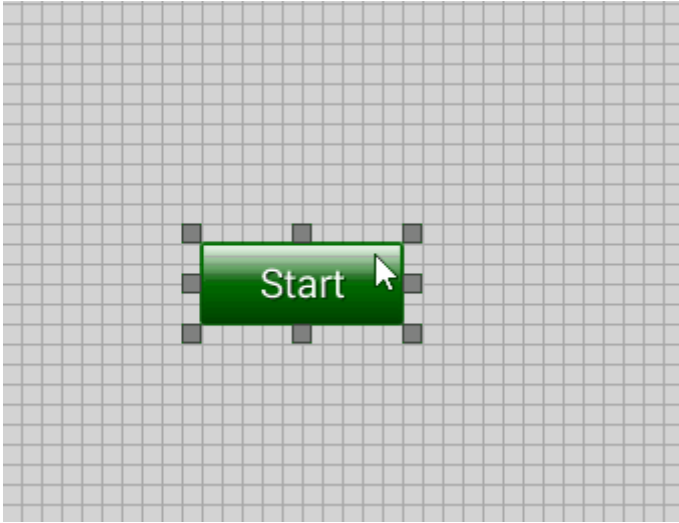
5.8 Hot keys and tips

Editor

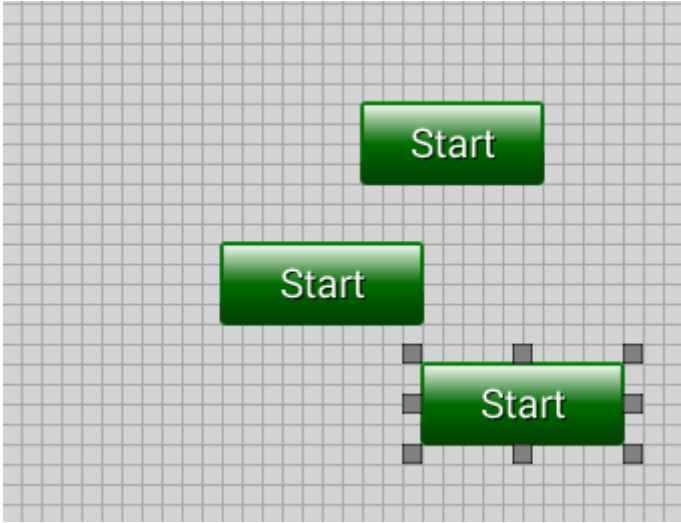
You can move objects by using arrow buttons of the keyboard (UP, DOWN, LEFT and RIGHT):



If CAPS LOCK is ON you can change dimension of the object by using arrow buttons of the keyboard:



You can select multiple graphical objects by holding CTRL keyboard button and clicking left mouse button on the objects:



Keyboard shortcut Windows and Linux	MacOS	Function
CTRL+N	^N	Create a new project.
CTRL+O	^O	Open project.
CTRL+S	^S	Save project.
CTRL+SHIFT+S	^+SHIFT+S	Save project as...
CTRL+SHIFT+L	^+SHIFT+L	Load project on remote desktop or Android device. TeslaSCADA2 Runtime should be started.

Keyboard shortcut Windows and Linux	MacOS	Function
CTRL+Z	^Z	Undo last action.
CTRL+Y	^Y	Redo last action.
CTRL+D	^D	Duplicate selected graphical object(s).
CTRL+X	^X	Cut selected graphical object(s).
CTRL+C	^C	Copy selected graphical object(s).
CTRL+P	^V	Paste selected graphical objects(s).
DEL	Fn+Delete(Bac kspace)	Delete selected graphical object(s).
CTRL+G	^G	Apply changes of the selected object for all objects with the same name.
F9	Fn+F9	Send to back selected graphical object(s).
SHIFT+F9	Fn+SHIFT+F9	Bring to front selected graphical object(s).
CTRL+F3	Fn+^F3	Align the selected objects to the left.
CTRL+F5	Fn+^F5	Center selected graphical objects horizontally.
CTRL+F7	Fn+^F7	Align the selected objects to the right.
CTRL+F4	Fn+^F4	Align the selected graphical objects to the top.
CTRL+F6	Fn+^F6	Center selected graphical objects vertically.
CTRL+F8	Fn+^F8	Align the selected graphical objects to the bottom.
CTRL+H	^H	Distribute the selected objects evenly horizontally.
CTRL+E	^E	Distribute the selected graphical objects evenly vertically.
F6	Fn+F6	Rotate clockwise selected object(s). To current rotation angle 90 degrees will be added.
SHIFT+F6	Fn+^F6	Rotate counter clockwise selected object(s). From current rotation angle 90 degrees will be subtracted.
ALT+W	ALT+W	Create a new screen in the project.
CTRL+SHIFT+R	SHIFT+^R	Create a new Modbus RTU server.
CTRL+SHIFT+M	SHIFT+^M	Create a new Modbus TCP(UDP) server.
CTRL+SHIFT+I	SHIFT+^I	Create a new Siemens server.
CTRL+SHIFT+A	SHIFT+^A	Create a new AllenBradley server.
CTRL+SHIFT+O	SHIFT+^O	Create a new OPC UA server.
CTRL+SHIFT+Q	SHIFT+^Q	Create a new MQTT server.

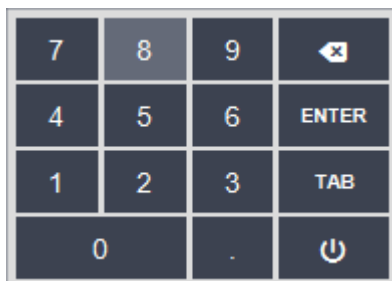
Keyboard shortcut Windows and Linux	MacOS	Function
CTRL+SHIFT+N	SHIFT+^N	Create a new Omron server.
ALT+S	ALT+S	Create a new script in the project.
ALT+T	ALT+T	Create a new tag in the project.
ALT+U	ALT+U	Create a new user in the project.
ALT+R	ALT+R	Create a new recipe in the project.
ALT+O	ALT+O	Add a new graphical object in the project.
CTRL+F11	Fn+^F11	Run simulation of the project.
CTRL+F12	Fn+^F12	Stop simulation of the project.
CTRL+F10	Fn+^F10	Logout and Login new user.
ALT+P	ALT+P	Open project properties window.

ST script editor

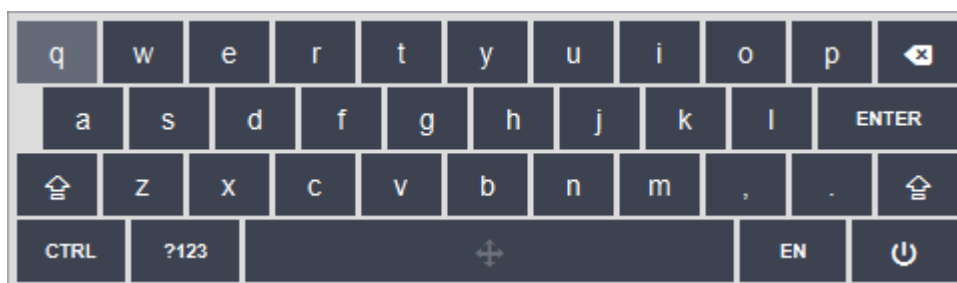
Keyboard shortcut Windows and Linux	MacOS	Function
CTRL+Z	^Z	Undo the last action.
CTRL+SHIFT+Z	SHIFT+^Z	Redo the undone action.
CTRL+SHIFT+>	CMD+SHIFT+>	Select the code to the right of the cursor.
CTRL+SHIFT+<-	CMD+SHIFT+<-	Select the code to the left of the cursor.
CTRL+X	CMD+X	Cut selected code.
CTRL+C	CMD+C	Copy selected code.
CTRL+V	CMD+V	Paste selected code.
CTRL+F11	Fn+^F11	Compile script.
Home/End		Move cursor to start / end of line.
CTRL+Home/ CTRL+End		Move cursor to start / end of script.
Shift + Home / Shift + End		Move cursor to start / end of line with selection.

Sensor screen

By checking menu item [Project](#)^[67] -> **Virtual keyboard** you can enter values on sensor screens. If it is checked, then when you click on an object available for entering numeric values, a numeric keypad will be displayed on the screen:



If you need to enter "-" or "," you have to long touch (or long click) on the virtual button "." and choose symbol you want. When you click on an object available for entering characters, a symbolic keyboard with support for Russian, English and special characters will be displayed on the screen:



6 Project

Create Project

To create a new project TeslaSCADA IDE must be started. Click on the [New](#)^[69] icon in the toolbar or click **menu item New** from the **main menu** [File](#)^[62]. You'll see the [project properties](#)^[100] window:

Edit Project

General

Project name: NewProject

Author: Administrator

Title: TeslaSCADA2 Runtime

Update interval(ms): 1000

☐ Use project protection

Password:

☐ Save tags values

Save DB name: savetagsdb

Max request attempts: 3

☐ Auto logout

Auto logout time (min): 15

Global images: Collection

Main menu: Setup

Description:

OK Cancel

Save project

To save project:

- Click on the **Save** or **Save as...** icon on the toolbar or select the menu item **File** and **Save** or **Save as....**. The first time you save a new project, you will be asked for a location.
- Now select the location and click the button Save (TeslaSCADA project extension .tsp2).

Open project

To open project:

- Click on the **Open** icon on the toolbar or select the menu item **File** and **Open**.
- Now select the project and click Open (TeslaSCADA project extension .tsp2).

Open project properties

To open [project properties](#) ¹⁰⁰:

1. Click on the [Properties](#) ⁷⁰ icon on the toolbar or select the menu item [Project](#) ⁶⁷ and **Properties**.

6.1 Project properties



The screenshot shows the 'Edit Project' dialog box with the 'General' tab selected. The dialog has a sidebar on the left with icons for General, Screens, Events/History, OPC UA, MQTT Publisher, Web-server, HTTP-server, Redundant server, and Cloud. The main area contains the following fields and options:

- Project name: NewProject
- Author: Administrator
- Title: TeslaSCADA2 Runtime
- Update interval(ms): 1000
- ☐ Use project protection
- Password: (empty field)
- ☐ Save tags values
- Save DB name: savetagsdb
- Max request attempts: 3
- ☐ Auto logout
- Auto logout time (min): 15
- Global images: Collection
- Main menu: Setup
- Description: (empty text area)

At the bottom right are 'OK' and 'Cancel' buttons.

Project properties are grouped in several tabs:

- [General](#) ¹⁰¹
- [Screens](#) ¹⁰⁶
- [Events/History](#) ¹⁰⁷
- [OPC UA](#) ¹²²
- [MQTT Publisher](#) ¹²⁵
- [Web-server](#) ¹²⁷
- [HTTP-server](#) ¹²⁷

- [Redundant server](#) 
- [Cloud](#) 

6.1.1 General tab

General tab contains general properties for the project.

Property	Description
Project name	Name of the project.
Author	Author of the project.
Title	Title of the project. We'll be shown instead of TeslaSCADA_Runtime caption.
Update interval	Update interval of the project. It's an interval for updating (redrawing) graphical objects of the current screen. Also with this interval scripts will be executed if "every cycle" is checked for ST script. For scripts with execution type "OnDataChange" scripts will be executed if tag's values is changed, if this tag is used in this script.
Use project protection	If you want to protect your project from opening and editing by non-authorised person check use project protection.
Password	Password for protecting your project.
Save tags values	Check if you want to save all tag's values when you close application and load them when you open your project.
Save DB name	Name of the database where tag's values will be saved.
Max request attempts	Number of maximum server requests before determining that the connection with the server has been lost.
Auto logout	If you want current user auto logout in setup minutes after login you have to check this property.
Auto logout time(min)	Time in minutes before auto logout happens.
Global images	Since 2.46 version all images of the project are stored in one global library. It needs to be beneath size of the project. To edit global images library click Collection button. You'll see the window:

Property	Description
	<div><div><div><div>Collection</div><div><div></div></div><div><div><div>Name:</div><div></div></div><div><div>Image:</div><div></div></div><div><div>Dimension:</div><div></div></div><div><div>Download</div><div>Open</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div><div><p>where:</p><ul style="list-style-type: none">• Name - name of the image.• Image - selected image.• Download - download selected image to disk.• Open - open new image file.• Add - add image to the collection.• Edit - edit image in the collection.• Remove - remove image from the collection.</div></div>
Main menu*	<p>You can use Main menu in your project that helps you to navigate through general screens of the project. Click Setup button to configure main menu. After clicking you'll see the window:</p>

Property	Description
	<div><div><div>Setup</div><div><div><div><input checked="" type="checkbox"/> Enable property</div><div><div>Y position:</div><div>0</div></div><div><div>Height:</div><div>400</div></div><div><div>Width:</div><div>200</div></div><div><div>Color:</div><div>Gray</div></div><div><div>Spacing:</div><div>5</div></div><div><div>Tail:</div><div>5</div></div><div><div>Menu buttons:</div><div>Collection</div></div></div><div><div>OK</div><div>Cancel</div></div></div></div></div> <div>where:</div> <div><ul style="list-style-type: none">• Y position - Main menu is slid from the left. Y position of the menu you setup in this field.• Height - height of the main menu.• Width - width of the main menu.• Color - background color of the main menu.• Spacing - spacing between buttons of the main menu.• Tail - tail of the main menu that appears on the screen.• Menu buttons - collection of the main menu buttons. After clicking button you'll see the window:• </div>

Property	Description
	<div><div><div><div>Collection</div><div><div></div></div><div><div>Screen:</div><div></div></div><div><div>Text:</div><div>Screen</div></div><div><div>Width:</div><div>200</div></div><div><div>Height:</div><div>30</div></div><div><div>Text color:</div><div><div>White</div></div></div><div><div>Fill color:</div><div><div>Gray</div></div></div><div><div>Type:</div><div>2D</div></div><div><div>Font type:</div><div>Roboto Regular</div></div><div><div>Underline:</div><div><input type="checkbox"/></div></div><div><div>Font size:</div><div>0</div></div><div><div>Text placement:</div><div>CENTER</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div> <div><p>ghere:</p><ul style="list-style-type: none">• Screen - screen you want to bind to the button. If you don't want to bind any screen left it empty.<p>Other properties are the same like general properties¹⁸¹ for the Button. Buttons Add, Edit and Remove let you change main menu buttons collection.</p></div>
Description	Optionally, specify a meaningful description of your project.

* Main menu works only on PC versions.

Lesson 7.1. SCADA for beginners. Project pr...
TeslaSCADA
MULTI-PLATFORM SOLUTION
Смотреть... Поделись

Project properties
General tab



Посмотреть на  YouTube <https://teslascada.com>

Project properties. General tab.

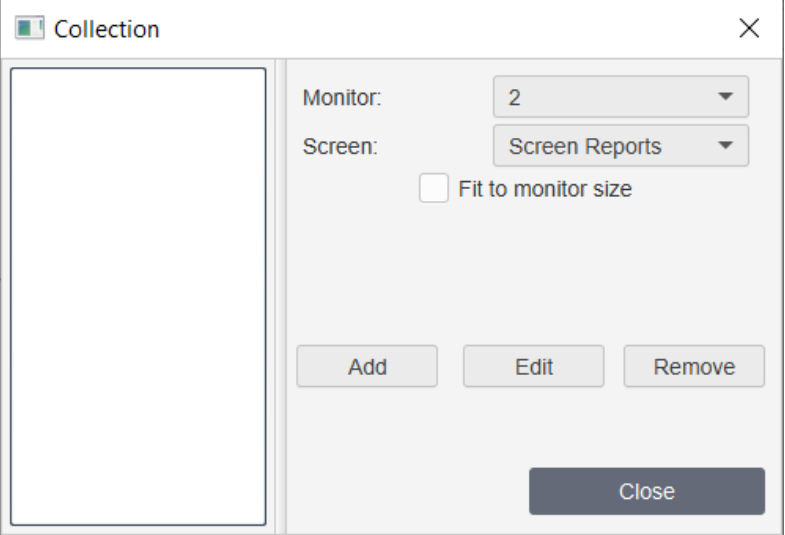
6.1.2 Screens tab

The screenshot shows the 'Edit Project' dialog box with the 'Screens' tab selected. The 'Screens' tab is highlighted with a red rectangle. The dialog contains the following settings:

- Start screen:** A dropdown menu.
- Use multi monitors:** A checkbox.
- Screens:** A button labeled 'Collection'.
- Default font:** A dropdown menu showing 'Roboto-Regular'.
- Screen dimensions:** Two input fields showing '800' and '600' with an 'X' separator.
- Runtime differs:** A checkbox.
- Screen dimensions:** Two input fields showing '800' and '600' with an 'X' separator.

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Property	Description
Start screen	Name of the start screen. When you create a new project the Start screen combobox is empty. You can choose the start screen after creating screens of the project.
Use multi monitors	If you want to use several monitors to display your project screens check this item.
Screens	To edit number of monitors to display screens of your project click button Collection . You'll see:

Property	Description
	 <p>where:</p> <ul style="list-style-type: none"> • Monitor - monitor's number • Screen - start screen for this monitor • Fit to monitor size - check if you want the screen is stretched to monitor's dimension.
Default font	Default font for all texts in the project. System font lets you use Chinese, Arabian and etc language symbols.
Screen dimensions	Default dimensions of your design screen in the screen dimensions fields. These values are also used for scaling your project. Be careful if these dimensions differs from the dimensions of the screens you develop, this may cause your project to display incorrectly on devices with different screen resolutions.
Runtime differs	If the screen dimensions of your target device differs check "runtime differs" and enter its screen dimensions.

6.1.3 Events/History tab

Events/History tab contains properties for general events and history databases, notification rules and sounds, report folder. Also it contains properties for E-mail client used for notifications by [E-mail](#)¹¹¹ and for [Telegram bot](#)¹¹³.

The screenshot shows the 'Edit Project' dialog box with the 'Events/History' tab selected. The left sidebar lists various project components: General, Screens, Events/History (highlighted with a red box), OPC UA, MQTT Publisher, Web-server, HTTP-server, Redundant server, and Cloud. The main area contains settings for the selected tab:

- Storage DB period:** Week
- Archive since:** Never
- Events DB name:** events
- History DB name:** history
- ☐ Use history table for every tag
- Username:** (empty field)
- Password:** (empty field)
- Notifications (priority<=):** 100
- Sounds:** Collection
- ☒ Show servers events
- Report folder:** C:\TeslaSCADA_IDE\app

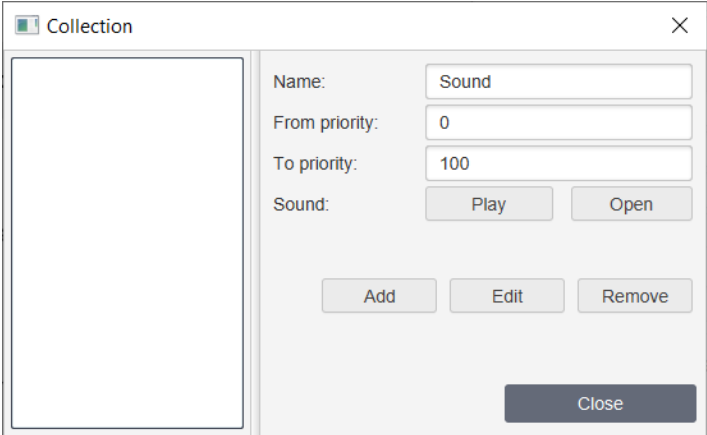
Below these settings are tabs for 'E-mail client', 'Telegram bot', 'Push notifications', and 'GSM modem'. The 'E-mail client' tab is active, showing:

- ☐ Use E-mail client
- Host:** smtp.gmail.com
- Port:** 587
- Type:** TLS
- From E-mail address:** (empty field)
- ☒ Authentication
- Username:** (empty field)
- Password:** (empty field)

At the bottom are 'OK' and 'Cancel' buttons.

Property	Description
Storage DB period	Select the time period during which data will be stored in general event and history databases.
Archive since	Select an archive period. The data collected before the archive period is stored in the archive database. The data collected for the selected period is stored in the main database. This improves performance when querying the underlying database.
Events DB name	<p>The name of the database that stores all information about events during project execution.</p> <ul style="list-style-type: none"> If you choose the simple name like events application will create SQLite database in the application directory¹⁸. If you choose names beginning with jdbc:mysql: like jdbc:mysql://192.168.0.104:3306/test, where test - name of the database, the application will connect to MySQL

Property	Description
	<p>database and create events table. How to install and setup MySQL you can read in MySQL*^[31] chapter.</p> <ul style="list-style-type: none"> • if you choose names beginning with jdbc:sqlserver: like jdbc:sqlserver://192.168.1.17:1433;databaseName=test where test - name of the database you want to connect, the application will connect to MSSQL*^[55] database and create events table. • if you choose names beginning with jdbc:postgresql: like jdbc:postgresql://192.168.1.17:5432/test where test name of the database you want to connect, the application will connect to PostgreSQL*^[58] database and create events table. • if you choose names beginning with jdbc:ucanaccess: like jdbc:ucanaccess:///C:\Users\fatkh\Downloads\events.aacdb where events.aacdb - name of the file you want to collect information, the application will connect to MS Access database and create events table.
History DB name	<p>The name of the database that stores general history information during project execution. It's also possible to store history information in History databases.</p> <ul style="list-style-type: none"> • If you choose the simple name like history application will create SQLite database in the application directory^[18]. • If you choose names beginning with jdbc:mysql: like jdbc:mysql://192.168.0.104:3306/test, where test - name of the database, the application will connect to MySQL database and create history table. How to install and setup MySQL you can read in MySQL*^[31] chapter. • if you choose names beginning with jdbc:sqlserver: like jdbc:sqlserver://192.168.1.17:1433;databaseName=test where test - name of the database you want to connect, the application will connect to MSSQL*^[55] database and create history table. • if you choose names beginning with jdbc:postgresql: like jdbc:postgresql://192.168.1.17:5432/test where test - name of the database you want to connect, the application will connect to PostgreSQL*^[58] database and create history table. • if you choose names beginning with jdbc:ucanaccess: like jdbc:ucanaccess:history where history - name of the file you

Property	Description
	want to collect information, the application will connect to MS Access database and create history table.
Use history table for every tag	If you check this property, for every tag that collects history information table will be created. This is helpful for big project with a lot of history information.
Username	Username for database (except SQLite database)
Password	Password for database (except SQLite database)
Notifications (priority <)	Events with a priority lower than this value will be notified about it by using the dialog box and sound. And also if E-mail client/ Telegram bot/ GSM modem/ Push are setup - by E-mail/Telegram/SMS/Push notifications.
Sounds	<p>Click Collection to set up sounds of events notifications depending on priority. After clicking you'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none"> • Name - name of the sound. • From priority and To priority - priority interval within which sound will play. • Play - play selected sound. • Open - open sound (wav) file. • Add - add sound to the collection. • Edit - edit sound. • Remove - remove sound from the collection.
Show servers events	Check if want to get notifications about disconnection, lost or restore servers. If you uncheck this property you'll not get notifications.

Property	Description
Report folder	The folder which all reports and screenshots will be written to by default

*** for mobile version it is possible to use only SQLite databases.**

6.1.3.1 E-mail client

If you want to notify users by E-mail you have to setup E-mail client:

Property	Description
Use E-mail client	Check if you want to use E-mail notifications about Alarms. All event messages that have priority less than Notifications(Priority<) will be sent by E-mail. You can also use function sendmail in ST script.
Host	E-mail host information.
Port	E-mail port information.
Type	Type of the connection - TLS or SSL.
From E-mail address	Which E-mail address the mail will be sent from
Authentication	Check if you use Username and Password.
Username	Username of the E-mail account.
Password	Password of the E-mail account.

Property	Description
Subject	<p>Subject of the E-mail. If you left it empty default subject will be used (it contains project name, tag name and other information).</p> <p>You can use keywords:</p> <p>{name} - name of the tag that send an alarm message.</p> <p>{server} - name of the PV input server.</p> <p>{message} - message is sent by tag's alarm.</p> <p>{group} - name of the tag's group.</p> <p>{subgroup} - name of tag's subgroup.</p> <p>{description} - tag's description.</p> <p>{value} - tag's value.</p> <p>{priority} - tag's message priority.</p> <p>{datetime} - current date and time (when alarm is happened).</p> <p>{projectname} - project name.</p> <p>{projectdescription} - project description.</p>
Message	<p>Message of the E-mail. If you left empty tag's message will be sent.</p> <p>You can use keywords:</p> <p>{name} - name of the tag that send an alarm message.</p> <p>{server} - name of the PV input server.</p> <p>{message} - message is sent by tag's alarm.</p> <p>{group} - name of the tag's group.</p> <p>{subgroup} - name of tag's subgroup.</p> <p>{description} - tag's description.</p> <p>{value} - tag's value.</p> <p>{priority} - tag's message priority.</p> <p>{datetime} - current date and time (when alarm is happened).</p> <p>{projectname} - project name.</p> <p>{projectdescription} - project description.</p>
To E-mail addresses	Which E-mail addresses the mail will be sent to. Use commas to separate addresses.
Depends on priority	If you want to use E-mail addresses depending on priority check this property and setup E-mail addresses depending on priority values:

Property	Description
	<div><div><div>Collection</div><div><div></div><div><div>Name: Emails</div><div>From: 0</div><div>To: 100</div><div>To E-mail address...</div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div>Close</div></div></div></div></div> <p>where:</p> <ul style="list-style-type: none">• Name - name of the E-mail range.• From - begin priority of the range.• To - end priority of the range.• To E-mail addresses - E-mail addresses separated by commas.

In some accounts, for example? in Gmail you have to make some setups before it would be possible to send E-mails. For Gmail you have to turn on the access for less secure apps, for example and also some other [settings](#).

6.1.3.2 Telegram bot

If you want to notify users by Telegram you have to setup Telegram bot:

E-mail client
Telegram bot
Push notifications
GSM modem

☒ Use Telegram Bot

Bot's name:

Bot's token:

Notifications (priority<=):

Message

OK
Cancel

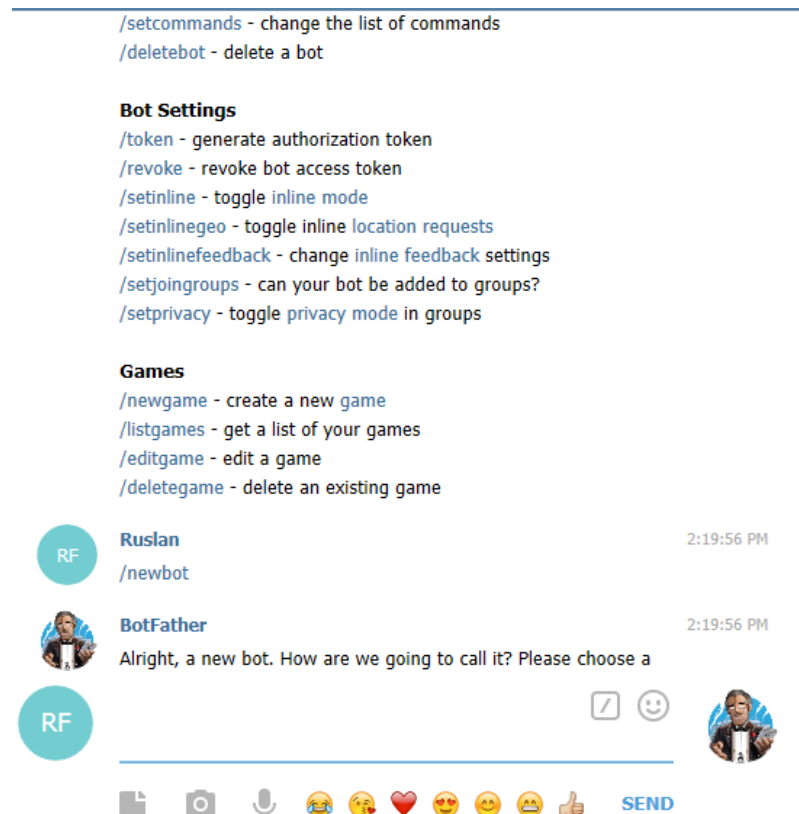
Property	Description
Use Telegram Bot	Check If you want to use Telegram notification in your project. All event messages that have priority less then Noti?cations(Priority<) ^[110] will use Telegram Bot to notify users.
Bot's name	Name of the Telegram bot. You'll get Telegram Bot's name from BotFather when creating your bot.
Bot's token	Token of the Telegram bot. You'll get Telegram Bot's token from BotFather when creating your bot.
Notifications (priority <)	Events with priority lower than this value will be noti?ed about it by using Telegram bot. If the value is less than 0 common Notifications (priority<) ^[110] will be used.
Message	<p>Message that will be sent to telegram bot. If this field is empty only tag message will be sent. If not empty this message will be sent. You can use keywords:</p> <p>{name} - name of the tag that sends an alarm message.</p> <p>{server} - name of the PV input server.</p> <p>{message} - message is sent by tag's alarm.</p> <p>{group} - name of the tag's group.</p> <p>{subgroup} - name of tag's subgroup.</p> <p>{description} - tag's description.</p> <p>{value} - tag's value.</p>

Property	Description
	{priority} - tag's message priority. {datetime} - current date and time (when alarm is happened). {projectname} - project name. {projectdescription} - project description.

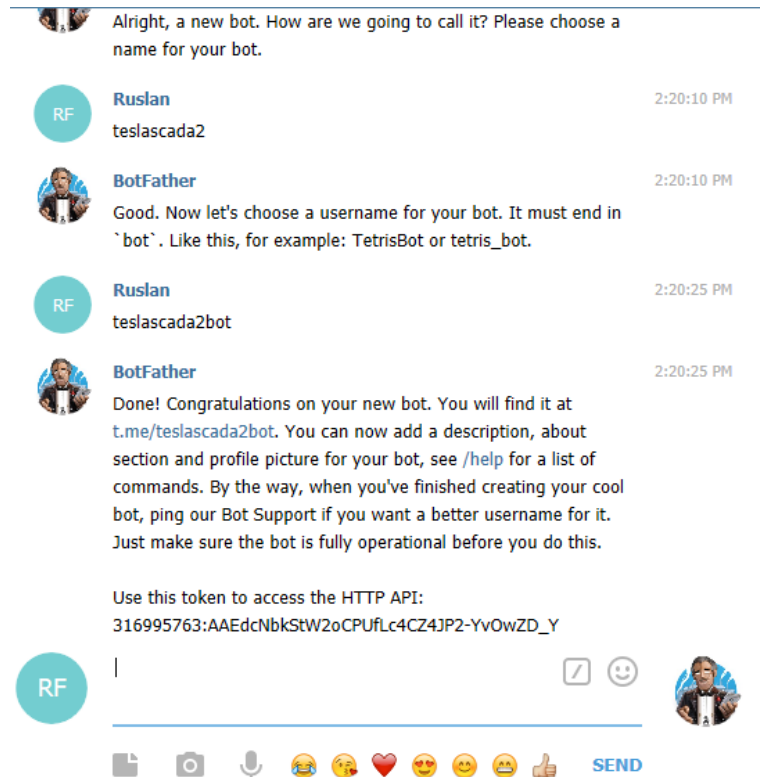
Before using telegram for notifications you have to [create telegram bot](#) ¹¹⁵.

6.1.3.2.1 Create Telegram Bot

If you want to get events notifications from your project in TeslaSCADA2 Runtime you can use Telegram messenger for this purpose. To send messages via Telegram, you need to make a preliminary configuration. First you need to create your own Telegram bot. To do this, you need to open the Telegram application, find a bot with the name "@BotFather", press the "Start" button and send the /newbot command to it:



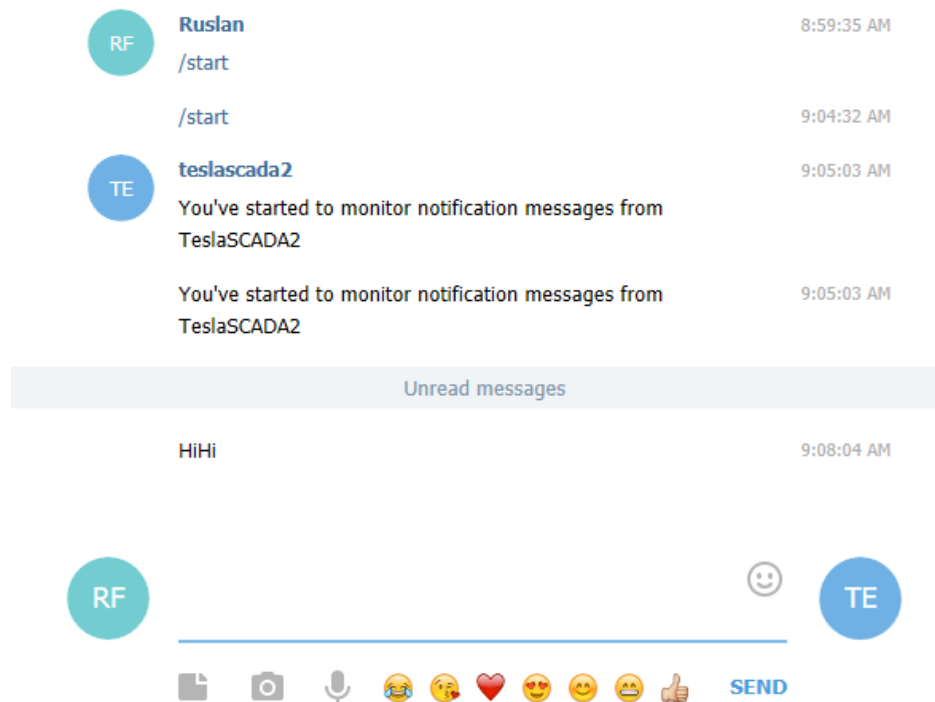
Next, you need to come up with a bot name and username (must end with the word "bot"). After that, the Token will be received:



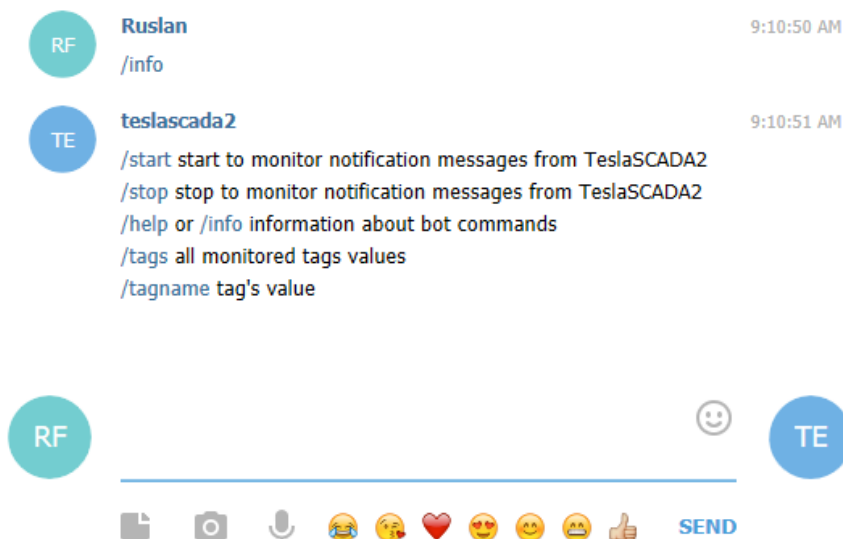
Next, in the **Project properties->Events/History tab**, check "Use Telegram Bot", enter bot's name and token:

E-mail client	Telegram bot	Push notifications	GSM modem
<input checked="" type="checkbox"/> Use Telegram Bot			
Bot's name:		<input type="text" value="teslascada2"/>	
Bot's token:		<input type="text" value="45nklUOP45dfreyuyq12klOIVCUMW12derls9I
slg;sfgk"/>	

Then you have to start TeslaSCADA2 Runtime and run this project (in TeslaSCADA IDE telegram bot doesn't work). After that, users who want to receive notifications should find our bot in Telegram and write **/start**:



To stop getting notification messages enter **/stop**. Also you can get some information from your project. To get possible command write **/info**:



Enter **/tags** to get current values of tags.

Enter name of the tag used in your project. You'll get information about value of this tag and if this tag supports history you'll get trend for last hour. You can choose other period by clicking proper button.

Important! Don't use underline in the name of the tags. Telegram have problems with working with this kind of names.

Important! At this moment you can use Telegram bot only on desktop versions of TeslaSCADA2 Runtime.

6.1.3.3 Push notifications

If you want to notify users by Push notifications messages you have to setup Push notifications and install TeslaSCADA2 Runtime mobile version on your Android or iOS devices.

E-mail client

Telegram bot

Push notifications

GSM modem

☒ Use push notifications

Topic:

Notifications (priority<=):

100

Title:

Message

OK

Cancel

Property	Description
Use push notifications	Check if you want to use push notifications. All event messages that have priority less then Notifications(priority<=) will be sent as push notifications on mobile devices.
Topic	Topic is used to subscribe mobile devices and send to

Property	Description
	this subscription by PC.
Notifications (priority<=)	If priority of the event message below this value push notification will be sent.
Title	<p>Title of the push notification.</p> <p>You can use keywords:</p> <p>{name} - name of the tag that send an alarm message.</p> <p>{server} - name of the PV input server.</p> <p>{message} - message is sent by tag's alarm.</p> <p>{group} - name of the tag's group.</p> <p>{subgroup} - name of tag's subgroup.</p> <p>{description} - tag's description.</p> <p>{value} - tag's value.</p> <p>{priority} - tag's message priority.</p> <p>{datetime} - current date and time (when alarm is happened).</p> <p>{projectname} - project name.</p> <p>{projectdescription} - project description.</p>
Message	<p>Message of the push notification</p> <p>You can use keywords:</p>

Property	Description
	<p>{name} - name of the tag that send an alarm message.</p> <p>{server} - name of the PV input server.</p> <p>{message} - message is sent by tag's alarm.</p> <p>{group} - name of the tag's group.</p> <p>{subgroup} - name of tag's subgroup.</p> <p>{description} - tag's description.</p> <p>{value} - tag's value.</p> <p>{priority} - tag's message priority.</p> <p>{datetime} - current date and time (when alarm is happened).</p> <p>{projectname} - project name.</p> <p>{projectdescription} - project description.</p>

6.1.3.4 GSM-modem

If you want to notify users by SMS messages you have to setup GSM-modem:

E-mail client
Telegram bot
Push notifications
GSM modem

☒ Use GSM modem

Port ID:

Baud rate:

9600

Flow control:

NONE

Data bits:

8

Stop bits:

1

Parity:

EVEN

To phone numbers:

OK
Cancel

Property	Description
Use GSM modem	Check if you want to use SMS notifications about Alarms. All event messages that have priority less then Notifications(priority<=) will be sent by SMS.
Port ID	ID of the COM port. If this port can not be open in TeslaSCADA2 Runtime other port will be tried to find and open.
Baud rate	Baud rate of the Common RTU server.
Flow control	Flow control of the port. It can be NONE, RTSCTS and XONXOF.
Data bits	Number of data bits. It can be 5, 6, 7 and 8.
Stop bits	Number of stop bits. It can be 1, 1.5 and 2.
Parity	Parity of the Common RTU. It can be NONE, EVEN, ODD, MARK and SPACE.
To phone numbers	Phone numbers separated by commas which SMS with alarms will be sent to.
Notifications (priority<=)	If priority of the event message below this value SMS will be sent. If this value <0 global Notifications (priority<=) will be used
Message	Message of the SMS. You can use keywords:

Property	Description
	<p>{name} - name of the tag that send an alarm message.</p> <p>{server} - name of the PV input server.</p> <p>{message} - message is sent by tag's alarm.</p> <p>{group} - name of the tag's group.</p> <p>{subgroup} - name of tag's subgroup.</p> <p>{description} - tag's description.</p> <p>{value} - tag's value.</p> <p>{priority} - tag's message priority.</p> <p>{datetime} - current date and time (when alarm is happened).</p> <p>{projectname} - project name.</p> <p>{projectdescription} - project description.</p>

6.1.4 OPC UA tab

OPC UA client settings

If you want to use OPC UA client certificate to connect to OPC UA servers in your project on the OPC UA tab enter Name of used/created certificate and Period(days) of validation if you create certificate:

Edit Project

General

Screens

Events/History

OPC UA

MQTT Publisher

Web-server

HTTP-server

Redundant server

Cloud

OPC UA client certificate

Name: TeslaSCADA2

Period(days): 3650

☒ Use OPC UA server

TCP port: 8666

Security mode:

☒ None

☒ BASIC128RSA15_SIGN

☒ BASIC128RSA15_SIGN_ENCRYPT

☒ BASIC256_SIGN

☒ BASIC256_SIGN_ENCRYPT

☒ BASIC256SHA256_SIGN

☒ BASIC256SHA256_SIGN_ENCRYPT

Policy:

☒ Anonymous

☒ Username/Password

Certificate name: opcuacertificate

Period(days): 3650

OK Cancel

The certificate is stored in the {app}/[private](#)^[18] directory.

OPC UA server settings

If you want to use [Client - Server architecture](#)^[12] in your system and use it with OPC UA server you have to check Use OPC UA sever:

Edit Project

General

Screens

Events/History

OPC UA

MQTT Publisher

Web-server

HTTP-server

Redundant server

Cloud

OPC UA client certificate

Name: TeslaSCADA2

Period(days): 3650

☒ Use OPC UA server

TCP port: 8666

Security mode:

☒ None

☒ BASIC128RSA15_SIGN

☒ BASIC128RSA15_SIGN_ENCRYPT

☒ BASIC256_SIGN

☒ BASIC256_SIGN_ENCRYPT

☒ BASIC256SHA256_SIGN

☒ BASIC256SHA256_SIGN_ENCRYPT

Policy:

☒ Anonymous

☒ Username/Password

Certificate name: opcuacertificate

Period(days): 3650

OK Cancel

Property	Description
Use OPC UA server	Check If you want to enable OPC UA server of TeslaSCADA2
TCP port	TCP port of your OPC UA server.
None	Check if you want to use None security mode in the server.
BASIC128RSA15_SIGN	Check if you want to use BASIC128RSA15_SIGN security mode in the server.
BASIC128RSA15_SIGN_ENCRYPT	Check if you want to use BASIC128RSA15_SIGN_ENCRYPT security mode in the server.
BASIC256_SIGN	Check if you want to use BASIC256_SIGN security mode in the server.

Property	Description
BASIC256_SIGN_ENCRYPT	Check if you want to use BASIC256_SIGN_ENCRYPT security mode in the server.
BASIC256SHA256_SIGN	Check if you want to use BASIC256SHA256_SIGN security mode in the server.
BASIC256SHA256_SIGN_ENCRYPT	Check if you want to use BASIC256SHA256_SIGN_ENCRYPT security mode in the server.
Anonymous	Check Anonymous if you want to use this policy in OPC UA server.
Username/Password	Check Username/Password if you want to use this policy in OPC UA server.
Certificate name	Certificate name of the OPC UA server.
Period(days)	The period during which the OPC UA server certificate will be valid.

6.1.5 MQTT Publisher tab

If you want to use [Client - Server architecture](#)¹²⁾ in your system and use it with MQTT broker you have to check Enable MQTT Publisher:

Edit Project

General

Screens

Events/History

OPC UA

MQTT Publisher

Web-server

HTTP-server

Redundant server

Cloud

☒ Enable MQTT Publisher

Broker URL:

Username:

Password:

Client ID:

Write topic format:

Read topic format:

QoS:

☒ Enable TLS/SSL

Protocol:

Certificate filename:

☒ Enable Client Certificate

Client Certificate:

Client Private Key:

Private Key Password:

☒ PEM Formatted

OK

Cancel

Property	Description
Enable MQTT Publisher	Check if you want to enable MQTT publisher.
Broker URL	Broker URL of the MQTT server.
Username	Username of the MQTT broker.
Password	Password of the MQTT broker.
Client ID	Some brokers need Client ID. If you left client ID unfilled publisher will generate ClientID itself.
Write topic format	Some cloud brokers need formatted topic. See IBM cloud example ⁵⁹⁸ . You can left this field empty.
Read topic format	Some cloud brokers need formatted topic. See IBM cloud example ⁵⁹⁸ . You can left this field empty.

Property	Description
QoS	Choose QoS of MQTT messages.
Enable TLS/SSL	Check Enable TLS/SSL if you want to use server certificate for encryption messages.
Certificate filename	Certificate filename. File should be placed in /private ¹⁸ folder in the directory where TeslaSCADA2 is installed.
Enable Client Certificate	Check if you want to use client certificate for encryption messages.
Client certificate	Client certificate filename. File also should be placed in /private ¹⁸ folder
Client Private key	Client private key filename. File also should be placed in /private ¹⁸ folder
Private key password	Private key password.
PEM formatted	Check if your certificate and key files are PEM formatted.

MQTT publisher will send tag's values collected during project running on MQTT broker you want. MQTT subscribers will collect this values and represent it on devices you want. If you don't use "Write topic format" and "Read topic format" fields publisher's topics consists of the «name of the project +/Tags/+tagname» for tags and «name of the project+/Events/+tagname» for events. If you use "Write topic format" and "Read topic format" tags replace {tagname} keyword.

6.1.6 Web-server tab

If you want to use Web-Server in your project click on the tab Web-Server and enable it. To have possibility to use Web-Server on the PC you want, Java 8(JRE) should be installed on it. For TeslaSCADA Runtime version below 2.41.2 Java version should be from 8.25 - 8.161. For TeslaSCADA Runtime starting from 2.41.2 version minimal Java version - 8.281. To check version of Java you have in command line write command `java -version`. In the response you'll get installed Java version. Also to have possibility to run Web-Server TeslaSCADA should be [installed](#)¹⁴ in the path without white spaces. You can use any modern browser to access to the Web-Server. The most recommended browser - Google Chrome.

Important! If you use Mac OS Big Sur and have problems with running Web server delete `/Library/Internet Plug-Ins/` folder on your disk and relogin.

Important! Web-Server is possible to use only in Evaluation version (project contains up to 16 tags) and in the Full version (if you activate a full license).

Edit Project

General

Screens

Events/History

OPC UA

MQTT Publisher

Web-server

HTTP-server

Redundant server

Cloud

☒ **Enable Web-server**

Host:

☒ **HTTP**

HTTP port:

☒ **HTTPS**

HTTPS port:

Truestore file:

Truestore password:

Keystore file:

Keystore password:

☒ **Use other project for WEB client**

Project:

OK **Cancel**

Property	Description
Enable Web-server	Check if you want to enable Web-server.
Host	Host of the Web-Server. Usually it's an IP address of PC where installed TeslaSCADA2 Runtime and Run configured project.
HTTP	Check HTTP if you want to use unsecured HTTP protocol to connect to Web-Server.
HTTP port	HTTP port used by Web-Server.
HTTPS	Check HTTPS if you want to use secured HTTPS protocol to connect to Web-Server.
HTTPS port	HTTPS port used by Web-Server.
Truestore file	It's a file where stored validated certificates. It should be with .jks or .keystore format.

Property	Description
Truestore password	Truestore password to have access to truestore ?le.
Keystore ?le	It's a ?le where stored certi?cates of the server. It should be with .jks or .keystore format.
Keystore password	Keystore password to have access to keystore ?le.
Use other project for WEB client	If you want to use other project for WEB client check this field.
Project	Choose path to the project for WEB client.

If you want to use self-signed certi?cates in keystore you'll have problems in accessing to WebServer by using the most popular browsers. You have to use certi?cates signed by CA to exclude these problems.

Important! Web-server create another instance of TeslaSCADA2 Runtime application to connect to the servers and databases of the project. It's not possible to use its functionality if your server doesn't let multiple connection (for example Modbus RTU lets only one app connects to the port). And it's not possible to use SQLite database at the same reason. To escape this problem use HTTP server and use HTTP client for WEB client. To do this check "Use other project for WEB client" and choose HTTP client project.

6.1.7 HTTP-server

If you want to use [Client - Server architecture](#)^[12] in your system and use it with HTTP-server you have to check Enable HTTP-server:

Edit Project

General

☒ Enable HTTP-server

Host: localhost

Port: 8000

Username:

Password:

☒ HTTPS

Keystore file:

Keystore password:

Create HTTP client: ...

HTTP-server

Redundant server

Cloud

OK Cancel

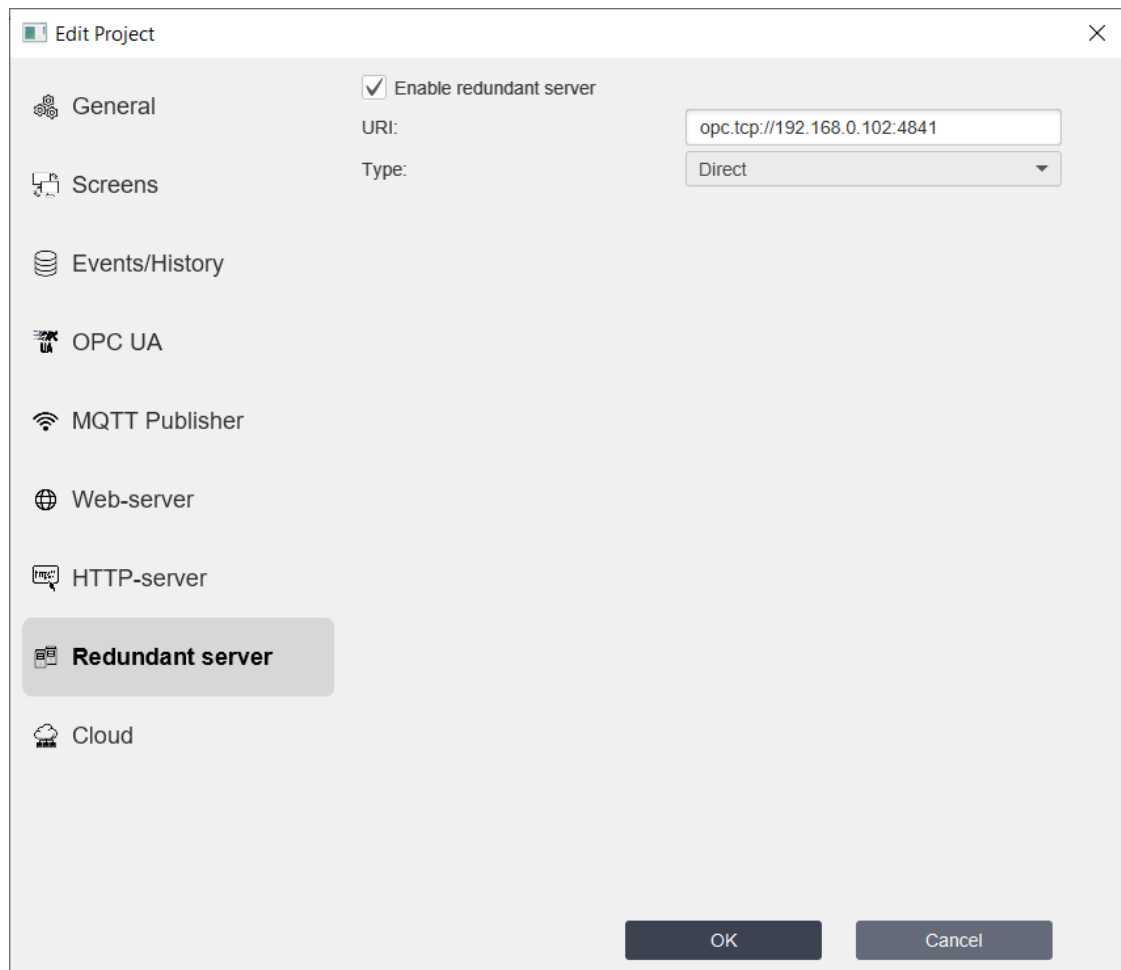
Below description of the properties:

Property	Description
Enable HTTP Server	Check if you want to enable HTTP server.
Host	Host or IP address of the HTTP server.
Username	Username of the HTTP server.
Password	Password of the HTTP server.
HTTPS	Check HTTPS if you want to use secured HTTPS protocol to connect to HTTP-server.
Keystore file	It's a file where stored certificates of the server. It should be with .jks or .keystore format. File placed in /private folder
Keystore password	Keystore password to have access to keystore file.

Property	Description
Create HTTP client	If you want to create HTTP client for connecting to this HTTP-server click this button.

6.1.8 Redundant server

If you want to use Redundant server in your project check **Enable redundant server**. Redundant server is based on OPC UA client. Primary server should use [OPC UA server](#)¹²² with Security mode is None and Anonymous policy:



Below description of the properties:

Property	Description
Enable redundant server	Check if you want to enable redundant server.
URI	OPC UA URI of the primary server.
Type	Type of the redundant server:

Property	Description
	<ul style="list-style-type: none"> - Direct - redundant server uses servers like primary server to get tag's values of the project. - OPC UA - redundant server uses OPC UA server of the primary server to get tag's values of the project.

6.1.9 Cloud

If you want to use Tesla Cloud in your project check **Enable cloud (To use cloud you have opened 7000 and 7001 ports on your device)** . Cloud lets you provide tags information from your project on the Tesla Cloud by using desktop TeslaSCADA2 Runtime and read this information by using browser or TeslaSCADA2 Runtime for desktop or mobile (only Android at this moment):

The screenshot shows the 'Edit Project' dialog box with the 'Cloud' tab selected. The 'Enable cloud' checkbox is checked. Below it are input fields for 'Username:', 'Password:', 'Create cloud client', 'Create HTTP client:', and 'Create MQTT client'. The 'Cloud' tab is highlighted in the left sidebar.

Below description of the properties:

Property	Description
Enable cloud	Check if you want to enable Tesla Cloud.
Username	Username of the user of Tesla Cloud.

Property	Description
Password	Password of the user of Tesla Cloud.
Create cloud client	If you want to create Cloud client for connecting to this Tesla Cloud click this button.
Create HTTP client	If you want to create HTTP client for TeslaCloud use this button.
Create MQTT client	If you want to create MQTT client for TeslaCloud use this button.

6.2 Screens

Create screen

To create a new screen select the menu item **Project**^[67] -> **New Screen** or choose **Screens**^[106] on the Project Window, click right button on it and choose **New Screen** item. You'll see the **screen properties**^[135] window:

Screen properties

Group:

Subgroup:

Name:

Comment:

Background color:

Screen type:

Scripts:

Screen dimension: X

Coordinates: X= Y=


Access level:

☐ Use password

Password:


OK Cancel

Open screen


To open screen on [Screens](#)  tab of the Project window:

- Right click on the screen you want to open and choose **Open** item.
- or
- Double click on the screen you want to open.

Copy screen

To copy screen on [Screens](#)  tab of the Project window right click on the screen you want to copy and choose **Copy** item.


Delete screen

To delete screen on [Screens](#)  tab of the Project window right click on the screen you want to delete and choose **Delete** item.

Open screen properties


To open [screen properties](#)  on [Screens](#)  tab of the Project window right click on the screen you want to open and choose **Screen properties** item.

Export screen


To export screen on [Screens](#)  tab of the Project window:

1. Right click on the screen you want to export and choose Export screen item.
2. Now select the location and click the button Save (TeslaSCADA2 screen extension .tsp2screen).

Import screen

To import screen on [Screens](#)  tab of the Project window:

1. Right click on the screen window and choose Import screen item.
2. Now select the screen file and click Open (TeslaSCADA screen extension .tsp2screen).

See **Project Window->**[Screens](#)  tab for more information about possible operation with screens.

6.2.1 Screen properties

Screen properties

Group:

Subgroup:

Name:

Screen0

Comment:

Background color:

Light Gray

Screen type:

General

Scripts:

Collection

Screen dimension:

800

 X

600

Coordinates:

X=

-1000

 Y=

-1000

Access level:

0

☐ Use password

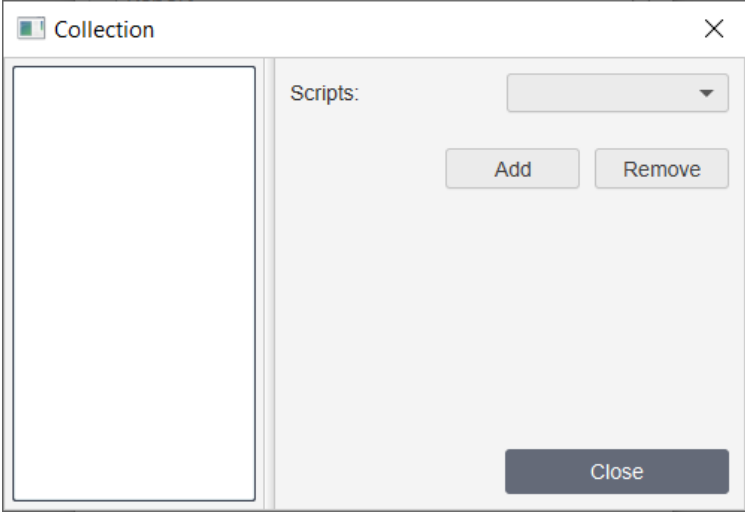
Password:

OK

Cancel

List of screen properties:

Property	Description
Group	Select group for the screen.
Subgroup	Select subgroup for the screen.
Name	Name of the screen.
Comment	Optionally specify a meaningful comment.
Background color	Background color of the screen.
Screen type	Select screen type of the screen - <i>General</i> or <i>Popup</i> .
Scripts	Click Collection to set up screen's scripts . After clicking you'll see the window:

Property	Description
	 <p>where:</p> <ul style="list-style-type: none"> • Scripts - list of available screen type scripts in the project. • Add - add script to the collection. • Remove - remove script from the collection.
Screen dimension	Width and height of the screen.
Coordinates	If you choose Popup screen you can enter position X and Y where this screen will be appeared. If you enter value < 0 the screen will appear at the center.
Access Level	Screen's access level. If this value greater then access level of the current user the screen couldn't be opened by this user.
Use password	Check if you want to use screen security.
Password	Only the user who knows the password will be able to open this window.

6.2.2 Designing screen

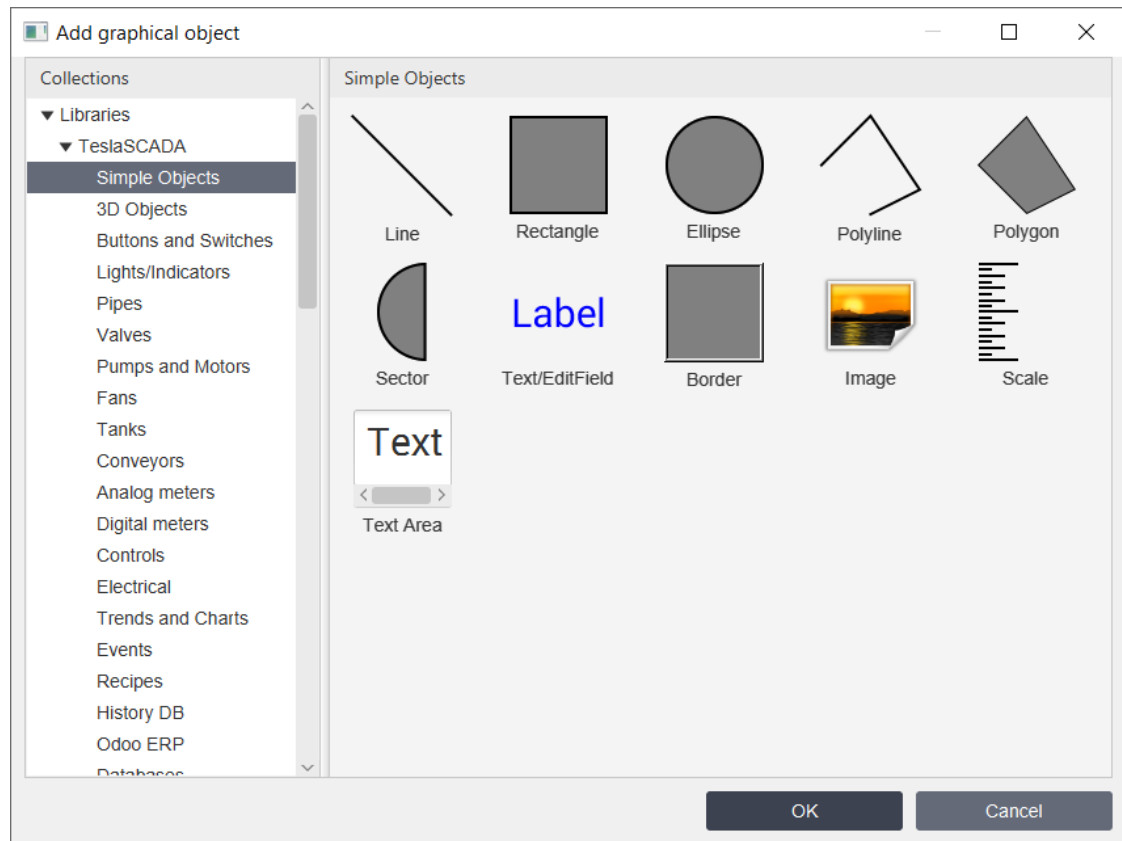
To start designing the screen you want, you need to double click on it or click right button on the [Project window](#)^[72] -> Screens and choose *Open screen* menu item.

Create graphical object

You can add new graphical object on the screen in several ways:

- Select the menu item [Project](#)^[67] and **New Object**.
- Click [New Object](#)^[70] button on the Toolbar.
- Click right button on the [Screen window](#)^[92] and choose **New object** menu item.
- Click right button on the [Canvas](#)^[91] and choose **New object** menu item.

You'll see the **Add graphical object** window:



Select library which object you want to use in your project (all libraries and their objects described below). Select object you can in several ways:

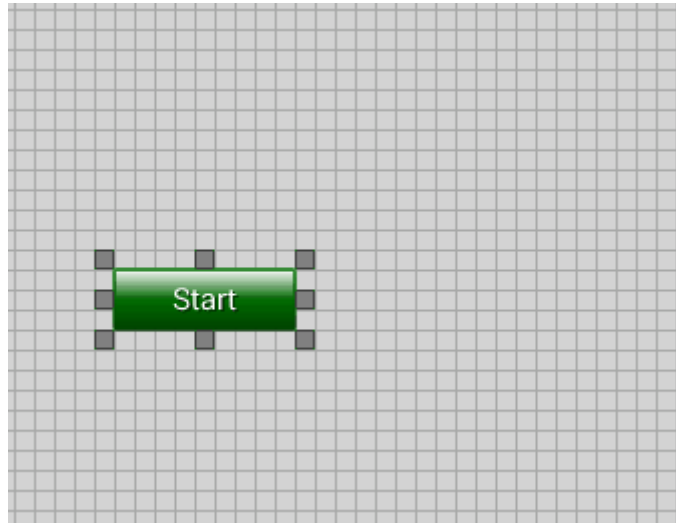
- By double clicking on the object.
- By clicking on the object (select rectangle will appear) and then clicking OK button.
- By clicking right button and choosing Select menu item.

Add graphical object window will disappear and you can select the location on the screen where you want to place the object.

Object information about its dimensions and coordinates you can find in the [status bar](#) on the right.

Resize graphical object

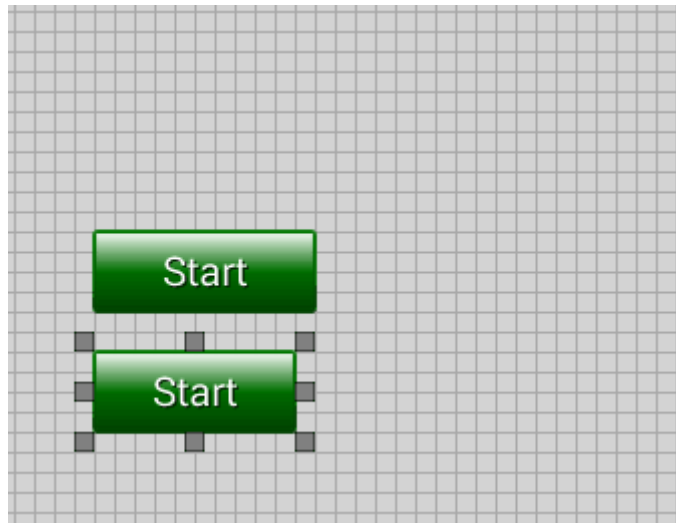
You can resize graphical object by clicking on it. Resize squares will appear and you can change dimensions of the object as you want.



Also you can resize object by using arrow [keys](#)^[94] on keyboard.

Select several objects

You can select several objects by using selecting rectangle or by clicking on objects you want to select and simultaneously holding CTRL key.



Move graphical object

You can move graphical objects by using Drag and Drop technology. You can also move objects by using arrow [keys](#)^[94] on keyboard.

Open graphical object properties

You can open graphical object properties on the [Screen Window](#)^[92] or on the [Canvas](#)^[91]. To open graphical object properties:

- Right click on the object you want to open and choose Object properties menu item.
- or

- Double click on the object, properties which you want to open.

Copy graphical object

You can copy graphical object:

- Right click on the object you want to copy and choose **Copy** menu item of the context menu.
- Select the object you want to copy and choose [Edit](#)^[63]->**Copy** menu item on the main menu.
- Select the object you want to copy and click [Copy](#)^[70] button on the [Toolbar](#)^[69].
- Use corresponding [hotkeys](#)^[96] for your operating system.

Cut graphical object

You can cut graphical object:

- Right click on the object you want to cut and choose Cut item of the context menu.
- Select the object you want to cut and choose [Edit](#)^[63]->Cut menu item on the main menu.
- Select the object you want to cut and click [Cut](#)^[70] button on the [Toolbar](#)^[69].
- Use corresponding [hotkeys](#)^[96] for your operating system.

Paste graphical object

You can paste (before you have to cut or copy) graphical object:

- Right click on the [Canvas](#)^[91] and choose Paste menu item of the context menu.
- Choose [Edit](#)^[63]->Paste menu item on the main menu.
- Click [Paste](#)^[70] button on the [Toolbar](#)^[69].
- Use corresponding [hotkeys](#)^[96] for your operating system.

Erase graphical object

You can erase graphical object:

- Right click on the object you want to erase and choose Erase menu item of the context menu.
- Select the object you want to erase and choose [Edit](#)^[63]->Erase menu item on the main menu.
- Right click on the object in the [Screen Window](#)^[92] and choose Delete object menu item.
- Use corresponding [hotkeys](#)^[96] for your operating system.

Duplicate graphical object

You can duplicate graphical object:

- Right click on the object you want to duplicate and choose Duplicate menu item of the context menu.
- Select the object you want to duplicate and choose [Edit](#)^[63]->Duplicate menu item on the main menu.

- Use corresponding [hotkeys](#)^[96] for your operating system.

Send to back graphical object

You can send to back graphical object relative to other objects of the screen:

- Right click on the object you want to send to back and choose Send to Back menu item of the context menu.
- Select the object you want to send to back and choose [Arrange](#)^[64]->Send to Back menu item on the main menu.
- Select the object you want to send to back and click [Send to Back](#)^[70] button on the [Toolbar](#)^[69].
- Use corresponding [hotkeys](#)^[96] for your operating system.

Bring to front graphical object

You can bring to front graphical object relative to other objects of the screen:

- Right click on the object you want to bring to front and choose Bring to Front menu item of the context menu.
- Select the object you want to bring to front and choose [Arrange](#)^[64]->Bring to Front menu item on the main menu.
- Select the object you want to bring to front and click [Bring to Front](#)^[70] button on the [Toolbar](#)^[69].
- Use corresponding [hotkeys](#)^[96] for your operating system.

Rotate clockwise graphical object

You can rotate clockwise graphical object clockwise:

- Select the object you want to rotate clockwise and click [Rotate Clockwise](#)^[71] button on the [Toolbar](#)^[69].
- Select the object you want to rotate clockwise and choose [Arrange](#)^[64]->Rotate Clockwise menu item on the main menu.
- Use corresponding [hotkeys](#)^[96] for your operating system.

Rotate counterclockwise graphical object

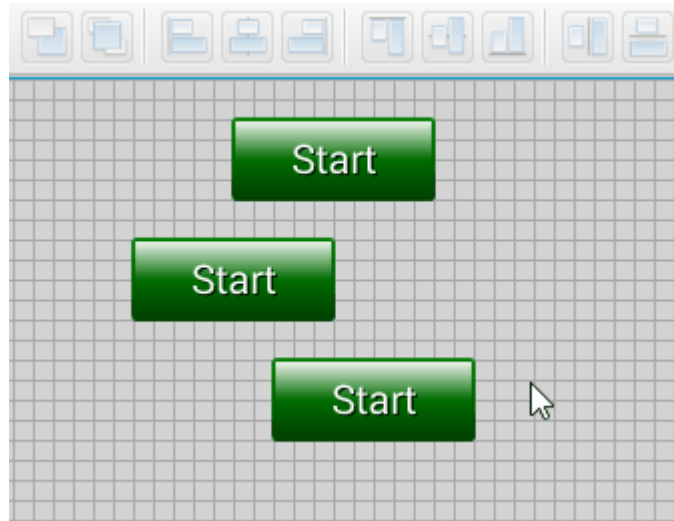
You can rotate counterclockwise graphical object clockwise:

- Select the object you want to rotate counterclockwise and click [Rotate CounterClockwise](#)^[71] button on the [Toolbar](#)^[69].
- Select the object you want to rotate counterclockwise and choose [Arrange](#)^[64]->Rotate CounterClockwise menu item on the main menu.
- Use corresponding [hotkeys](#)^[96] for your operating system.

Align graphical objects

You can align objects relative to each other on the screen. Choose objects you want to align by [selecting square or by clicking on objects you want to select and simultaneously holding CTRL key](#)^[138]. And:

- Choose [Arrange](#)^[64] -> Align menu items on the main menu.
- Click [Align buttons](#)^[70] on the [Toolbar](#)^[69].
- Right click on selecting square and choose Align menu item of the context menu.
- Use corresponding [hotkeys](#)^[96] for your operating system.

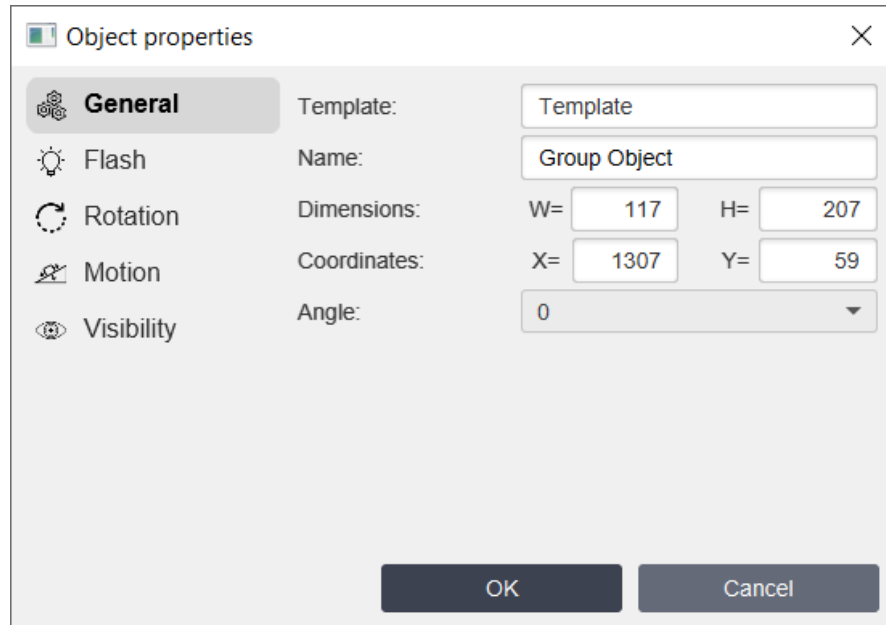


For more information about each alignment operation you can read above in section Start TeslaSCADA IDE -> [Toolbar](#)^[70].

Group graphical objects

You can group objects. Choose objects you want to group by [selecting square or by clicking mouse left button and simultaneously holding CTRL button](#)^[138]. And:

- Select [Arrange](#)^[64] -> Group objects menu item on the main menu.
- Click [Group objects](#)^[71] button on the [Toolbar](#)^[69].
- Right click on selecting square and choose Group objects menu item on the context menu.
- You can edit properties of this group object by double clicking or by choosing context menu properties menu item:



You can change name of the group object, coordinates, dimensions and enter template name. Later you can apply changes for the template by clicking appropriate [main menu item](#)^[63].

Ungroup graphical objects

You can ungroup objects. Choose group of objects you want to ungroup by clicking on it . And:

- Select [Arrange](#)^[64]->Ungroup objects menu item on the main menu.
- Click [Ungroup objects](#)^[71] button on the [Toolbar](#)^[69].
- Right click on selecting square and choose Ungroup objects menu item on the context menu.

Copy properties

You can copy properties of the object. This possibility lets to copy all properties of the object excluding General properties and place them into clipboard. You can do it by:

- Right clicking on the object which properties you want to copy and choose Copy properties menu item.

Paste properties

You can paste properties of the object. This possibility lets to paste all properties that were placed into the clipboard by using Copy properties. You can do it by:

- Right clicking on the object to which you want to copy the properties from the clipboard.

Add object into the Group

You can add selected object into the group of the objects:

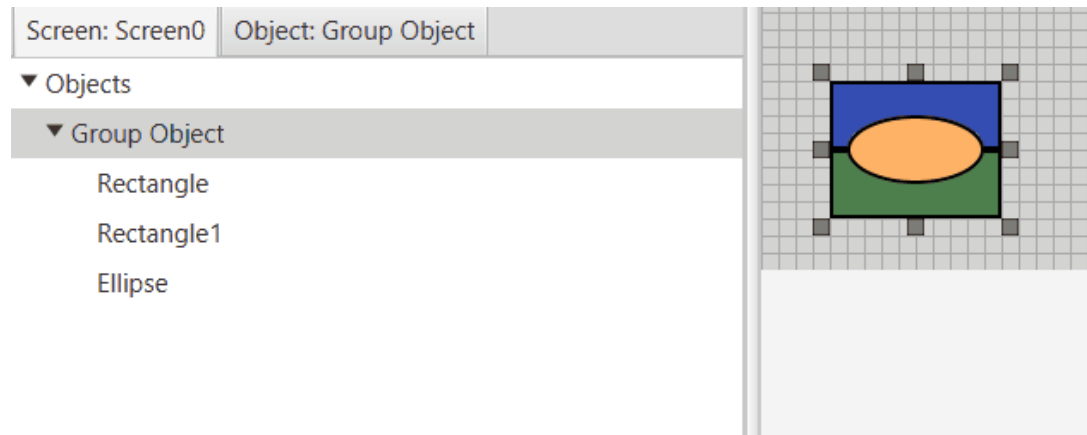
1. Choose object you want to add into the group.
2. And in the screen window drag and drop object into the group object.



Remove object from the Group

You can remove object from the group of the objects:

1. Choose object you want to remove from the group in the screen window.
2. And in the screen window drag and drop object on the name of the group object.



Virtual keyboard

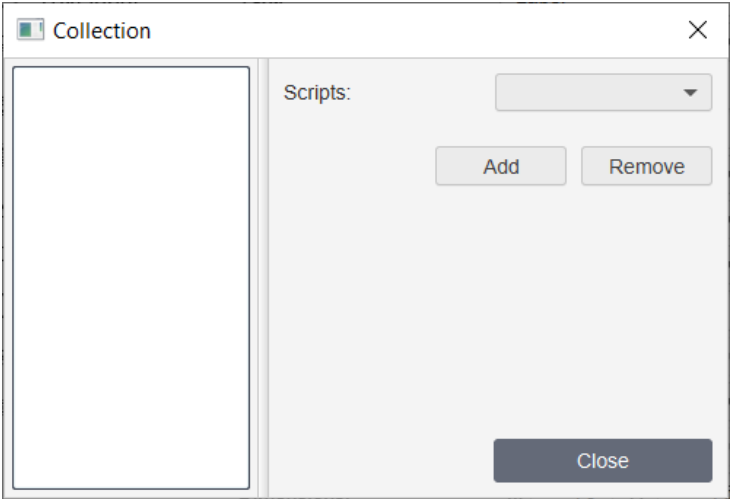
If you want to develop your project on [Sensor screen](#)^[98] you can turn on virtual keyboard. You can do by checking [Project](#)^[67] -> Virtual keyboard menu item on the main menu.

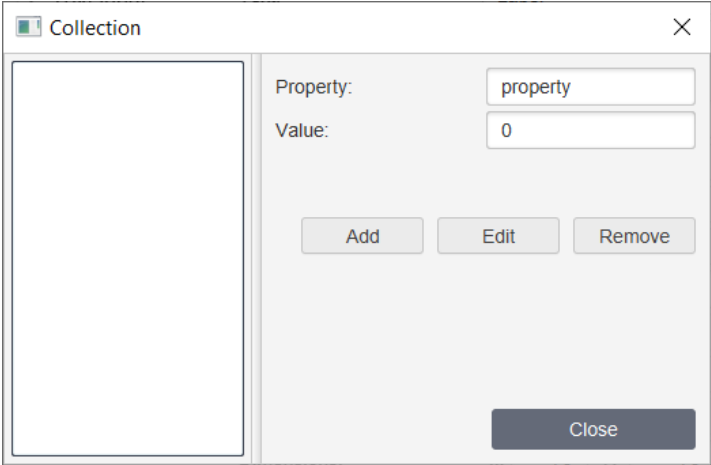
6.2.3 Graphical objects

Every graphical object has several group of properties. All properties you can edit in [Property sheet](#)^[97] or in Object settings window (you can get this window by double clicking

on the object). The description of every group of properties you can find below in the chapter - [Properties](#)^[344]. In this chapter we describe one group for every object - General.

This group is responsible for the appearance of the object, contains scripts and user-defined properties. Every object has the following properties:

Property	ST script field*	Description
Name	name	Name of the object. You can use indirect name by using group name. To do this use curve braces {}. For example, if group's name is "group" and you enter in the field {group}name and you'll get name of the object is "groupname".
Dimensions	width	Dimensions of the graphical object. Enter width of the object in the W(width) field and enter height of the object in the H(height) field.
	height	
Coordinates	posx	Coordinates of the graphical object. Write x coordinates of the object in the X(posx) field and enter y coordinates of the object in the Y(posy) field.
	posy	
Angle	angle	Select the rotation angle of the object (0, 90, 180, 270).
Type		Select the type of the object - 2D or 3D.
Scripts		<p>Click Collection to add scripts for the Object. After clicking Collection button you'll see the following window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Scripts^[393] - list of object type scripts. ▪ Add - add script to the object.

Property	ST script field*	Description
		<ul style="list-style-type: none"> ▪ Remove - remove script from the object.
User-Defined**		<p>Click Collection to add user-defined properties for the Object. After clicking Collection button you'll see the following window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Property - name of the user-defined property. ▪ Value - value of the user-defined property. ▪ Add - add user-defined property to the object. ▪ Edit - edit user-defined property of the object. ▪ Remove - remove user-defined property from the object.

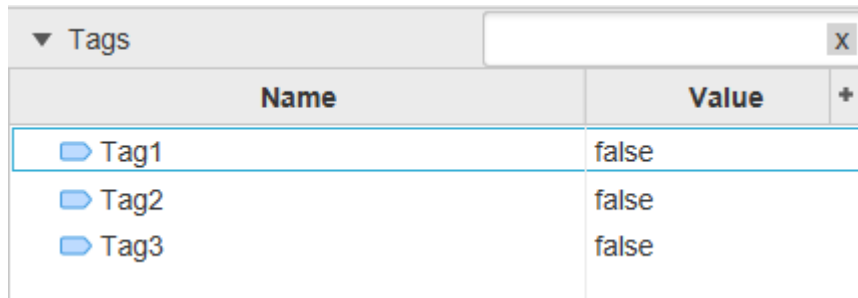
* This field is used in ST scripts. For example: **Objects.Button.width = 100**. In this script command width of the object with name **Button** become **100**.

** User-defined properties can be used in indirect properties tag names and in scripts. Below is described how to do it.

User-defined properties

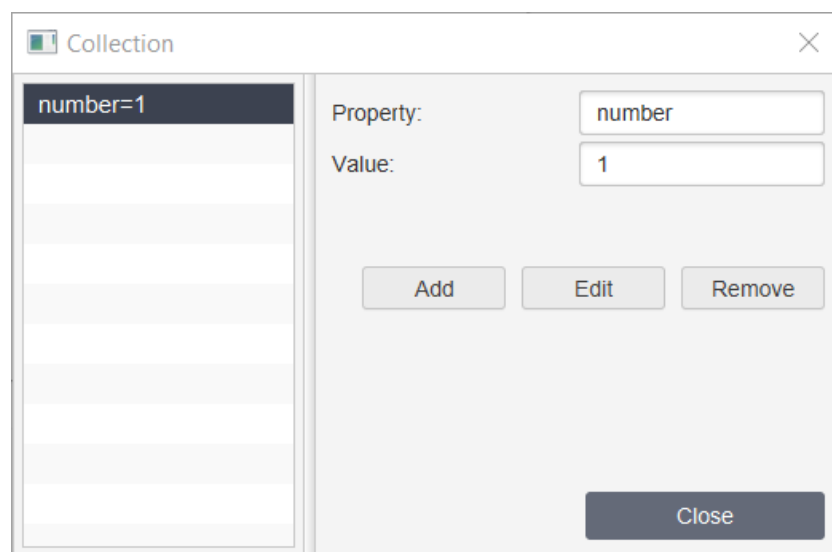
We have several same type objects, each object has one tag. We can setup only one object and then duplicate this object and correct only value of User-defined property in new objects. Look at the example.

Let's create several tags: (one for each object):



Name	Value
Tag1	false
Tag2	false
Tag3	false

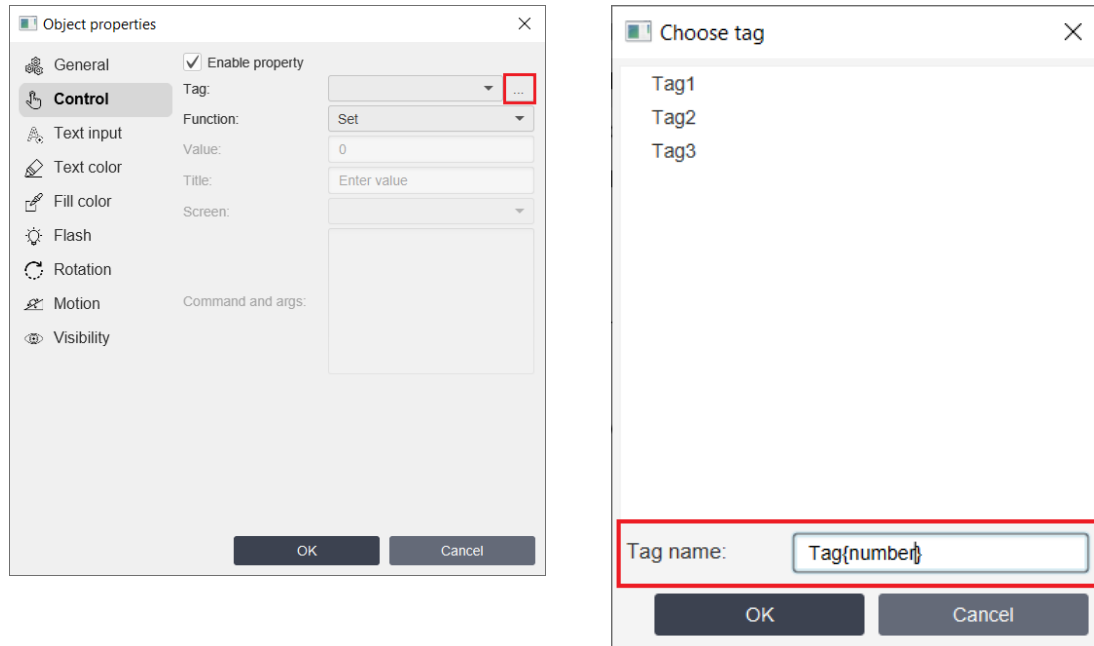
Let's create an object - Button, set user-defined property "number" and set its value "1" (because we want to bind this object to Tag1):



Property	Value
number	1

Buttons: Add, Edit, Remove, Close

Then you have to close Collection window and click "OK" to close Object properties window add save this user-defined property for this object. Now open object properties window again to bind this object to the tag (or you can do it in [Property sheet](#)⁹¹). Open Choose tag window for some of the property that bind to the tag:



You can use indirect Tag name by using user-defined property number we created and using curve braces {}. Tag name will be looked Tag{number}. For this object number equal 1. So the tag name will be Tag1. So we have a configured object.

Now we can copy this object (Button in our case) and change user-defined property number to bind these objects to other tags. The easiest way to do it change property number in [Property sheet](#)⁹¹.

Screen: Scre... **Object: Button**

Font type: Roboto Regular

Underline: ☐

Font size: 0

Text placement: CENTER

Text color: ☐ White

Fill color: ☒ Green

Type: 3D

Animation: ☒

Width: 75.0

Height: 37.0

Position X: 555.0

Position Y: 182.0

Angle: 0

Scripts: Collection

number: 2

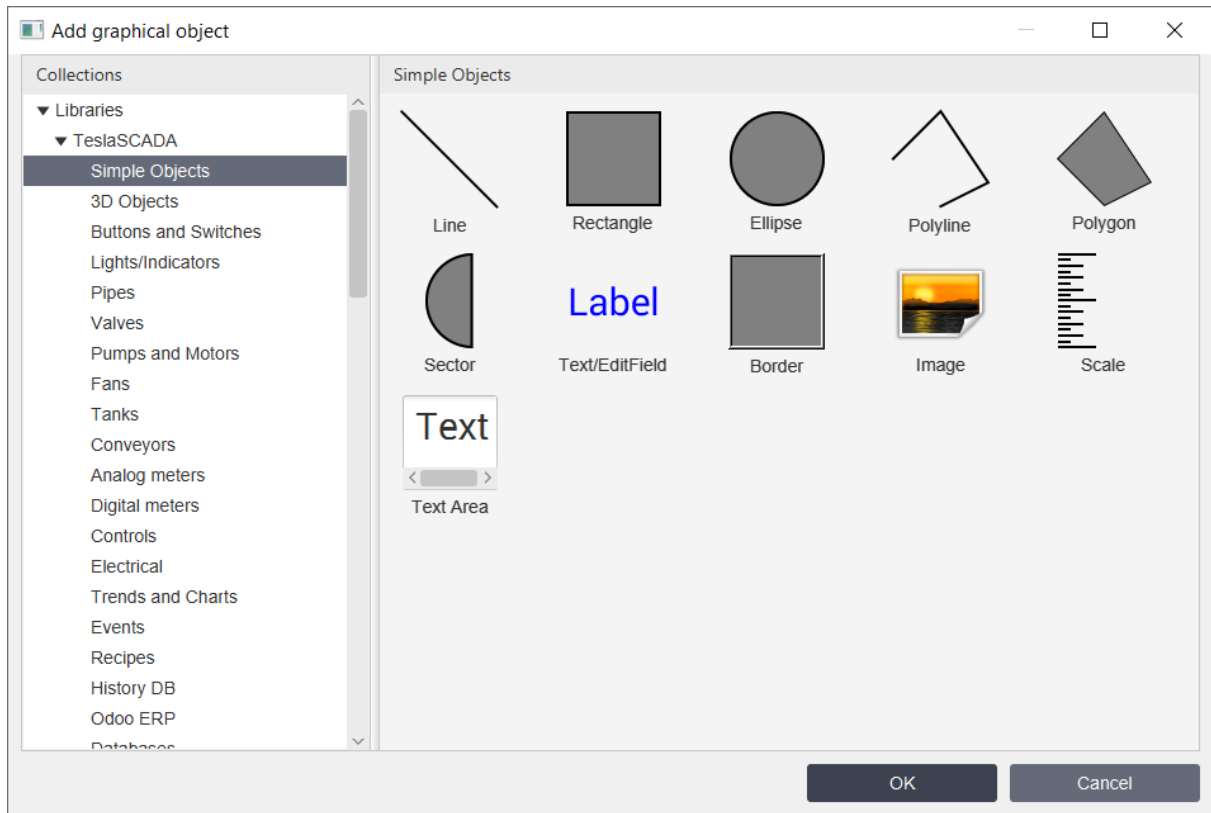
This is very helpful if you develop big project with similar objects and tags.

Also you can use user-defined property in ST scripts. For example, you have user-defined property "description" with some description of the object and want to display it on the screen with some Text object type when, for example, you click on this object. You have to create ST script with execution type - OnClick and add this script to the object which description you want to display. Script code will look like below:

Objects.Text.text = Objects.this.description;

Important! If you change user-defined property in ST script it will not affect on the indirect tag name of the object's property.

6.2.3.1 Simple objects library



Simple objects library contains the following objects:

- [Line](#) ¹⁵⁰
- [Rectangle](#) ¹⁵¹
- [Ellipse](#) ¹⁵²
- [Polyline](#) ¹⁵³
- [Polygon](#) ¹⁵⁶
- [Sector](#) ¹⁵⁸
- [Text/EditField](#) ¹⁵⁹
- [Border](#) ¹⁶¹
- [Image](#) ¹⁶²
- [Scale](#) ¹⁶³
- [Text Area](#) ¹⁶⁵

6.2.3.1.1 Line

Object properties

General

Name: Line

Line width: 2

Color: Black

Line style: Solid

Beginmarker: Flat

Endmarker: Flat

Dimensions: W= 75 H= 75

Coordinates: X= 1033 Y= 149

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#))

Property	ST script field	Description
Line width	linewid th	Width of the line.
Color	color	Color of the line.
Line style	linestyl e	Style of the line: <ul style="list-style-type: none"> ▪ Solid ▪ Dash ▪ Dot ▪ DashDot
Beginmarker	beginm arker	Marker of the line's begin: <ul style="list-style-type: none"> ▪ Flat ▪ Arrow ▪ Square ▪ Circle

Property	ST script field	Description
Endmarker	endmarker	Marker of the line's end: <ul style="list-style-type: none"> ▪ Flat ▪ Arrow ▪ Square ▪ Circle

Properties from the "**Line Color**" tab are described [here](#)³⁵⁰.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.1.2 Rectangle

The screenshot shows the 'Object properties' dialog box for a 'Rectangle' object. The 'General' tab is selected, showing various properties and their values:

- Name:** Rectangle
- Line width:** 2
- Color:** Black
- Fill:** true
- Fill color:** Gray
- Dimensions:** W= 75, H= 75
- Coordinates:** X= 154, Y= 126
- Angle:** 0
- Scripts:** Collection
- User-defined:** Collection

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Line width	linewid th	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill rectangle.
Fill color	fillcolor	Fill color of the rectangle.

Properties from the "Line Color" tab are described [here](#)^[350].

Properties from the "Fill Color" tab are described [here](#)^[352].

Properties from the "Filling" tab are described [here](#)^[354].

Properties from the "Flash" tab are described [here](#)^[345].

Properties from the "Rotation" tab are described [here](#)^[347].

Properties from the "Motion" tab are described [here](#)^[348].

Properties from the "Visibility" tab are described [here](#)^[349].

6.2.3.1.3 Ellipse

Object properties

General

Name: Ellipse

Line color: Line width: 2

Fill color: Color: Black

Flash: Fill: true

Rotation: Fill color: Gray

Motion: Dimensions: W= 75 H= 75

Visibility: Coordinates: X= 338 Y= 115

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143])

Property	ST script field	Description
Line width	linewid th	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill ellipse.
Fill color	fillcolor	Color of the ellipse's filling.

Properties from the "Line Color" tab are described [here](#)³⁵⁰.

Properties from the "Fill Color" tab are described [here](#)³⁵².

Properties from the "Flash" tab are described [here](#)³⁴⁵.

Properties from the "Rotation" tab are described [here](#)³⁴⁷.

Properties from the "Motion" tab are described [here](#)³⁴⁸.

Properties from the "Visibility" tab are described [here](#)³⁴⁹.

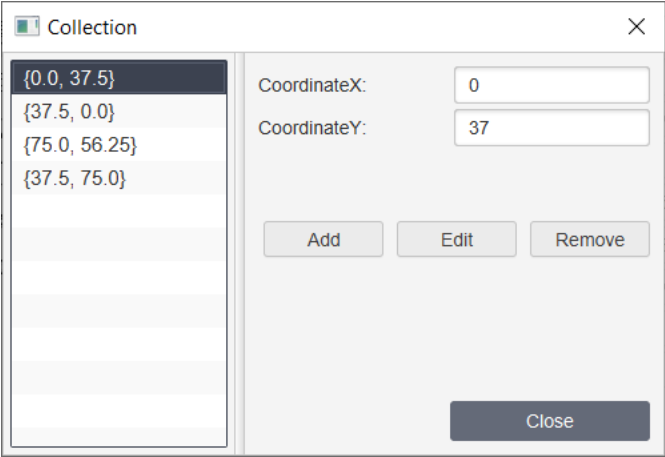

6.2.3.1.4 Polyline

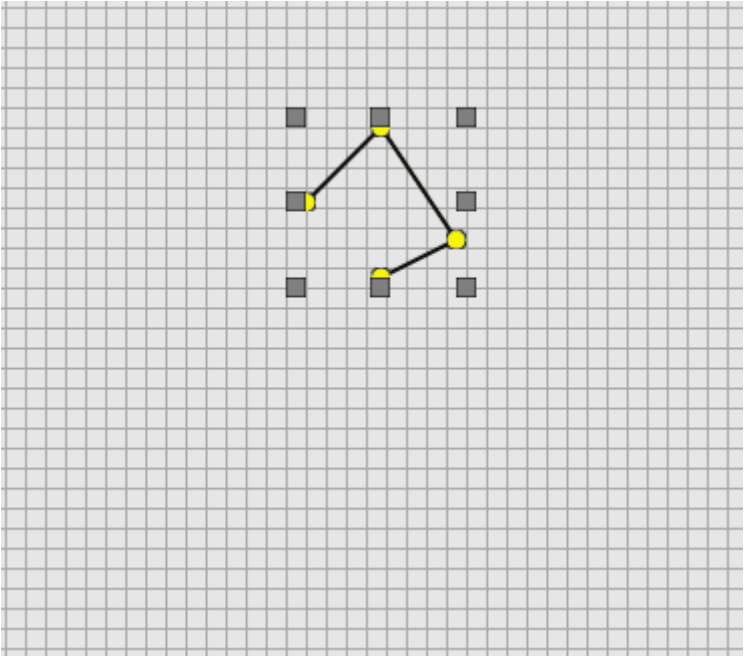
The screenshot shows the 'Object properties' dialog box for a 'Polyline' object. The 'General' tab is active. The properties are as follows:

- Name:** Polyline
- Line width:** 2
- Color:** Black
- Hotspots:** Collection
- Dimensions:** W= 75, H= 75
- Coordinates:** X= 636, Y= 66
- Angle:** 0
- Scripts:** Collection
- User-defined:** Collection

At the bottom are 'OK' and 'Cancel' buttons.

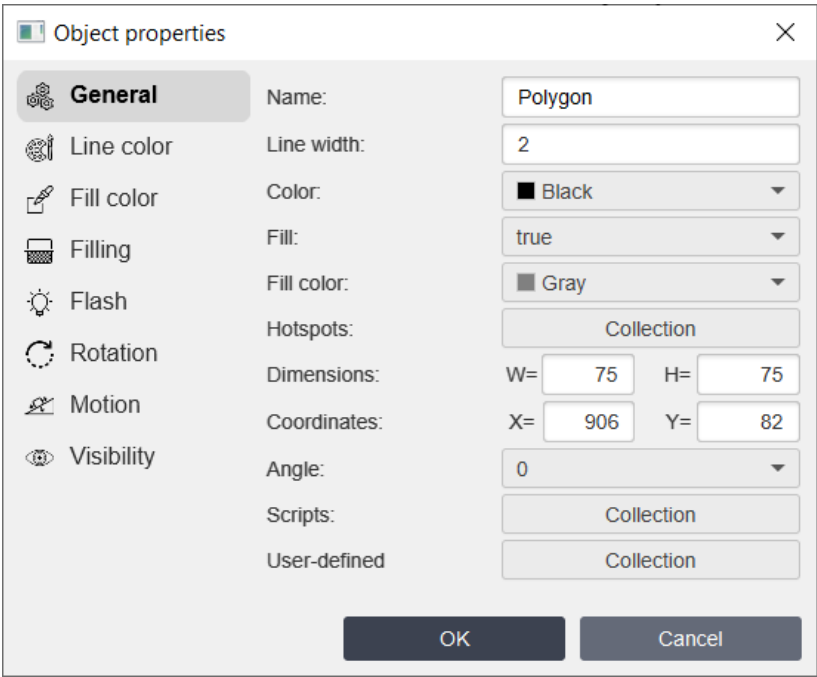
Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Line width	linewidth	Width of the line.
Color	color	Color of the line.
Hotspots		<div>When you click Collection button the Collection window will appear:</div> <div></div> <div>where:</div> <div><ul style="list-style-type: none">▪ CoordinateX - X coordinate of the polyline's node.▪ CoordinateY - Y coordinate of the polyline's node.▪ Add - add a new polyline's node.▪ Edit - edit the polyline's node.▪ Remove - remove the polyline's node.</div> <div>You can also edit polyline's nodes on the Canvas </div>

Property	ST script field	Description
		

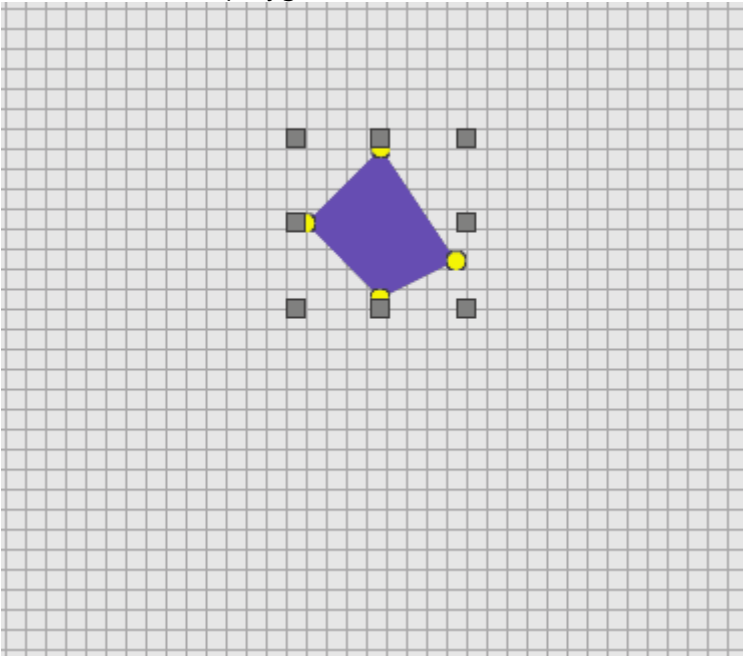
Properties from the **"Line Color"** tab are described [here](#)³⁵⁰.
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.
Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.
Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.1.5 Polygon



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill polygon.
Fill color	fillcolor	Color of the polygon's filling.
Hotspots		When you click Collection button the Collection window will appear:

Property	ST script field	Description
		<div><div><div><div>Collection</div><div><div><div>{0.0, 37.5}</div><div>{37.5, 0.0}</div><div>{75.0, 56.25}</div><div>{37.5, 75.0}</div></div><div><div>CoordinateX:</div><div>0</div></div><div><div>CoordinateY:</div><div>37</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div><div><div>where:</div><div><div><div>▪ CoordinateX - X coordinate of the polygon's node.</div><div>▪ CoordinateY - Y coordinate of the polygon's node.</div><div>▪ Add - add a new polygon's node.</div><div>▪ Edit - edit the polygon's node.</div><div>▪ Remove - remove the polygon's node.</div></div><div><div>You can also edit polygon's nodes on the Canvas:</div><div></div></div></div></div></div>

Properties from the "Line Color" tab are described [here](#)³⁵⁰.
Properties from the "Fill Color" tab are described [here](#)³⁵².
Properties from the "Filling" tab are described [here](#)³⁵⁴.

Properties from the **"Flash"** tab are described [here](#)^[345].
 Properties from the **"Rotation"** tab are described [here](#)^[347].
 Properties from the **"Motion"** tab are described [here](#)^[348].
 Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.1.6 Sector

The screenshot shows the 'Object properties' dialog box for a 'Sector' object. The 'General' tab is active. The following properties are visible:

- Name:** Sector
- Line width:** 2
- Color:** Black
- Fill:** true
- Fill color:** Gray
- Start angle:** 90.0
- Rotation angle:** 180.0
- Dimensions:** W= 75, H= 75
- Coordinates:** X= 796, Y= 292
- Angle:** 0
- Scripts:** Collection
- User-defined:** Collection

Buttons for 'OK' and 'Cancel' are located at the bottom right.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143])

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill sector.
Fill color	fillcolor	Color of the sector's filling.
Start angle	startangle	Start angle of the sector. 0 degrees is the right middle point of the dimensions rectangle.
Rotation angle	rotationangle	Counterclockwise rotation angle of the sector.

Properties from the **"Line Color"** tab are described [here](#)^[350].
 Properties from the **"Fill Color"** tab are described [here](#)^[352].
 Properties from the **"Flash"** tab are described [here](#)^[345].
 Properties from the **"Rotation"** tab are described [here](#)^[347].

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.1.7 Text/EditField

The screenshot shows the 'Object properties' dialog box for a 'Text/EditField' object. The 'General' tab is active, displaying a list of properties on the left and their values on the right. The properties include Name, Text, Font type, Underline, Font size, Text placement, Text color, Border, Border width, Border color, Fill, Fill color, Dimensions, Coordinates, Angle, Scripts, and User-defined. The values are: Name: Text/EditField1, Text: Label, Font type: Roboto Regular, Underline: false, Font size: 30, Text placement: CENTER, Text color: Blue, Border: false, Border width: 2, Border color: Black, Fill: false, Fill color: White, Dimensions: 75x75, Coordinates: X=1031, Y=98, Angle: 0, Scripts: Collection, and User-defined: Collection. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Text	text	Text displayed on the screen by using this object.
Font type	fonttype	Type of the text's font.
Underline	underline	Check if you want to underline the text.
Font size	fontsize	Size of the text's font.
Text placement	textplacement	Placement of the text: <ul style="list-style-type: none"> ▪ Left ▪ Center ▪ Right

Property	ST script field	Description
Text color	textcolor	Color of the text.
Border	useborder	Select use or not use border for the text.
Border width	linewidth	Width of the border's line.
Border color	bordercolor	Color of the border's line.
Fill	fill	Select fill or not fill text's background.
Fill color	fillcolor	Color of the text's background.

Also for all text/editfield objects you can use fields in ST scripts:

- **textbefore** - text before the value.
- **textafter** - text after the value.
- **decimalpos** - decimal position for the value. Properties from the "**Line Color**" tab are described [here](#)³⁵⁰.

Properties from the "**Text input**" tab are described [here](#)³⁵⁹.

Properties from the "**Output value**" tab are described [here](#)³⁶².

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Line Color**" tab are described [here](#)³⁵⁰.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.1.8 Border

Object properties

General

Name: Border

Line width: 2

Inner: true

Fill: true

Fill color: Gray

Dimensions: W= 75 H= 75

Coordinates: X= 863 Y= 242

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Line width	linewidth	Width of the border.
Inner	inner	Select style of the border - Inner or not.
Fill	fill	Select fill or not fill the border.
Fill color	fillcolor	Color of the border.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

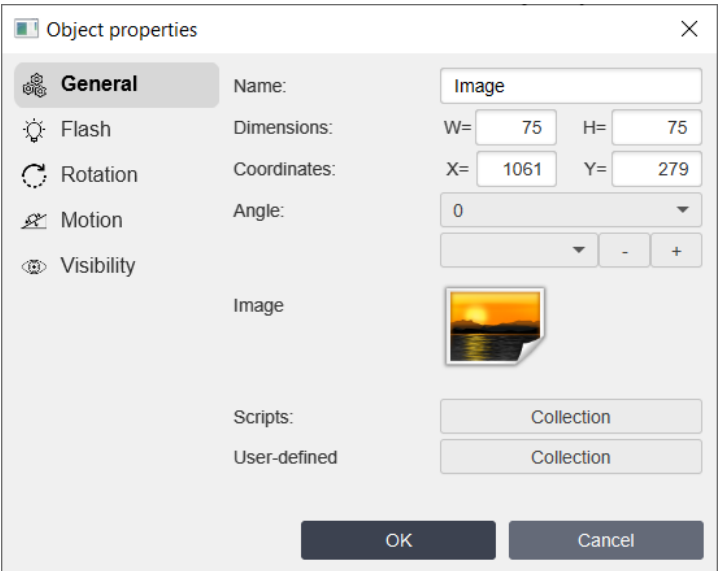
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.1.9 Image



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Image		Select image you want to add to the project by clicking "+" button. File dialog will appeare. Choose file with image you want to add to the project and click Open button. You can use GIF files if you want.

Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.1.10 Scale

Object properties

General

Name: Scale

Line color: Line width: 2

Flash: Color: Black

Rotation: Border: false

Motion: Scale №2: true

Visibility: Scale №3: true

Scale interval №1: 2

Scale interval №2: 4

Scale interval №3: 2

Marker №1 size: 30

Marker №2 size: 20

Marker №3 size: 10

Type: Left

☐ Use digital:

Minimum: 0.0

Maximum: 100.0

Font size: 0

Decimal position: 0

Dimensions: W= 75 H= 75

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³)

Property	ST script field	Description
Line width	linewidth	Width of the line.
Color	color	Color of the border and scale lines.
Border	useborder	Select use or not use border for the scale.
Scale ? 2	scale2	Select use or not second scale.
Scale ? 3	scale3	Select use or not third scale.

Property	ST script field	Description
Scale interval ? 1	scaleinterval 1	Interval of the main scale.
Scale interval ? 2	scaleinterval 2	Interval of the second scale.
Scale interval ? 3	scaleinterval 3	Interval of the third scale.
Marker ? 1 size	sizemarkers1	Width of the main scale.
Marker ? 2 size	sizemarkers2	Width of the second scale.
Marker ? 3 size	sizemarkers3	Width of the third scale.
Type	type	Type of the scale: <ul style="list-style-type: none"> ▪ Left ▪ Right ▪ Top ▪ Bottom
Use digital	usedigit	Check if you want to bind numeration to the main scale.
Minimum	min	Minimum value for the main scale.
Maximum	max	Maximum value for the main scale.
Decimal position	decimalpos	Decimal position for the scale numbers.

Properties from the "Line Color" tab are described [here](#)³⁵⁰.

Properties from the "Flash" tab are described [here](#)³⁴⁵.

Properties from the "Rotation" tab are described [here](#)³⁴⁷.

Properties from the "Motion" tab are described [here](#)³⁴⁸.

Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.1.11 Text Area

Object properties

General

Name: Text Area

Text: Text

Font type: Roboto Regular

Font size: 30

Use file: ☐

Filename:

Editable: ☐

Dimensions: W= 75 H= 75

Coordinates: X= 209 Y= 312

Angle: 0

Scripts: Collection

User-defined: Collection

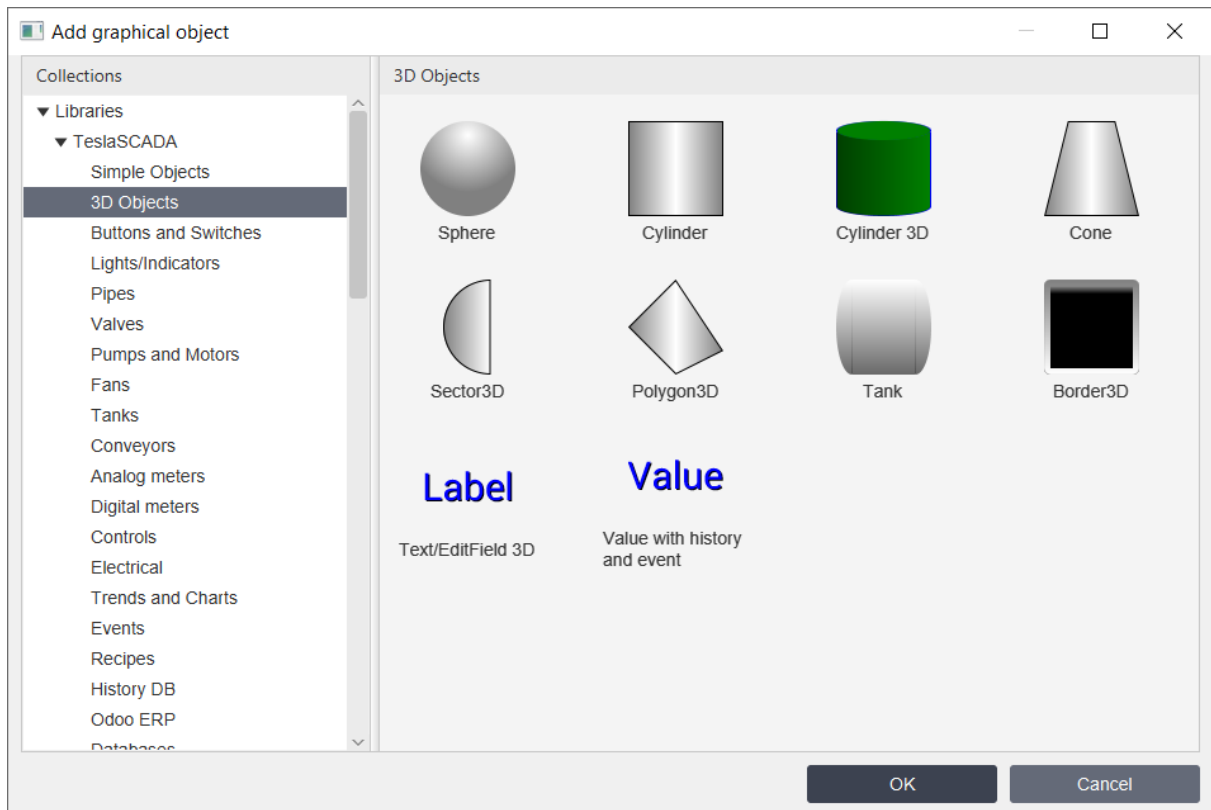
OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Text	text	Text displayed on the screen by using this object.
Font type	fonttype	Type of the text's font.
Font size	fontsize	Size of the text's font.
Use file	usefile	Use or not file to load it in the text area.
Filename	filename	Name of the file you want to load in the text area. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in DB ¹⁴⁸⁾ folder of the application.
Editable	editable	Check if you want to edit the text area.

Properties from the **"Flash"** tab are described [here](#)^[345].
Properties from the **"Rotation"** tab are described [here](#)^[347].
Properties from the **"Motion"** tab are described [here](#)^[348].
Properties from the **"Visibility"** tab are described [here](#)^[349].

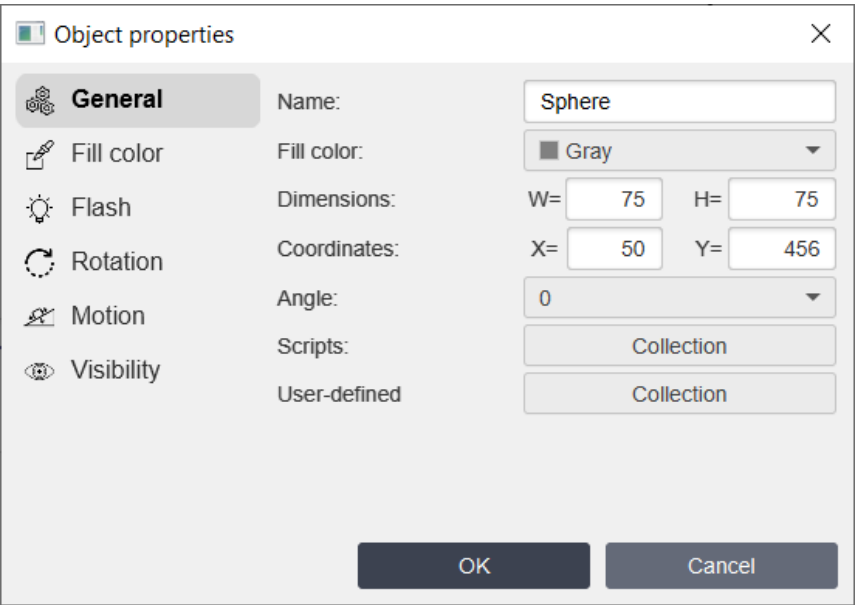
6.2.3.2 3D Objects library



3D objects library contains the following objects:

- [Sphere](#)^[167]
- [Cylinder](#)^[168]
- [Cylinder 3D](#)^[168]
- [Cone](#)^[169]
- [Sector 3D](#)^[170]
- [Polygon 3D](#)^[171]
- [Tank](#)^[173]
- [Border 3D](#)^[174]
- [Text/EditField 3D](#)^[175]
- [Value with History and Event](#)^[177]

6.2.3.2.1 Sphere

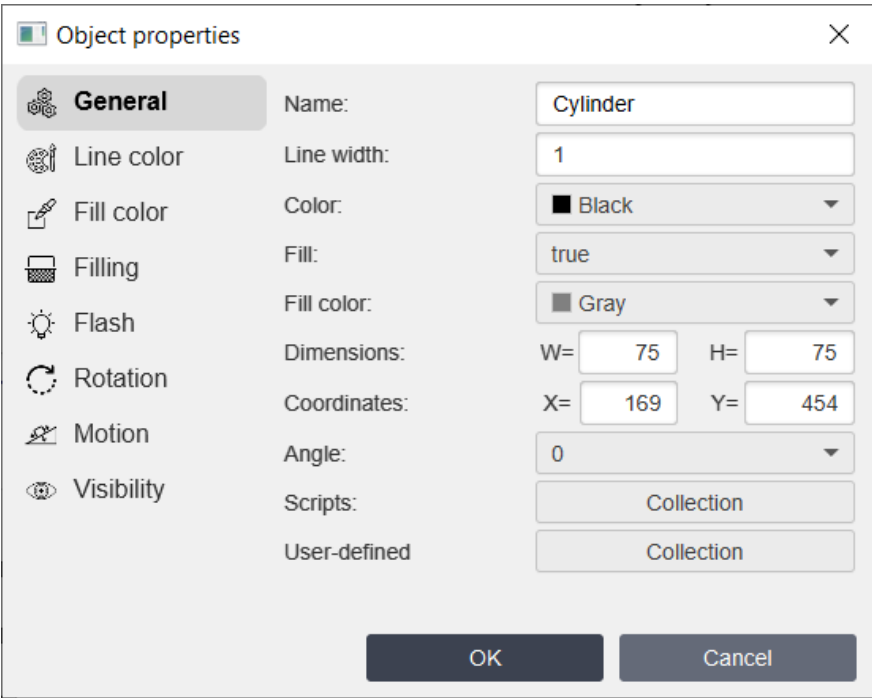


Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Fill color	fillcolor	Color of the sphere.

- Properties from the "Fill Color" tab are described [here](#)³⁵²⁾.
- Properties from the "Flash" tab are described [here](#)³⁴⁵⁾.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷⁾.
- Properties from the "Motion" tab are described [here](#)³⁴⁸⁾.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹⁾.

6.2.3.2.2 Cylinder

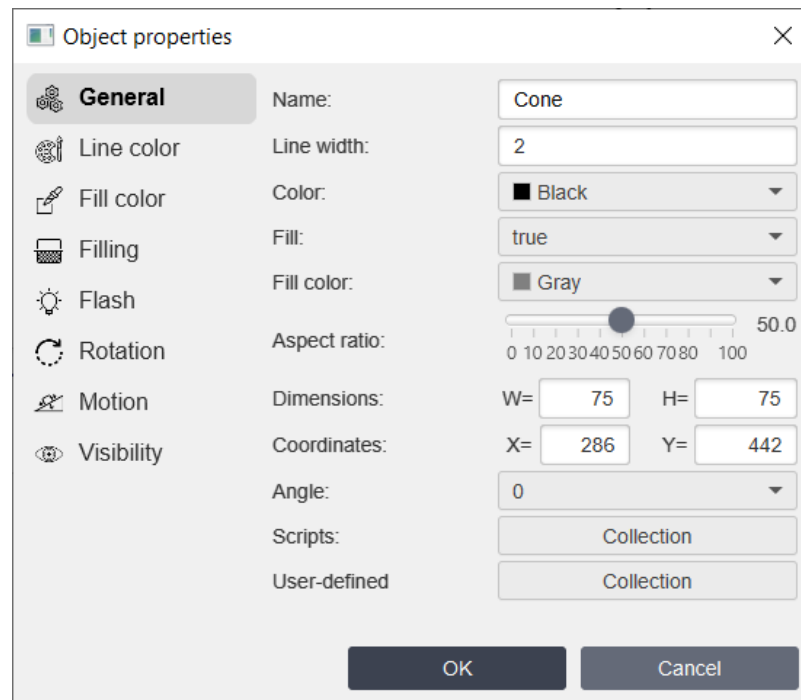


Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill cylinder.
Fill color	fillcolor	Fill color of the cylinder.

Properties from the "Line Color" tab are described [here](#)³⁵⁰.
Properties from the "Fill Color" tab are described [here](#)³⁵².
Properties from the "Filling" tab are described [here](#)³⁵⁴.
Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.2.3 Cone



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill cone.
Fill color	fillcolor	Fill color of the cone.
Aspect ratio	aspectratio	Aspect ratio of the cone.

Properties from the "**Line Color**" tab are described [here](#)³⁵⁰⁾.

Properties from the "**Fill Color**" tab are described [here](#)³⁵²⁾.

Properties from the "**Filling**" tab are described [here](#)³⁵⁴⁾.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵⁾.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷⁾.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸⁾.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹⁾.

6.2.3.2.4 Sector 3D

The screenshot shows the 'Object properties' dialog box for an object named 'Sector3D'. The 'General' tab is selected, showing various configuration options. The 'Line color' is set to 'Black', 'Fill' is 'true', and 'Fill color' is 'Gray'. The 'Start angle' is 90.0 and the 'Rotation angle' is 180.0. Dimensions are set to W=75 and H=75. Coordinates are X=398 and Y=445. The 'Angle' is set to 0. There are buttons for 'Collection' and 'User-defined' scripts, and 'OK' and 'Cancel' buttons at the bottom.

Property	Value
Name	Sector3D
Line width	1
Line color	Black
Fill	true
Fill color	Gray
Start angle	90.0
Rotation angle	180.0
Dimensions (W)	75
Dimensions (H)	75
Coordinates (X)	398
Coordinates (Y)	445
Angle	0
Scripts (Collection)	Collection
Scripts (User-defined)	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill sector.
Fill color	fillcolor	Color of the sector's filling.
Start angle	startangle	Start angle of the sector. 0 degrees is the right middle point of the dimensions rectangle.
Rotation angle	rotationangle	Counterclockwise rotation angle of the sector.

Properties from the "Line Color" tab are described [here](#)³⁵⁰.

Properties from the "Fill Color" tab are described [here](#)³⁵².

Properties from the "Flash" tab are described [here](#)³⁴⁵.

Properties from the "Rotation" tab are described [here](#)³⁴⁷.

Properties from the "Motion" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.2.5 Polygon 3D

Object properties

General

Name: Polygon3D

Line width: 2

Color: Black

Fill: true

Fill color: Gray

Hotspots: Collection

Dimensions: W= 75 H= 75

Coordinates: X= 462 Y= 441

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill polygon.
Fill color	fillcolor	Color of the polygon's filling.
Hotspots		When you click Collection button the Collection window will appear:

Property	ST script field	Description
		<div><div><div><div>Collection</div><div><div><div>{0.0, 37.5}</div><div>{37.5, 0.0}</div><div>{75.0, 56.25}</div><div>{37.5, 75.0}</div></div><div><div>CoordinateX:</div><div>0</div></div><div><div>CoordinateY:</div><div>37</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div><div><p>where:</p><ul style="list-style-type: none">• CoordinateX - X coordinate of the polygon's node.• CoordinateY - Y coordinate of the polygon's node.• Add - add a new polygon's node.• Edit - edit the polygon's node.• Remove - remove the polygon's node.<p>You can also edit polygon's nodes on the Canvas⁹¹:</p><div></div></div></div></div>

Properties from the "Line Color" tab are described [here](#)³⁵⁰.
Properties from the "Fill Color" tab are described [here](#)³⁵².
Properties from the "Filling" tab are described [here](#)³⁵⁴.

Properties from the **"Flash"** tab are described [here](#)³⁴⁵.
 Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.
 Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
 Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.2.6 Tank

The screenshot shows the 'Object properties' dialog box for a 'Tank' object. The 'General' tab is active. The properties are as follows:

Property	Value
Name	Tank
Fill color	Light Gray
Ratio	3.0
Vertical	false
Type	3D
Dimensions	W= 75, H= 75
Coordinates	X= 554, Y= 441
Angle	0
Scripts	Collection
User-defined	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Fill color	fillcolor	Color of the tank.
Ratio	ratio	Ratio of the tank.
Vertical	vertical	Select vertical or horizontal tank's type.

Properties from the **"Fill Color"** tab are described [here](#)³⁵².
 Properties from the **"Flash"** tab are described [here](#)³⁴⁵.
 Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.
 Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
 Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.2.7 Border 3D

Object properties

General

Name: Border3D

Line color: Line width: 5

Fill color: Color: Gray

Flash: Fill: true

Rotation: Fill color: Black

Motion: Corner radius: 10.0

Visibility: Glass: true

Dimensions: W= 75 H= 75

Coordinates: X= 647 Y= 444

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143])

Property	ST script field	Description
Line width	linewidth	Width of the border.
Color	color	Color of the border.
Fill	fill	Select fill or not fill the border.
Fill color	fillcolor	Fill color of the border.
Corner radius	cornerradius	Radius of the border's corner.
Glass	glass	Select use or not glass effect.

Properties from the "**Line Color**" tab are described [here](#)^[350].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.2.8 Text/EditField 3D

The screenshot shows the 'Object properties' dialog box for the 'Text/EditField 3D' object. The 'General' tab is selected, showing various configuration options. The 'Name' is 'Text/EditField 3D', 'Text' is 'Label', 'Font type' is 'Roboto Regular', 'Underline' is unchecked, 'Font size' is '30', 'Text placement' is 'CENTER', 'Text color' is 'Blue', 'Border' is 'false', 'Border width' is '8', 'Border color' is 'Gray', 'Fill' is 'false', 'Fill color' is 'Black', 'Dimensions' are '75' by '75', 'Coordinates' are 'X= 742' and 'Y= 432', 'Angle' is '0', 'Scripts' is 'Collection', and 'User-defined' is 'Collection'. The 'OK' and 'Cancel' buttons are at the bottom.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)⁽¹⁴³⁾)

Property	ST script field	Description
Text	text	Text displayed on the screen by using this object.
Font type	fonttype	Type of the text's font.
Underline	underline	Check if you want to underline the text.
Font size	fontsize	Size of the text's font.
Text placement	textplacement	Placement of the text: <ul style="list-style-type: none"> ▪ Left ▪ Center ▪ Right
Text color	textcolor	Color of the text.

Property	ST script field	Description
Border	useborder	Select use or not use border for the text.
Border width	linewidth	Width of the border's line.
Border color	bordercolor	Color of the border's line.
Fill	fill	Select fill or not fill text's background.
Fill color	fillcolor	Color of the text's background.

Also for all text/editfield objects you can use fields in ST scripts:

- **textbefore** - text before the value.
- **textafter** - text after the value.
- **decimalpos** - decimal position for the value.

Properties from the "**Text input**" tab are described [here](#)³⁵⁹.

Properties from the "**Output value**" tab are described [here](#)³⁶².

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Line Color**" tab are described [here](#)³⁵⁰.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.2.9 Value with history and event

Object properties

General

Name: Value with history and event

Text: Value

Font type: Roboto Regular

Font size: 30

Text placement: Center

Text color: Blue

Border: false

Border width: 2

Border color: Black

Fill: false

Fill color: White

Type: 3D

Dimensions: W= 75 H= 75

Coordinates: X= 838 Y= 428

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Text	text	Text displayed on the screen by using this object.
Font type	fonttype	Type of the text's font.
Font size	fontsize	Size of the text's font.
Text placement	textplacement	Placement of the text: <ul style="list-style-type: none"> ▪ Left ▪ Center ▪ Right
Text color	textcolor	Color of the text.

Property	ST script field	Description
Border	useborder	Select use or not use border for the text.
Border width	linewidth	Width of the border's line.
Border color	bordercolor	Color of the border's line.
Fill	fill	Select fill or not fill text's background.
Fill color	fillcolor	Color of the text's background.

Properties from the **"Grid"** tab are described [here](#) ¹⁷⁹.

Properties from the **"Text input"** tab are described [here](#) ³⁵⁹.

Properties from the **"Text Color"** tab are described [here](#) ³⁵⁵.

Properties from the **"Line Color"** tab are described [here](#) ³⁵⁰.

Properties from the **"Fill Color"** tab are described [here](#) ³⁵².

Properties from the **"Flash"** tab are described [here](#) ³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#) ³⁴⁷.

Properties from the **"Motion"** tab are described [here](#) ³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#) ³⁴⁹.

6.2.3.2.9.1 Grid

Object properties

General

Grid

Text input

Text color

Line color

Fill color

Flash

Rotation

Motion

Visibility

Side: Right

Line width: 1

Curve color: Blue

Line style: Solid

Horizontally: 6

Vertically: 5

Grid width: 225

Grid height: 225

Font size: 10

Mark color: Black

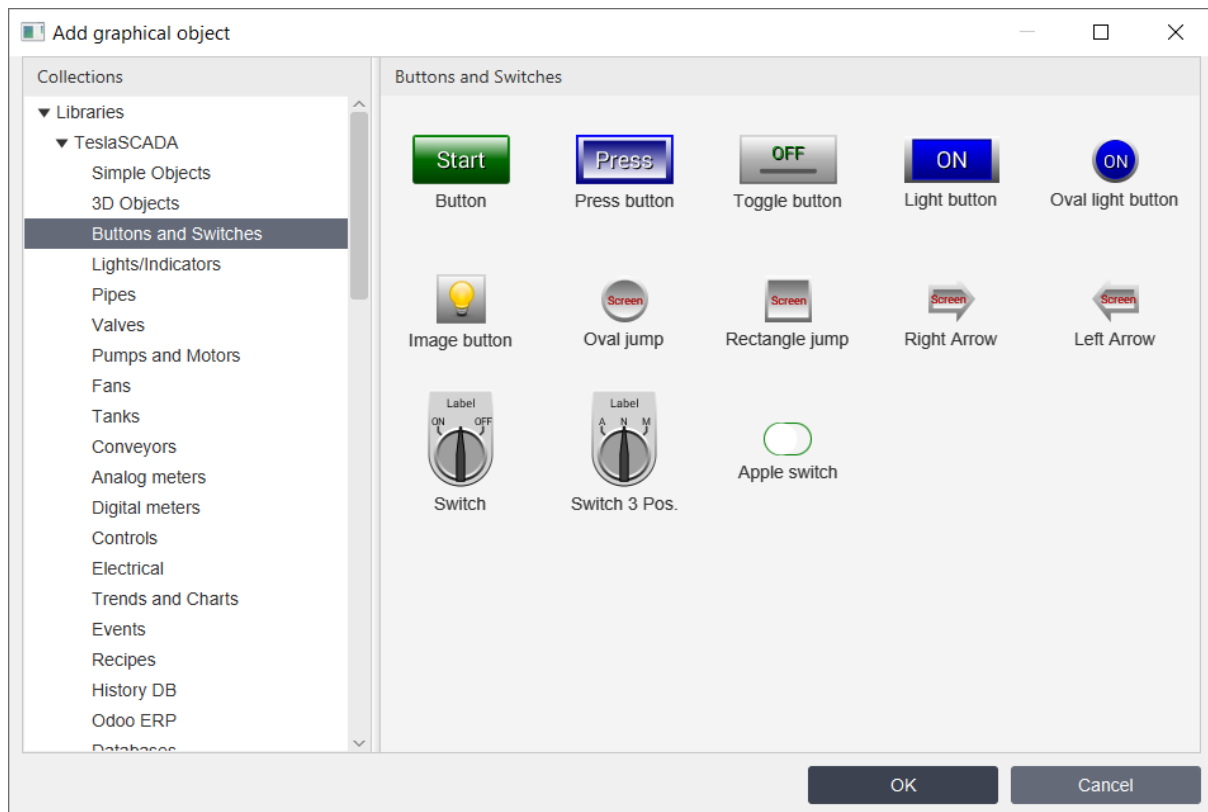
Time format: HH:mm

OK Cancel

Property	ST script field	Description
Side	side	Choose side of placement of the trend and event table: <ul style="list-style-type: none"> ▪ Right ▪ Left ▪ Top ▪ Bottom ▪ RightTop ▪ LeftTop
Line width		Line width of the curve.
Curve color	gridlinecolor	Choose curve's color
Line style	linestyle	Style of the line: <ul style="list-style-type: none"> ▪ Solid ▪ Dash ▪ Dot

Property	ST script field	Description
		▪ DashDot
Horizontally	horizontally	Number of trend's horizontal grid lines.
Vertically	vertically	Number of trend's vertical grid lines.
Grid width	gridwidth	Width of the trend and event table.
Grid height	gridheight	Height of the trend and event table.
Font size	fontsize	Font size of the trend's marks.
Mark color	markcolor	Color of the marks.
Time format	timeformat	Time format of the trend's time.

6.2.3.3 Buttons and Switches library



Buttons and Switches library contains the following objects:

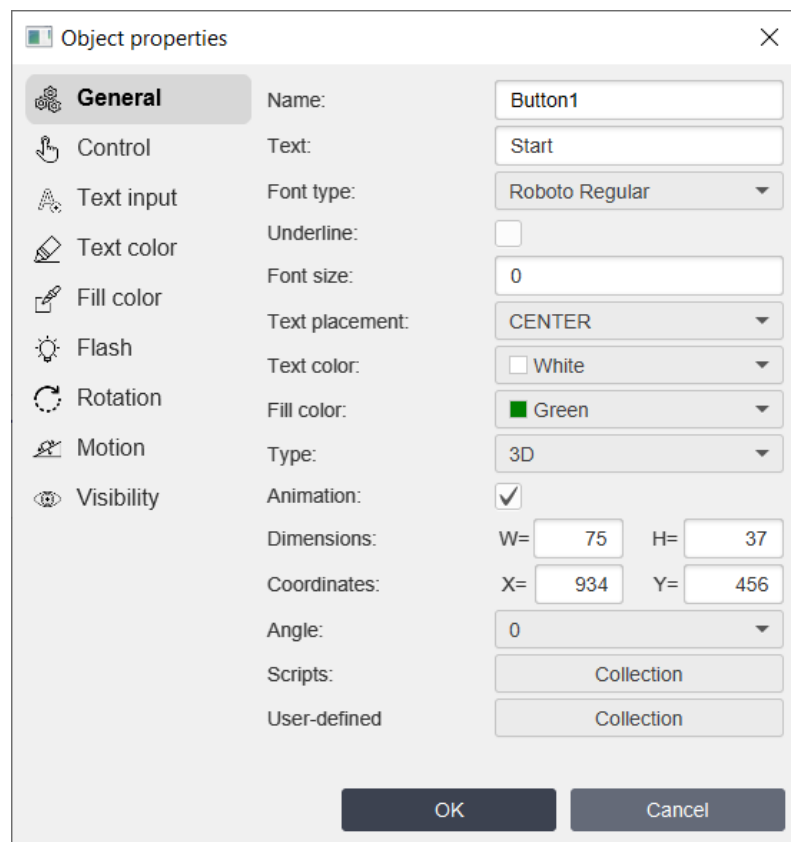
- [Button](#) ¹⁸¹
- [Press button](#) ¹⁸¹
- [Toggle button](#) ¹⁸¹
- [Light button](#) ¹⁸¹
- [Oval light button](#) ¹⁸¹
- [Image button](#) ¹⁸³

- [Oval jump button](#)¹⁸¹
- [Rectangle jump button](#)¹⁸¹
- [Right Arrow](#)¹⁸¹
- [Left Arrow](#)¹⁸¹
- [Switch](#)¹⁸⁴
- [Switch 3 Pos](#)¹⁸⁶
- [Apple switch](#)¹⁸⁵

All Buttons and Arrows except Image button have the same General group properties. Below we describe there only for 5 graphical objects - Button, Image button, Switch, Apple switch and Three position Switch.

6.2.3.3.1 Button

This section applies to the following objects: Button, Press button, Light button, Oval light button, Oval jump button , Rectangle jump button, Right Arrow, Left Arrow.



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Text	text	Text displayed on the button.
Font type	fonttype	Type of the button text's font.
Underline	underline	Check if you want to underline the text.
Font size	fontsize	Size of the button text's font.
Text placement	textplacement	Placement of the button text: <ul style="list-style-type: none"> ▪ Left ▪ Center ▪ Right
Text color	textcolor	Color of the text.
Fill color	fillcolor	Color of the button.

Properties from the "**Control**" tab are described [here](#)³⁵⁷.

Properties from the "**Text input**" tab are described [here](#)³⁵⁹.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

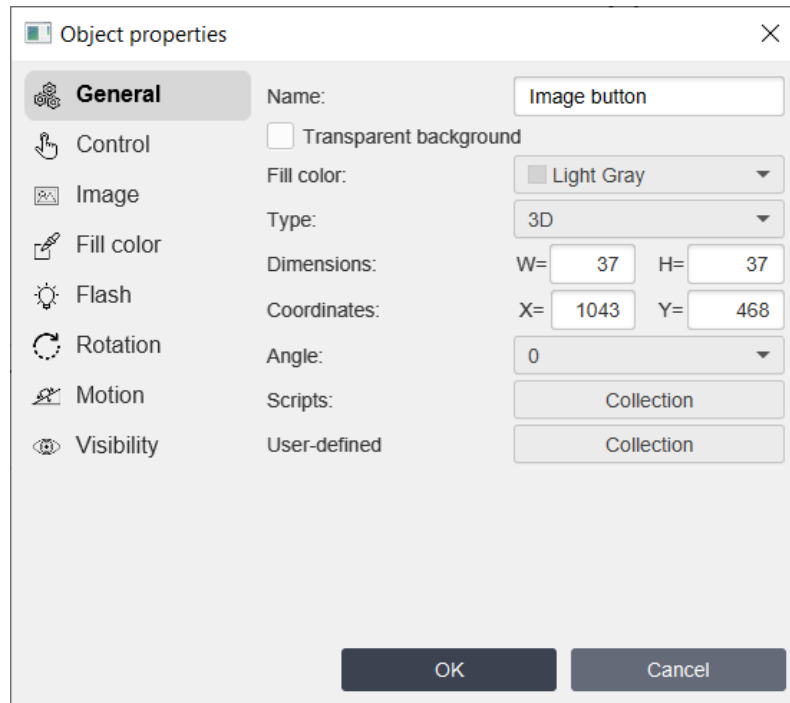
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.3.2 Image button



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Transparent background		Make background transparent.
Fill color	fillcolor	Color of the button.

Properties from the "**Control**" tab are described [here](#)³⁵⁷⁾.

Properties from the "**Image**" tab are described [here](#)³⁶⁴⁾.

Properties from the "**Fill Color**" tab are described [here](#)³⁵²⁾.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵⁾.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷⁾.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸⁾.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹⁾.

6.2.3.3.3 Switch

Object properties

General

Name: Switch

Text: Label

Text color: Black

Fill color: Light Gray

Text ON: ON

Text OFF: OFF

Type: 3D

Dimensions: W= 50 H= 75

Coordinates: X= 1099 Y= 444

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143])

Property	ST script field	Description
Text	text	Text displayed on the switch.
Text color	textcolor	Color of the text.
Fill color	fillcolor	Color of the switch background.
Text ON	texton	Label for ON position of the switch.
Text OFF	textoff	Label for OFF position of the switch.

Properties from the "**Switch control**" tab are described [here](#)^[372].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.3.4 Apple switch

Object properties

General

Name: Apple switch

Text:

Text color: Black

Fill color: Green

Text ON:

Text OFF:

Type: 2D

Function: Toggle

Dimensions: W= 37 H= 25

Coordinates: X= 1198 Y= 460

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143])

Property	ST script field	Description
Text	text	Text displayed on the switch.
Text color	textcolor	Color of the text.
Fill color	fillcolor	Color of the switch background.
Text ON	texton	Label for ON position of the switch.
Text OFF	textoff	Label for OFF position of the switch.
Function	clicktype	Choose Function type: <ul style="list-style-type: none"> ▪ Toggle ▪ Push

Properties from the "**Switch control**" tab are described [here](#)^[372].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the **"Motion"** tab are described [here](#)^[348].

Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.3.5 Three position switch

The screenshot shows the 'Object properties' dialog box for a 'Three position switch'. The 'General' tab is selected. The properties are as follows:

Property	Value
Name:	Switch 3 Pos.
Text:	Label
Text color:	Black
Fill color:	Light Gray
Text ON:	A
Text OFF:	M
Text Neutral:	N
Type:	3D
Dimensions:	W= 50 H= 75
Coordinates:	X= 1270 Y= 462
Angle:	0
Scripts:	Collection
User-defined	Collection

Buttons: OK, Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143])

Property	ST script field	Description
Text	text	Text displayed on the switch.
Text color	textcolor	Color of the text.
Fill color	fillcolor	Color of the switch background.
Text ON	texton	Label for ON position of the switch.
Text OFF	textoff	Label for OFF position of the switch.
Text Neutral	textneutral	Label for Neutral position of the switch.

Properties from the **"Switch control"** tab are described [here](#)^[373].

Properties from the **"Fill Color"** tab are described [here](#)^[352].

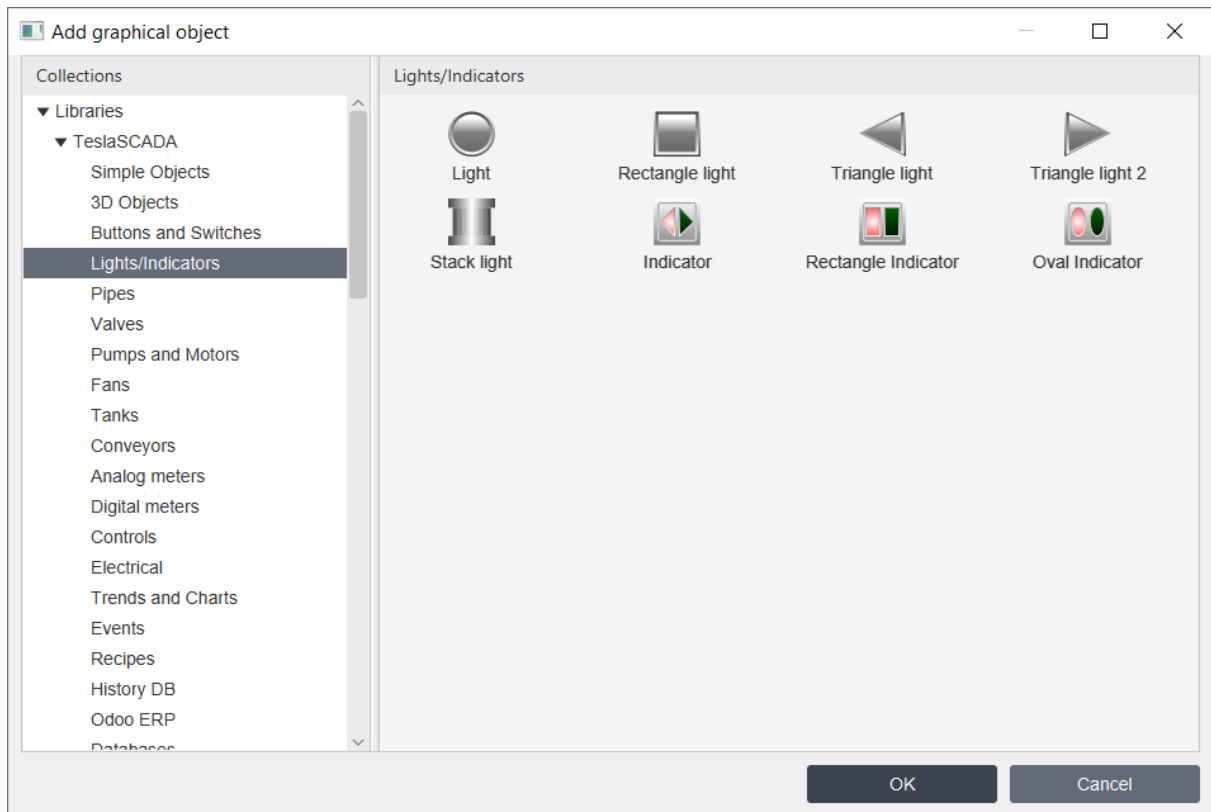
Properties from the **"Flash"** tab are described [here](#)^[345].

Properties from the **"Rotation"** tab are described [here](#)^[347].

Properties from the **"Motion"** tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.4 Lights/Indicators library

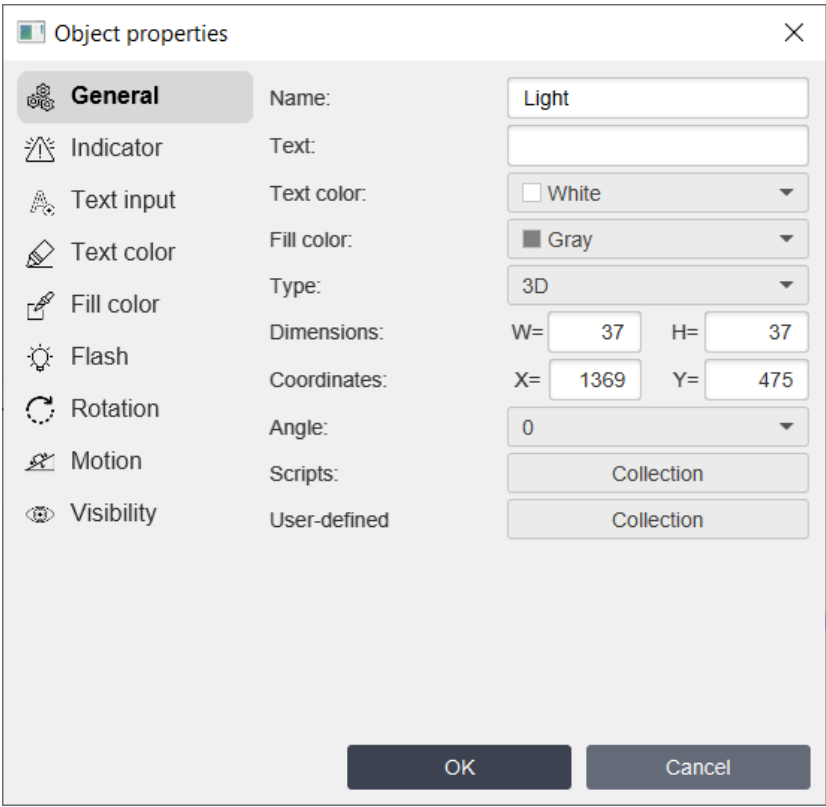


Lights/Indicators library contains the following objects:

- [Light](#)¹⁸⁸
- [Rectangle light](#)¹⁸⁸
- [Triangle light](#)¹⁸⁸
- [Triangle light 2](#)¹⁸⁸
- [Stack light](#)¹⁸⁸
- [Indicator](#)¹⁸⁹
- [Rectangle Indicator](#)¹⁸⁹
- [Oval Indicator](#)¹⁸⁹

All lights have the same General group properties and all indicators have the same General group properties. Below we'll describe them only for two graphical objects - Light and Indicator.

6.2.3.4.1 Light



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Text	text	Text displayed on the light.
Text color	textcolor	Color of the text.
Fill color	fillcolor	Color of the light.

- Properties from the "Indicator" tab are described [here](#)³⁶²⁾.
- Properties from the "Text input" tab are described [here](#)³⁵⁹⁾.
- Properties from the "Text Color" tab are described [here](#)³⁵⁵⁾.
- Properties from the "Fill Color" tab are described [here](#)³⁵²⁾.
- Properties from the "Flash" tab are described [here](#)³⁴⁵⁾.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷⁾.
- Properties from the "Motion" tab are described [here](#)³⁴⁸⁾.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹⁾.

6.2.3.4.2 Indicator

Object properties

General

Indicator

Flash

Rotation

Motion

Visibility

Name:

Color TRUE:

Color FALSE:

Type:

Dimensions:

Coordinates:

Angle:

Scripts:

User-defined

Indicator

Green

Red

3D

W= 37H= 37

X= 85Y= 646

0

Collection

Collection

OK

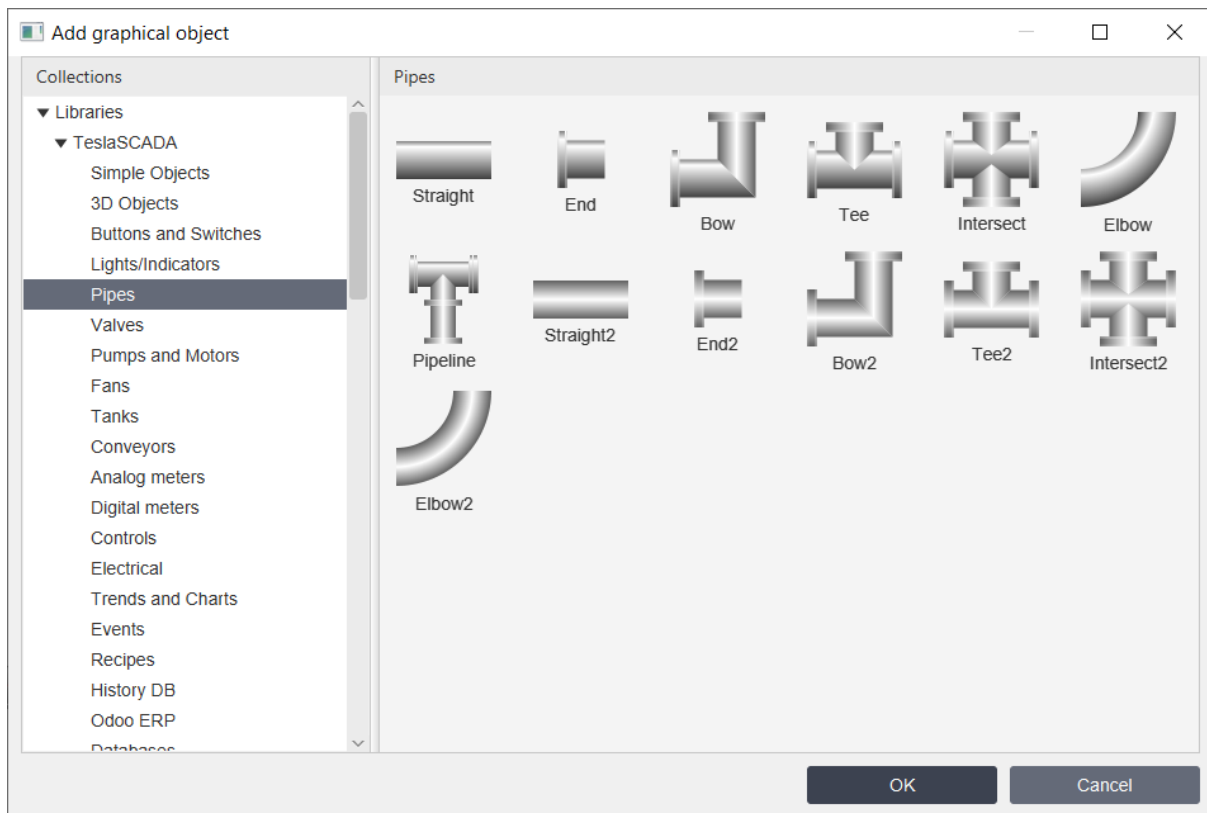
Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)














Property	ST script field	Description
Color TRUE	truecolor	Color TRUE of the indicator.
Color FALSE	falsecolor	Color FALSE of the indicator.

- Properties from the "Indicator" tab are described [here](#)³⁶².
- Properties from the "Flash" tab are described [here](#)³⁴⁵.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷.
- Properties from the "Motion" tab are described [here](#)³⁴⁸.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.5 Pipes library



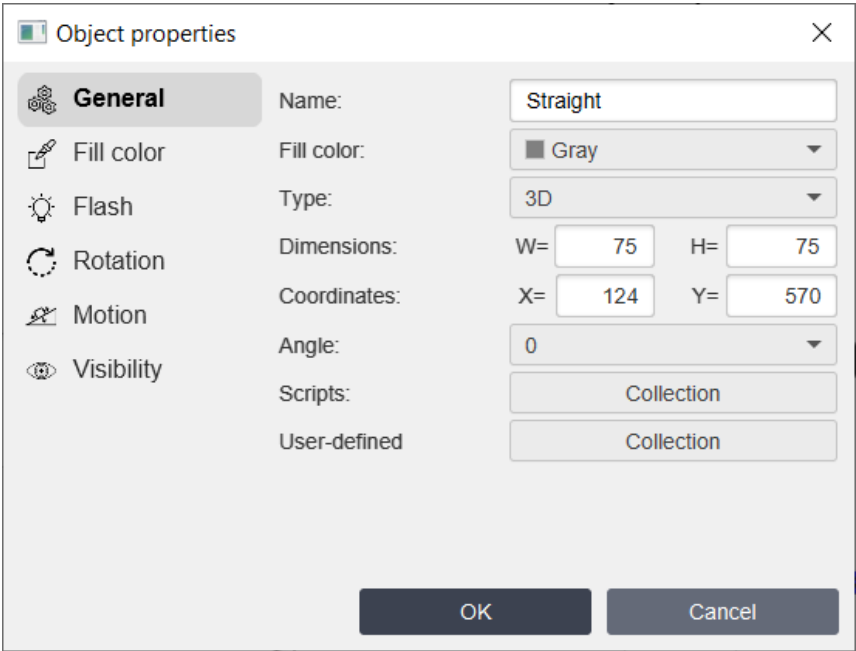
Pipes library contains the following pipes objects:

- [Straight](#)  191
- [End](#)  191
- [Bow](#)  191
- [Tee](#)  191
- [Intersect](#)  191
- [Elbow](#)  191
- [Pipeline](#)  192
- [Straight2](#)  191
- [End2](#)  191
- [Bow2](#)  191
- [Tee2](#)  191
- [Intersect2](#)  191
- [Elbow2](#)  191

All pipes have the same General group properties. Below we'll describe them only for two graphical objects - Straight and Pipeline .

6.2.3.5.1 Pipe

This section applies to the following objects: Straight, End, Bow, Tee, Intersect, Elbow, Straight2, End2, Bow2, Tee2, Intersect2, Elbow2.





Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)


Property	ST script field	Description
Fill color	fillcolor	Color of the pipe.


- Properties from the "Fill Color" tab are described [here](#)³⁵²⁾.
- Properties from the "Flash" tab are described [here](#)³⁴⁵⁾.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷⁾.
- Properties from the "Motion" tab are described [here](#)³⁴⁸⁾.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹⁾.


6.2.3.5.2 Pipeline


 Object properties


 **General**

 Fill color

 Flash

 Rotation

 Motion

 Visibility

Name:

Pipeline

Pipe width:

20

Color:

Gray

Type:

3D

Hotspots:

Collection

Dimensions:

W=75H=75

Coordinates:

X=255Y=560

Angle:

0

Scripts:

Collection

User-defined

Collection

OK

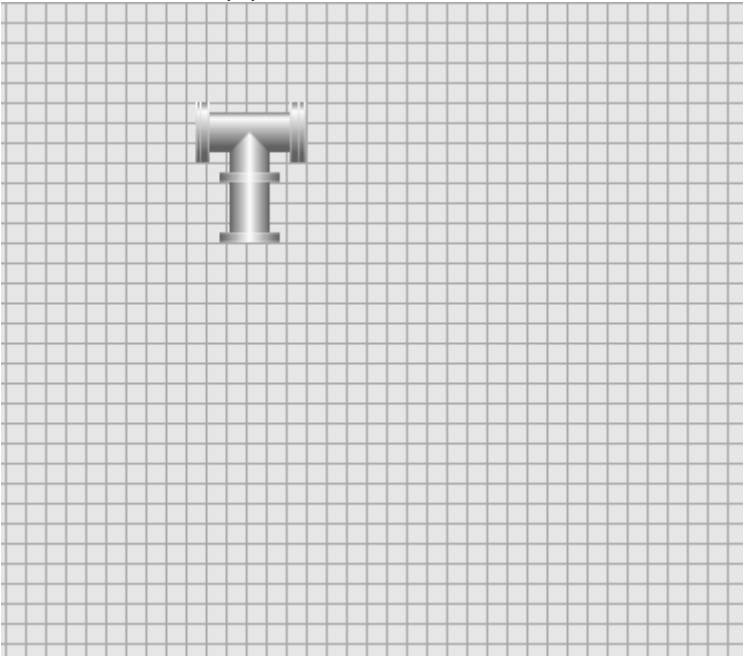
Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#))

Property	ST script field	Description
Pipe width	linewidthh	Pipe width of the pipeline.
Color	fillcolor	Color of the pipeline.
Hotspots		When you click Collection button the Collection window will appear:

TeslaSCADA2 IDE User manual

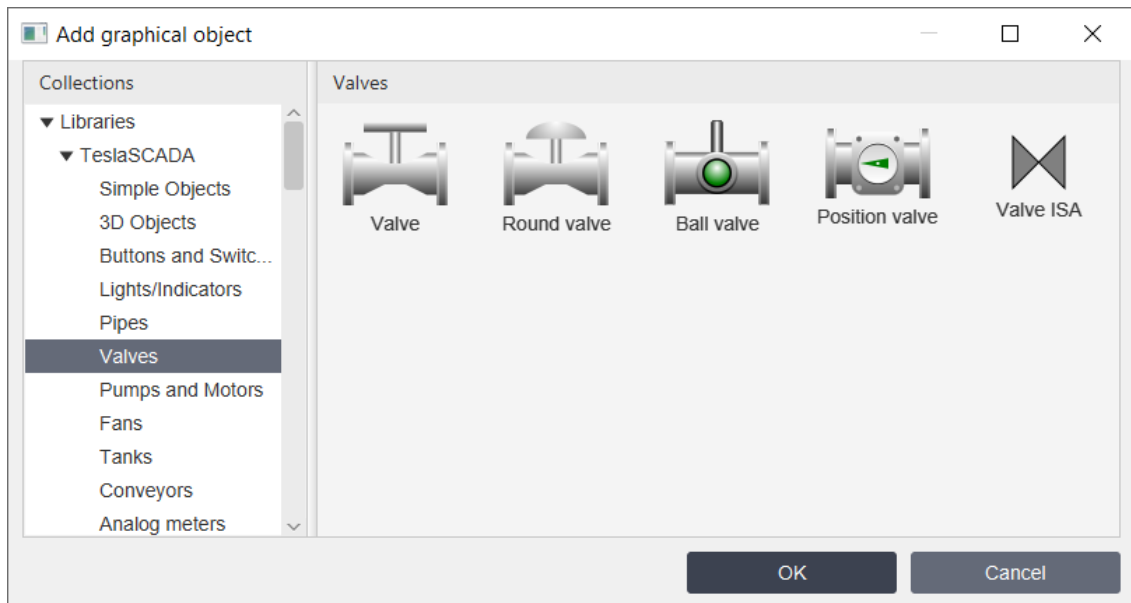
© 2024 LLC Tesla

Property	ST script field	Description
		<div><div><div><div>Collection</div><div><div><div>{0.0, 37.5}</div><div>{37.5, 0.0}</div><div>{75.0, 56.25}</div><div>{37.5, 75.0}</div></div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div><div><div>CoordinateX:</div><div>0</div><div>CoordinateY:</div><div>37</div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div>Close</div></div></div></div><div><div>Where:</div><div><div><div>▪ CoordinateX - X coordinate of the pipeline's node.</div><div>▪ CoordinateY - Y coordinate of the pipeline's node.</div><div>▪ Add - add a new pipeline's node.</div><div>▪ Edit - edit the pipeline's node.</div><div>▪ Remove - remove the pipeline's node.</div></div></div><div><div>You can also edit pipeline's nodes on the Canvas</div><div></div></div></div></div>

Properties from the "Fill Color" tab are described [here](#)³⁵²
Properties from the "Flash" tab are described [here](#)³⁴⁵
Properties from the "Rotation" tab are described [here](#)³⁴⁷

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.6 Valves library



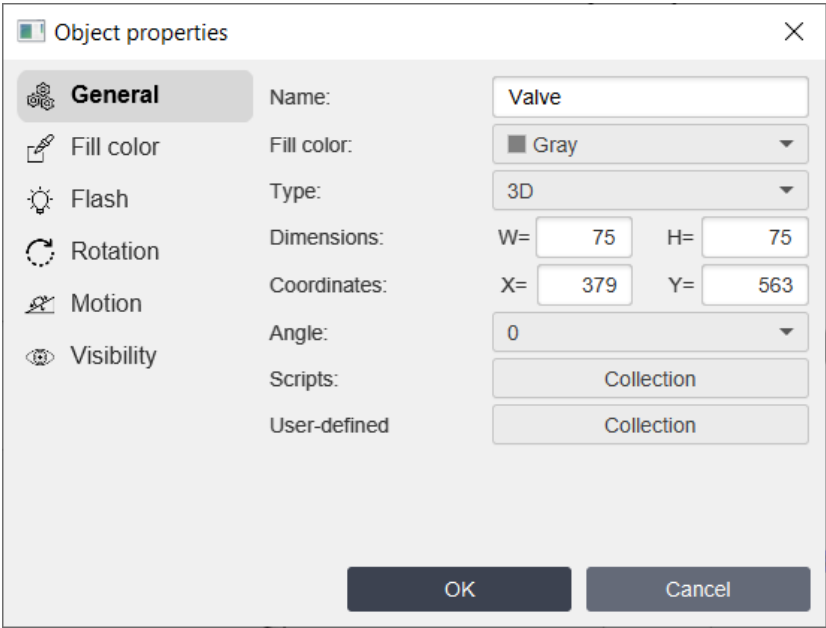
Valves library contains the following objects:

- [Valve](#)¹⁹⁴
- [Round valve](#)¹⁹⁴
- [Ball valve](#)¹⁹⁶
- [Position valve](#)¹⁹⁷
- [Valve ISA](#)¹⁹⁴

Valve, Round valve and Valve ISA have the same General properties.

6.2.3.6.1 Valve

This section applies to the following objects: Valve, Round valve and Valve ISA.

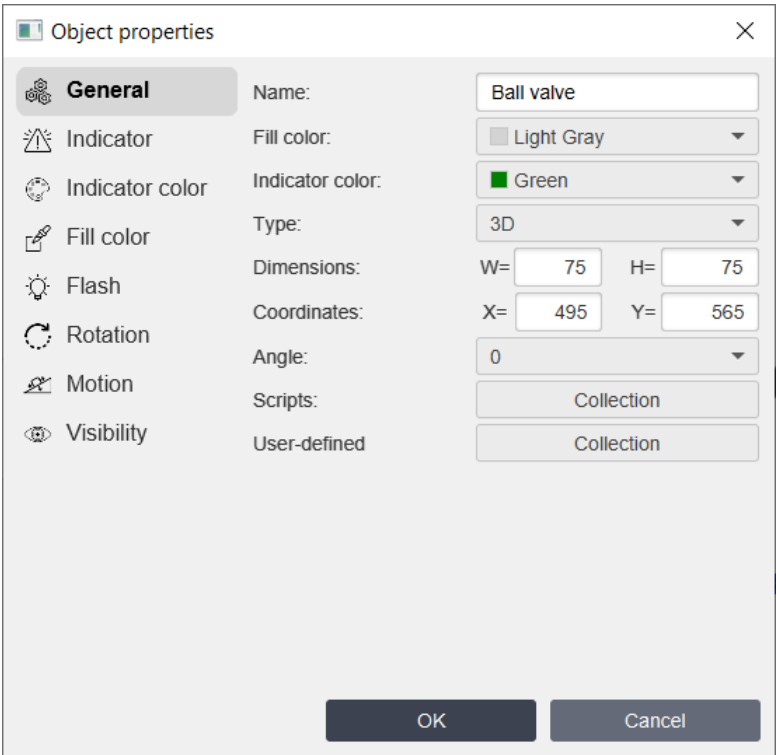


Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Fill color	fillcolor	Color of the valve.

- Properties from the "Fill Color" tab are described [here](#)³⁵².
- Properties from the "Flash" tab are described [here](#)³⁴⁵.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷.
- Properties from the "Motion" tab are described [here](#)³⁴⁸.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.6.2 Ball valve



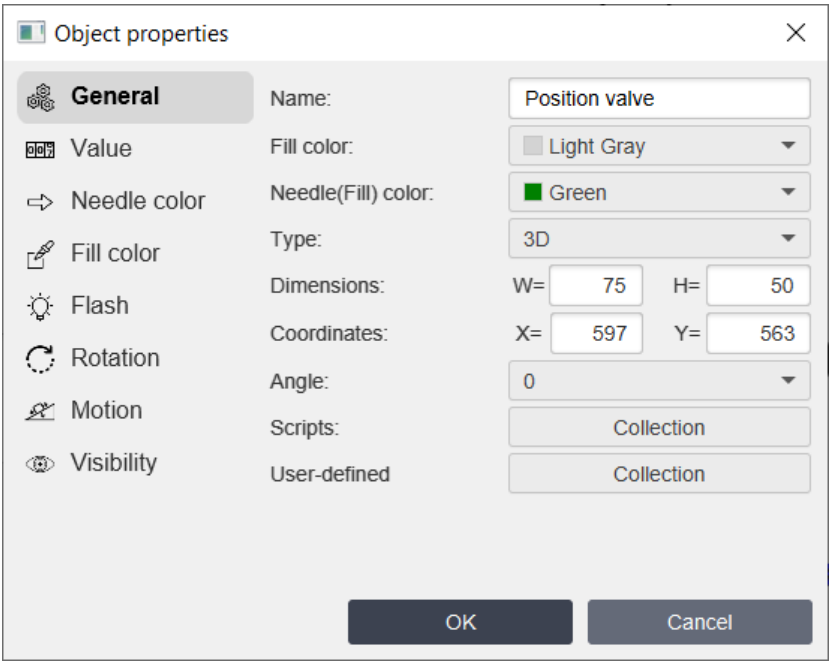
Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Fill color	fillcolor	Color of the valve.
Indicator color	indicatorcolor	Color of the indicator (ball).

Indicator color property like other color properties.

- Properties from the "Indicator" tab are described [here](#)³⁶².
- Properties from the "Indicator color" tab are described [here](#)³⁶⁶.
- Properties from the "Text input" tab are described [here](#)³⁵⁹.
- Properties from the "Fill Color" tab are described [here](#)³⁵².
- Properties from the "Flash" tab are described [here](#)³⁴⁵.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷.
- Properties from the "Motion" tab are described [here](#)³⁴⁸.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.6.3 Position valve



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Fill color	fillcolor	Color of the valve.
Needle(fill) color	indicatorcolor	Color of the needle.

Properties from the "Value" tab are the same as for analog meters and described [here](#)³⁶⁹.

Properties from the "Needle color" tab are described [here](#)³⁶⁶.

Properties from the "Fill Color" tab are described [here](#)³⁵².

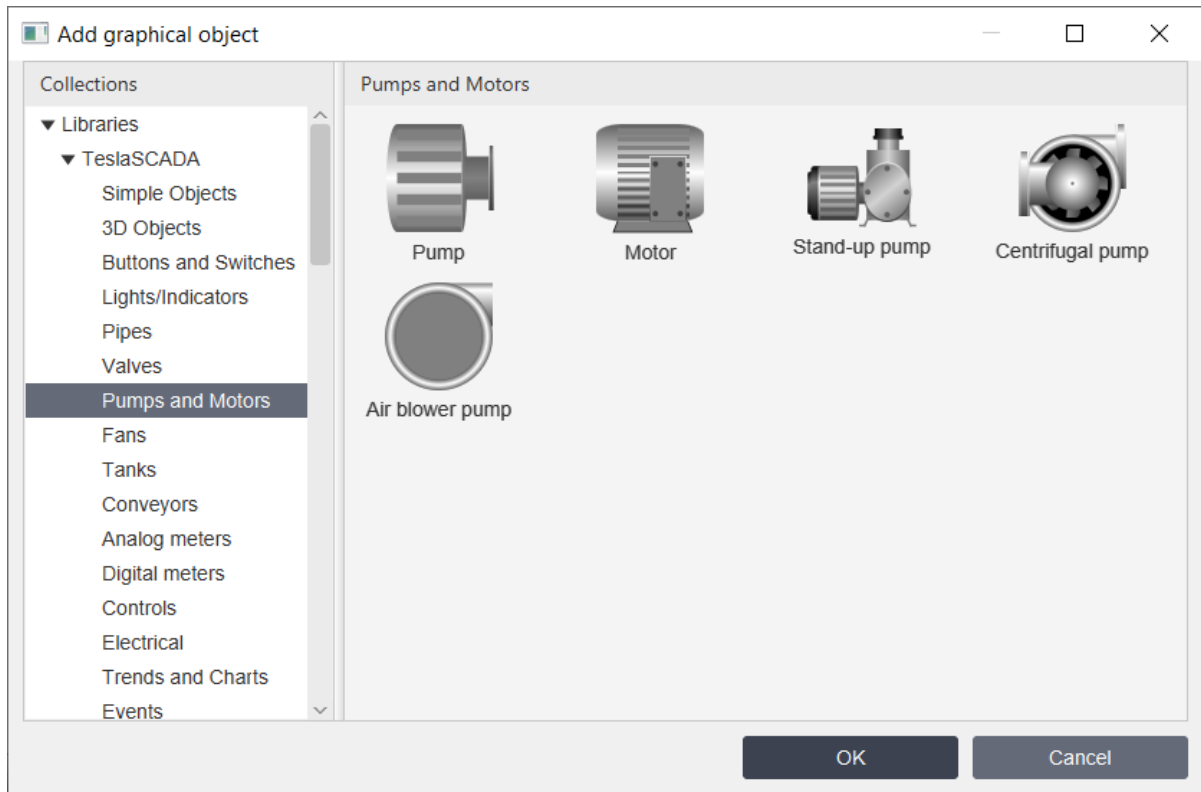
Properties from the "Flash" tab are described [here](#)³⁴⁵.

Properties from the "Rotation" tab are described [here](#)³⁴⁷.






Properties from the "Motion" tab are described [here](#)³⁴⁸.

Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.7 Pumps and Motors library



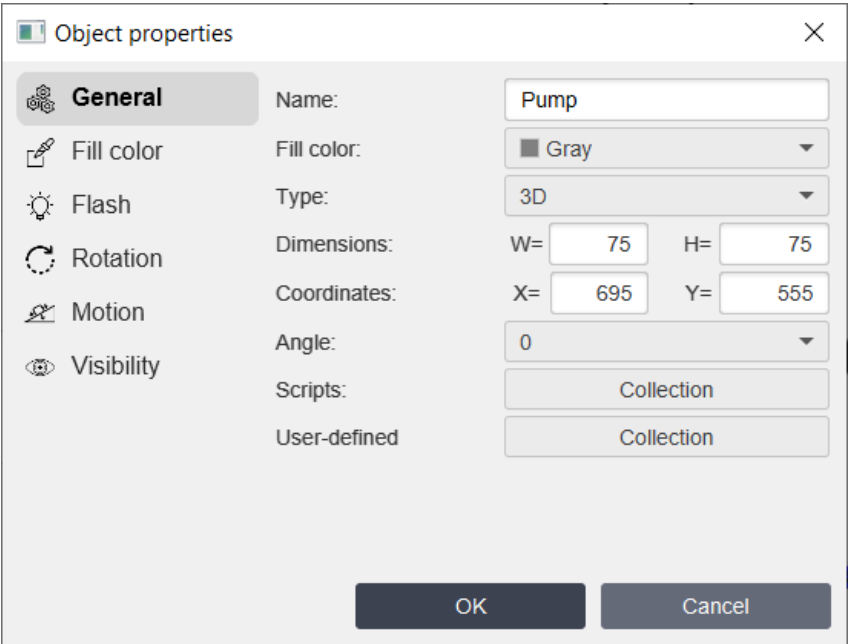
Pumps library contains the following objects:

- [Pump](#) 
- [Motor](#) 
- [Stand-up pump](#) 
- [Centrifugal pump](#) 
- [Air blower pump](#) 

All pumps have the same General group properties. Below we'll describe only for one graphical object - Pump.

6.2.3.7.1 Pump

This section applies to the following objects: Pump, Motor, Stand-up pump, Centrifugal pump, Air blower pump.

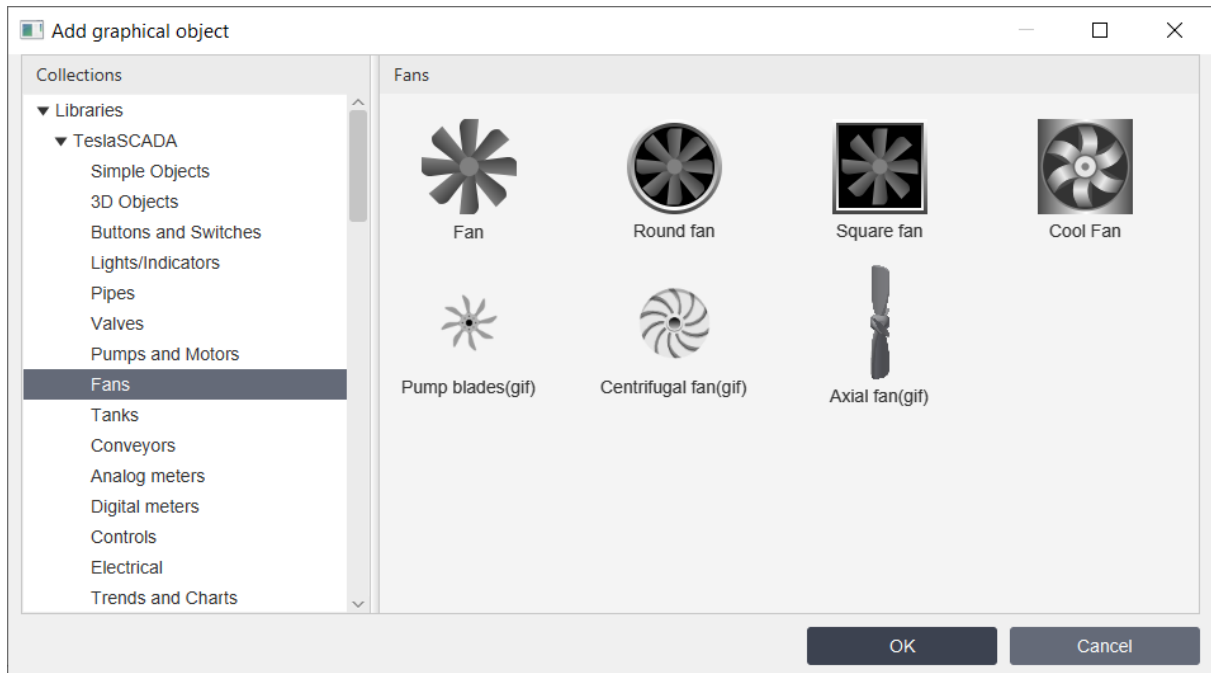


Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).








Property	ST script field	Description
Fill color	fillcolor	Color of the pump.

- Properties from the "Fill Color" tab are described [here](#)³⁵².
- Properties from the "Flash" tab are described [here](#)³⁴⁵.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷.
- Properties from the "Motion" tab are described [here](#)³⁴⁸.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.8 Fans library



Fans library contains the following objects:

- [Fan](#) 
- [Round fan](#) 
- [Square fan](#) 
- [Cool fan](#) 
- [Pump blades\(gif\)](#) 
- [Centrifugal fan\(gif\)](#) 
- [Axial fan\(gif\)](#) 

All fans have the same properties. (gif) means that non-vector graphics are used to draw this graphic object. That means you can't change fill color of this object. For animation use 'gif' files.

6.2.3.8.1 Fan

This section applies to the following objects: Fan, Round fan, Square fan, Cool fan, Pump blades(gif), Centrifugal fan(gif), Axial fan(gif).

Object properties

General

Name: Fan

Fill color: Gray

Rotation: ClockWise

Type: 3D

Dimensions: W= 75 H= 75

Coordinates: X= 839 Y= 559

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Fill color	fillcolor	Color of the fan.
Rotation	rotation	Rotation of the fan - clockwise or counter clockwise.

Set up Rotation ind. properties to rotate fan. Also it's possible to use speed property in ST script for changing speed animation. For vector graphic it's changed proportional of the value. For 'gif' animation that depends on the value. For speed below 5000 used fast speed animation, for speed is equal 5000 used medium speed animation and for speed is greater than 5000 used slow speed animation.

Properties from the "**Rotarion indicator**" tab are described [here](#)³⁶³.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

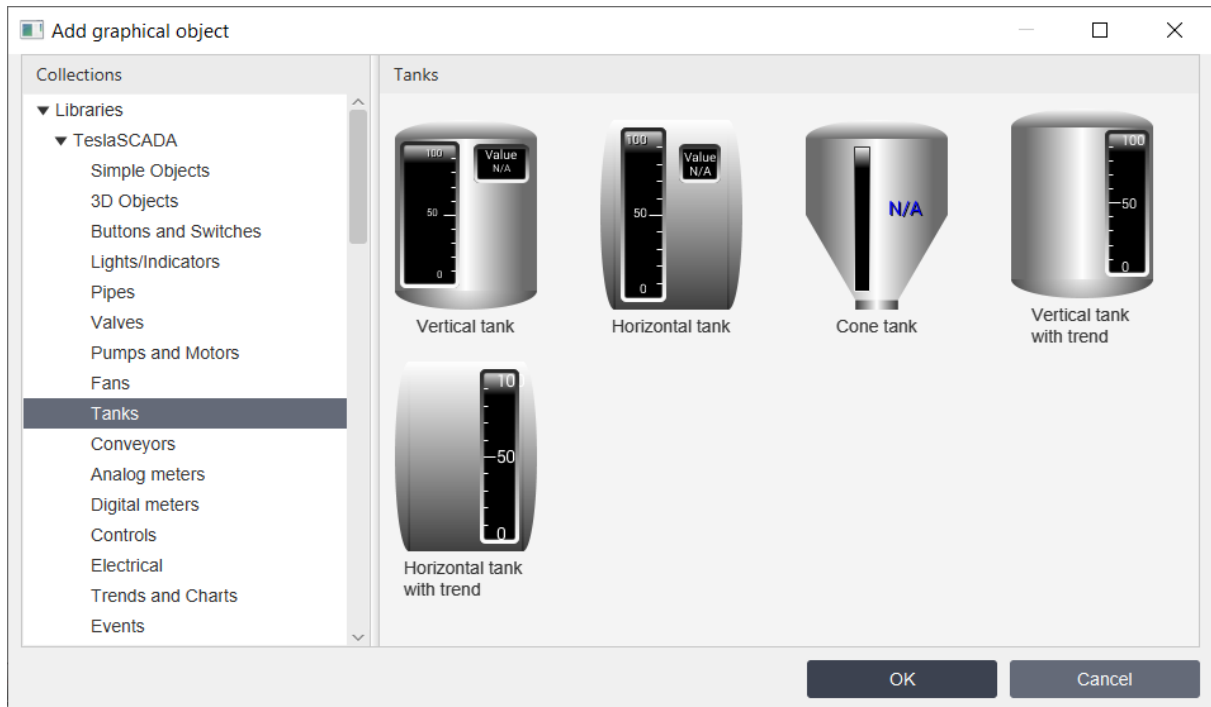
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.9 Tanks library



Tanks library contains the following objects:

- [Vertical tank](#) ²⁰²
- [Horizontal tank](#) ²⁰²
- [Cone tank](#) ²⁰²
- [Vertical tank with trend](#) ²⁰²
- [Horizontal tank with trend](#) ²⁰²

All tanks have the same General group properties. Below we'll describe them only for one graphical object - Vertical tank.

6.2.3.9.1 Vertical tank

This section applies to the following objects: Vertical tank, Horizontal tank, Cone tank, Vertical tank with trend, Horizontal tank with trend.

Object properties

General

Name: Vertical tank

Color: Gray

Fill color: Blue

Text: Value

Font size: 0

Type: 3D

Dimensions: W= 112 H= 150

Coordinates: X= 872 Y= 514

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³)

Property	ST script field	Description
Color	color	Background color of the tank .
Fill color	fillcolor	Filling color of the tank.
Text	text	Text displayed on the tank.
Font size	fontsize	Size of the text's font.

Properties from the "**Filling**" tab are described [here](#)³⁵⁴.

Properties from the "**Indicator color**" tab are described [here](#)³⁶⁶.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

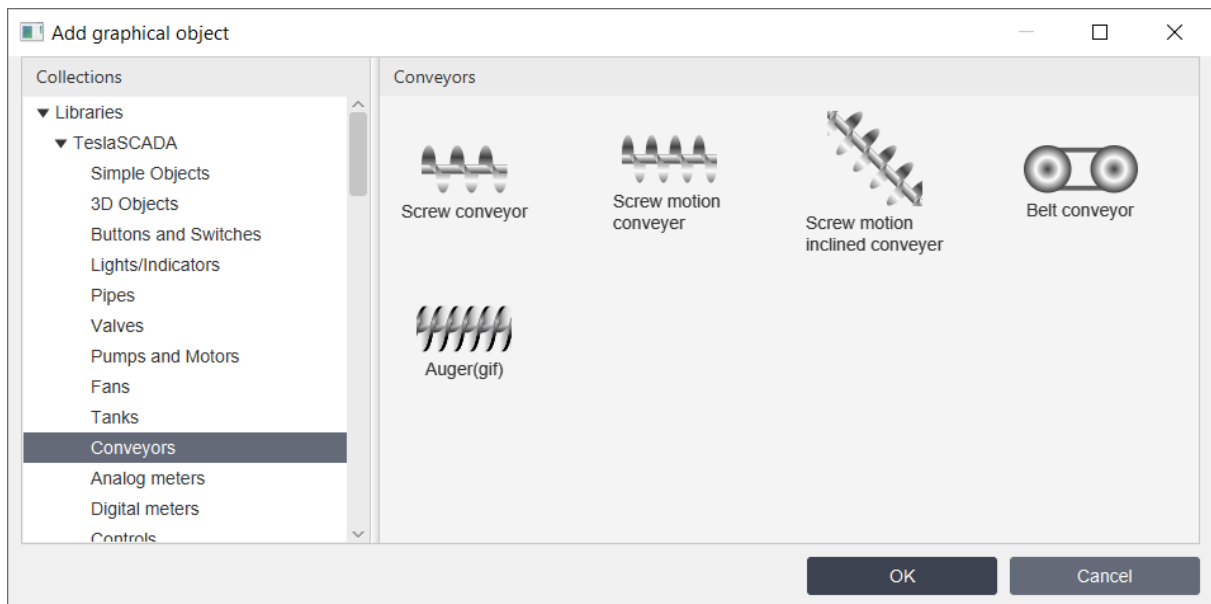
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.10 Conveyers library



Conveyers library contains the following objects:

- [Screw conveyor](#) ²⁰⁴
- [Screw motion conveyor](#) ²⁰⁵
- [Screw motion inclined conveyor](#) ²⁰⁵
- [Belt conveyor](#) ²⁰⁴
- [Auger\(gif\)](#) ²⁰⁵

Screw conveyor and Belt conveyor have the same General group properties. Below we'll describe them only for one graphical object - Belt conveyor. Screw motion conveyor, Screw motion inclined conveyor and Auger(gif) have the same General group properties. Below we'll describe it only for one graphical object - Screw motion conveyor.

(gif) means that non-vector graphics are used to draw this graphic object. That means you can't change fill color of this object. For animation use gif files.

6.2.3.10.1 Belt conveyor

This section applies to the following objects: Screw conveyor and Belt conveyor.

Object properties

General

Name: Screw conveyor

Fill color: Gray

Incline: false

Type: 3D

Dimensions: W= 75 H= 75

Coordinates: X= 1029 Y= 541

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Fill color	fillcolor	Color of the conveyor.
Incline	inclined	Choose incline or not conveyor.

Properties from the **"Fill Color"** tab are described [here](#)³⁵².

Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.10.2 Screw motion conveyor

This section applies to the following objects: Screw motion conveyor, Screw motion inclined conveyor and Auger(gif).

Object properties

General

Name: Screw motion conveyor

Rotation ind. Fill color: Gray

Fill color Rotation: ClockWise

Flash Type: 3D

Rotation Dimensions: W= 75 H= 75

Motion Coordinates: X= 1153 Y= 564

Visibility Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾)

Property	ST script field	Description
Fill color	fillcolor	Color of the fan.
Rotation	rotation	Rotation(Motion) of the screw conveyor - right or left.

Set up Rotation ind. properties to rotate(movement) conveyor. Also it's possible to use speed property in ST script for changing speed animation. For vector graphic it's changed proportional of the value. For gif animation that depends on the value. For speed below 5000 used fast speed animation, for speed is equal 5000 used medium speed animation and for speed is greater than 5000 used slow speed animation.

Properties from the "**Rotation indicator**" tab are described [here](#)³⁶³⁾.

Properties from the "**Fill Color**" tab are described [here](#)³⁵²⁾.

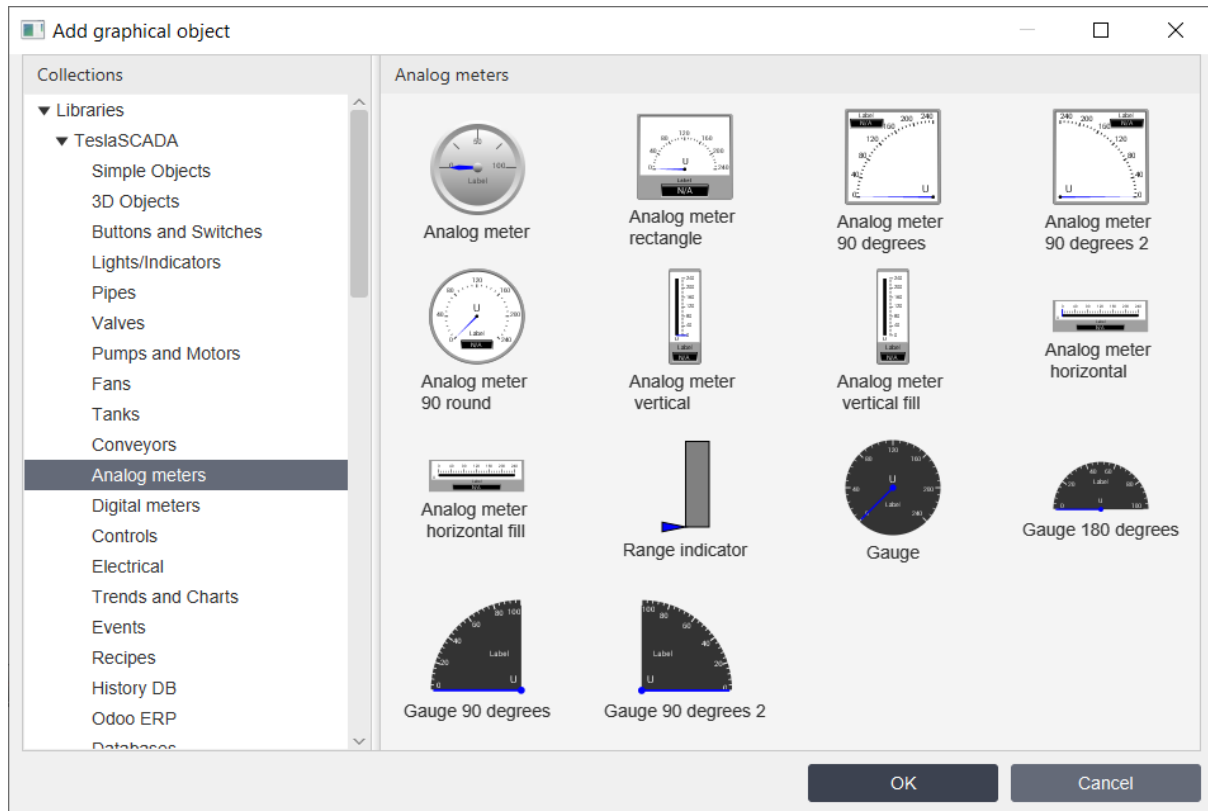
Properties from the "**Flash**" tab are described [here](#)³⁴⁵⁾.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷⁾.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸⁾.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹⁾.

6.2.3.11 Analog meters library



Analog meters library contains the following objects:

- [Analog meter](#) ²⁰⁸
- [Analog meter rectangle](#) ²¹⁰
- [Analog meter 90 degrees](#) ²¹⁰
- [Analog meter 90 degrees 2](#) ²¹⁰
- [Analog meter 90 round](#) ²¹⁰
- [Analog meter vertical](#) ²¹⁰
- [Analog meter vertical ? II](#) ²¹⁰
- [Analog meter horizontal](#) ²¹⁰
- [Analog meter horizontal ? II](#) ²¹⁰
- [Range Indicator](#) ²⁰⁹
- [Gauge](#) ²¹⁰
- [Gauge 180 degrees](#) ²¹⁰
- [Gauge 90 degrees](#) ²¹⁰
- [Gauge 90 degrees 2](#) ²¹⁰

Below you can find description 3 objects from analog meters library. The rest objects have the same properties.

6.2.3.11.1 Analog meter

Object properties

General

Name: Analog meter

Needle(Fill) color: Blue

Fill color: Gray

Text: Label

Dimensions: W= 75 H= 75

Coordinates: X= 1232 Y= 549

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Needle(fill) color	color	Color of the needle.
Fill color	fillcolor	Color of the meter.
Text	text	Text of the label.

Properties from the "**Value**" tab are described [here](#)^[369].

Properties from the "**Needle color**" tab are described [here](#)^[366].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.11.2 Range indicator

Object properties

General

Name: Range indicator

Needle(Fill) color: Blue

Fill color: Gray

Border color: Black

Type: Left

Dimensions: W= 37 H= 75

Coordinates: X= 1314 Y= 552

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Needle(fill) color	color	Color of the needle.
Fill color	fillcolor	Color of the range's background.
Border color	bordercolor	Color of the border.
Type	type	Type of the indicator: <ul style="list-style-type: none"> ▪ Left ▪ Right

Properties from the "**Value**" tab are described [here](#)^[370].

Properties from the "**Needle color**" tab are described [here](#)^[366].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.11.3 Other analog meters

This section applies to the following objects: Analog meter, Analog meter rectangle, Analog meter 90 degrees, Analog meter 90 degrees 2, Analog meter 90 round, Analog meter vertical, Analog meter vertical ?II, Analog meter horizontal, Analog meter horizontal ?II, Gauge, Gauge 180 degrees, Gauge 90 degrees, Gauge 90 degrees 2.

Object properties

General

Name: Analog meter vertical fill

Needle(Fill) color: Blue

Border color: #a0a0a0

Text: Label

Unit: U

No of intervals: 6

Use digital: ☒

Dimensions: W= 25 H= 75

Coordinates: X= 164 Y= 662

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Color	color	Color of the needle.
Border color	bordercolor	Color of the border.
Text	text	Text of the label.
Unit	unit	Text of the measured value's unit.
? of intervals	interval	The number of meter's intervals.

Property	ST script field	Description
Use digital	usedigital	Check it if you want to use also digital meter.

Properties from the **"Value"** tab are described [here](#)³⁶⁹ (for meters).

Properties from the **"Value"** tab are described [here](#)³⁷⁰ (for gauges).

Properties from the **"Needle color"** tab are described [here](#)³⁶⁶.

Properties from the **"Border color"** tab are described [here](#)³⁶⁶.

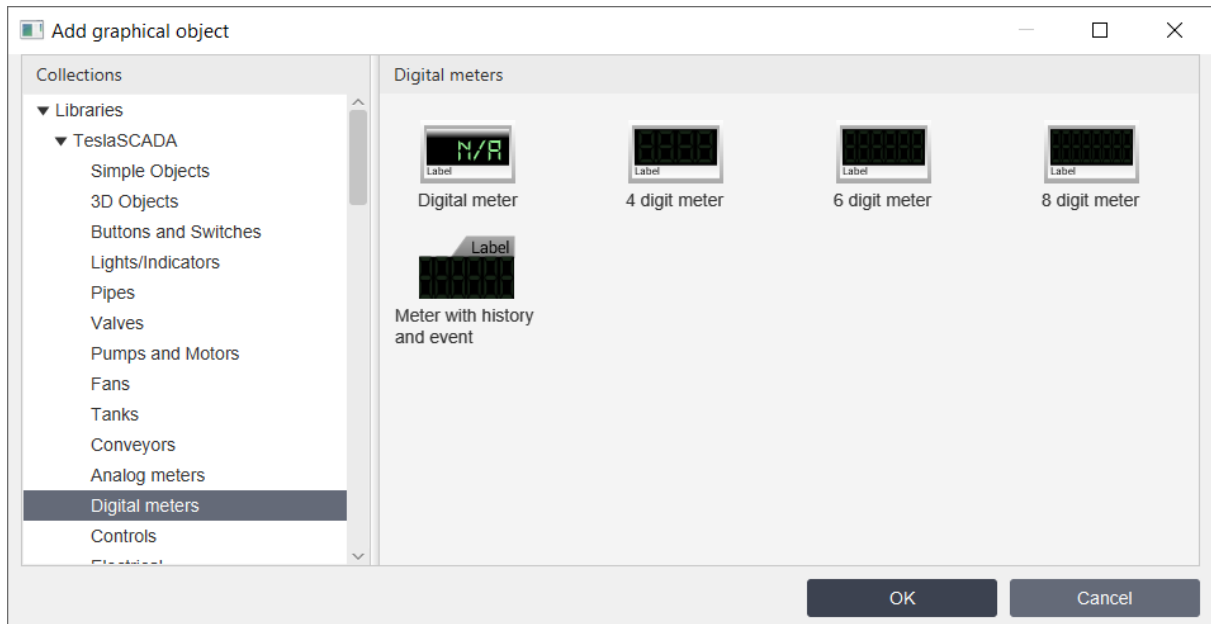
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.12 Digital meters library



Digital meters library contains the following objects:

- [Digital meter](#)²¹¹
- [4 digit meter](#)²¹¹
- [6 digit meter](#)²¹¹
- [8 digit meter](#)²¹¹
- [Meter with history and event](#)²¹¹

All digital meters have the same general properties.

6.2.3.12.1 Digital meter

This section applies to the following objects: Digital meter, 4 digit meter, 6 digit meter, 8 digit meter, Meter with history and event.

Object properties

General

Name: Digital meter

Text: Label

Text color: Light Green

Border color: Dark Gray

Fill color: Black

Type: 3D

Dimensions: W= 75 H= 50

Coordinates: X= 240 Y= 664

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Text	text	Text of the label.
Text color	textcolor	Color of the meter's digits.
Fill color	fillcolor	Color of the meter's background.
Border color	bordercolor	Color of the meter's border.
Side	side	This property only for Meter with history and event. You can choose where history trend or event table will appear after clicking on meter.

Properties from the "**Text input**" tab are described [here](#)³⁵⁹.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Border color**" tab are described [here](#)³⁶⁶.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

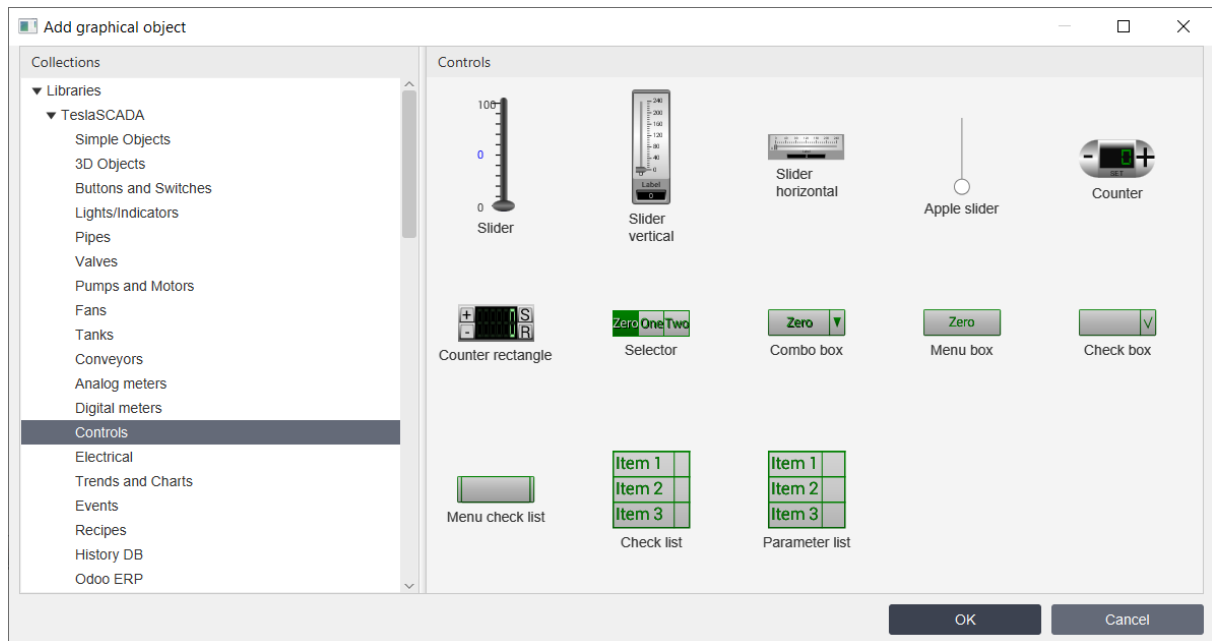
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.13 Controls library



Controls library contains the following objects:

- [Slider](#)²¹⁴
- [Slider vertical](#)²¹⁵
- [Slider horizontal](#)²¹⁵
- [Counter](#)²¹⁶
- [Counter rectangle](#)²¹⁶
- [Apple slider](#)²¹⁴
- [Selector](#)²¹⁷
- [ComboBox](#)²¹⁷
- [MenuBox](#)²¹⁸
- [CheckBox](#)²¹⁹
- [MenuCheckList](#)²²¹
- [CheckList](#)²¹⁹
- [Parameter list](#)²²³

6.2.3.13.1 Slider and Apple slider

The screenshot shows the 'Object properties' dialog box for a slider object. The 'General' tab is active. The properties are as follows:

Property	Value
Name	Slider
Color	Gray
Fill color	Blue
Type	3D
Dimensions	W= 37, H= 112
Coordinates	X= 370, Y= 685
Angle	0
Scripts	Collection
User-defined	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Color	color	Color of the slider's background.
Fill color	fillcolor	Color of the slider's filling.

Properties from the "**Control**" tab are described [here](#)³⁶⁷.

Properties from the "**Indicator color**" tab are described [here](#)³⁶⁶.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.13.2 Slider vertical and horizontal

Object properties

General

Name: Slider vertical

Color: #c8c8c8

Fill color: #c8c8c8

Text: Label

Unit: U

No of intervals: 6

Use digital: ☒

Type: 3D

Dimensions: W= 37 H= 112

Coordinates: X= 469 Y= 646

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Color	color	Color of the slider's background.
Fill color	fillcolor	Color of the slider's filling.
Text	text	Text of the label.
Unit	unit	Specify the unit of measure for the tag value
? of intervals	interval	The number of slider's intervals.
Use digital	usedigital	Check it if you want to use also digital meter.

Properties from the "**Control**" tab are described [here](#)³⁶⁷.

Properties from the "**Indicator color**" tab are described [here](#)³⁶⁶.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.13.3 Counter and Counter rectangle

The screenshot shows the 'Object properties' dialog box for a Counter object. The 'General' tab is active. The properties are as follows:

Property	Value
Name	Counter
Color	Gray
Text color	Green
Type	3D
Dimensions	W= 75, H= 37
Coordinates	X= 599, Y= 684
Angle	0
Scripts	Collection
User-defined	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Color	color	Color of the counter's background.
Text color	textcolor	Color of the counter's digits.

Properties from the **"Control"** tab are described [here](#)³⁶⁸.

Properties from the **"Indicator color"** tab are described [here](#)³⁶⁶.

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.

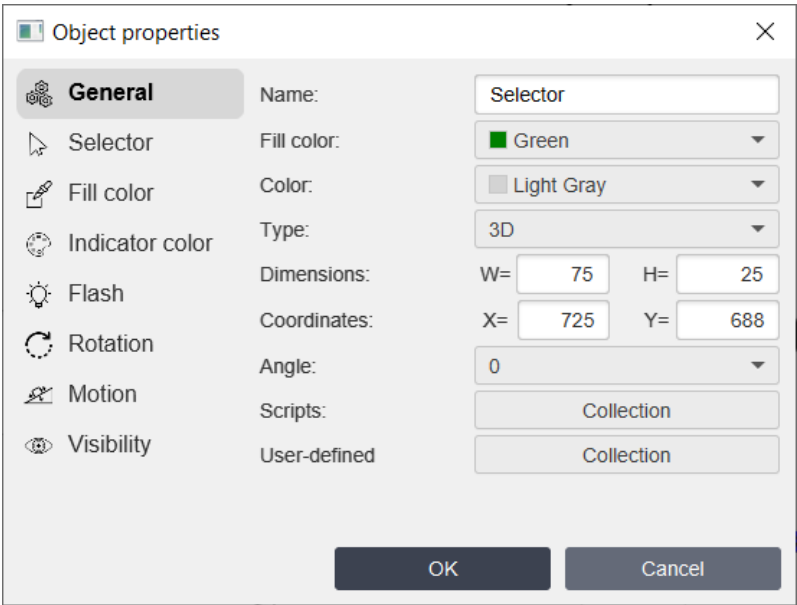
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.13.4 Selector and Combo box



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Fill Color	fillcolor	Color of the selected object's item background .
Color	color	Color of the non-selected object's item background.

- Properties from the "Selector" tab are described [here](#)³⁷⁴.
- Properties from the "Fill Color" tab are described [here](#)³⁵².
- Properties from the "Indicator color" tab are described [here](#)³⁶⁶.
- Properties from the "Flash" tab are described [here](#)³⁴⁵.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷.
- Properties from the "Motion" tab are described [here](#)³⁴⁸.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.13.5 Menu box

The screenshot shows the 'Object properties' dialog box for a 'Menu box' object. The 'General' tab is active. The properties are as follows:

Property	Value
Name	Menu box
Fill color	Light Gray
Color	Green
Type	3D
Expand type	Vertically
Animation	<input checked="" type="checkbox"/>
Dimensions	W= 75, H= 25
Coordinates	X= 845, Y= 708
Angle	0
Scripts	Collection
User-defined	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Fill Color	fillcolor	Color of the selected object's item background .
Color	color	Color of the non-selected object's item background.
Expand type	expandedtype	Expanded type of the menu: <ul style="list-style-type: none"> ▪ horizontally ▪ vertically
Animation	animation	Check if you want to animate expanding of the menu.

Properties from the "**Selector**" tab are described [here](#)^[374].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Indicator color**" tab are described [here](#)^[366].

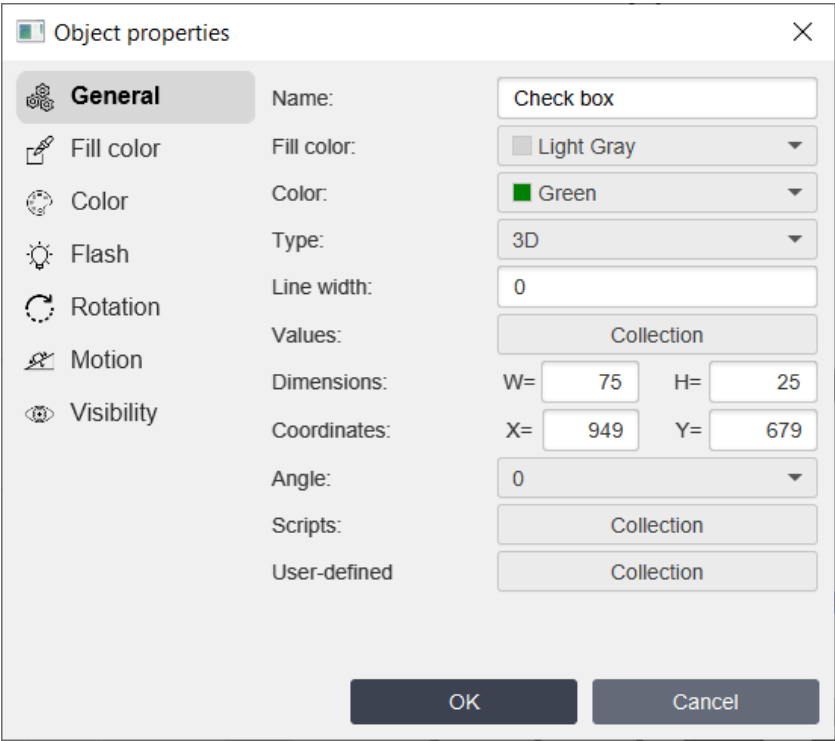
Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.13.6 Check box and Check list



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Fill Color	fillcolor	Color of the selected object's item background .
Color	color	Color of the text.
Line width	linewidth	Width of the border's line.
Value		After clicking Collection you'll see window:

Property	ST script field	Description
		<div><div><div>Collection</div><div><div>(=1.0,0.0)-->Item 1</div><div>(=1.0,0.0)-->Item 2</div><div>(=1.0,0.0)-->Item 3</div></div><div><div>Tag:</div><div>Value:</div><div>Uncheck value:</div><div>Text:</div></div><div><div><div></div></div><div><div>1.0</div></div><div><div>0.0</div></div><div><div>Item 1</div></div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div> <div>where:<ul style="list-style-type: none">▪ Tag - choose tag for the object's menu item.▪ Value - value which will be written after selecting the item of the object's menu.▪ Uncheck value - value which will be written after unselecting the item of the object's menu.▪ Text - enter text for the object's menu item.</div>

Properties from the "Fill Color" tab are described [here](#)³⁵².
Properties from the "Color" tab are described [here](#)³⁶⁶.
Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.13.7 Menu check list

The screenshot shows the 'Object properties' dialog box for an object named 'Menu check list'. The 'General' tab is selected. The properties are as follows:

Property	Value
Name:	Menu check list
Fill color:	Light Gray
Color:	Green
Type:	3D
Expand type:	Vertically
Line width:	0
Animation:	<input type="checkbox"/>
Values:	Collection
Dimensions:	W= 75, H= 25
Coordinates:	X= 1060, Y= 663
Angle:	0
Scripts:	Collection
User-defined	Collection

Buttons: OK, Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Fill Color	fillcolor	Color of the selected object's item background .
Color	color	Color of the text.
Expand type	expandedt ype	Expanded type of the menu: <ul style="list-style-type: none"> ▪ horizontally ▪ vertically
Animation	animation	Check if you want to animate expanding of the menu.
Line width	linewidth	Width of the border's line.
Value		After clicking Collection you'll see window:

Property	ST script field	Description
		<div><div><div>Collection</div><div><div>(=1.0,0.0)-->Item 1</div><div>(=1.0,0.0)-->Item 2</div><div>(=1.0,0.0)-->Item 3</div></div><div><div>Tag:</div><div>Value:</div><div>Uncheck value:</div><div>Text:</div></div><div><div><div></div></div><div><div>1.0</div></div><div><div>0.0</div></div><div><div>Item 1</div></div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div> <div>where:<ul style="list-style-type: none">▪ Tag - choose tag for the object's menu item.▪ Value - value which will be written after selecting the item of the object's menu.▪ Uncheck value - value which will be written after unselecting the item of the object's menu.▪ Text - enter text for the object's menu item.</div>

Properties from the "Fill Color" tab are described [here](#)³⁵².
Properties from the "Color" tab are described [here](#)³⁶⁶.
Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.13.8 Parameter list

Object properties

General

Fill color

Color

Flash

Rotation

Motion

Visibility

Name:

Fill color:

Color:

Type:

Line width:

Values:

☐ Write simultaneously

Dimensions:

Coordinates:

Angle:

Scripts:

User-defined

Parameter list

Light Gray

Green

3D

0

Collection

W=75

H=75

X=1162

Y=633

0

Collection

Collection

OK

Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾).

Property	ST script field	Description
Fill Color	fillcolor	Color of the selected object's item background .
Color	color	Color of the text.
Line width	linewidth	Width of the border's line.
Value		After clicking Collection you'll see window:

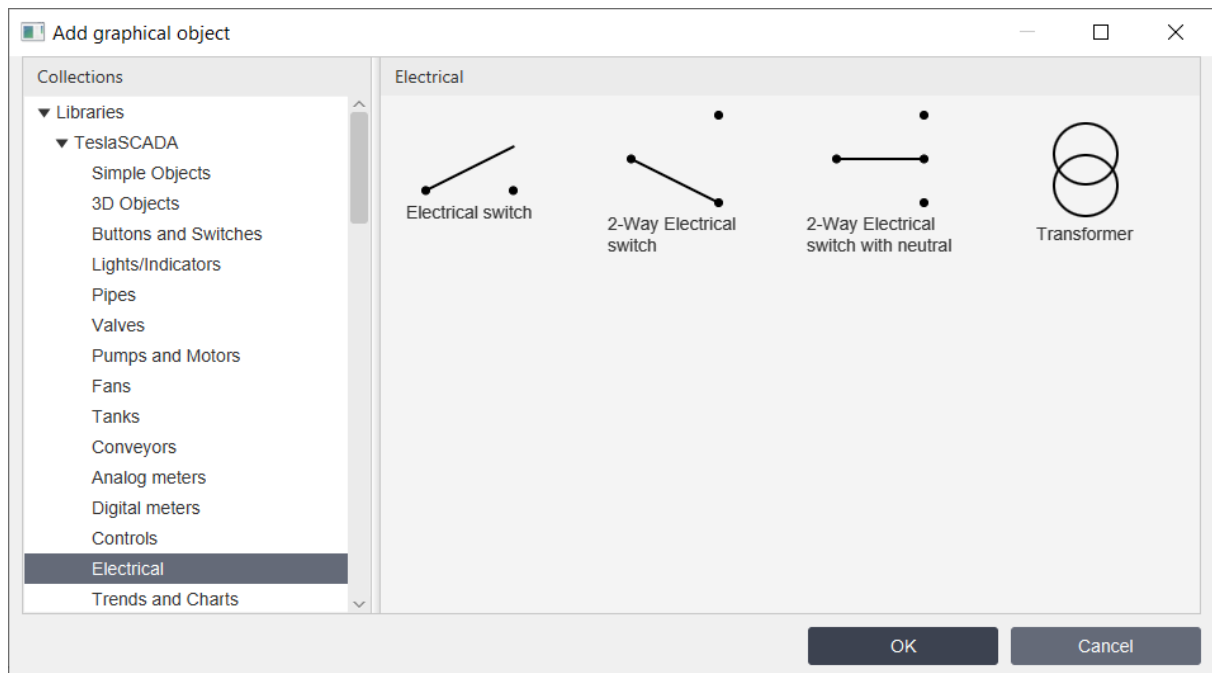
TeslaSCADA2 IDE User manual

© 2024 LLC Tesla





Property	ST script field	Description
		<div><div><div>Collection</div><div><div>Item 1</div><div>Item 2</div><div>Item 3</div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div>Tag:</div><div></div><div>...</div><div>Name:</div><div>Item 1</div><div>Decimal position:</div><div>0</div><div>Add</div><div>Edit</div><div>Remove</div><div>Close</div></div></div></div> <p>where:</p> <ul style="list-style-type: none">▪ Tag - choose tag for the parameter list item.▪ Name - name of the parameter list item.▪ Decimal position - decimal position for the parameter list item's values.
Write simultaneously	simultaneously	Check to enter values simultaneously in tags.

Properties from the "Fill Color" tab are described [here](#)³⁵².
Properties from the "Color" tab are described [here](#)³⁶⁶.
Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.14 Electrical library



Electrical library contains the following objects:

- [Electrical switch](#)  225
- [2-Way Electrical switch](#)  225
- [2-Way Electrical switch with neutral](#)  225
- [Transformer](#)  227

Below description of the Electrical switch. All other switches have the same properties.

6.2.3.14.1 Electrical switch

This section applies to the following objects: Electrical switch, 2-Way Electrical switch, 2-Way Electrical switch with neutral.

Object properties

General

Name: Electrical switch

Switch control: ☐

Line width: 2

Line color: Black

Flash: ☐

Beginmarker: Circle

Endmarker: Circle

Rotation: ☐

Dimensions: W= 75 H= 75

Motion: ☐

Coordinates: X= 1296 Y= 675

Visibility: ☐

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Line width	linewidth	Width of the switch line.
Color	color	Color of the switch line.
Beginmarker	beginmarker	Marker of the switch line's begin: <ul style="list-style-type: none"> ▪ Flat ▪ Arrow ▪ Square ▪ Circle
Endmarker	endmarker	Marker of the switch line's end: <ul style="list-style-type: none"> ▪ Flat ▪ Arrow ▪ Square ▪ Circle

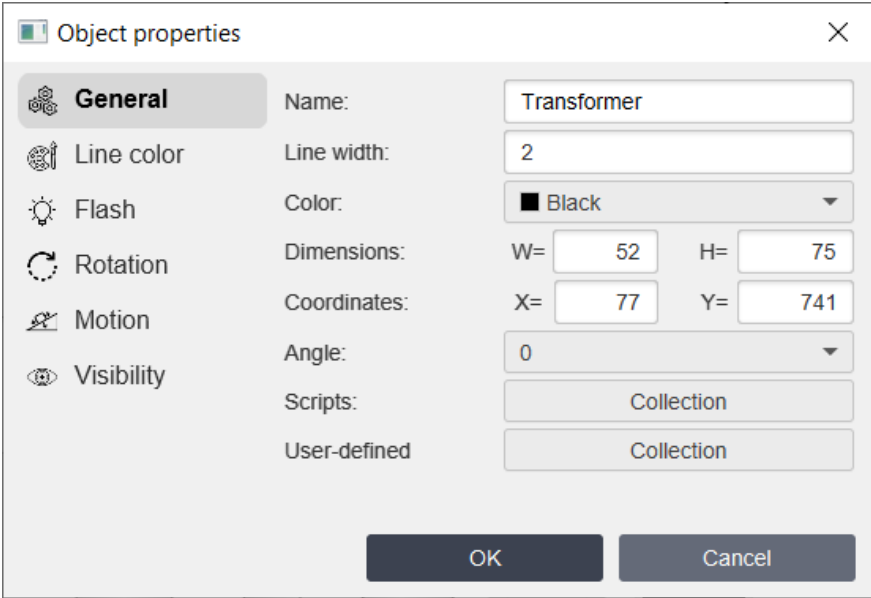
Properties from the "**Switch control**" tab are described [here](#)^[372].

Properties from the "**Line Color**" tab are described [here](#)^[350].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.14.2 Transformer

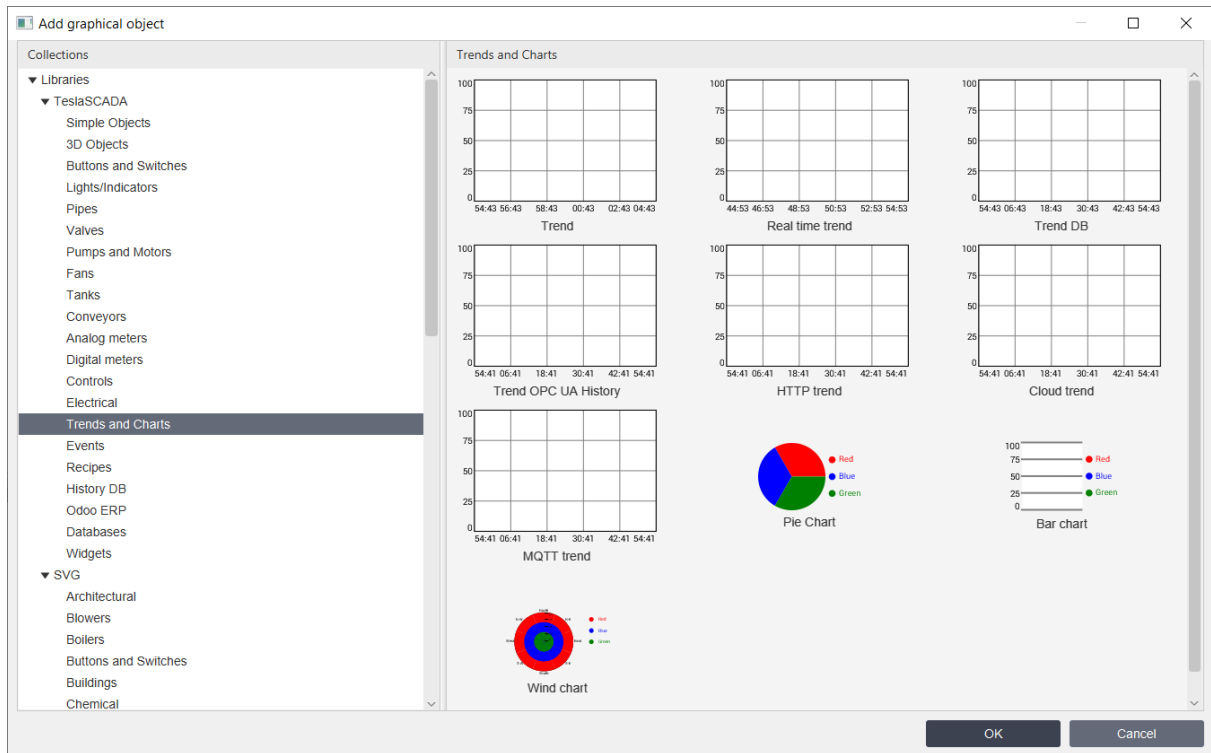


Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Line width	linewidth	Width of the transformer's line.
Color	color	Color of the transformer's line.

Properties from the "Line Color" tab are described [here](#)³⁵⁰.
Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.15 Trends and Charts library



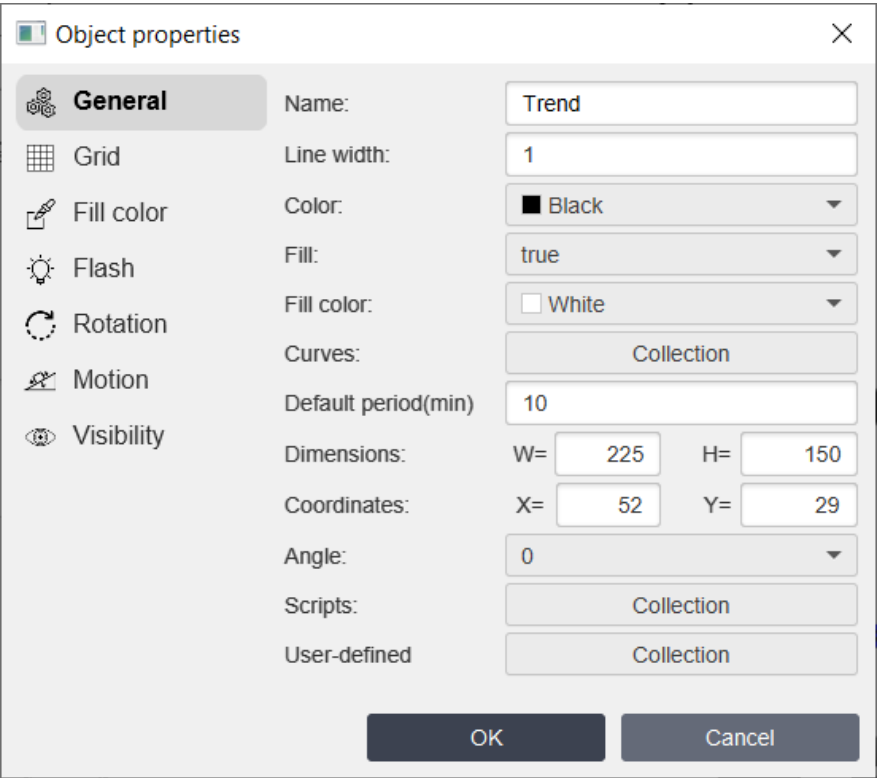
Trends library contains the following objects:

- [Trend](#) ²²⁸
- [Real time trend](#) ²²⁸
- [Trend DB](#) ²²⁸
- [Trend OPC UA History](#) ²²⁸
- [HTTP trend](#) ²²⁸
- [Cloud trend](#) ²²⁸
- [MQTT trend](#) ²²⁸
- [Pie Chart](#) ²³²
- [Bar chart](#) ²³⁴
- [Wind chart](#) ²³⁶

Trend and Real time trend draw curves based on tags that use history data collection (check [Enable history](#) ⁴⁷⁴ in Tags properties). Trend DB draws curves based on tags that use data stored in [general database](#) ¹⁰⁷ (check [Store in DB](#) ⁴⁷⁴ in Tags properties). Trend OPC UA History draws curves based on tags that are binded to OPC UA nodes supported Historyzing property. All trends have the same General and Grid group properties. Below we'll describe them only for one graphical object - Trend.

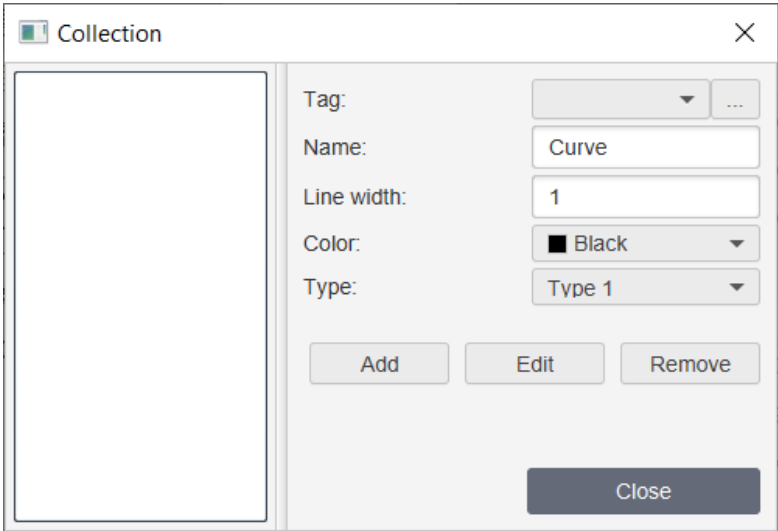
6.2.3.15.1 Trend

This section applies to the following objects: Trend, Real time trend, Trend DB, Trend OPC UA History, HTTP trend, Cloud trend, MQTT trend.



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill trend.
Fill color	fillcolor	Fill color of the trend.
Curves		After clicking Collection you'll see window:

Property	ST script field	Description
		 <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this curve. ▪ Name - name of the curve. ▪ Line with - curve's line width. ▪ Color - curve's line color. ▪ Type - line's type: <ul style="list-style-type: none"> ✓ Type 1 - just draw the line. ✓ Type 2 - draw line with ?lling till axis X. ✓ Type 3 - draw a ladder line. ✓ Type 4 - draw a ?lled ladder line.
Default period (min)	default period	Default time period of the trend (end time - begin time).
History DB *		History database name of the HTTP server for HTTP history DB trend.
Auto refresh *		Check it if you want to auto refresh HTTP history DB trend.

* Available only in HTTP history DB trend.

Also Trend object has several properties that you can't setup by using settings dialog window, but you can setup by using ST script:

- **begin** - start time for trend information. Time is represented in minutes from current period. (start time = current time - begin).

- **end** - ?nish time for trend information. Time is represented in minutes from current period. (?nish time = current time - end).
- **title** - title for the trend's report representation.
- **? lename** - name of the report's ?le.
- **number** - report's frequency of writing values.
- **savereport** - when this value become true trend's report will be created.
- **begindatetime** - start time for trend information. Time is represented in milliseconds from 1 January 1970.
- **enddatetime** - ?nish time for trend information. Time is represented in milliseconds from 1 January 1970.
- **disablesavereport** - disable "Save report" button in the dialog.
- **disableprint** - disable "Print" report button in the dialog.

Properties from the "**Grid**" tab are described [here](#)²³¹.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

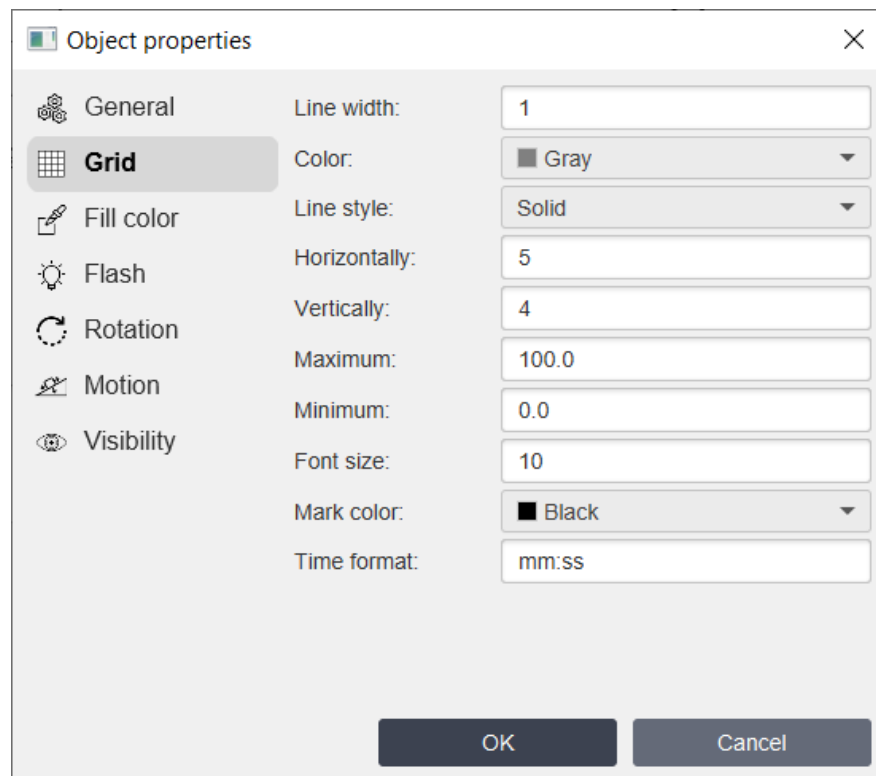
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.15.1.1 Grid



Property	ST script field	Description
Line width		Width of grid's lines .
Color	gridlinecolor	Color of grid's lines.
Line style	linestyle	Style of the line: <ul style="list-style-type: none"> ▪ Solid ▪ Dash ▪ Dot ▪ DashDot
Horizontally	horizontally	Number of trend's horizontal grid lines.
Vertically	vertically	Number of trend's vertical grid lines.
Maximum	maximum	Maximum of the trend's value.
Minimum	minimum	Minimum of the trend's value.
Font size	fontsize	Font size of the trend's marks.
Mark color	markcolor	Color of the marks.
Time format	timeformat	Time format of the trend's time.

6.2.3.15.2 Pie chart

Object properties

General

Name: Pie Chart

Sectors: Collection

Use legends: ☒

Donut: ☐

Dimensions: W= 150 H= 75

Coordinates: X= 346 Y= 43

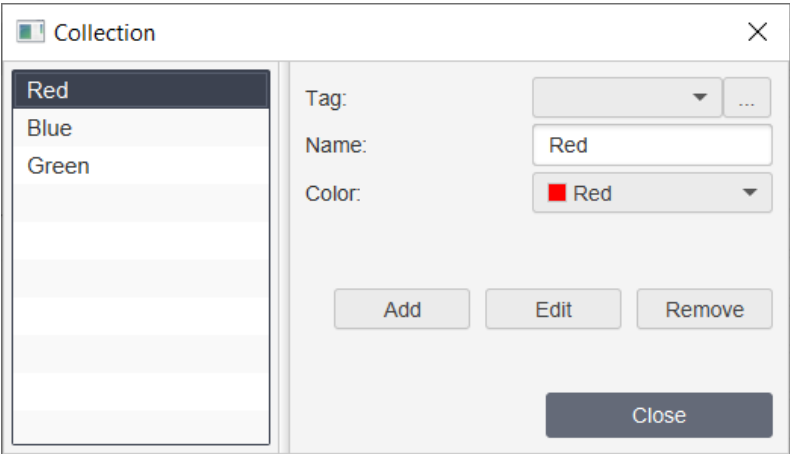
Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³).

Property	ST script field	Description
Sectors		<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this chart's sector. ▪ Name - name of the sector. ▪ Color - sector's color.
Use legends	uselegends	Check it if you want to add legends to the chart.
Donut	donut	Check it if you want to use ring type chart.

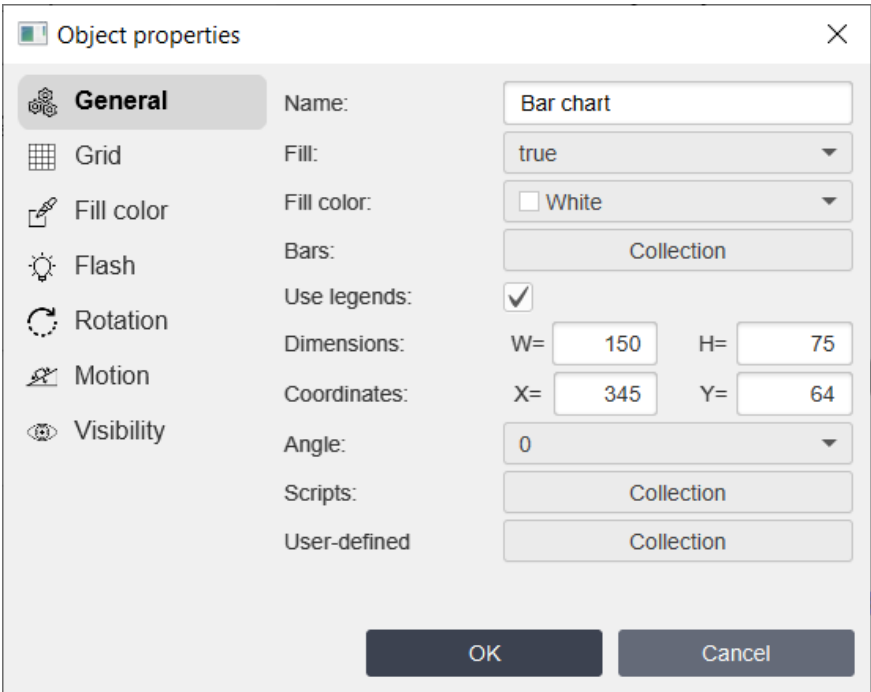
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.15.3 Bar chart



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾).

Property	ST script field	Description
Fill	fill	Select fill or not fill bar chart.
Fill color	fillcolor	Fill color of the bar chart.
Bars		<p>After clicking Collection you'll see window:</p> <p>where:</p>

Property	ST script field	Description
		<ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this bar. ▪ Name - name of the bar chart. ▪ Color - bar's color.
Use legends	uselegends	Check it if you want to add legends to the bar chart.

Properties from the "**Grid**" tab are described [here](#)²³⁵.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.15.3.1 Grid

The screenshot shows the 'Object properties' dialog box with the 'Grid' tab selected. The dialog has a sidebar with icons for General, Grid, Fill color, Flash, Rotation, Motion, and Visibility. The main area contains the following settings:

- Line width: 2
- Color: Gray
- Line style: Solid
- Vertically: 4
- Maximum: 100.0
- Minimum: 0.0
- Font size: 10
- Mark color: Black

At the bottom, there are 'OK' and 'Cancel' buttons.

Property	ST script field	Description
Line width		Width of grid's lines .

Property	ST script field	Description
Color		Color of grid's lines.
Line style	linestyle	Style of the line: <ul style="list-style-type: none"> ▪ Solid ▪ Dash ▪ Dot ▪ DashDot
Vertically	vertically	Number of trend's vertical grid lines.
Maximum	maximum	Maximum of the bar chart's value.
Minimum	minimum	Minimum of the bar chart's value.
Font size	fontsize	Font size of the trend's marks.
Mark color	markcolor	Color of the marks.

6.2.3.15.4 Wind chart

Object properties

General

Flash

Rotation

Motion

Visibility

Name:

Wind chart

Sectors:

Collection

Use legends:

☒

Vertically:

4

Minimum:

0.0

Maximum:

80.0

Row number:

8

Legends:

[North, N-E, East, S-E, South, S-

Dimensions:

W= 150 H= 75

Coordinates:

X= 554 Y= 54

Angle:

0

Scripts:

Collection

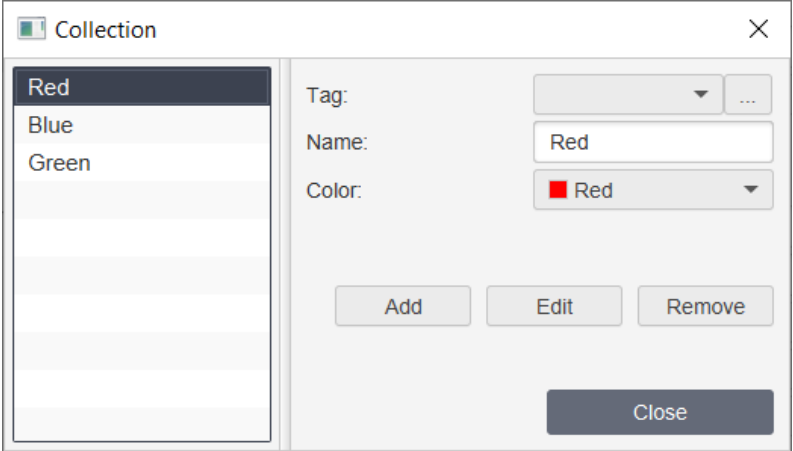
User-defined

Collection

OK

Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Sectors		<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this chart's sector. ▪ Name - name of the sector. ▪ Color - sector's color.
Use legends	uselegends	Check it if you want to add legends to the chart.
Vertically	vertically	Enter number of scale ticks.
Minimum	minimum	Minimum of chart's value.
Maximum	maximum	Maximum of chart's value.
Row number	number	Number of wing's directions.
Legends	legends	Legends for wing's directions.

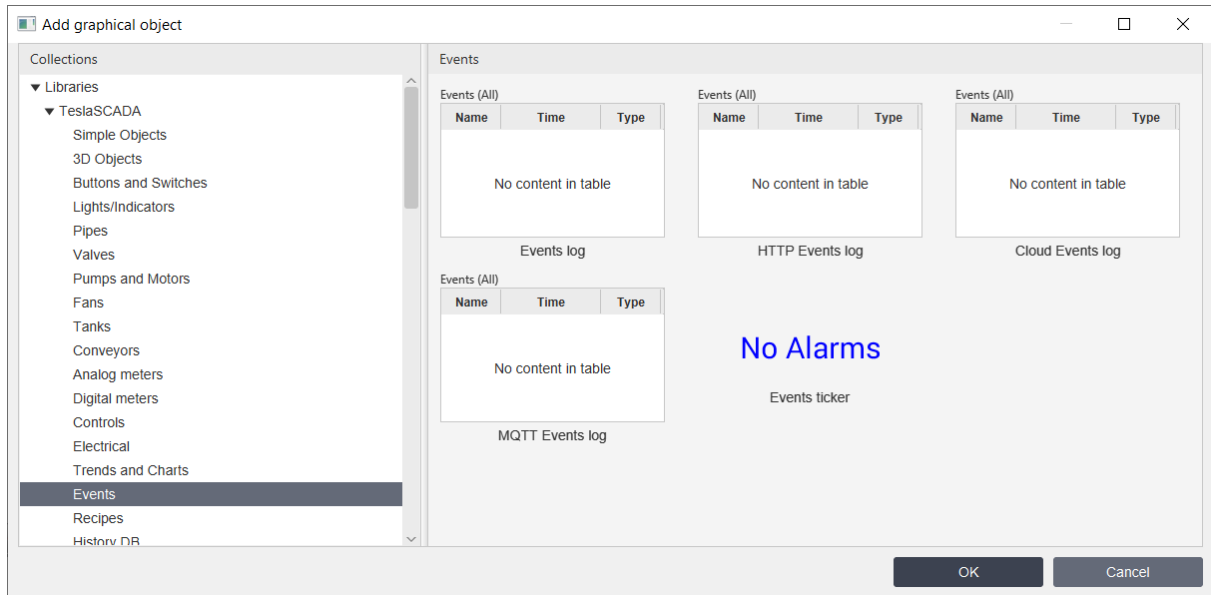
Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.16 Events library



Events library contains the following object:

- [Events log](#)²³⁸
- [HTTP Events log](#)²³⁸
- [Cloud Events log](#)²³⁸
- [MQTT Events log](#)²³⁸
- [Events ticker](#)²⁴³

Events log collects tag's events (check [Enable alarms](#)⁴⁷³) and check events you want to collect in Tags properties). Events will be collected in events database. You can setup it in **Project properties**-> [Events/History tab](#)¹⁰⁷.

6.2.3.16.1 Events log

This section applies to the following objects: Events log, HTTP Events log, Cloud Events log, MQTT Events log.

Object properties

General

Name: Events log

Use title: ☒

Title: Events

Font size: 12

Acknowledge color: White

Unacknowledge color: Light Blue

Priority colors: Collection

Time format: d/MM hh:mm:ss

Only active state: ☐

Only unacknowledged: ☐

Dimensions: W= 225 H= 150

Coordinates: X= 393 Y= 273

Angle: 0

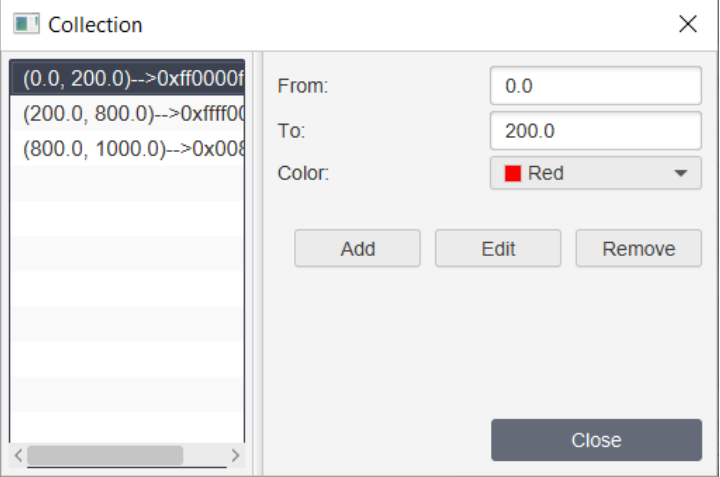
Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Use title	usetitle	Use title for the table or not.
Title	title	Title of the table.
Font size	fontsize	Size of the text's font.
Acknowledge color	ackcolor	Row's background color of the acknowledged events
Unacknowledge color	unackcolor	Row's background color of not unacknowledged events
Priority colors		After clicking Collection button you'll see the window:

Property	ST script field	Description
		 <p>where:</p> <ul style="list-style-type: none"> ▪ From - the priority⁴⁷³ of the event from which is used this color. ▪ To - the priority⁴⁷³ of the event to which is used this color. ▪ Color - color of the event text. ▪ Add - add a new color priority range. ▪ Edit - edit selected color priority range. ▪ Remove - remove selected color priority range.
Time format	timeformat	Time format of the text in time column.
Only active state	onlyactivestate	Display only active state of the events.
Only unacknowledged	onlyunack	Display only unacknowledged events.
HTTP server*		Choose HTTP server.

***Only for HTTP History DB table**

Also Event log object has several properties that you can't setup by using settings dialog box, but you can setup by using ST script:

- **begin** - start time for log information. Time represented in minutes from current period. (start time = current time - begin).
- **end** - ?nish time for log information. Time represented in minutes from current period. (?nish time = current time - end).

- **? lename** - name of the report's ?le.
- **savereport** - when this value become true trend's report will be created.
- **enbegin** - enable start time for the filter of the event log information.
- **enend** - enable finish time for the filter of the event log information.
- **enprbegin** - enable priority begin for the filter of the event log information.
- **enprend** - enable priority end for the filter of the event log information.
- **beginpriority** - begin priority for the filter of the event log information.
- **endpriority** - end priority for the filter of the event log information.
- **begindatetime** - start time for trend information. Time represented in milliseconds from 1 January 1970.
- **enddatetime** - ?nish time for trend information. Time represented in milliseconds from 1 January 1970.
- **disablesavereport** - disable "Save report" button in the dialog.
- **disableprint** - disable "Print" report button in the dialog.

Properties from the "**Columns**" tab are described [here](#) ²⁴².

Properties from the "**Flash**" tab are described [here](#) ³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#) ³⁴⁷.

Properties from the "**Motion**" tab are described [here](#) ³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#) ³⁴⁹.

6.2.3.16.1.1 Columns

Object properties

General

Columns

Flash

Rotation

Motion

Visibility

☒ Name

☒ Time

☒ Type

☒ Priority

☒ Message

☒ Value

☐ Ack.time

Title:

Title:

Title:

Title:

Title:

Title:

Title:

Name

Time

Type

Priority

Message

Value

Ack.time

Width:

Width:

Width:

Width:

Width:

Width:

Width:

60

100

60

40

180

60

0

OK

Cancel

Property	ST script field	Description
Enable (not shown)		Enable or disable correspondent column: <ul style="list-style-type: none">NameTimeTypePriorityMessageValueAck.time
Title	<div>nametitle</div> <div>timetitle</div> <div>typetitle</div> <div>prioritytitle</div> <div>messagetitle</div> <div>valuetitle</div> <div>acktimetitle</div>	Title of the corresponding column.

Property	ST script field	Description
Width	namewidth timewidth typewidth prioritywidth messagewidth h valuewidth acktimewidth h	Width of the corresponding column.

6.2.3.16.2 Events ticker

The screenshot shows the 'Object properties' dialog box for the 'Events ticker' object. The 'General' tab is active, displaying the following settings:

- Name:** Events ticker
- Speed(ms):** 2000
- From priority:** 0
- To priority:** 100
- Text:** No Alarms
- Font type:** Roboto Regular
- Underline:** ☐
- Font size:** 30
- Text color:** Blue
- Border:** false
- Border width:** 2
- Border color:** Black
- Fill:** false
- Fill color:** White
- Dimensions:** W= 150, H= 75
- Coordinates:** X= 843, Y= 106
- Angle:** 0
- Scripts:** Collection
- User-defined:** Collection

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Speed(ms)	speed	Speed of the running text.
From priority	beginpriority	Begin priority of the shown events.
To priority	endpriority	End priority of the shown events.
Text	defaulttext	Default text displayed. It's shown if events in selected priority range are not available.
Font type	fonttype	Type of the text's font.
Underline	underline	Check if you want to underline the text.
Font size	fontsize	Size of the text's font.
Text color	textcolor	Color of the text.
Border	useborder	Select use or not use border for the text.
Border width	linewidth	Width of the border's line.
Border color	bordercolor	Color of the border's line.
Fill	fill	Select fill or not fill text's background.
Fill color	fillcolor	Color of the text's background.

Also for all text/editfield objects you can use fields in ST scripts:

- **eventscount** - number of events are shown in the events ticker.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Line Color**" tab are described [here](#)³⁵⁰.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

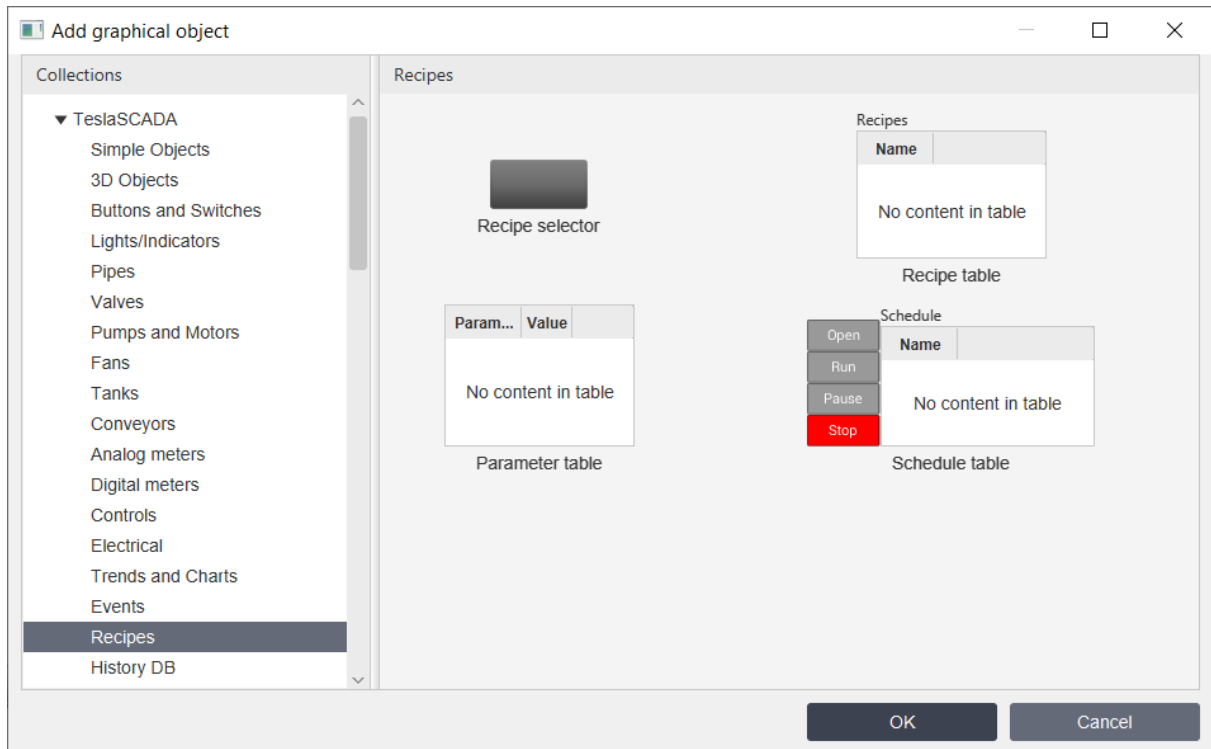
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

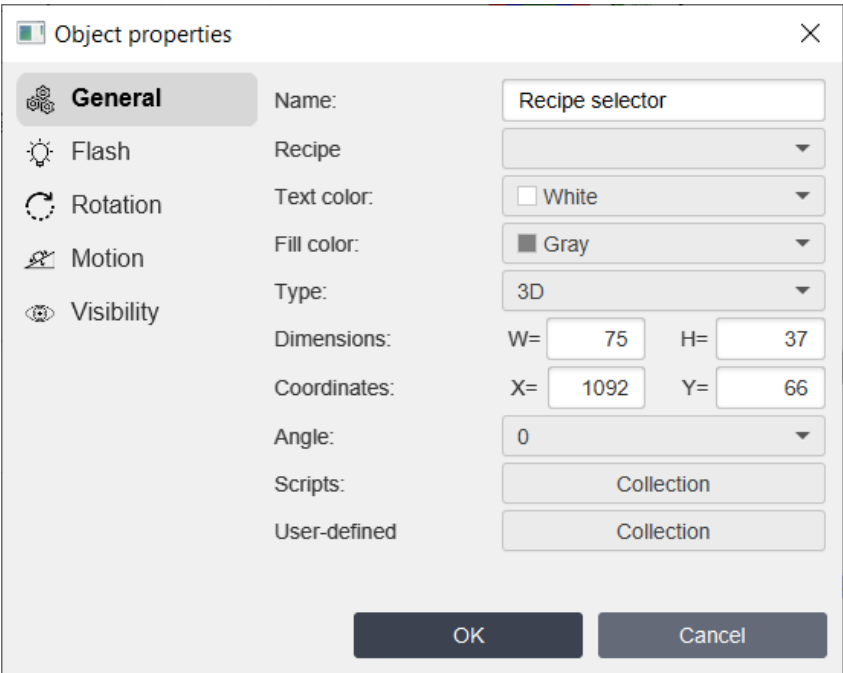
6.2.3.17 Recipes library



Recipes library contains the following objects that works with [recipes](#) ⁴⁸³ databases:

- [Recipe table](#) ²⁴⁷
- [Recipe selector](#) ²⁴⁶
- [Parameter table](#) ²⁴⁸
- [Schedule table](#) ²⁵⁰

6.2.3.17.1 Recipe selector



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Recipe	recipename	Choose Recipe ⁴⁸³ you want to bind to the selector. During running you can select ?elds of the recipe database by clicking on the recipe selector.
Text color	textcolor	Color of the text.
Fill color	fillcolor	Color of the selector.

Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.17.2 Recipe table

The screenshot shows the 'Object properties' dialog box for a 'Recipe table' object. The 'General' tab is selected. The properties are as follows:

- Name:** Recipe table
- Recipe:** (Dropdown menu)
- Use title:** ☒
- Title:** Recipes
- Font size:** 12
- Name column width:** 60
- Other column width:** 40
- Dimensions:** W= 150, H= 112
- Coordinates:** X= 781, Y= 81
- Angle:** 0
- Scripts:** Collection
- User-defined:** Collection

Buttons: OK, Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Recipe	recipename	Choose Recipe ⁴⁸³ you want to bind to the table. During running you can add, edit and delete ?elds of the recipe database by clicking right button on the table and choosing operation.
Use title	usetitle	Use title for the table.
Title	title	Title of the table.
Font size	fontsize	Size of the text's font.
Name column width	namecolumn width	Set width of the name's column.
Other column width	othercolumn width	Set width of other columns.

Also Recipe Table object has several properties that you can't setup by using settings dialog window, but you can setup by using ST script:

- **ownumber** - number of the row is chosen (clicked) by user.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

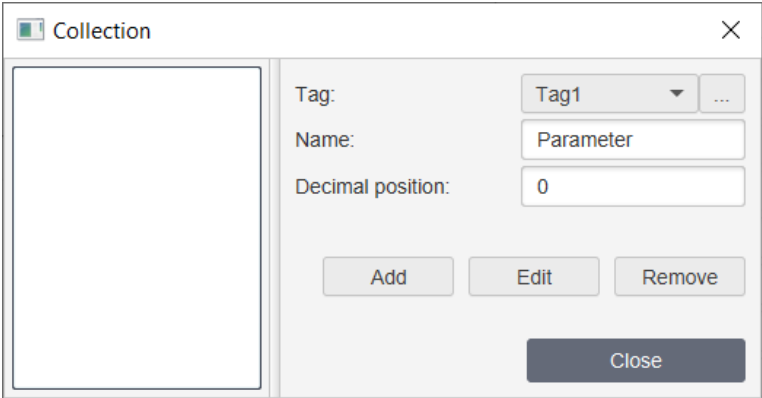
Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.17.3 Parameter table

The screenshot shows the 'Object properties' dialog window for a 'Parameter table' object. The 'General' tab is selected. The dialog contains various configuration options for the object, including its name, parameter column, value column, and dimensions. The 'Parameters' section shows a 'Collection' button. The 'Dimensions' section shows 'W= 150' and 'H= 112'. The 'Coordinates' section shows 'X= 1250' and 'Y= 320'. The 'Angle' is set to 0. The 'Scripts' section shows a 'Collection' button. The 'User-defined' section shows a 'Collection' button. The 'OK' and 'Cancel' buttons are at the bottom.

Property	Value
Name:	Parameter table
Parameter column:	Parameter
Value column:	Value
Use DB value:	<input type="checkbox"/>
DB value column:	DB value
Recipe:	
Row number:	0
Font size:	12
Name column width:	60
Other column width:	40
Parameters:	Collection
Dimensions:	W= 150 H= 112
Coordinates:	X= 1250 Y= 320
Angle:	0
Scripts:	Collection
User-defined:	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Parameter column	parametercolumn	Parameter column name.
Value column	valuecolumn	Value column name.
Use DB value	usedb	Check it if you want to use DB value column.
DB value column	dbcolumn	DB value column name.
Recipe	recipe name	Choose Recipe ⁴⁸³ you want to bind to the table
Row number	row number	Row number of the database which be used in DB value column.
Font size	font size	Size of the text's font.
Name column width	name column width	Set width of the name's column.
Other column width	other column width	Set width of other columns.
Parameters		<p>After clicking Collection button you'll see the window:</p>  <p>where:</p>

Property	ST script field	Description
		<ul style="list-style-type: none"> ▪ Tag - tag you want to bind to the table's parameter. ▪ Name - name of the parameter. ▪ Decimal position - decimal position for the tag's value. ▪ Add - add parameter. ▪ Edit - edit parameter. ▪ Remove - remove parameter.

Properties from the "**Row number**" tab are described [here](#)^[376].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.17.4 Schedule table

The screenshot shows the 'Object properties' dialog box for a 'Schedule table'. The 'General' tab is selected. The dialog contains the following fields and options:

- Name:** Schedule table
- Default schedule:** (dropdown menu)
- Title:** Schedule
- Font size:** 12
- Name column width:** 60
- Other column width:** 40
- Time interval:** (dropdown menu) ...
- ☐ Repeat
- Dimensions:** W= 225 H= 112
- Coordinates:** X= 1005 Y= 269
- Angle:** 0 (dropdown menu)
- Scripts:** Collection
- User-defined:** Collection

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Default schedule	recipename	Choose default schedule from Recipes ⁴⁸³ you want to bind to the table. During running you can add, edit and delete ?elds of the recipe database by clicking right button on the table and choosing operation
Title	title	Title of the table.
Font size	fontsize	Size of the text's font.
Name column width	namecolumn width	Set width of the name's column.
Other column width	othercolumn width	Set width of other columns.
Time interval	timertagname	Choose time interval tag. Depending of this tag's value will be duration of the next step(row) of the schedule table.
Repeat	repeat	Check it if you want to repeat all schedule steps (rows).

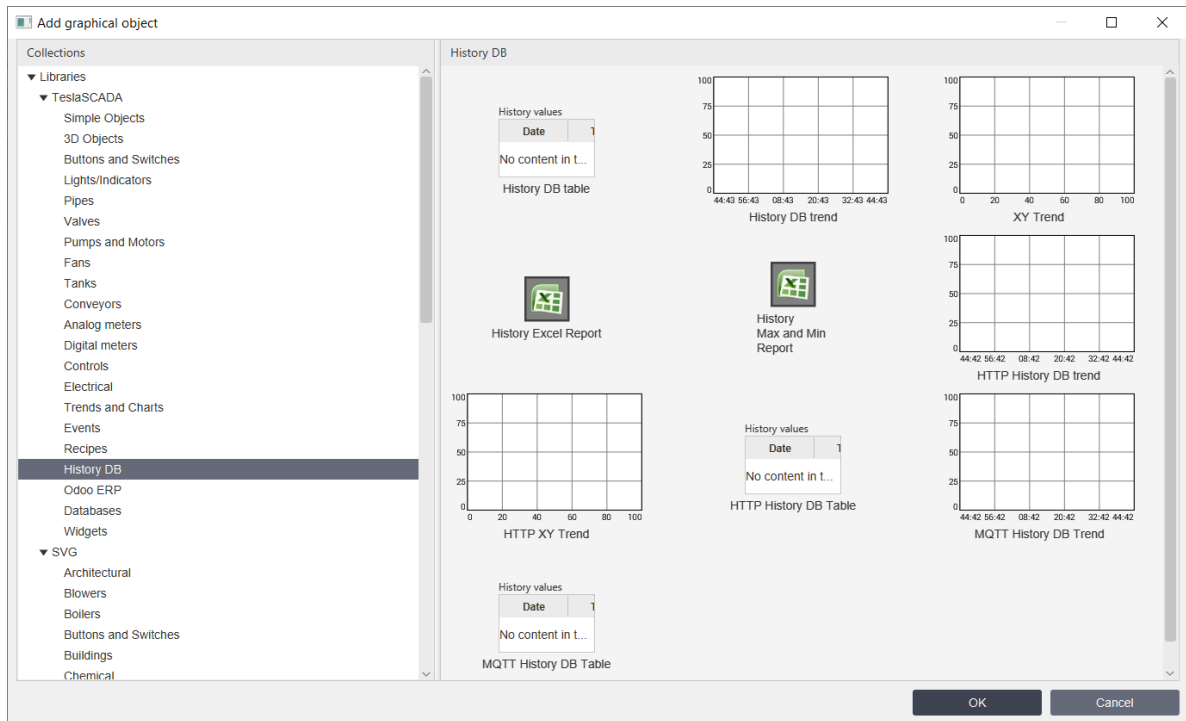
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.18 History DB library



History DB library contains the following objects that works with [History DB](#) ⁴⁸⁵ databases:

- [History DB table](#) ²⁵²
- [History DB trend](#) ²⁵⁵
- [XY Trend](#) ²⁵⁸
- [History Excel Report](#) ²⁶³
- [History Max and Min Report](#) ²⁶³
- [HTTP history DB trend](#) ²⁵⁵
- [HTTP XY Trend](#) ²⁵⁸
- [HTTP History DB table](#) ²⁵²
- [MQTT history DB trend](#) ²⁵⁵
- [MQTT History DB table](#) ²⁵²

6.2.3.18.1 History DB table

This section applies to the following objects: History DB table, HTTP History DB table, MQTT History DB table.

Object properties

General

Name: History DB table

History DB: [dropdown]

Use title: ☒

Title: History values

Font size: 12

Date and time type: 2 columns [dropdown]

Format: HH:mm:ss

Date column width: 80

Time column width: 80

Other column width: 60

Decimal position: 0

Auto refresh: ☐

Time order by: ASC [dropdown]

Dimensions: W= 112 H= 75

Coordinates: X= 60 Y= 260

Angle: 0 [dropdown]

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
History DB	historydbname	Choose History DB ⁴⁸⁵ you want to bind to the table.
Use title	usetitle	Use title for the table or not.
Title	title	Title of the table.
Font size	fontsize	Size of the text's font.

Property	ST script field	Description
Date and Time type	type	Date and time type representation (2 columns or 1 column)
Format	timeformat	Date and time format
Date column width	datecolumn width	Set width of the date's column.
Time column width	timecolumn with	Set width of the time's column.
Other column width	othercolumn width	Set width of other columns.
Decimal position	decimalpos	Decimal position of tag's values entered in the table.
Auto refresh	autorefresh	Check it if you want to update table every time when new tag's value added into database.
Time order by	orderby	Choose time order by of the database rows: <ul style="list-style-type: none"> ▪ ASC ▪ DESC
*HTTP server		Choose HTTP server

***Only for HTTP History DB table**

Also History DB Table object has several properties that you can't setup by using settings dialog window, but you can setup by using ST script:

- **begin** - start time for table information. Time represented in minutes from current period. (start time = current time - begin).
- **end** - ?nish time for table information. Time represented in minutes from current period. (?nish time = current time - end).
- **? lename** - name of the report's ?le.
- **savereport** - when this value become true trend's report will be created.
- **begindatetime** - start time for trend information. Time represented in milliseconds from 1 January 1970.
- **enddatetime** - ?nish time for trend information. Time represented in milliseconds from 1 January 1970.
- **disablesavereport** - disable "Save report" button in the dialog.
- **disableprint** - disable "Print" report button in the dialog.

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.18.2 History DB trend

This section applies to the following objects: History DB trend, HTTP history DB trend, MQTT history DB trend.

Object properties

General

Name: History DB trend

History DB: [dropdown]

Line width: 1

Color: Black

Fill: true

Fill color: White

Auto refresh: ☐

Default period(min): 60

Curves: Collection

Dimensions: W= 225 H= 150

Coordinates: X= 1212 Y= 174

Angle: 0

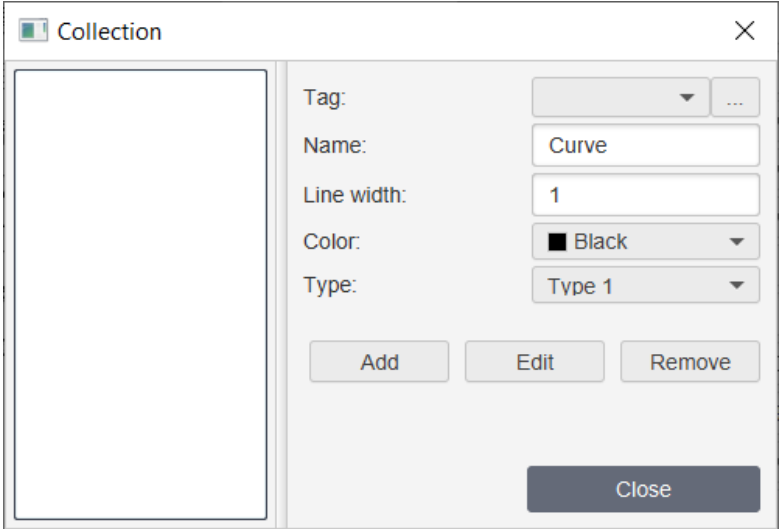
Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
History DB	history dbname	Choose History DB ⁴⁸⁵ you want to bind to the trend.
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.

Property	ST script field	Description
Fill	fill	Select fill or not fill trend.
Fill color	fillcolor	Fill color of the trend.
Auto refresh	autorefresh	Check it if you want to update trend every time when new tag's value added into database.
Curves		<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this curve. ▪ Name - name of the curve. ▪ Line with - curve's line width. ▪ Color - curve's line color. ▪ Type - line's type: <ul style="list-style-type: none"> ✓ Type 1 - just draw the line. ✓ Type 2 - draw line with ?lling till axis X. ✓ Type 3 - draw a ladder line. ✓ Type 4 - draw a ?lled ladder line.
Default period (min)	default period	Default time period of the trend (end time - begin time).

Also History DB Trend object has several properties that you can't setup by using settings dialog box, but you can setup by using ST script:

- **begin** - start time for trend information. Time represented in minutes from current period. (start time = current time - begin).
- **end** - ?nish time for trend information. Time represented in minutes from current period. (?nish time = current time - end).
- **? lename** - name of the report's ?le.
- **savereport** - when this value become true trend's report will be created.
- **begindatetime** - start time for trend information. Time is represented in milliseconds from 1 January 1970.
- **enddatetime** - ?nish time for trend information. Time is represented in milliseconds from 1 January 1970.
- **disablesavereport** - disable "Save report" button in the dialog.
- **disableprint** - disable "Print" report button in the dialog.
- **duration** - duration of the history time line in minutes. It works only when auto refresh is enabled. End time will be current time and begin time will be current time minus duration in minutes.

Properties from the "**Grid**" tab are described [here](#)²⁵⁷.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.18.2.1 Grid

The screenshot shows the 'Object properties' dialog box with the 'Grid' tab selected. The dialog has a sidebar on the left with icons for General, Grid, Fill color, Flash, Rotation, Motion, and Visibility. The main area contains the following settings:

Line width:	1
Color:	Gray
Line style:	Solid
Horizontally:	5
Vertically:	4
Maximum:	100.0
Minimum:	0.0
Font size:	10
Mark color:	Black
Time format:	mm:ss

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Line width		Width of grid's lines .
Color		Color of grid's lines.
Line style	linestyle	Style of the line: <ul style="list-style-type: none">▪ Solid▪ Dash▪ Dot▪ DashDot
Horizontally	horizontally	Number of trend's horizontal grid lines.
Vertically	vertically	Number of trend's vertical grid lines.
Maximum	maximum	Maximum of the trend's value.
Minimum	minimum	Minimum of the trend's value.
Font size	fontsize	Font size of the trend's marks.
Mark color	markcolor	Color of the marks.
Time format	timeformat	Time format of the trend's time.

6.2.3.18.3 XY Trend

This section applies to the following objects: XY Trend, HTTP XY Trend.

Object properties

General

Name: XY Trend

History DB: [Dropdown]

Line width: 1

Color: Black

Fill: true

Fill color: White

Auto refresh: [Checkbox]

Default period(min): 60

Axis X tag: [Dropdown] ...

Curves: Collection

Dimensions: W= 225 H= 150

Coordinates: X= 684 Y= 376

Angle: 0

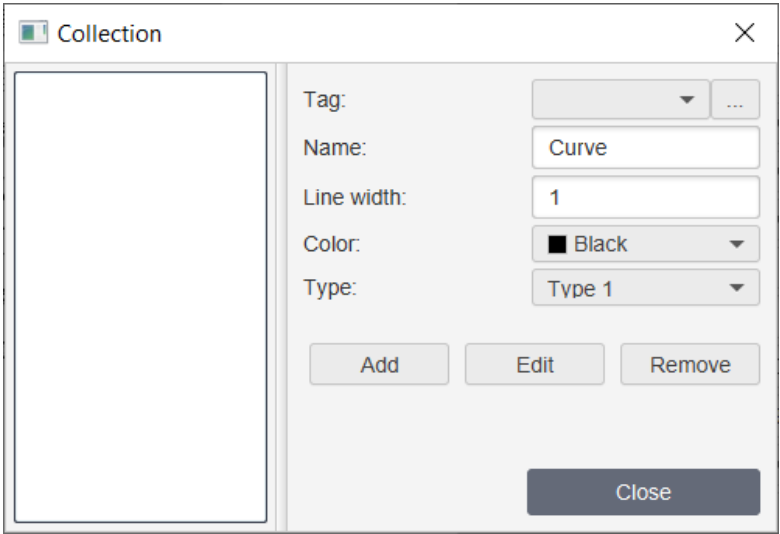
Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³).

Property	ST script field	Description
History DB*	history dbname	Choose History DB ⁴⁸⁵ you want to bind to the trend.
Line width	linewidth	Width of the border's line.
Color	color	Color of the border's line.
Fill	fill	Select fill or not fill trend.

Property	ST script field	Description
Fill color	fillcolor	Fill color of the trend.
Auto refresh	autorefresh	Check it if you want to update trend every time when new tag's value added into database.
Curves		<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this curve. ▪ Name - name of the curve. ▪ Line with - curve's line width. ▪ Color - curve's line color. ▪ Type - line's type: <ul style="list-style-type: none"> ✓ Type 1 - just draw the line. ✓ Type 2 - draw line with ?lling till axis X. ✓ Type 3 - draw a ladder line. ✓ Type 4 - draw a ?lled ladder line.
Default period (min)	default period	Default time period of the trend (end time - begin time).
Axis X tag	tagname	Bind tag to axis X of the trend.

***For HTTP XY Trend you have to enter History DB manually.**

Also History DB Trend object has several properties that you can't setup by using settings dialog box, but you can setup by using ST script:

- **begin** - start time for trend information. Time represented in minutes from current period. (start time = current time - begin).
- **end** - finish time for trend information. Time represented in minutes from current period. (finish time = current time - end).
- **filename** - name of the report's file.
- **savereport** - when this value become true trend's report will be created.

Properties from the "**Grid**" tab are described [here](#)^[261].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.18.3.1 Grid

The screenshot shows the 'Object properties' dialog box with the 'Grid' tab selected. The dialog has a sidebar on the left with icons for General, Grid, Fill color, Flash, Rotation, Motion, and Visibility. The main area contains the following settings:

Property	Value
Line width:	1
Color:	Gray
Line style:	Solid
Horizontally:	5
Vertically:	4
Maximum:	100.0
Minimum:	0.0
Maximum X:	100.0
Minimum X:	0.0
Font size:	10
Mark color:	Black

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Line width		Width of grid's lines .
Color		Color of grid's lines.
Line style	linestyle	Style of the line: <ul style="list-style-type: none"> ▪ Solid ▪ Dash ▪ Dot ▪ DashDot
Horizontally	horizontally	Number of trend's horizontal grid lines.
Vertically	vertically	Number of trend's vertical grid lines.
Maximum	maximum	Maximum of the trend's value.
Minimum	minimum	Minimum of the trend's value.
Maximum X	maximumx	Maximum of the axis X trend's value.
Minimum X	minimumx	Minimum of the axis X trend's value.
Font size	fontsize	Font size of the trend's marks.
Mark color	markcolor	Color of the marks.
Time format	timeformat	Time format of the trend's time.

6.2.3.18.4 History Excel report and History Max and Min report

Object properties

General

Name: History Excel Report

History DB: [dropdown]

Title: History values

Title 2: Organization

Decimal position: 0

☐ Transparent background

Fill color: Gray

Type: 2D

Dimensions: W= 50 H= 50

Coordinates: X= 693 Y= 306

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾).

Property	ST script field	Description
History DB	historydbname	Choose History DB ¹⁴⁵⁾ you want to bind to the table.
Title	title	Title of the table of the report.
Title 2	title2	Second title of the table of the report.
Decimal position	decimalpos	Decimal position of tag's values entered in the report's table.
Transparent background	transparent	Check it if you want to make background of the button invisible.
Fill color	fillcolor	Fill color of the report's button.

Also reports object has several properties that you can't setup by using settings dialog window, but you can setup by using ST script:

- **begin** - start time for report information. Time is represented in minutes from current period. (start time = current time - begin).
- **end** - finish time for report information. Time is represented in minutes from current period. (finish time = current time - end).
- **filename** - name of the report's file.
- **savereport** - when this value becomes true trend's report will be created.

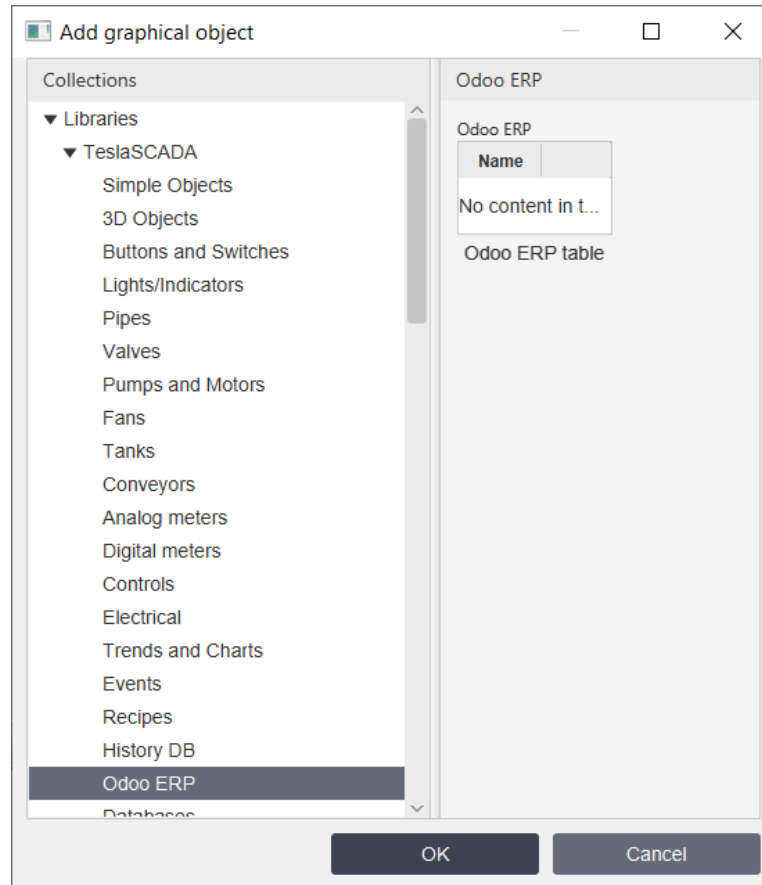
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.19 Odoo ERP



Odoo ERP library contains the following object:

- [Odoo ERP table](#)²⁶⁵

Odoo ERP table collects rows of Odoo ERP.

6.2.3.19.1 Odoo ERP table

Object properties

General

Name: Odoo ERP table

Odoo ERP: [Dropdown]

Model: [Dropdown]

Title: Odoo ERP

Font size: 12

Fields: Collection

Filters: Collection

Function: Collection

Auto refresh: ☐

Dimensions: W= 112 H= 75

Coordinates: X= 228 Y= 809

Angle: 0 [Dropdown]

Scripts: Collection

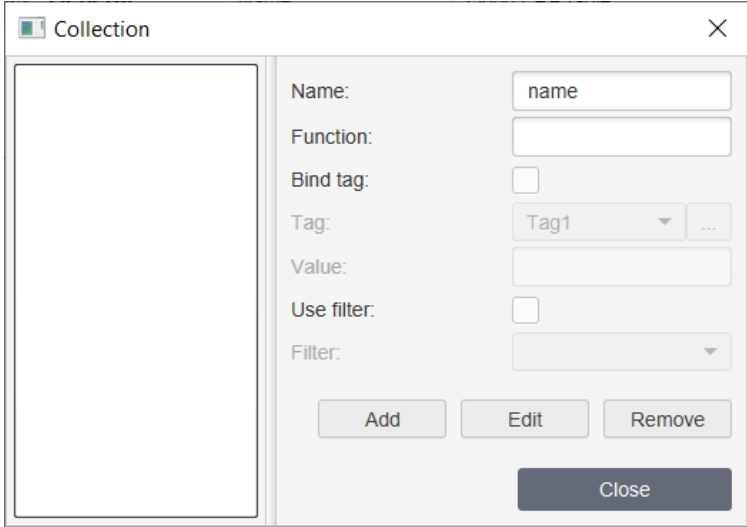
User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³).

Property	ST script field	Description
Odoo ERP	odooer pname	Choose Odoo ERP ⁴⁸⁸ bind to this table.
Model	model name	Choose model of the Odoo ERP.
Title	title	Title of the table.
Font size	fontsize	Font size of the table's texts.
Fields		After clicking Fields Collection button you'll see the window:

Property	ST script field	Description
		<div><div><div><div>Collection</div><div><div><div>Name</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div>Name:</div><div><input type="text" value="Name"/></div><div>Field:</div><div><input type="text" value="name"/></div><div>Width:</div><div><input type="text" value="60"/></div><div>Use relation:</div><div><input type="checkbox"/></div><div>Read only:</div><div><input checked="" type="checkbox"/></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div></div><p>where:</p><ul style="list-style-type: none">• Name - name of the ?eld.• Field - field of the Odoo ERP model.• Width - width of the ?eld’s column.• Use relation - check it to get data from the relation model.• Read only - check it if you don’t want to let change ?eld.</div>
Filters		<p>After clicking Filters Collection button you’ll see the window:</p> <div><div><div><div>Collection</div><div><div><div></div></div><div><div>Name:</div><div><input type="text" value="Name"/></div><div>Field:</div><div><input type="text" value="name"/></div><div>Comparison:</div><div><input "="" type="text" value="!="/></div><div>Value:</div><div><input type="text"/></div><div>Use:</div><div><input type="checkbox"/></div><div>Color:</div><div><div>■ Black</div></div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div></div> <p>where:</p> <ul style="list-style-type: none">▪ Name - name of the ?lter.▪ Field - field of the Odoo ERP model.▪ Comparison - choose comparison operation for the ?lter.▪ Value - value for the comparison.

Property	ST script field	Description
		<ul style="list-style-type: none"> ▪ Use - check it if you want to use this ?Iter for the table by default. ▪ Color - choose color for rows that ?ts for this ?Iter conditions.
Functions		<p>After clicking Functions Collection button you'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Name - name of the function. ▪ Function - function of the Odoo ERP model. ▪ Bind tag - check it if you want to bind the tag to the button. ▪ Tag - choose tag for the function. ▪ Value - value that will be written to the tag. ▪ Use ?Iter - check it to bind button of the function to the ?Iter (if check the button enable if ?Iter condition is TRUE). ▪ Filter - choose filter bind to the function.
Auto refresh	autorefresh	Check it to refresh table automatically.

When you click on the row of the table you select the row and you can use it in the script by using ?elds: **selectrow?eld** and **selectrowvalue**. At ?rst you should select ?eld of the row and then get or set value of the row.

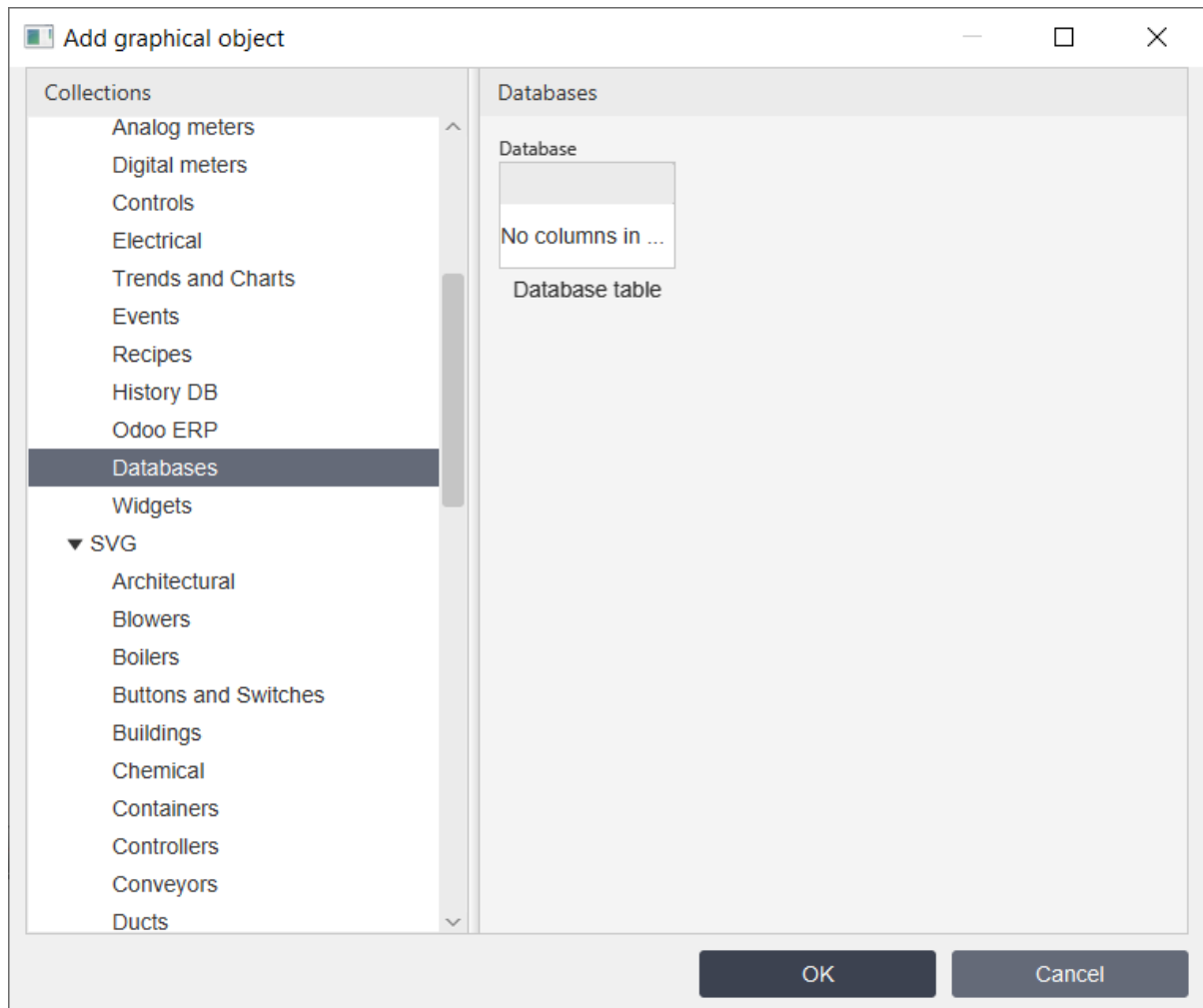
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.20 Databases library



Databases library contains the following object:

- [Database table](#)²⁶⁹

6.2.3.20.1 Database table

Object properties

General

Flash

Rotation

Motion

Visibility

Name:

Database:

Username:

Password:

Table name:

Use title:

Title:

Font size:

Column width:

Use custom columns:

Columns

Dimensions:

Coordinates:

Angle:

Scripts:

User-defined

Database table

☒

Database

12

60

☐

Collection

W= 112 H= 75

X= 569 Y= 740

0

Collection

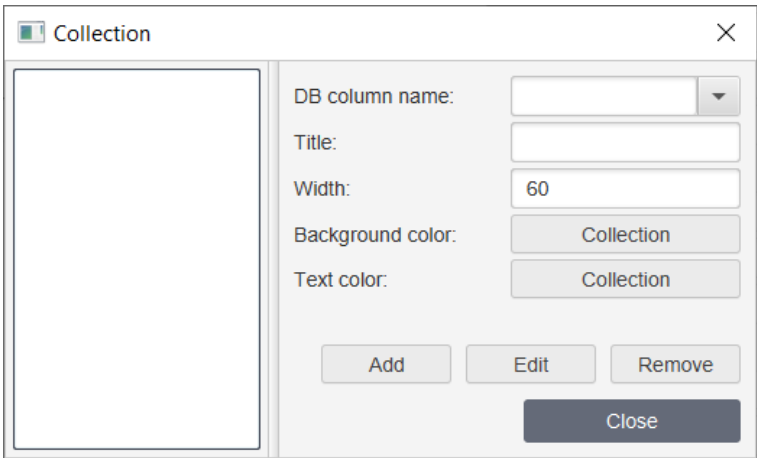
Collection

Cancel

OK

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Database	databas ename	Database name. If database name contains "jdbc:mysql" it means address of MySQL ^[31] database. If database name contains "jdbc:mssql" it means address of MSSQL ^[55] database. If database name contains "jdbc:postgresql" it means address of PostgreSQL ^[58] database. If database name doesn't contain "jdbc" it means address of SQLite ^[29] database.

Property	ST script field	Description
Username	username	Username for MySQL database.
Password	password	Password for MySQL database.
Table name	tablename	Name of the table.
Use title	usetitle	Use title for the table or not.
Title	title	Title of the table.
Font size	fontsize	Size of the text's font.
Column width	columnwidth	Set width of the columns.
Use custom columns		Check if you want to use custom columns.
Columns		<div data-bbox="667 1068 1419 1522">  </div> <p>where:</p> <ol style="list-style-type: none"> 1. DB column name - database column name of the DB you use for table. 2. Title - title you want to use for column. 3. Width - width of the column. 4. Background color - background color of the cell depending on value range. 5. Text color - text color of the cell depending on value range.

Also Database Table object has several properties that you can't setup by using settings dialog window, but you can setup by using ST script:

- **disablesavereport** - disable "Save report" menu item in the context menu.
 - **disableprint** - disable "Print" menu item in the context menu.
 - **columnwidtharray** - use this value to setup different column widths. Example: `Objects.Databasetable.columnwidtharray = "[10, 150, 150, 200]";`
 - **rownumber** - number of the row is chosen (clicked) by user.
 - **resultset*** - if you want to fill data from [Result set](#)^[441], set name of the result set here.
 - **csv*** - if you want to fill data from .csv file set name of the file here (if you use just name the file will be gotten from the [DB](#)^[18] folder. You can use the full path also).
- *resultset and csv don't works on iOS version.**

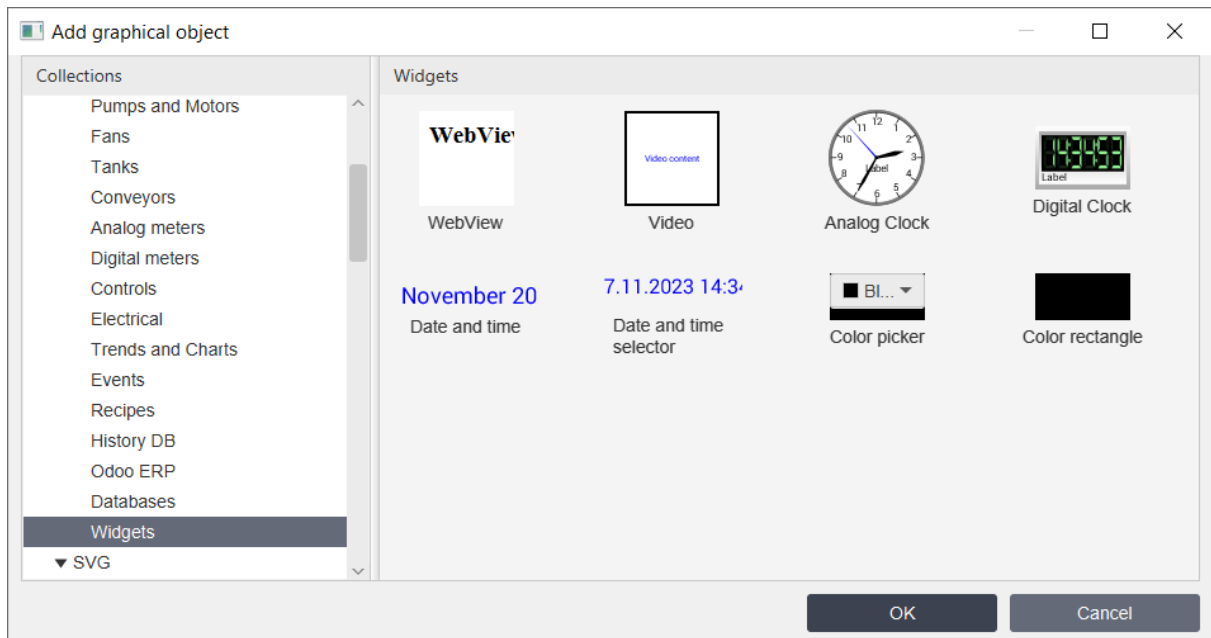
Properties from the **"Flash"** tab are described [here](#)^[345].

Properties from the **"Rotation"** tab are described [here](#)^[347].

Properties from the **"Motion"** tab are described [here](#)^[348].

Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.21 Widgets library

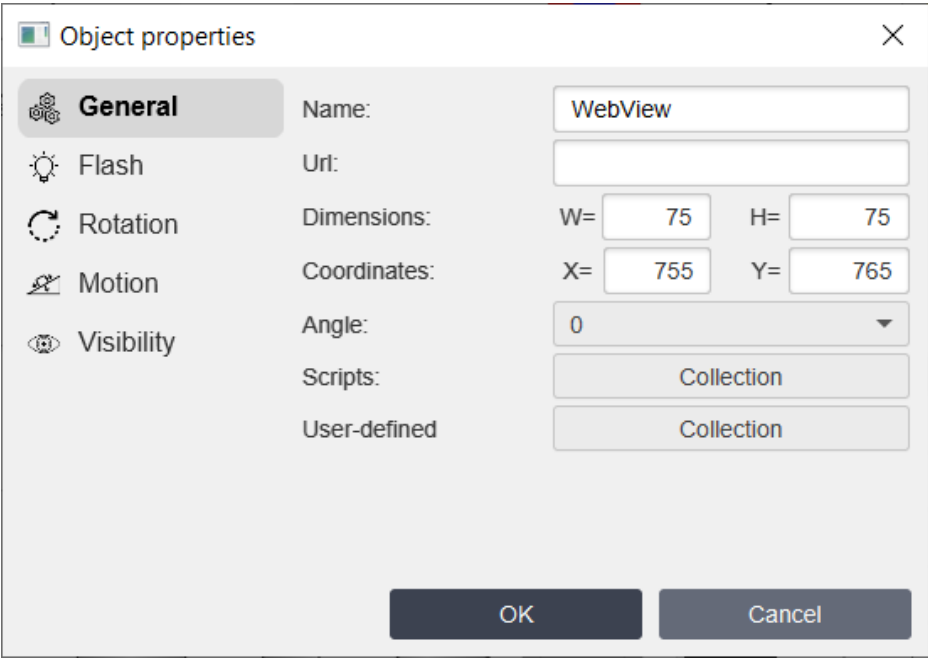


Widgets library contains the following object:

- [WebView](#)^[272]
- [Video](#)^[273]
- [Analog Clock](#)^[275]
- [Digital Clock](#)^[276]
- [Date and time](#)^[277]
- [Date and time selector](#)^[279]
- [Color picker](#)^[278]

- [Color rectangle](#)²⁷⁸

6.2.3.21.1 WebView



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Url	url	Url of the internet resource.

- Properties from the "**Flash**" tab are described [here](#)³⁴⁵.
- Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.
- Properties from the "**Motion**" tab are described [here](#)³⁴⁸.
- Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.21.2 Video

Object properties

General

Name: Video

Url:

Type: MP4,FXM,FLV,HLS

☒ Authentication

Username:

Password:

☐ Motion detect

Time interval: 1000

Axis X tag: ...

☐ Auto save image

Save condition: Tag.PV>Difference

Difference: 0.0

Dimensions: W= 75 H= 75

Coordinates: X= 871 Y= 745

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³).

Property	ST script field	Description
Url	url	Url of the internet resource.
Type	type	Type of the video signal: <ul style="list-style-type: none"> ▪ MP4,FXM,FLV,HLS ▪ MJPEG ▪ JPEG ▪ RTSP*
Authentication	security	Check it if your video camera use username and password for login.
Username	username	Username of the authentication.

Property	ST script field	Description
Password	password	Password of the authentication.
Motion detect	motiondetect	Check it for detecting motion by using this camera.
Time interval	interval	Time interval in ms for comparing 2 frames.
Tag		Choose tag for writing the value of comparing 2 frames in %.
Auto save image	autosaveimage	Check it if you want to save images from video camera depending on the value of motion detect.
Save condition	savecondition	Choose save condition.
Difference	diff	Difference between 2 frames in % during motion detect.

Properties Authentication, Username, Password, Motion detect, Time interval, Tag, Auto save image, Save condition, Difference is used only PC versions. These features doesn't work on Android and iOS.

* RTSP protocol can be used only on PC. You should install [VLC media player](#) for your OS to have possibility to use this protocol.

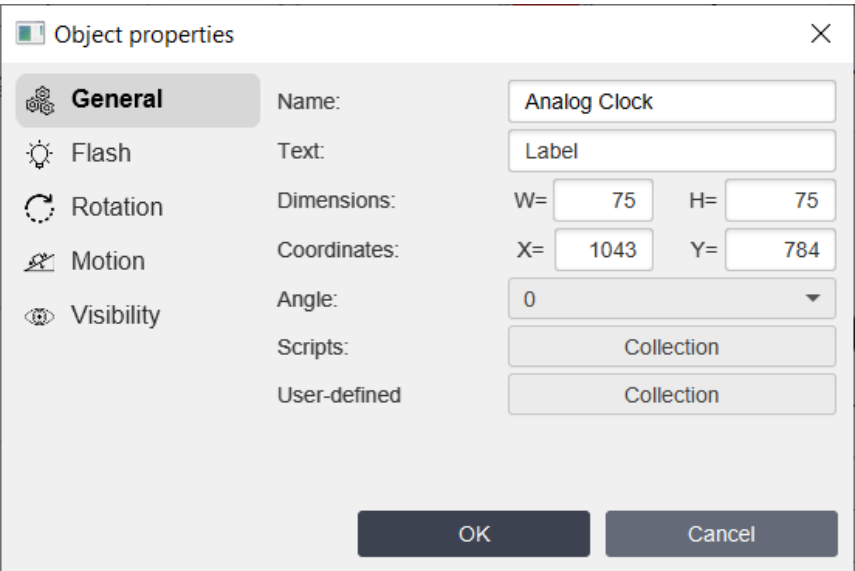
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.21.3 Analog clock



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Text	text	Text of the clock's label.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.
Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.
Properties from the "**Motion**" tab are described [here](#)³⁴⁸.
Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.21.4 Digital clock

Object properties

General

Name: Digital Clock

Text: Label

Text color: Light Green

Border color: Dark Gray

Fill color: Black

Type: 3D

Dimensions: W= 75 H= 50

Coordinates: X= 1191 Y= 775

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Text	text	Text of the label.
Text color	textcolor	Color of the clock's digits.
Fill color	fillcolor	Color of the clock's background.
Border color	bordercolor	Color of the clock's border.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Border color**" tab are described [here](#)³⁶⁶.

Properties from the "**Fill Color**" tab are described [here](#)³⁵².

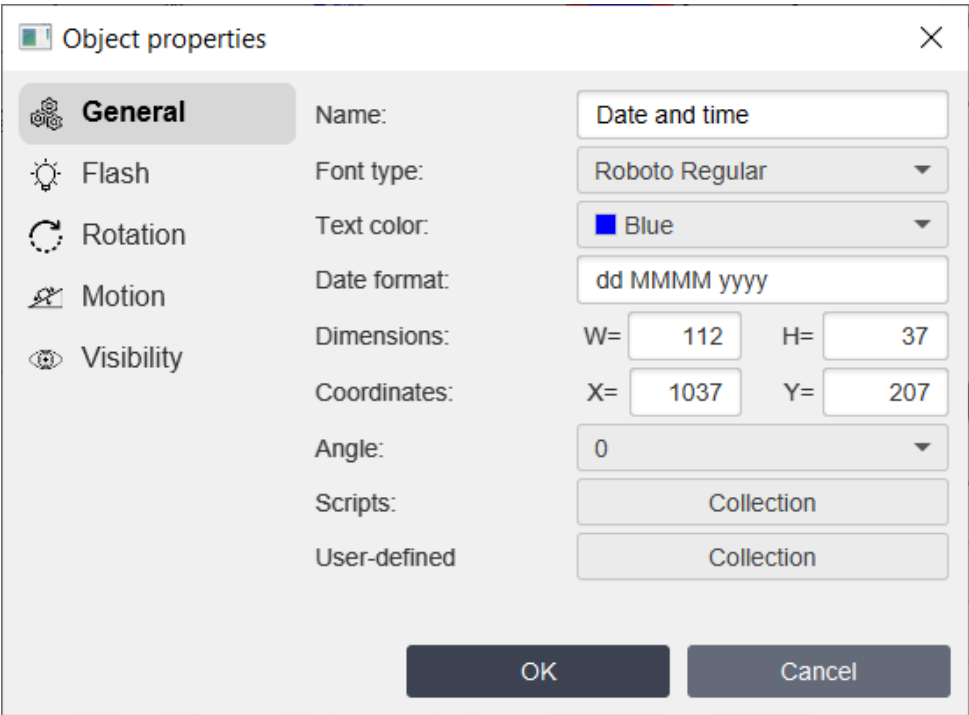
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.21.5 Date and time



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Font type	fonttype	Type of the text's font.
Text color	textcolor	Color of the text.
Date format	timeformat	Time format of the date and time object.

Properties from the "Flash" tab are described [here](#)³⁴⁵.
Properties from the "Rotation" tab are described [here](#)³⁴⁷.
Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.21.6 Color picker and Color rectangle

Object properties

General

Name: Color picker

Red color: ...

Green color: ...

Blue color: ...

Opacity: ...

Use rectangle: ☒

Dimensions: W= 75 H= 37

Coordinates: X= 483 Y= 821

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Red color	redcolortags tring	Choose Red color tag.
Green color	greencolor tagstring	Choose Green color tag.
Blue color	bluecolortag string	Choose Blue color tag.
Opacity	opacitycolor tagstring	Choose Opacity tag.

Properties from the "**Flash**" tab are described [here](#)^[345].

Properties from the "**Rotation**" tab are described [here](#)^[347].

Properties from the "**Motion**" tab are described [here](#)^[348].

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.21.7 Date and time selector

The screenshot shows the 'Object properties' dialog box with the 'General' tab selected. The object is named 'Date and time selector'. The format is 'dd.MM.yyyy HH:mm'. The font type is 'Roboto Regular', font size is '16', and text placement is 'CENTER'. The text color is 'Blue', border is 'false', border width is '2', border color is 'Black', fill is 'false', and fill color is 'White'. The dimensions are W=112 and H=37. The coordinates are X=1020 and Y=765. The angle is '0'. There are two 'Collection' buttons for 'Scripts' and 'User-defined'.

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Format	text	Date and time format of the selector.
Font type	fonttype	Type of the text's font.
Underline	underline	Check if you want to underline the text.
Font size	fontsize	Size of the text's font.
Text placement	textplacement	Placement of the text: <ul style="list-style-type: none"> ▪ Left ▪ Center ▪ Right

Property	ST script field	Description
Text color	textcolor	Color of the text.
Border	useborder	Select use or not use border for the text.
Border width	linewidth	Width of the border's line.
Border color	bordercolor	Color of the border's line.
Fill	fill	Select fill or not fill text's background.
Fill color	fillcolor	Color of the text's background.

Properties from the **"Output value"** tab are described [here](#)^[362].

Properties from the **"Text Color"** tab are described [here](#)^[355].

Properties from the **"Line Color"** tab are described [here](#)^[350].

Properties from the **"Fill Color"** tab are described [here](#)^[352].

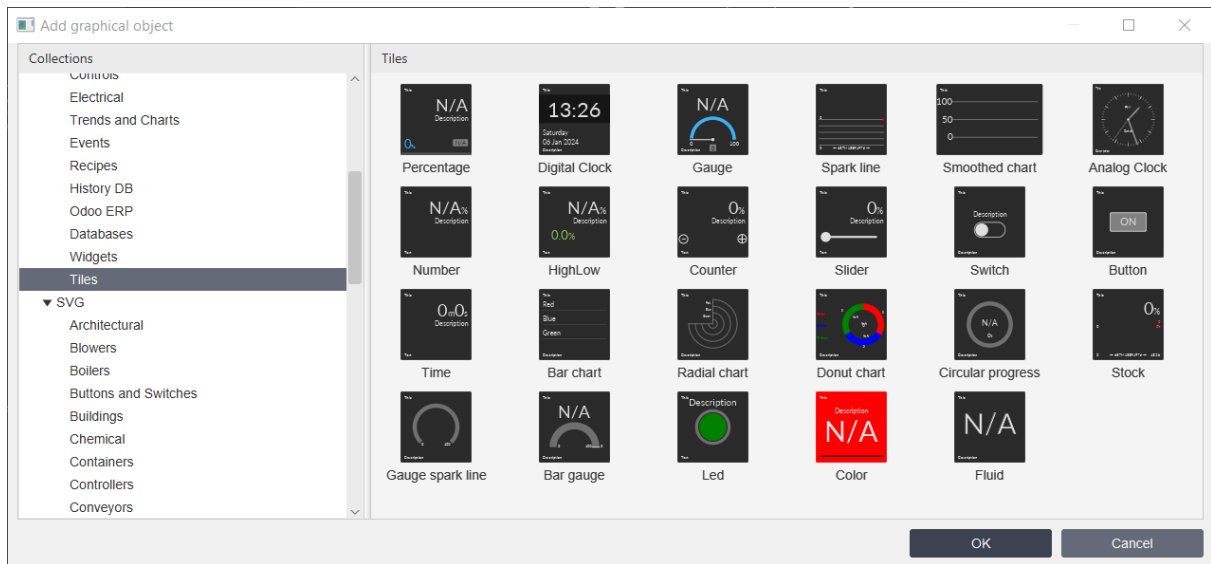
Properties from the **"Flash"** tab are described [here](#)^[345].

Properties from the **"Rotation"** tab are described [here](#)^[347].

Properties from the **"Motion"** tab are described [here](#)^[348].

Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.22 Tiles

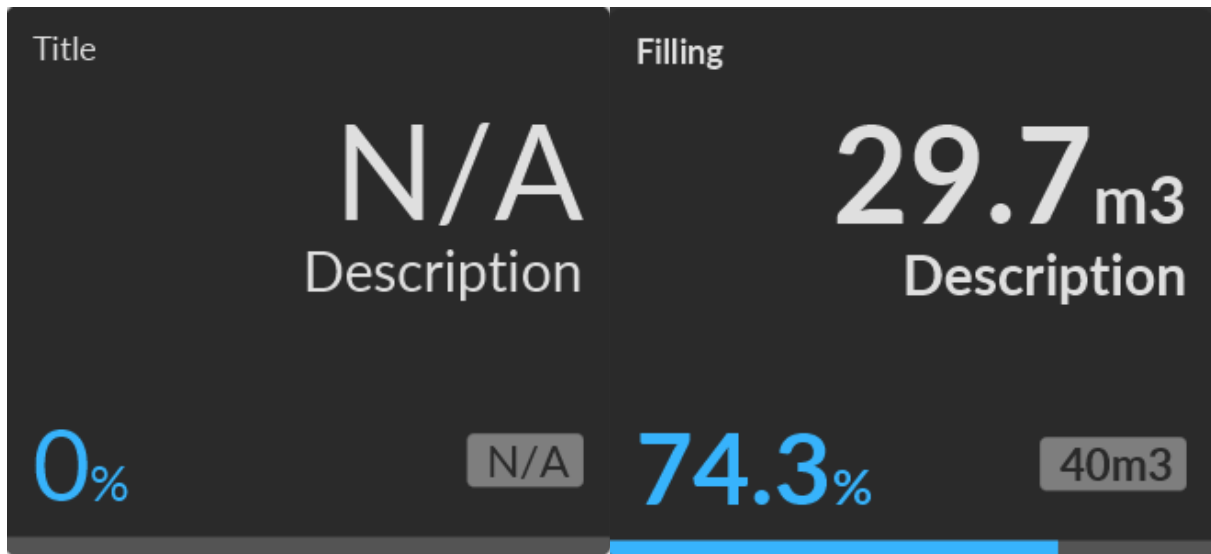


Tiles library contains the following object:

- [Percentage](#)^[281]
- [Digital Clock](#)^[283]
- [Gauge](#)^[285]
- [Spark line](#)^[287]
- [Smoothed chart](#)^[290]
- [Analog Clock](#)^[294]

- [Number](#) 295
- [HighLow](#) 297
- [Counter](#) 299
- [Slider](#) 301
- [Switch](#) 303
- [Button](#) 305
- [Time](#) 307
- [Bar chart](#) 309
- [Radial chart](#) 312
- [Donut chart](#) 315
- [Circular progress](#) 318
- [Stock](#) 320
- [Gauge spark line](#) 322
- [Bar gauge](#) 324
- [Led](#) 326
- [Color](#) 328
- [Fluid](#) 330

6.2.3.22.1 Percentage



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Percentage

Background color: #2a2a2a

Text color: #dfdfdf

Fill color: #37b3fc

Font type: Lato Regular

Title: Title

Description: Description

Unit:

Dimensions: W= 75 H= 75

Coordinates: X= 561 Y= 55

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

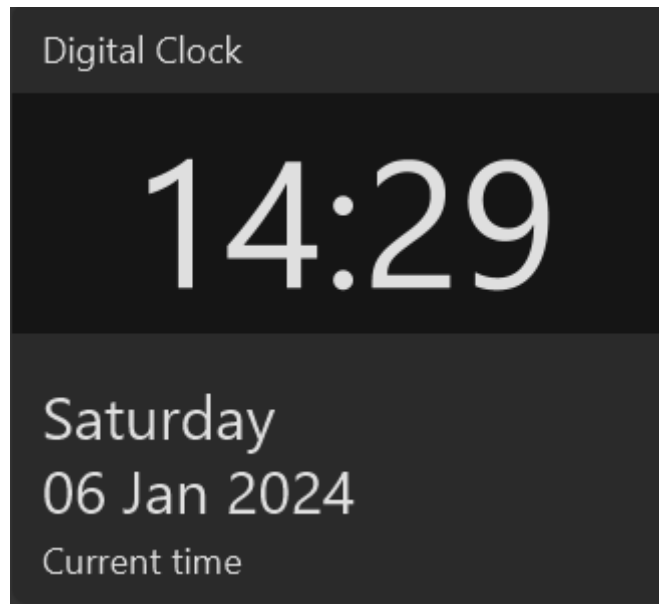
Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the percentage bar
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Value**" tab are described [here](#)³⁶⁹.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the **"Fill Color"** tab are described [here](#)³⁵²
Properties from the **"Text Color"** tab are described [here](#)³⁵⁵
Properties from the **"Flash"** tab are described [here](#)³⁴⁵
Properties from the **"Rotation"** tab are described [here](#)³⁴⁷
Properties from the **"Motion"** tab are described [here](#)³⁴⁸
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹

6.2.3.22.2 DigitalClockTile



Object properties

General

Name: Digital Clock

Back. color: Background color: #2a2a2a

Text color: #dfdfdf

Flash: Font type: Lato Regular

Rotation: Title: Title

Motion: Description: Description

Visibility: Date format: dd MMM YYYY

Time format: HH:mm

Dimensions: W= 75 H= 75

Coordinates: X= 484 Y= 485

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Date format	dateformat	Specify date format
Time format	timeformat	Specify time format

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.

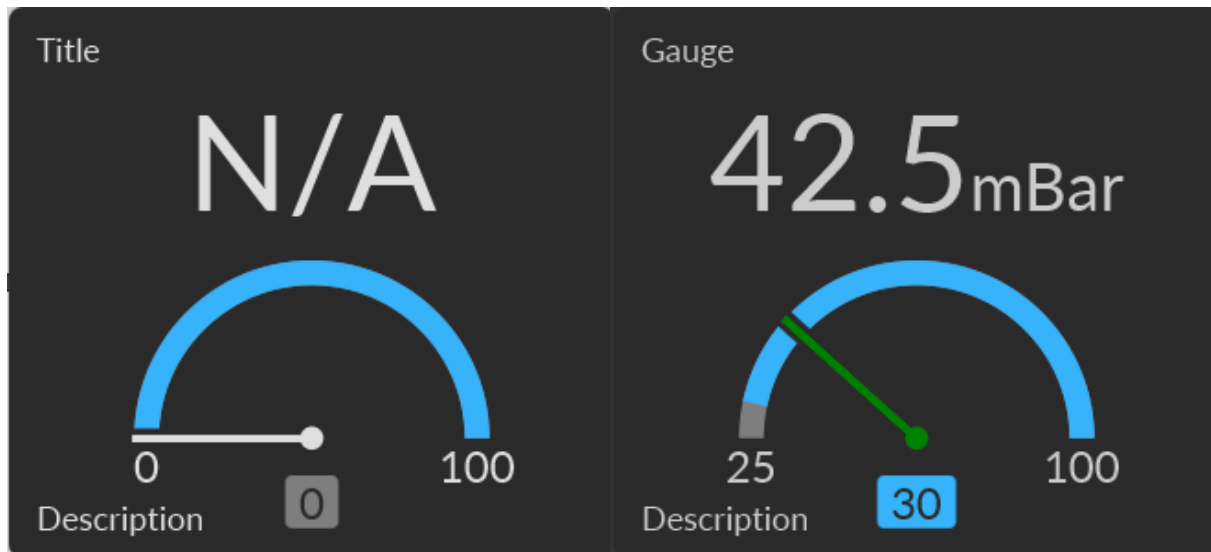
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.3 GaugeTile



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Gauge

Needle color: [icon]

Value: [icon]

Back. color: [icon]

Fill color: [icon]

Text color: [icon]

Flash: [icon]

Rotation: [icon]

Motion: [icon]

Visibility: [icon]

Background color: #2a2a2a

Text color: #dfdfdf

Fill color: #37b3fc

Needle(Fill) color: #dfdfdf

Font type: Lato Regular

Title: Title

Description: Description

Unit: [text box]

Threshold: 0.0

Dimensions: W= 75 H= 75

Coordinates: X= 319 Y= 565

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the arc of the gauge
Needle(fill) color	needlecolor	Specify needle color
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Property	ST script field	Description
Threshold	threshold	Specify the tag value up to which the arc color will be highlighted in a different color.

Properties from the **"Needle color"** tab are described [here](#)^[366].

Properties from the **"Value"** tab are described [here](#)^[370].

Properties from the **"Back. color"** tab are described [here](#)^[366].

Properties from the **"Fill Color"** tab are described [here](#)^[352].

Properties from the **"Text Color"** tab are described [here](#)^[355].

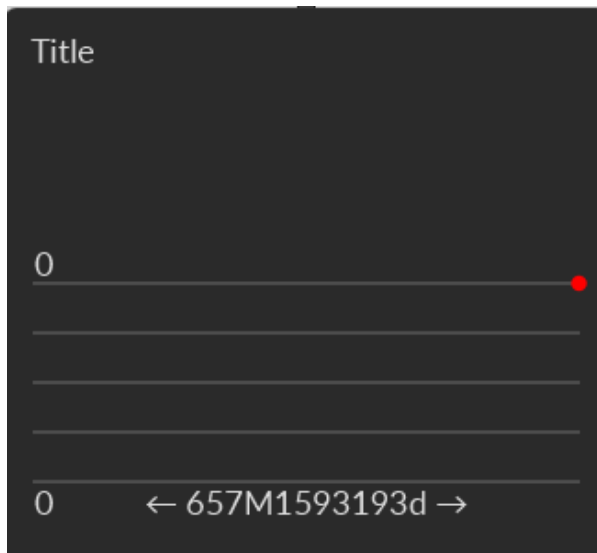
Properties from the **"Flash"** tab are described [here](#)^[345].

Properties from the **"Rotation"** tab are described [here](#)^[347].

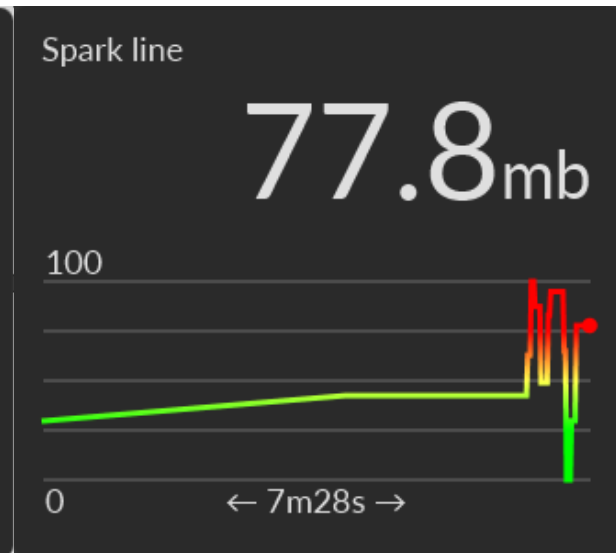
Properties from the **"Motion"** tab are described [here](#)^[348].

Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.22.4 Sparkline



pic. 1 - object image



pic. 2 - object image in a project

Object properties

General

Name: Spark line

Back. color: Background color: #2a2a2a

Text color: Text color: #d9d9d9

Flash: Tag: ...

Rotation: Line color: Red

Motion: Ranges: Collection

Visibility: Font type: Lato Regular

Title: Title

Default period(min): 10

Decimal position: 0

Unit: %

Dimensions: W= 75 H= 75

Coordinates: X= 699 Y= 544

Angle: 0

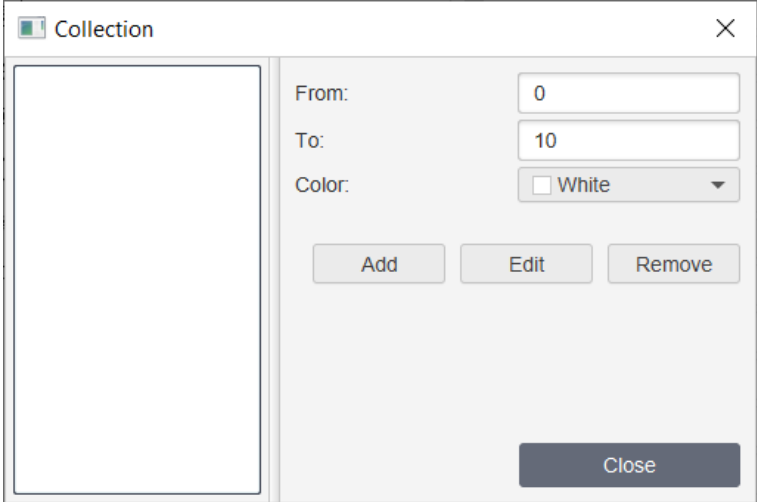
Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Tag	tagname	Enter tagname
Line color	linecolor	Specify the color of the line

Property	ST script field	Description
Ranges	---	<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> • From - enter the value from which curve will have color of this range. • To - enter the value to which curve will have color of this range. • Color - choose color for this range.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Default period (min)	defaultperiod	Default time period of the trend (end time - begin time).
Decimal position	decimalpos	Decimal position of tag's values
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

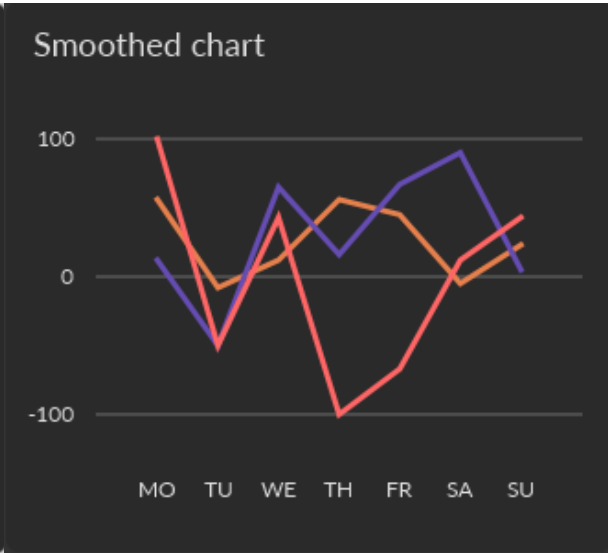
Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "Motion" tab are described [here](#)³⁴⁸.
Properties from the "Visibility" tab are described [here](#)³⁴⁹.

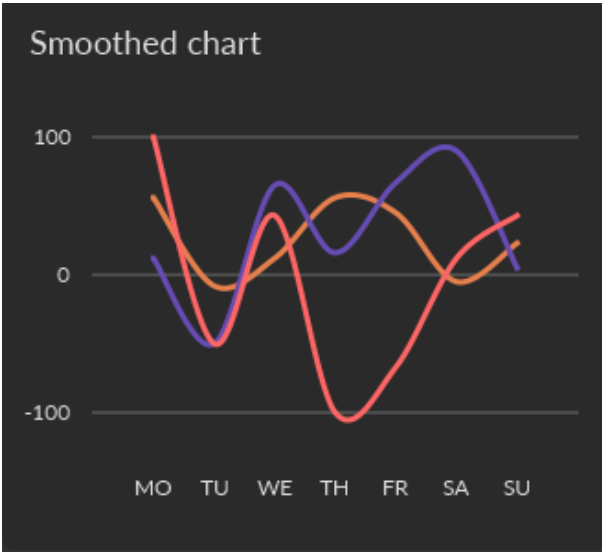
6.2.3.22.5 Smoothed chart



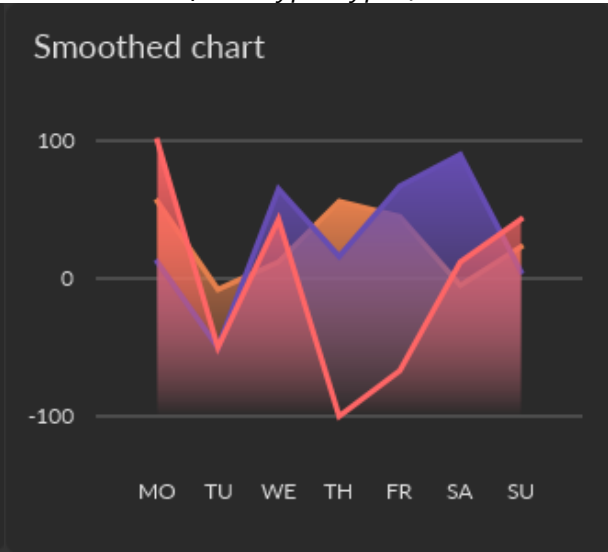
pic. 1 - object image



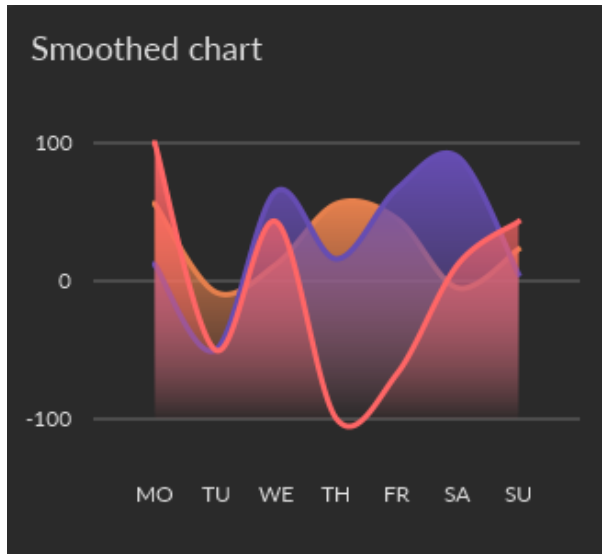
pic. 2 - object image in a project (Smoothing - false; Type -type1)



pic. 2 - object image in a project (Smoothing - true; Type -type1)



pic. 2 - object image in a project (Smoothing - false; Type -type2)



pic. 2 - object image in a project (Smoothing - true; Type -type2)

Object properties

General

Name: Smoothed chart

Back. color: Background color: #2a2a2a

Text color: #dfdfdf

Flash: Smoothing: ☒

Rotation: Font size: 10

Motion: Type: Type 1

Visibility: Font type: Lato Regular

Title: Title

Minimum: 0.0

Maximum: 100.0

Sectors: Collection

Sectors: Collection

Sectors: Collection

Dimensions: W= 112 H= 75

Coordinates: X= 100 Y= 535

Angle: 0

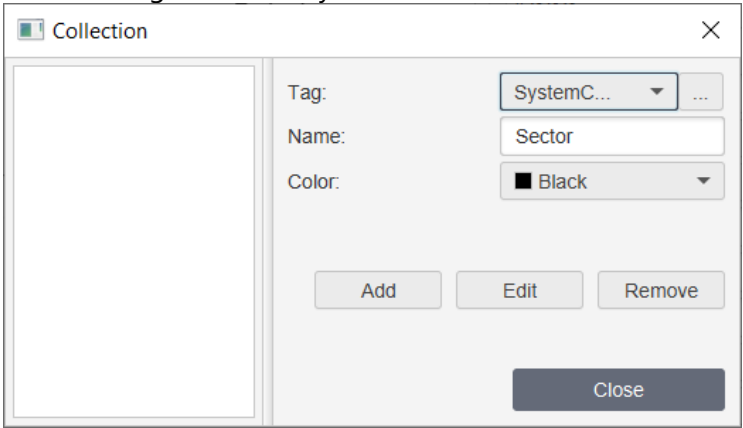
Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Smoothing	smoothing	Check if you want the line on the chart to be smooth

Property	ST script field	Description
Font size	fontsize	Specify font size
Type	charttype	Specify the chart type (type 1 - line, type 2 - area chart)
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Minimum	minimum	Specify the minimum value
Maximum	maximum	Specify the maximum value
Sectors	---	<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this bar. ▪ Name - name of the bar chart sector. ▪ Color - bar's color.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.22.6 Analog clock



Object properties

General

Back. color

Text color

Flash

Rotation

Motion

Visibility

Name: Analog Clock

Background color: #2a2a2a

Text color: #dfdfdf

Font type: Lato Regular

Title: Title

Description: Description

Dimensions: W= 75 H= 75

Coordinates: X= 453 Y= 491

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#) ¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

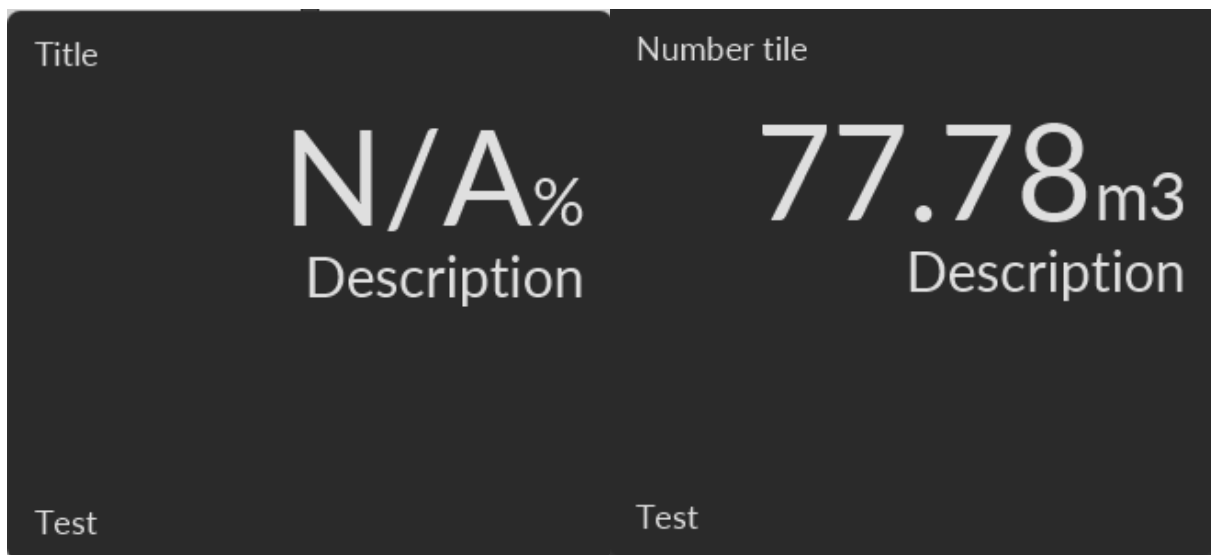
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.22.7 Number



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Number

Background color: #2a2a2a

Text color: #dfdfdf

Font type: Lato Regular

Title: Title

Text: Test

Description: Description

Unit: %

Dimensions: W= 75 H= 75

Coordinates: X= 615 Y= 333

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

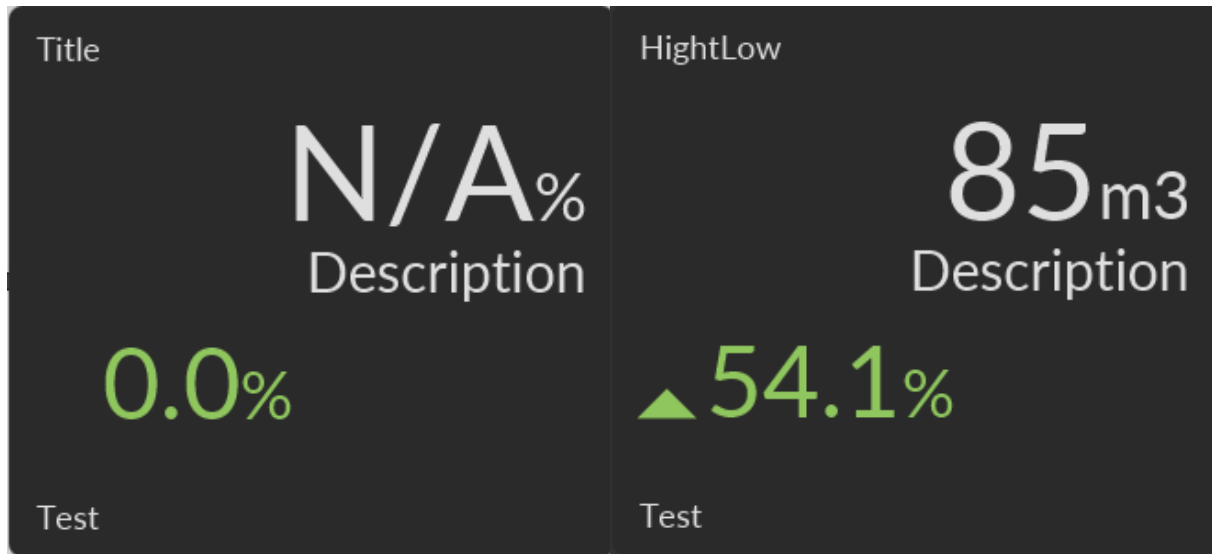
Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³⁾).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Value**" tab are described [here](#)³⁶⁹⁾.

Properties from the "**Back. color**" tab are described [here](#)^[366].
Properties from the "**Text Color**" tab are described [here](#)^[355].
Properties from the "**Flash**" tab are described [here](#)^[345].
Properties from the "**Rotation**" tab are described [here](#)^[347].
Properties from the "**Motion**" tab are described [here](#)^[348].
Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.22.8 HighLow



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: HighLow

Background color: #2a2a2a

Text color: #dfdfdf

Font type: Lato Regular

Title: Title

Text: Test

Description: Description

Unit: %

Dimensions: W= 75 H= 75

Coordinates: X= 363 Y= 455

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Value**" tab are described [here](#)^[369].

Properties from the "**Back. color**" tab are described [here](#)^[366].

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.

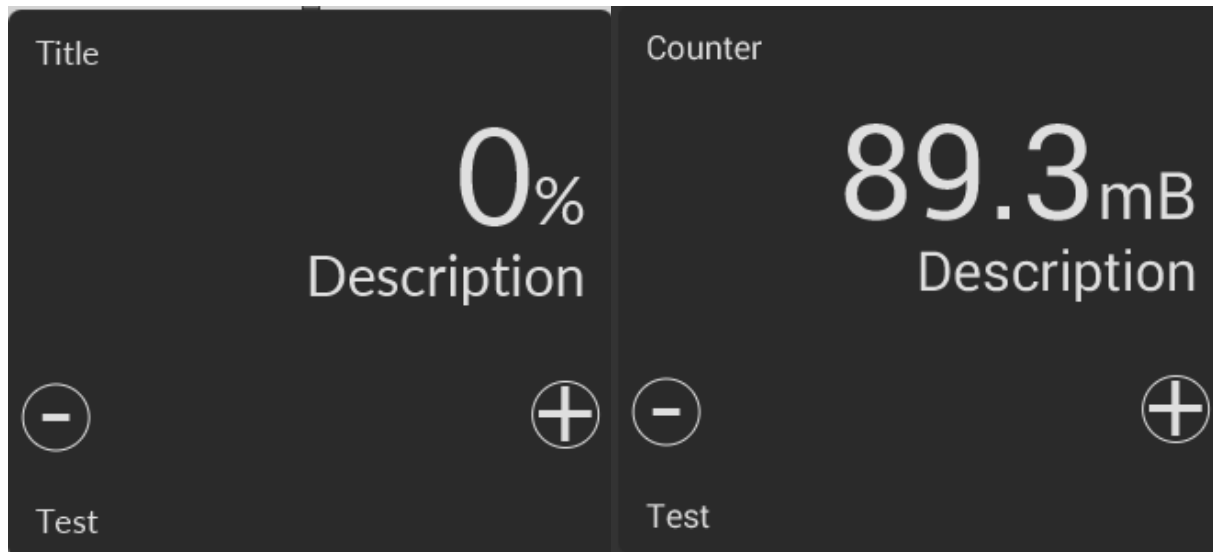
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.9 Counter



pic. 1 - object image

pic. 2 - object image in a project

The 'Object properties' dialog box is shown with the 'General' tab selected. The properties are as follows:

Property	Value
Name	Counter
Background color	#2a2a2a
Text color	#dfdfdf
Font type	Lato Regular
Title	Title
Text	Test
Description	Description
Unit	%
Dimensions (W, H)	W= 75, H= 75
Coordinates (X, Y)	X= 507, Y= 155
Angle	0
Scripts	Collection
User-defined	Collection

Buttons: OK, Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "Control" tab are described [here](#)³⁶⁸.

Properties from the "Back. color" tab are described [here](#)³⁶⁶.

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.

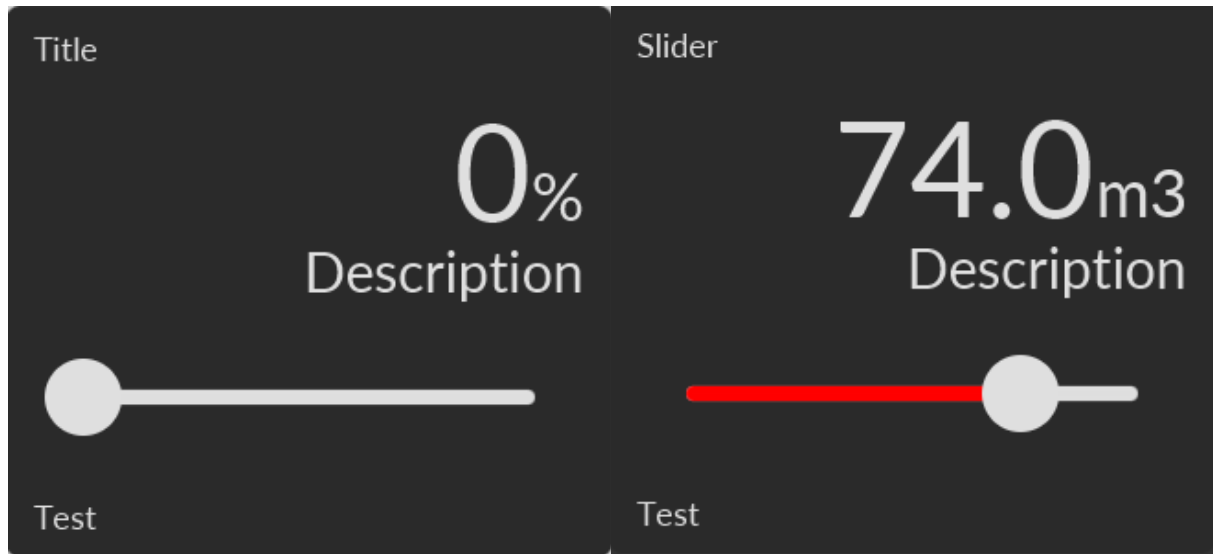
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.10 Slider



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Slider

Background color: #2a2a2a

Text color: #dfdfdf

Fill color: #37b3fc

Font type: Lato Regular

Title: Title

Text: Test

Description: Description

Unit: %

Dimensions: W= 75 H= 75

Coordinates: X= 711 Y= 131

Angle: 0

Scripts: Collection

User-defined: Collection

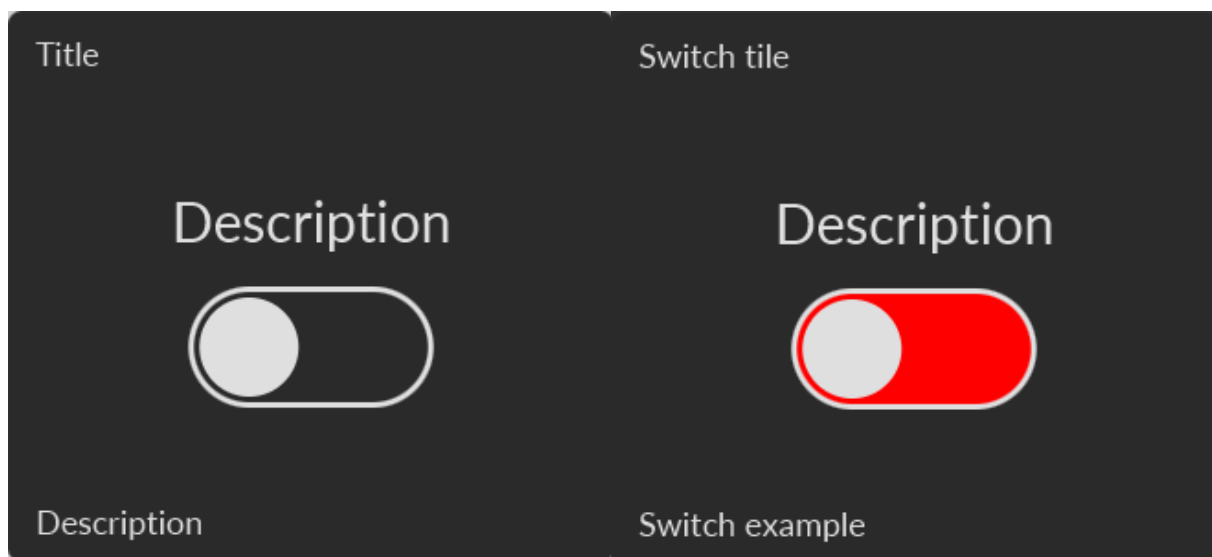
OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the bar that displays the tag value
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Control**" tab are described [here](#)^[367].
Properties from the "**Back. color**" tab are described [here](#)^[366].
Properties from the "**Fill Color**" tab are described [here](#)^[352].
Properties from the "**Text Color**" tab are described [here](#)^[355].
Properties from the "**Flash**" tab are described [here](#)^[345].
Properties from the "**Rotation**" tab are described [here](#)^[347].
Properties from the "**Motion**" tab are described [here](#)^[348].
Properties from the "**Visibility**" tab are described [here](#)^[349].

6.2.3.22.11 Switch



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Switch control

Back. color

Fill color

Text color

Flash

Rotation

Motion

Visibility

Name: Switch

Background color: #2a2a2a

Text color: #dfdfdf

Fill color: #2a2a2a

Font type: Lato Regular

Title: Title

Text: Description

Description: Description

Dimensions: W= 75 H= 75

Coordinates: X= 684 Y= 46

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the bar that displays the tag value
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary

Properties from the "**Switch control**" tab are described [here](#)^[372].

Properties from the "**Back. color**" tab are described [here](#)^[366].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.

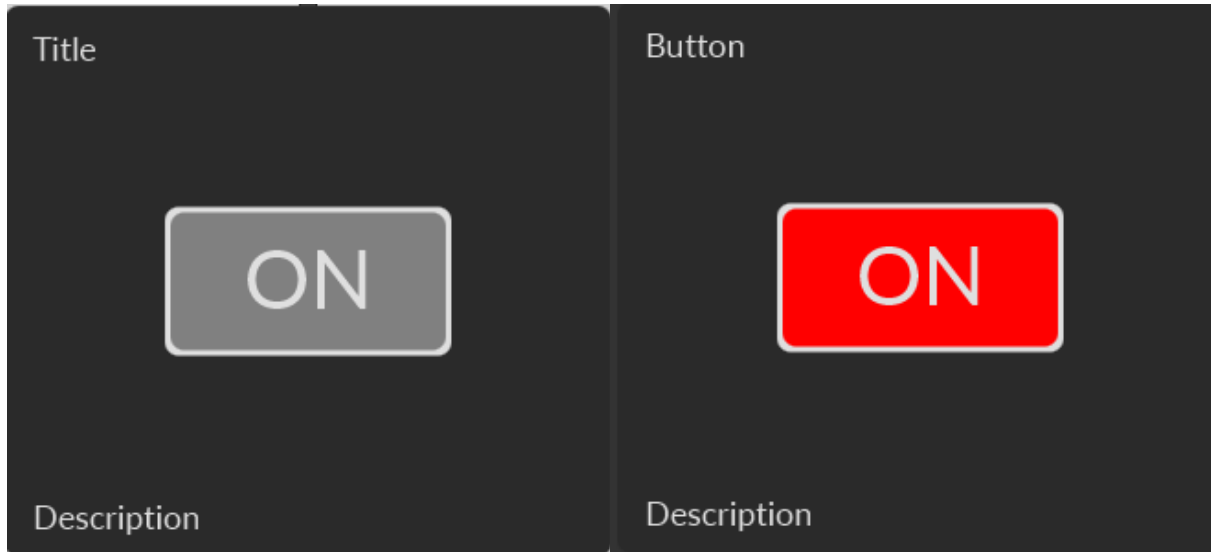
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.12 Button



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Button1

Background color: #2a2a2a

Text color: #dfdfdf

Fill color: Gray

Font type: Lato Regular

Title: Title

Text: Description

Description: ON

Dimensions: W= 75 H= 75

Coordinates: X= 44 Y= 494

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the bar that displays the tag value
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary

Properties from the "**Control**" tab are described [here](#)³⁵⁷.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the "**Fill Color**" tab are described [here](#)³⁵²
Properties from the "**Text Color**" tab are described [here](#)³⁵⁵
Properties from the "**Flash**" tab are described [here](#)³⁴⁵
Properties from the "**Rotation**" tab are described [here](#)³⁴⁷
Properties from the "**Motion**" tab are described [here](#)³⁴⁸
Properties from the "**Visibility**" tab are described [here](#)³⁴⁹

6.2.3.22.13 Time



Object properties

General

Name: Time

Back. color: Background color: #2a2a2a

Text color: #dfdfdf

Flash: Font type: Lato Regular

Rotation: Title: Title

Motion: Text: Test

Visibility: Description: Description

Tag: ...

Dimensions: W= 75 H= 75

Coordinates: X= 332 Y= 478

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Text	text	Set tile's text
Description	description	Set tile's description if necessary
Tag	tagname	Enter tagname

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.
Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.
Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.14 Bar chart



pic. 1 - object image

pic. 2 - object image in a project

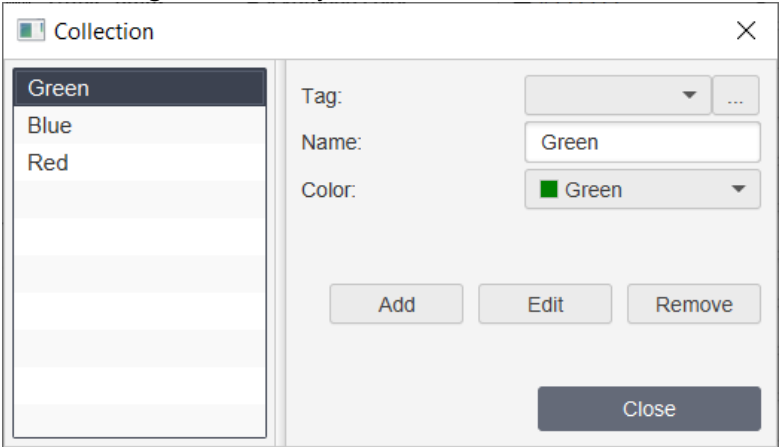
The screenshot shows the 'Object properties' dialog box for a bar chart object. The 'General' tab is selected. The properties are as follows:

Property	Value
Name:	Bar chart1
Back. color:	#2a2a2a
Text color:	#dfdfdf
Use legends:	<input checked="" type="checkbox"/>
Decimal position:	0
Font type:	Lato Regular
Title:	Title
Minimum:	0.0
Maximum:	100.0
Sectors:	Collection
Dimensions:	W= 75 H= 75
Coordinates:	X= 715 Y= 329
Angle:	0
Scripts:	Collection
User-defined	Collection

Buttons: OK, Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Use legends	uselegends	Check it if you want to add legends to the bar chart.

Property	ST script field	Description
Decimal position	decimalpos	Decimal position of tag's values entered in the table.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Minimum	text	Set tile's text
Maximum	description	Set tile's description if necessary
Sectors	---	<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this bar. ▪ Name - name of the bar chart sector. ▪ Color - bar's color.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

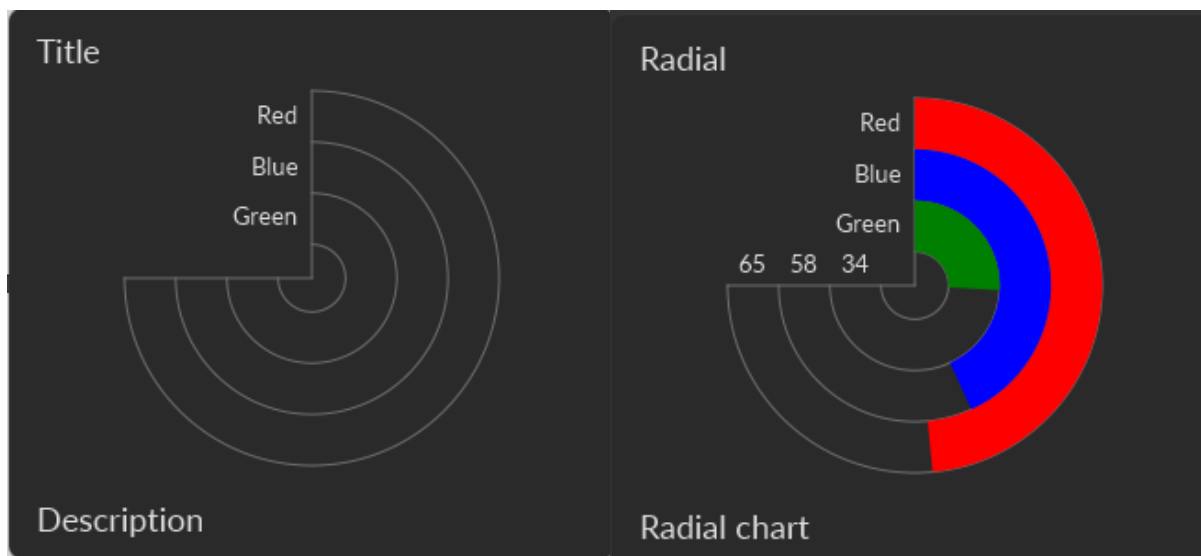
Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

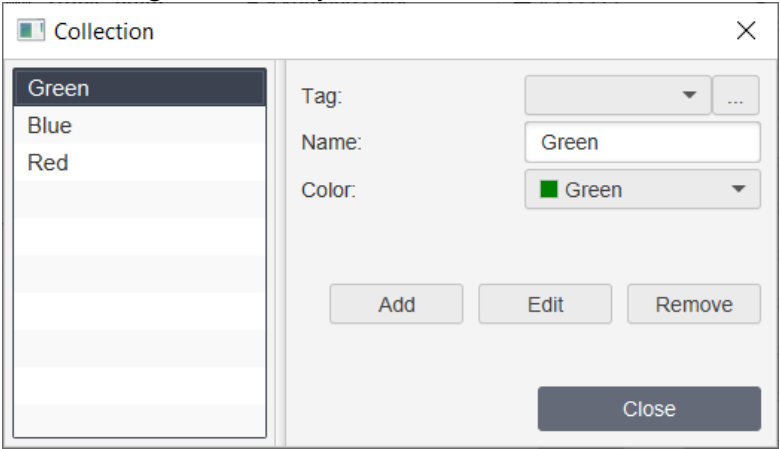
6.2.3.22.15 Radial chart*pic. 1 - object image**pic. 2 - object image in a project*

The screenshot shows the 'Object properties' dialog box for an object named 'Radial chart1'. The 'General' tab is selected, showing various configuration options. The background color is set to #2a2a2a and the text color is #dfdfdf. The 'Use legends' checkbox is checked. The decimal position is 0, and the font type is Lato Regular. The description is 'Description', the title is 'Title', the minimum value is 0.0, and the maximum value is 100.0. The sectors are set to 'Collection'. The dimensions are W=330 and H=300. The coordinates are X=91 and Y=183. The angle is 0. The scripts and user-defined fields are also set to 'Collection'. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Property	Value
Name:	Radial chart1
Background color:	#2a2a2a
Text color:	#dfdfdf
Use legends:	<input checked="" type="checkbox"/>
Decimal position:	0
Font type:	Lato Regular
Description:	Description
Title:	Title
Minimum:	0.0
Maximum:	100.0
Sectors:	Collection
Dimensions:	W= 330 H= 300
Coordinates:	X= 91 Y= 183
Angle:	0
Scripts:	Collection
User-defined	Collection

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.

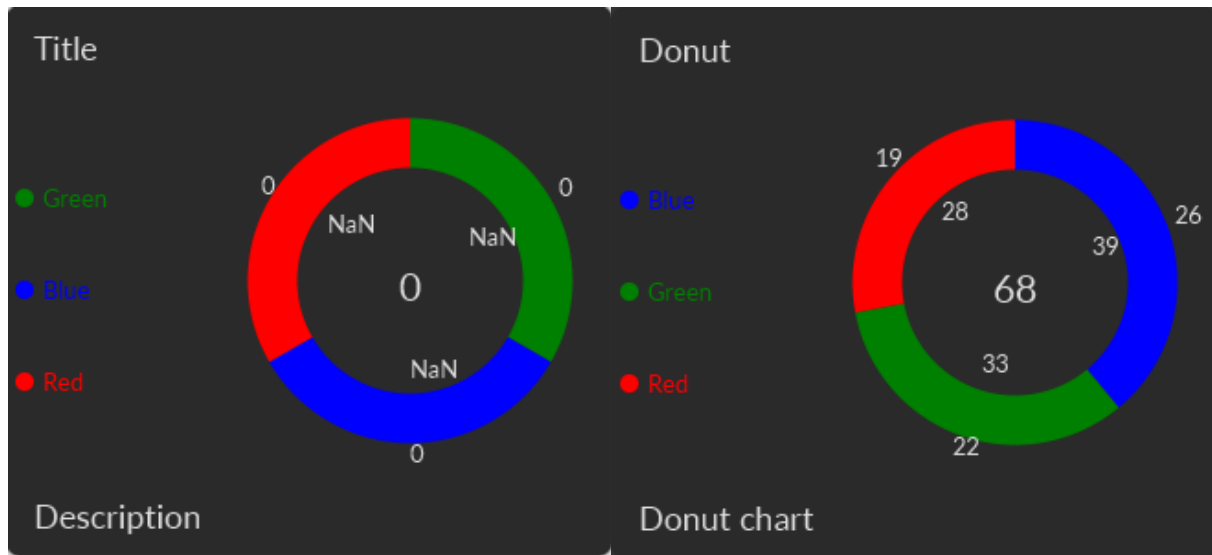
Property	ST script field	Description
Use legends	uselegends	Check it if you want to add legends to the bar chart.
Decimal position	decimalpos	Decimal position of tag's values entered in the table.
Font type	fonttype	Type of the text's font.
Description	description	Set tile's description if necessary
Title	title	Set tile's title
Minimum	text	Set tile's text
Maximum	description	Set tile's description if necessary
Sectors	---	<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this bar. ▪ Name - name of the bar chart sector. ▪ Color - bar's color.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.


Properties from the **"Flash"** tab are described [here](#)³⁴⁵.
Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.
Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.


6.2.3.22.16 Donut chart





pic. 1 - object image


pic. 2 - object image in a project


 Object properties


 General


 Back. color

 Text color

 Flash

 Rotation

 Motion

 Visibility

Name:

Donut chart1

Background color:

#2a2a2a

Text color:

#dfdfdf

Use legends:

☒

Decimal position:

0

Font type:

Lato Regular

Description:

Description

Title:

Title

Minimum:

0.0

Maximum:

100.0

Sectors:

Collection

Dimensions:

W=330H=300

Coordinates:

X=111Y=140

Angle:

0

Scripts:

Collection

User-defined

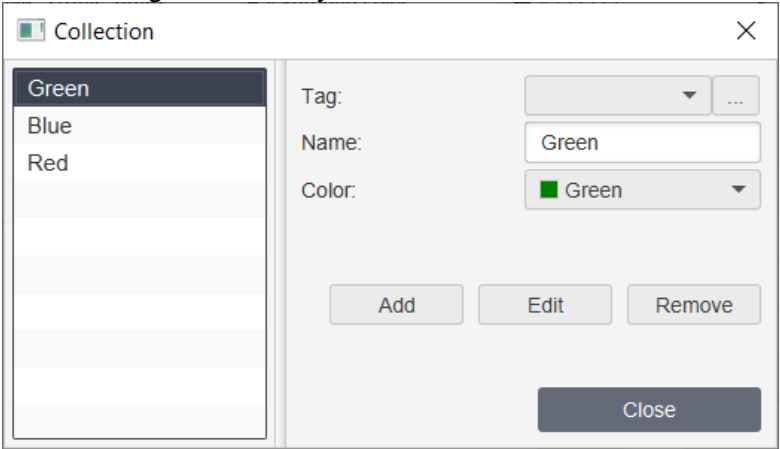
Collection

OK

Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Use legend	uselegends	Check it if you want to add legends to the bar chart.

Property	ST script field	Description
s		
Decimal position	decimalpos	Decimal position of tag's values entered in the table.
Font type	fonttype	Type of the text's font.
Description	description	Set tile's description if necessary
Title	title	Set tile's title
Minimum	text	Set tile's text
Maximum	description	Set tile's description if necessary
Sectors	---	<p>After clicking Collection you'll see window:</p>  <p>where:</p> <ul style="list-style-type: none"> ▪ Tag - tag that you want to bind to this bar. ▪ Name - name of the bar chart sector. ▪ Color - bar's color.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

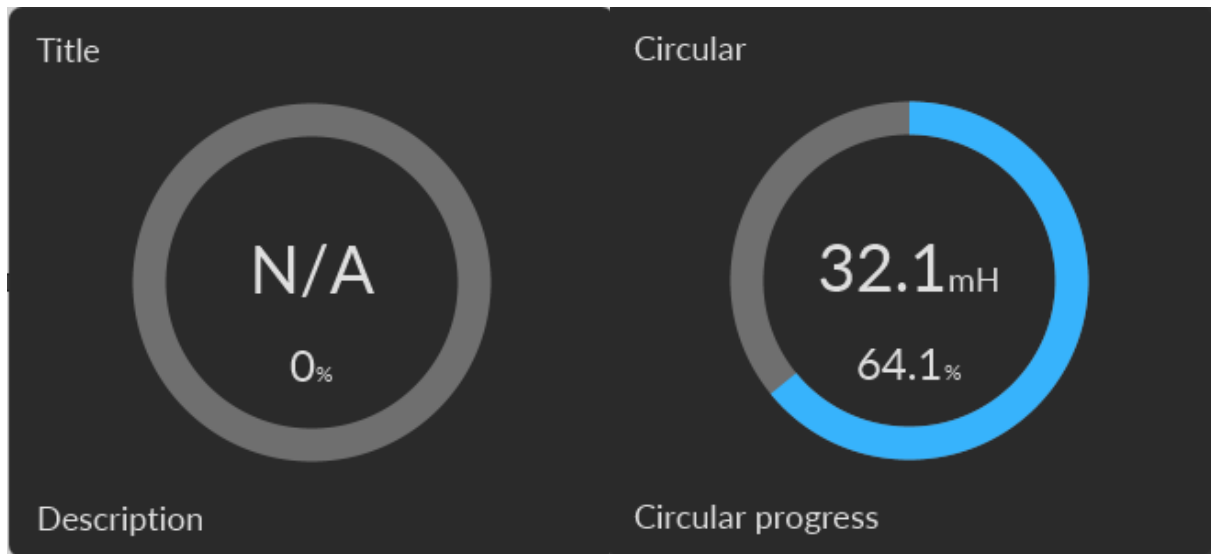
Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.17 Circular progress



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Circular progress1

Background color: #2a2a2a

Text color: #dfdfff

Fill color: #37b3fc

Font type: Lato Regular

Title: Title

Description: Description

Unit:

Dimensions: W= 330 H= 300

Coordinates: X= 133 Y= 131

Angle: 0

Scripts: Collection

User-defined: Collection

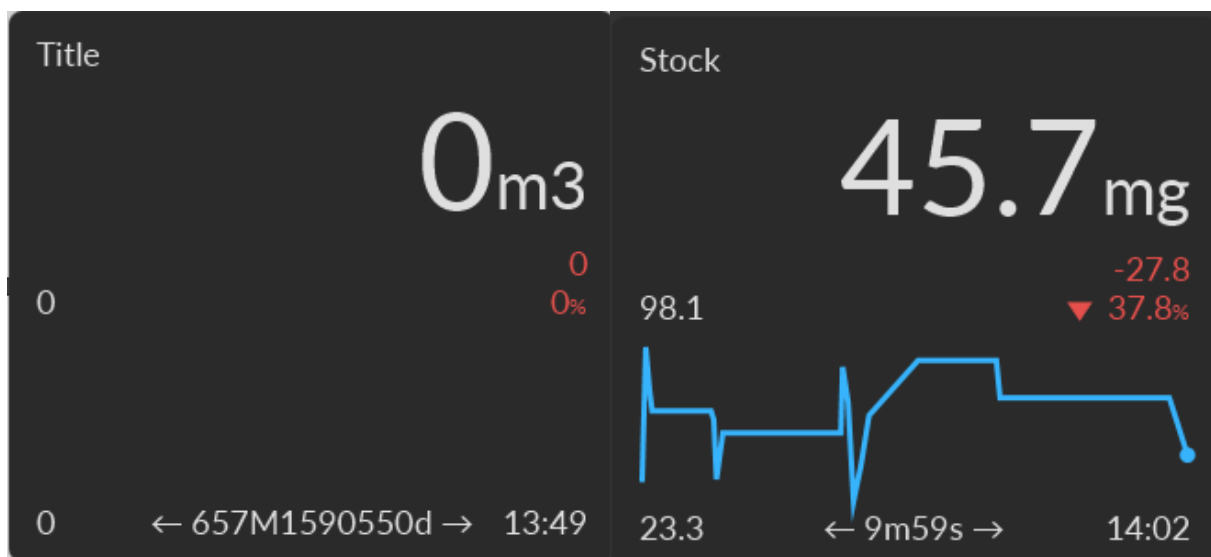
OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the arc of the object that shows tag value
Font type	fonttype	Type of the text's font.
Description	description	Set tile's description if necessary
Title	title	Set tile's title
Unit	unit	Specify the unit of measure for the tag value

Properties from the **"Value"** tab are described [here](#)^[369].
Properties from the **"Back. color"** tab are described [here](#)^[366].
Properties from the **"Fill Color"** tab are described [here](#)^[352].
Properties from the **"Text Color"** tab are described [here](#)^[355].
Properties from the **"Flash"** tab are described [here](#)^[345].
Properties from the **"Rotation"** tab are described [here](#)^[347].
Properties from the **"Motion"** tab are described [here](#)^[348].
Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.22.18 Stock



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Stock1

Background color: #2a2a2a

Text color: #dfdfdf

Line color: #37b3fc

Font type: Lato Regular

Title: Title

Default period(min): 10

Unit: m3

Dimensions: W= 330 H= 300

Coordinates: X= 107 Y= 104

Angle: 0

Scripts: Collection

User-defined: Collection

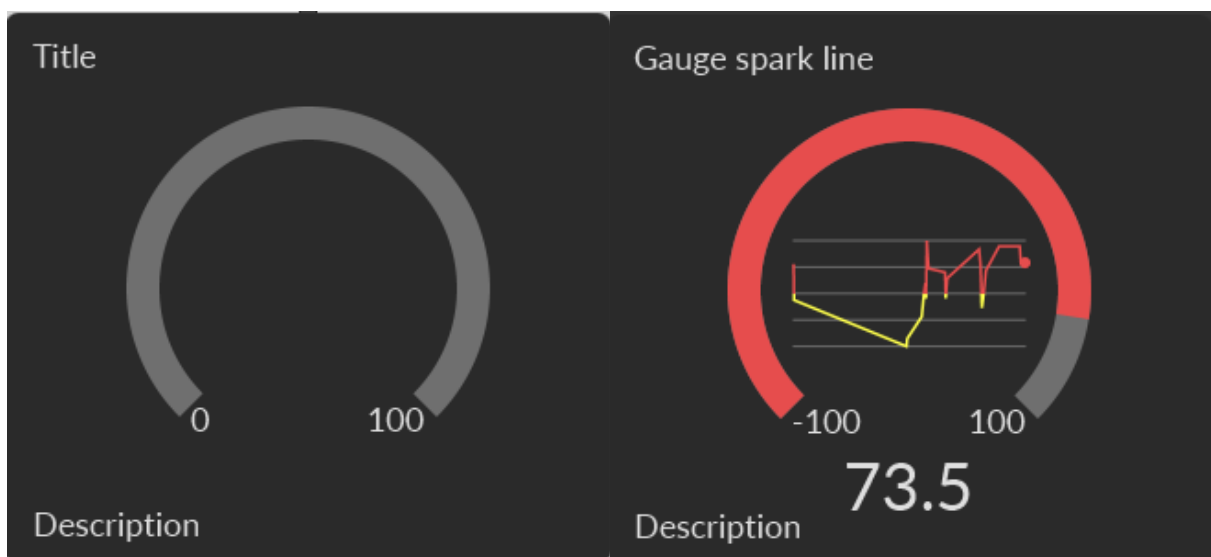
OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Line color	linecolor	Specify the color of the line
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Default period (min)	defaultperiod	Default time period of the trend (end time - begin time).
Unit	unit	Specify the unit of measure for the tag value

Properties from the **"Value"** tab are described [here](#)^[369].
Properties from the **"Back. color"** tab are described [here](#)^[366].
Properties from the **"Line Color"** tab are described [here](#)^[350].
Properties from the **"Text Color"** tab are described [here](#)^[355].
Properties from the **"Flash"** tab are described [here](#)^[345].
Properties from the **"Rotation"** tab are described [here](#)^[347].
Properties from the **"Motion"** tab are described [here](#)^[348].
Properties from the **"Visibility"** tab are described [here](#)^[349].

6.2.3.22.19 Gauge spark line



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Gauge spark line1

Background color: #2a2a2a

Text color: #dfdfdf

Font type: Lato Regular

Title: Title

Description: Description

Dimensions: W= 330 H= 300

Coordinates: X= 123 Y= 124

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary

Properties from the "**Value**" tab are described [here](#)³⁷⁰.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the "**Text Color**" tab are described [here](#)³⁵⁵.

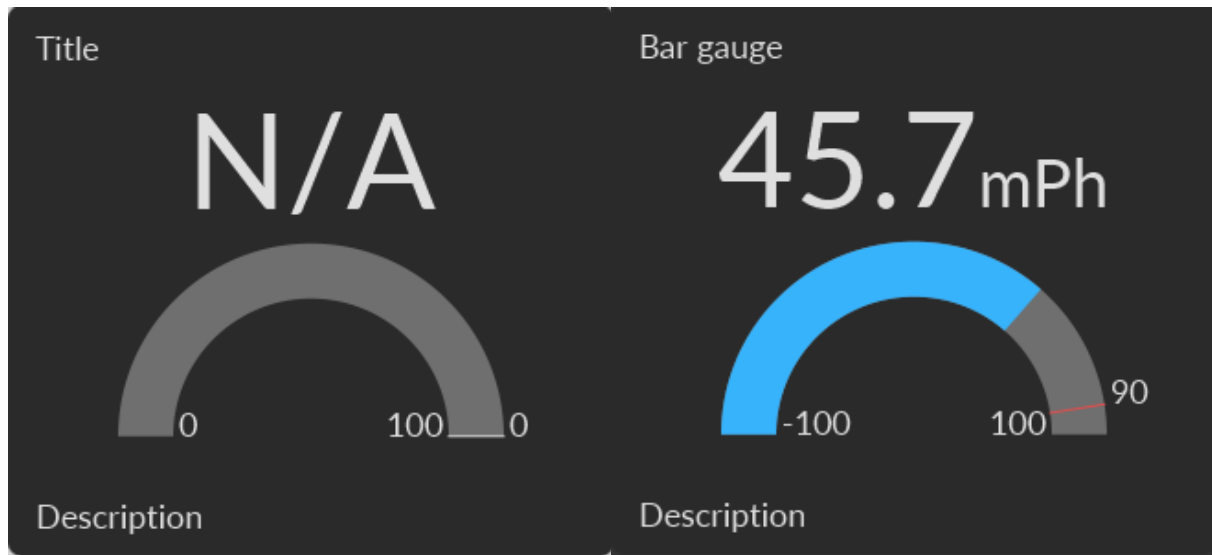
Properties from the "**Flash**" tab are described [here](#)³⁴⁵.

Properties from the "**Rotation**" tab are described [here](#)³⁴⁷.

Properties from the "**Motion**" tab are described [here](#)³⁴⁸.

Properties from the "**Visibility**" tab are described [here](#)³⁴⁹.

6.2.3.22.20 Bar gauge

*pic. 1 - object image**pic. 2 - object image in a project*

Object properties

General

Name: Bar gauge1

Needle color: Background color: #2a2a2a

Value: Text color: #dfdfdf

Back. color: Fill color: #37b3fc

Fill color: Needle(Fill) color: #dfdfdf

Text color: Font type: Lato Regular

Flash: Title: Title

Rotation: Description: Description

Motion: Unit:

Visibility: Threshold: 0.0

Dimensions: W= 330 H= 300

Coordinates: X= 145 Y= 143

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the arc of the gauge
Needle(fill) color	needlecolor	Specify needle color
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title

Property	ST script field	Description
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value
Threshold	threshold	Specify the tag value up to which the arc color will be highlighted in a different color.

Properties from the **"Needle color"** tab are described [here](#) ³⁶⁶.

Properties from the **"Value"** tab are described [here](#) ³⁶⁹.

Properties from the **"Back. color"** tab are described [here](#) ³⁶⁶.

Properties from the **"Fill Color"** tab are described [here](#) ³⁵².

Properties from the **"Text Color"** tab are described [here](#) ³⁵⁵.

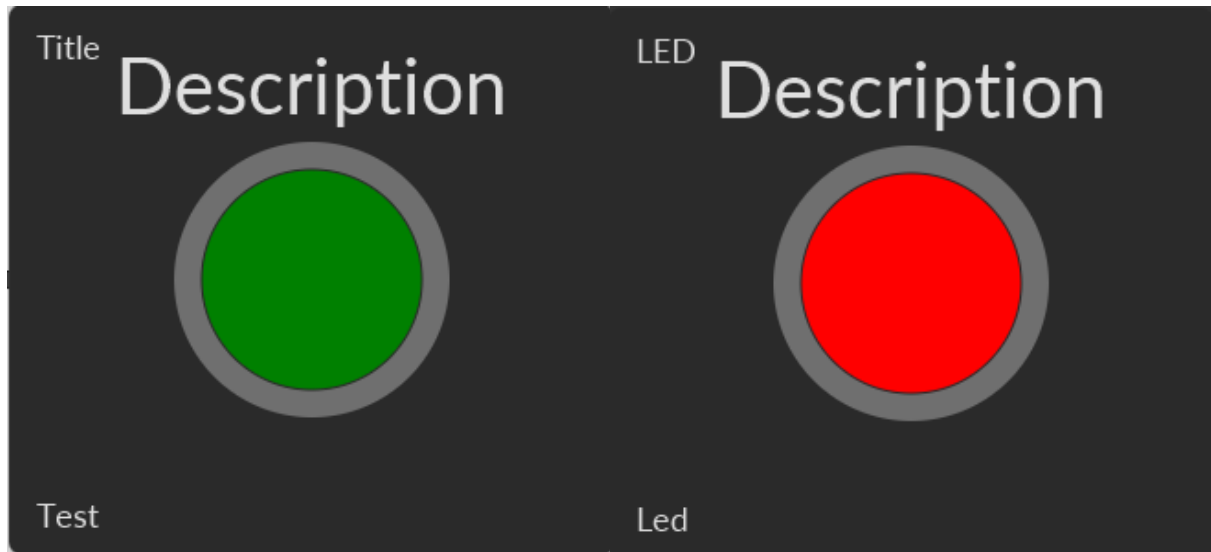
Properties from the **"Flash"** tab are described [here](#) ³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#) ³⁴⁷.

Properties from the **"Motion"** tab are described [here](#) ³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#) ³⁴⁹.

6.2.3.22.21 Led



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name:

Back. color:

Text color:

Fill color:

Font type:

Title:

Description:

Text:

Dimensions: W= H=

Coordinates: X= Y=

Angle:

Scripts:

User-defined:

OK Cancel

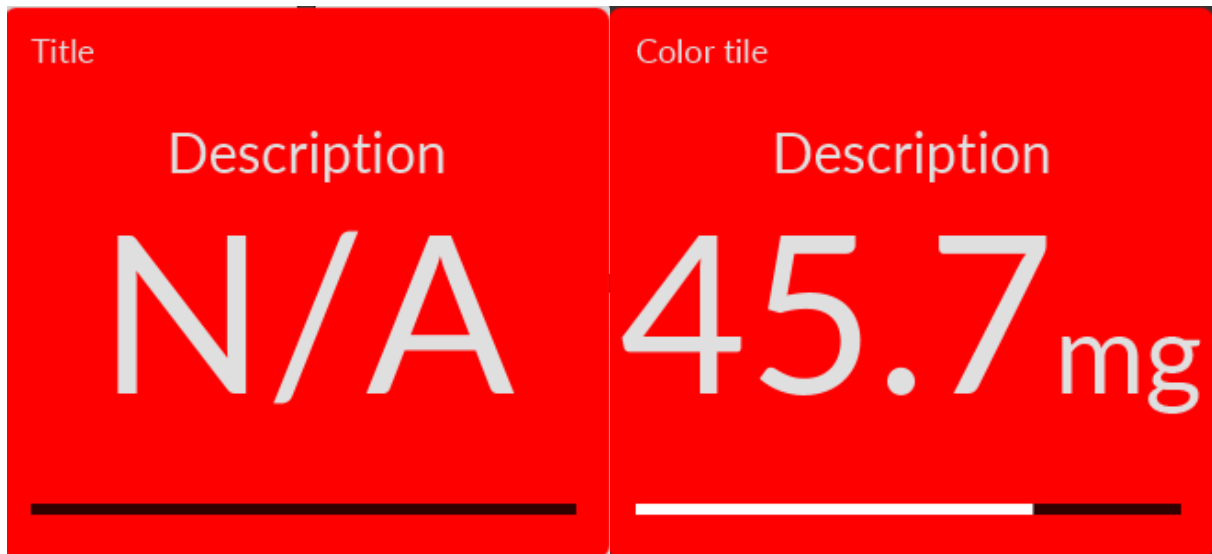
Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the arc of the gauge
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Text	text	Text displayed on the object.

Properties from the "**Back. color**" tab are described [here](#)³⁶⁶.

Properties from the **"Fill Color"** tab are described [here](#)³⁵²
Properties from the **"Text Color"** tab are described [here](#)³⁵⁵
Properties from the **"Flash"** tab are described [here](#)³⁴⁵
Properties from the **"Rotation"** tab are described [here](#)³⁴⁷
Properties from the **"Motion"** tab are described [here](#)³⁴⁸
Properties from the **"Visibility"** tab are described [here](#)³⁴⁹

6.2.3.22.22 Color



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Color1

Background color: Red

Text color: #dfdfdf

Fill color: White

Font type: Lato Regular

Title: Title

Description: Description

Unit:

Dimensions: W= 330 H= 300

Coordinates: X= 170 Y= 94

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)^[143]).

Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the line that shows tag value
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Value**" tab are described [here](#)^[369].

Properties from the "**Back. color**" tab are described [here](#)^[366].

Properties from the "**Fill Color**" tab are described [here](#)^[352].

Properties from the **"Text Color"** tab are described [here](#)³⁵⁵.

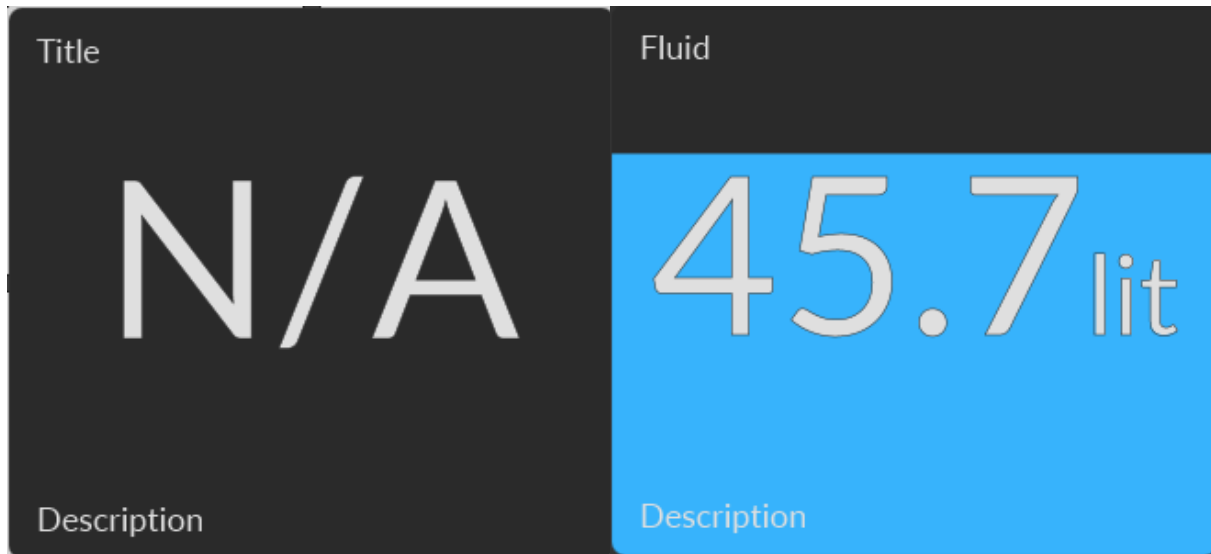
Properties from the **"Flash"** tab are described [here](#)³⁴⁵.

Properties from the **"Rotation"** tab are described [here](#)³⁴⁷.

Properties from the **"Motion"** tab are described [here](#)³⁴⁸.

Properties from the **"Visibility"** tab are described [here](#)³⁴⁹.

6.2.3.22.23 Fluid



pic. 1 - object image

pic. 2 - object image in a project

Object properties

General

Name: Fluid1

Background color: #2a2a2a

Text color: #dfdfdf

Fill color: #37b3fc

Font type: Lato Regular

Title: Title

Description: Description

Unit:

Dimensions: W= 330 H= 300

Coordinates: X= 147 Y= 164

Angle: 0

Scripts: Collection

User-defined: Collection

OK Cancel

Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

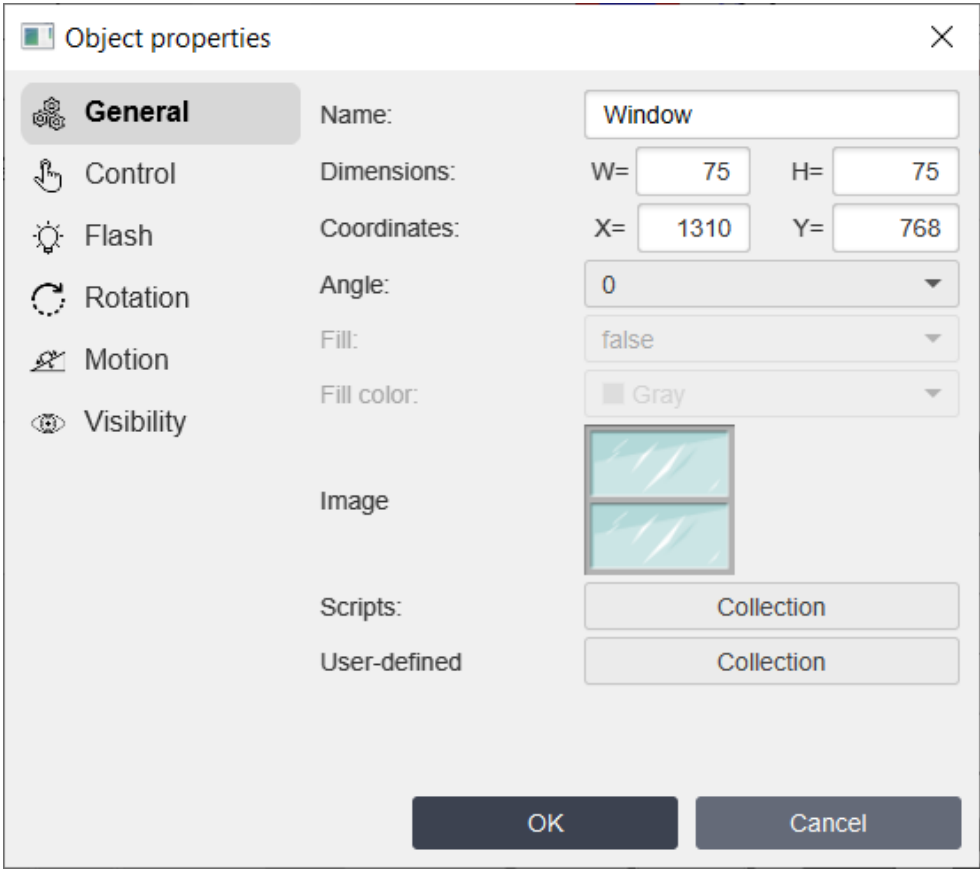
Property	ST script field	Description
Background color	bgcolor	Color of the background of the tile
Text color	textcolor	Color of the text.
Fill color	fillcolor	Specify the color of the line that shows tag value
Font type	fonttype	Type of the text's font.
Title	title	Set tile's title
Description	description	Set tile's description if necessary
Unit	unit	Specify the unit of measure for the tag value

Properties from the "**Value**" tab are described [here](#)³⁶⁹.

- Properties from the "Back. color" tab are described [here](#)³⁶⁶.
- Properties from the "Fill Color" tab are described [here](#)³⁵².
- Properties from the "Text Color" tab are described [here](#)³⁵⁵.
- Properties from the "Flash" tab are described [here](#)³⁴⁵.
- Properties from the "Rotation" tab are described [here](#)³⁴⁷.
- Properties from the "Motion" tab are described [here](#)³⁴⁸.
- Properties from the "Visibility" tab are described [here](#)³⁴⁹.

6.2.3.23 SVG objects library

All SVG library objects have similar properties except for the ability to change the fill color. Below is a description of these properties:



Let's look at the "General" properties of this object (properties not listed in the table are common to all objects, you can read about them [here](#)¹⁴³).

Property	ST script field	Description
Fill	usefillcolor	Select fill or not fill SVG.

Property	ST script field	Description
Fill color	fillcolor	Fill color of the SVG object.
Image		This is only for demonstration purposes.

Properties from the **"Flash"** tab are described [here](#)^[345].

Properties from the **"Rotation"** tab are described [here](#)^[347].

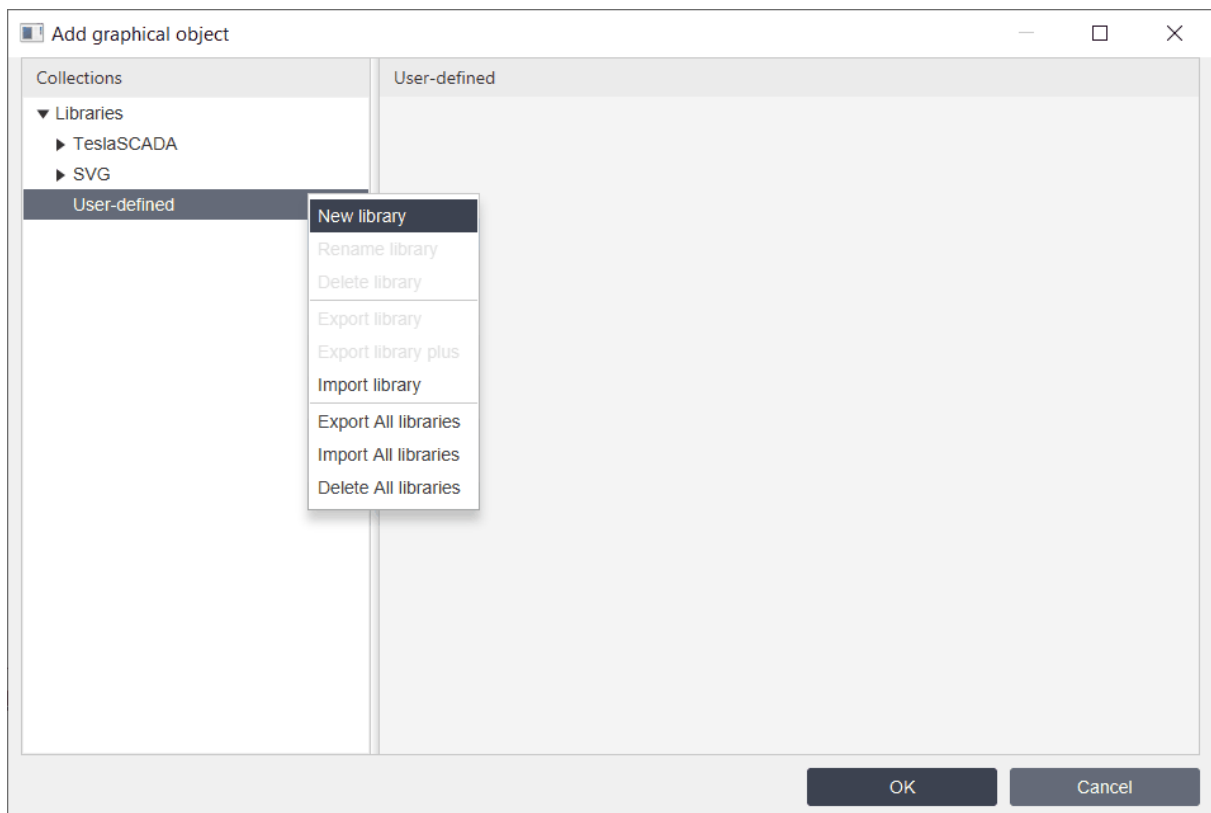
Properties from the **"Motion"** tab are described [here](#)^[348].

Properties from the **"Visibility"** tab are described [here](#)^[349].

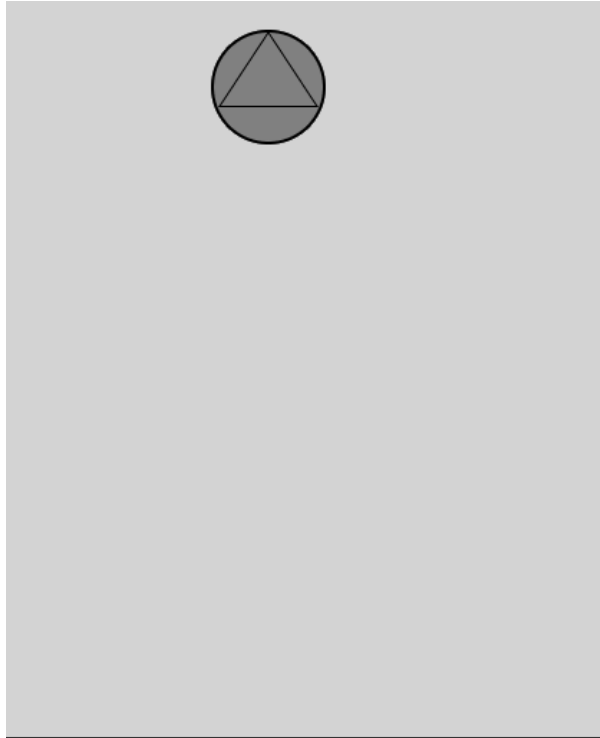
6.2.4 User-defined library

Create user-defined library

You can create your own library by clicking right button on User-defined section of the [Add graphical object](#)^[136] window and choosing New library menu item:



You can add graphical object in your library by clicking right button on the object on [Canvas](#)^[91] or [Screen window](#)^[92] and choosing **Add to Library->You library** menu item:



You can Select, Rename or Delete created object in your library by clicking right button on it and selecting correspondent menu item.

Rename user-de? ned library

To rename library right click on the library you want to rename and choose Rename library menu item.

Delete user-de? ned library

To delete library right click on the library you want to delete and choose Delete library menu item.

Export user-de? ned library

To export library:

1. Right click on the library you want to export and choose **Export library** menu item.
2. Now select the location and click the button Save (TeslaSCADA library extension .tsp2lib).

Library with all objects will be exported in the the file. You can use this file to import library with all objects in a new project.

If you want to export your library with all objects and scripts that used these objects and also with screens that called from this object (for example button that call screen or popup screen) or with screens that called from the scripts bind to the object, you need to use **Export library plus** menu item. In this case in exported file you'll have object, scripts and screens.

Import user-defined library

To import library:

1. Right click on the User-defined section and choose Import library menu item.
2. Now select the library file and click Open (TeslaSCADA library extension .tsp2lib).

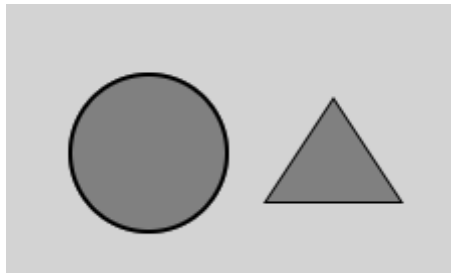
If exported file contains only objects (you use **Export library** menu item during exporting library) only library with objects are created in the Add graphical object. If exported file also contains script and/or screens (you use **Export library plus** menu item during exporting library) these scripts and screens are added in the project and you can see them in the [Project window](#)^[72].

Below you can find example how to create complex object with script and screen are bound to it. And how to add this object to the user-defined library, how to export this library and how to import it in the new project.

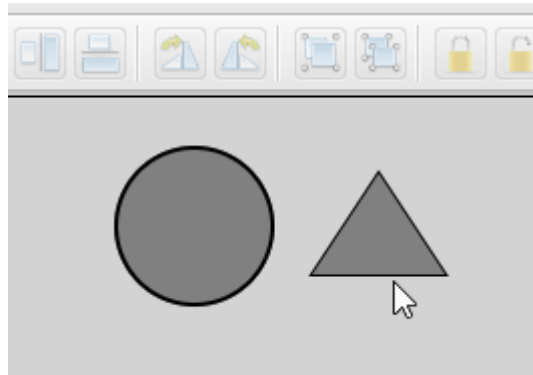
6.2.4.1 Example: How to create new graphical object

In this example, we will create a group graphic object ([Ellipse](#)^[152] + [Polygon](#)^[156]), configure it so that when the tag value changes, the fill color of the ellipse changes, and also when you click on the ellipse, a pop-up window appears with information about the state and description of the object. And then we will add the configured object to the User-defined library. We export the object along with its settings in order to use it in other projects.

Let's create a pump object consisting of two simple objects: an [Ellipse](#)^[152] and a [Polygon](#)^[156].



Let's group these objects:



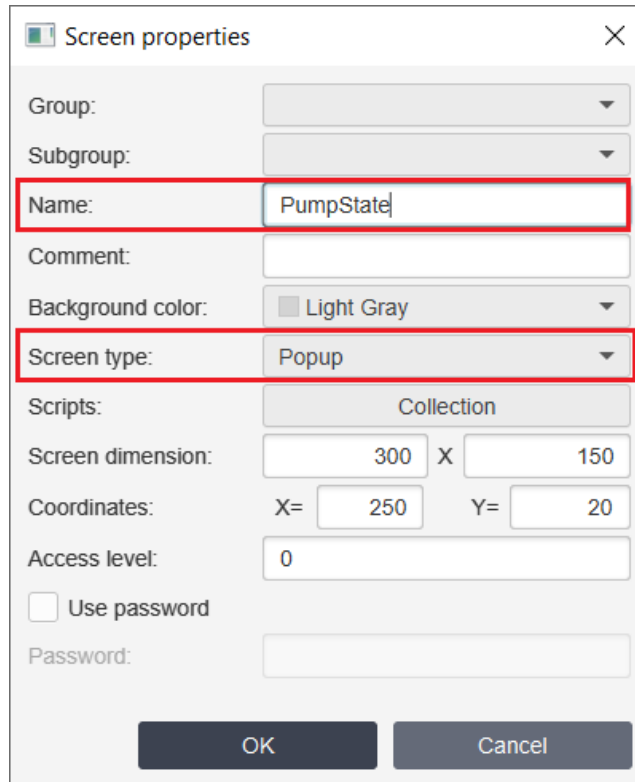
Let's add 3 tags to our project:

Tags		X
Name	Value	+
Pump1State	false	
Pump2State	false	
Pump3State	false	

We want the color of the ellipse to be bound to a tag with the following name: *Pump{number}State*, where {number} is the number of the graphic object instance.

We want a popup window to appear with status information and a description when the user clicks on an object.

So let's create a pop-up window (screen):



Screen properties

Group:

Subgroup:

Name:

Comment:

Background color:

Screen type:

Scripts:

Screen dimension: X

Coordinates: X= Y=

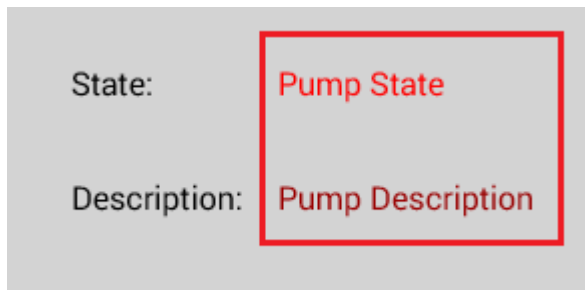
Access level:

☐ Use password

Password:

OK Cancel

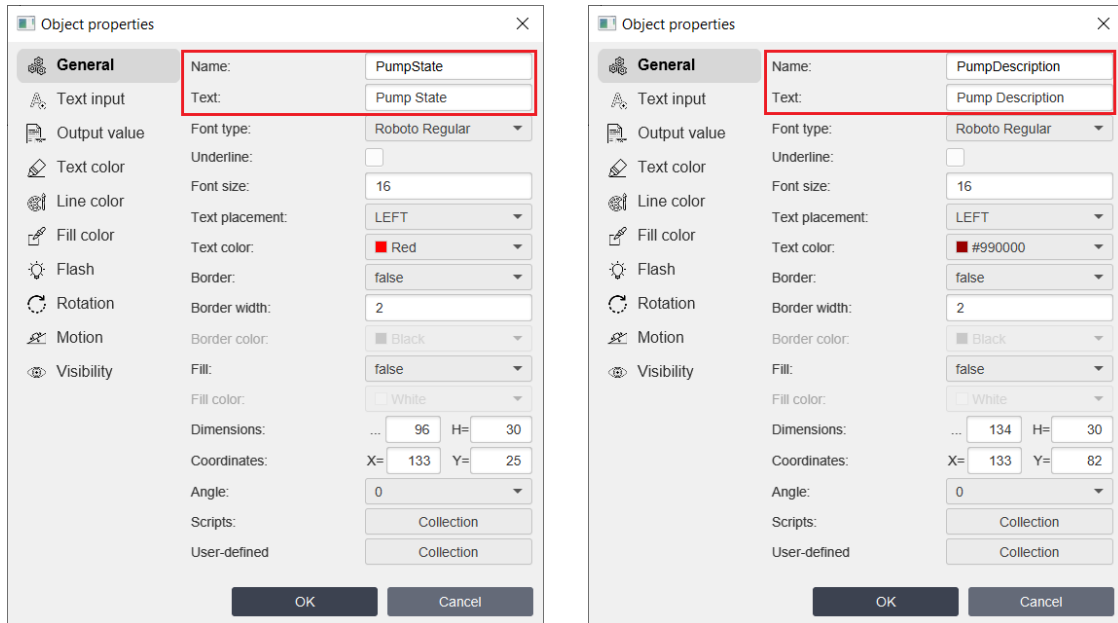
Let's add graphic objects to the screen:



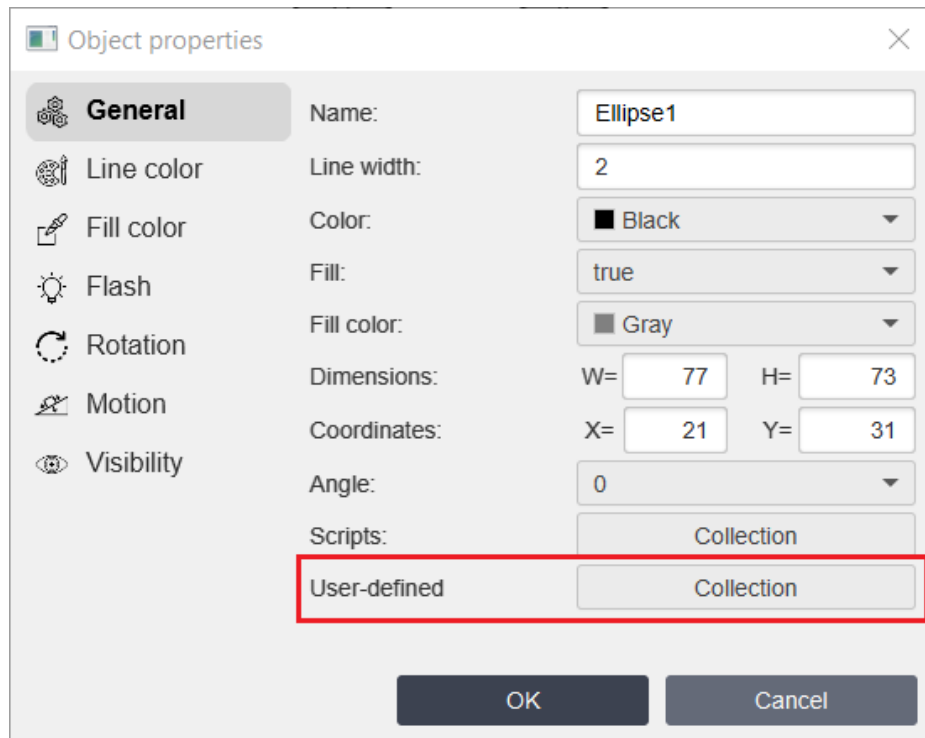
State: **Pump State**

Description: **Pump Description**

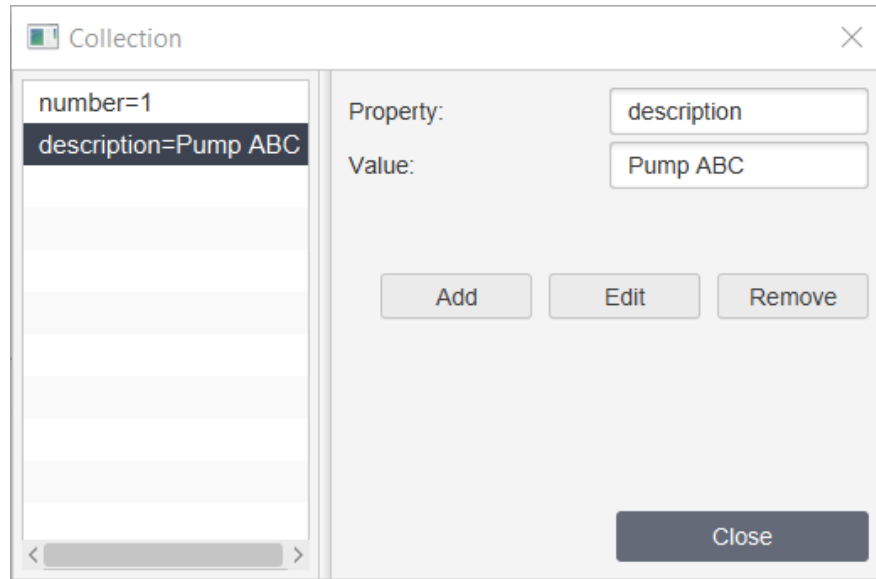
where objects named PumpState and PumpDescription will display information about the state of the pump and its description:



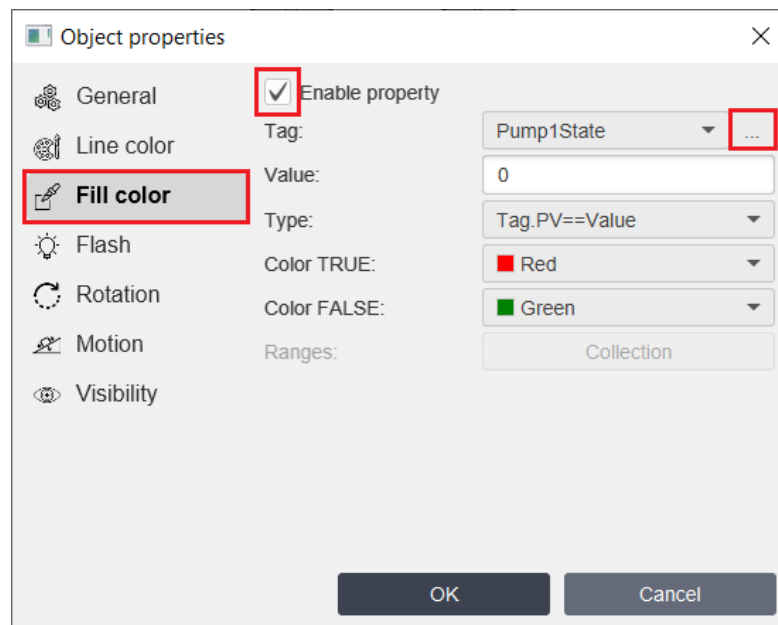
For the ellipse object, we will create user-defined properties - number and description. To do this, double-click on the Ellipse object. (or select the Object Properties menu item from the context menu) in the [Screen window](#)⁹². The Object Properties window appears:



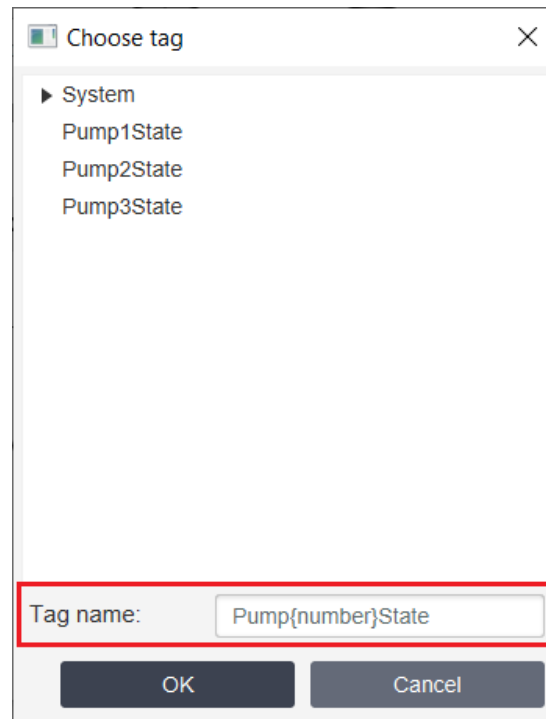
Click "**Collection**" properties "User-defined" and add our properties:



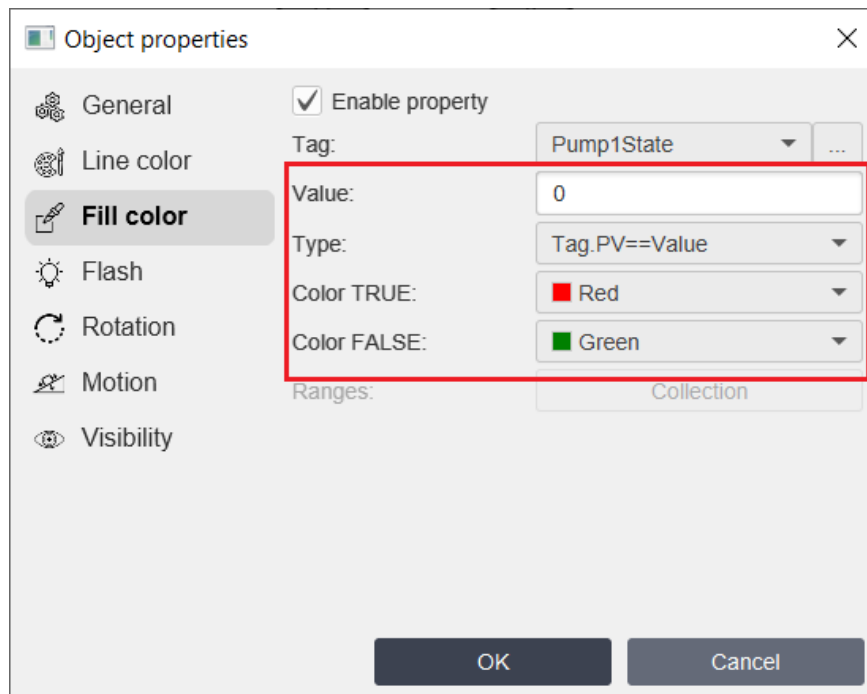
Then close this window and the Object Properties window by clicking "OK" to confirm these changes. Then open the Ellipse properties window and open the Fill Color tab:



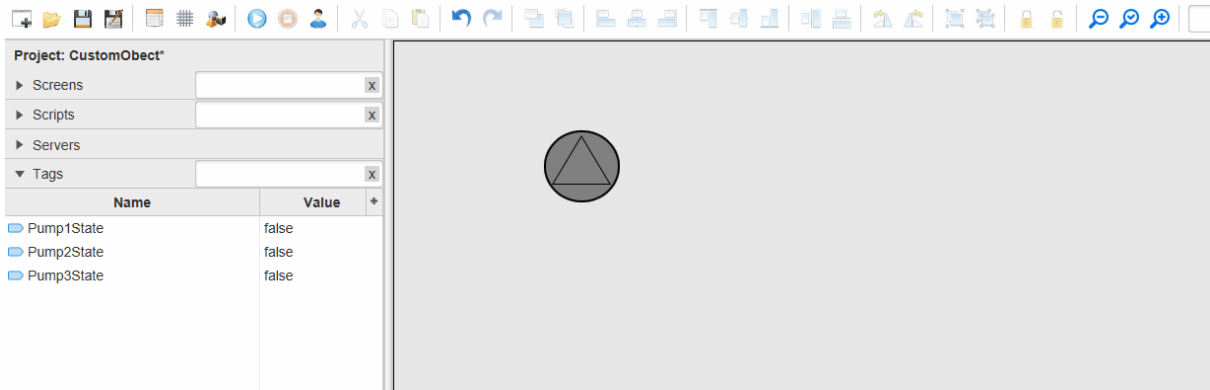
In order to bind a tag to this property, click "..." and write down Pump{number}State, where {number} is our custom property:



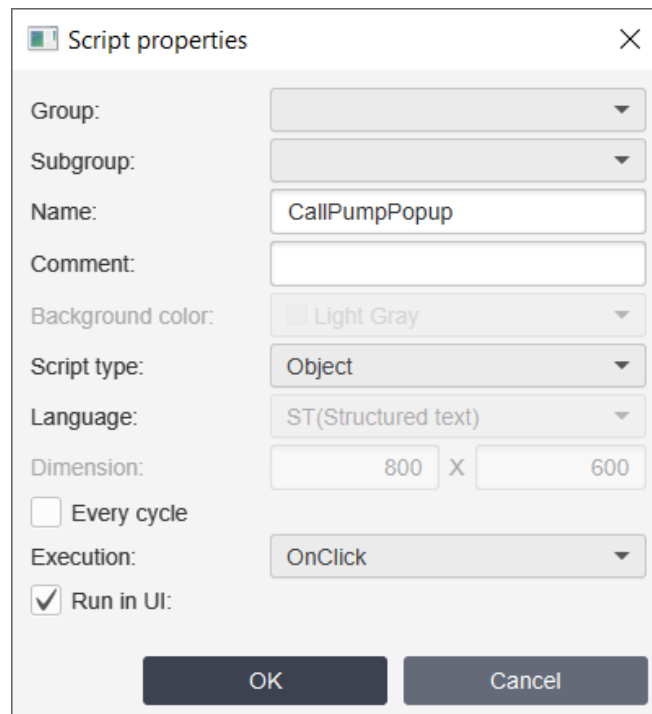
Click "OK" and close the tag selection window. Because the custom property "number" is set to 1, the Fill Color property will be bound to the Pump1State tag. Leave the remaining parameters as default (Tag value = 0 (false) : red fill color; Tag value != 0 (true) : green fill color)



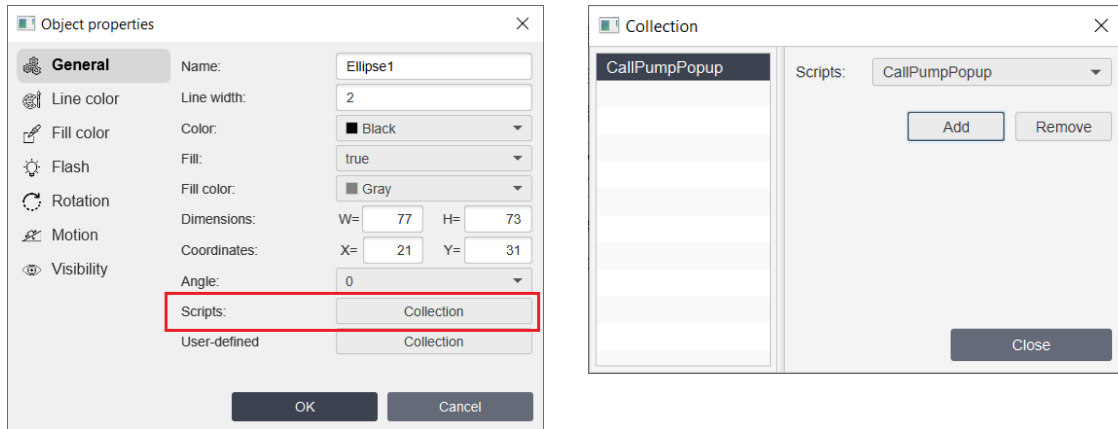
Click "OK" to close the object properties window. Let's check the functionality. Run the simulation and change the value of the Pump1State tag from false to true:



In order to trigger the popup window, let's create a script:




Let's add this script to the Ellipse object scripts:

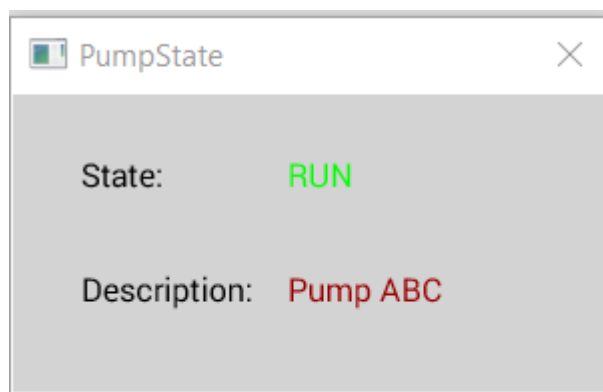


The script text looks like this:

```
1 string statetagname = "Pump" + Objects.this.number + "State";
2 bool state = gettagvalue(statetagname, "false");
3 if (state==1) {
4     Objects.PumpState.text = "RUN";
5     Objects.PumpState.textcolor = Color.GREEN;
6 }else{
7     Objects.PumpState.text = "STOP";
8     Objects.PumpState.textcolor = Color.RED;
9 }
10 Objects.PumpDescription.text = Objects.this.description;
11 callpopup("PumpState");
```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

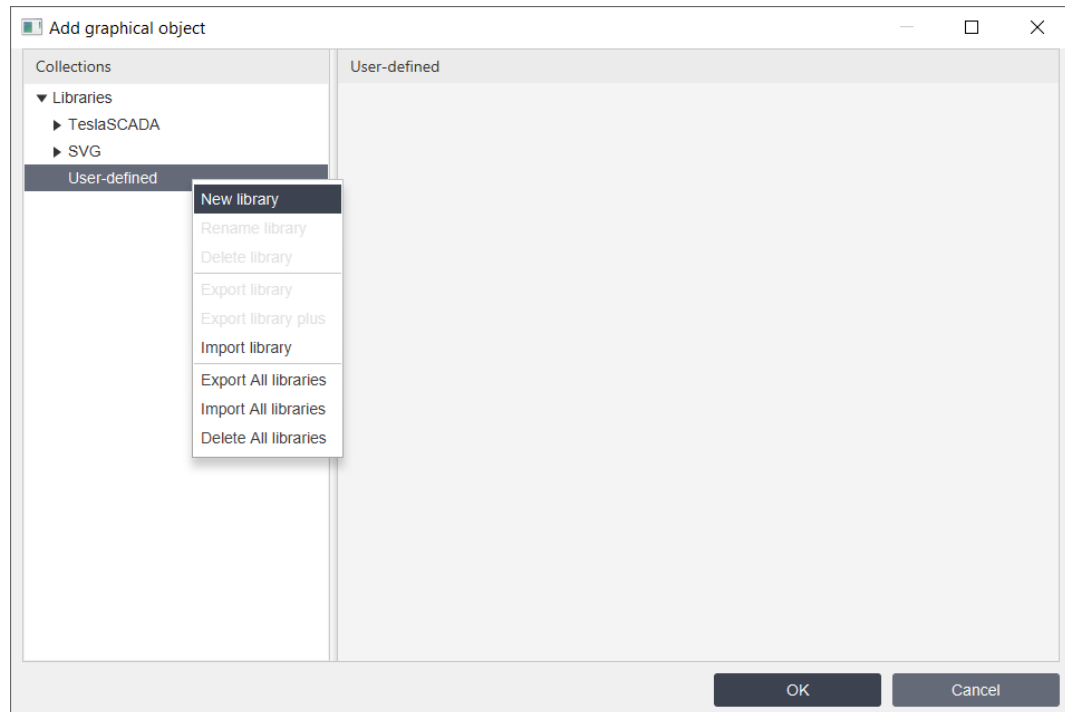
Now when you click on the ellipse, you will see a pop-up window (depending on the tag value).



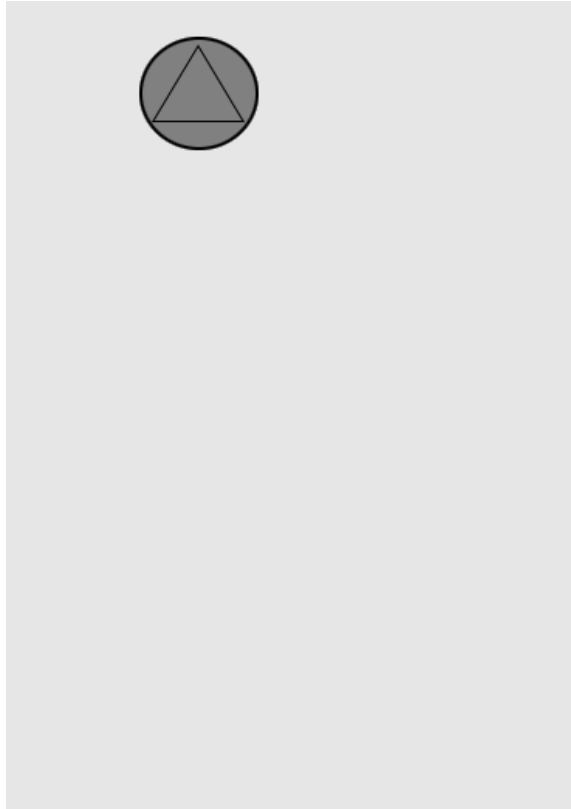
You can create another pump instance and change the number user-defined property to 2 or 3 to bind the pumps to the Pump2State and Pump3State tags. You can also change the description property for each pump ellipse:

Attention! It might be better to use Polygon to store user-defined properties and mouse click action because it is above Ellipse. Or, alternatively, you can use a transparent Button over the entire group object and use its OnClick action.

Now we can add this graphic object to our library. First you need to create a library: open the "Add Graphic Object" window, select "User-defined" and right-click on "New Library":



Give the library a name, for example "Pumps". Let's add an object to the library:



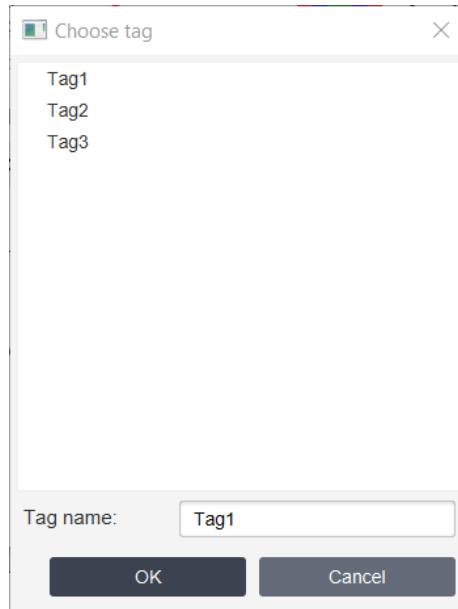
Now let's export the library: open the Add graphic object window and find our library, right-click on it and select the "**Export library plus**" menu item. A file dialog box will appear, enter the name of the library and click "Save". This library will be saved with the object as well as the saved script that we created in this project and the popup. Now, if you want to use this object with this popup and script, just import it into another project.

Important! In the new project you need to create the same tag names.

You can download the example project [here](#).

6.2.5 Properties

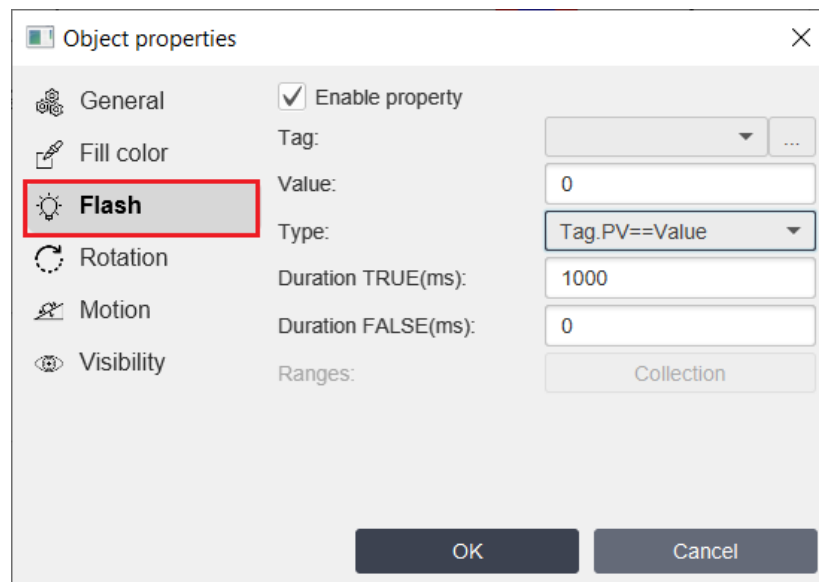
Every graphical object has several group of properties. To use property of the graphical object check **Enable Property**. You can select [tag](#)^[81] to bind to the property by using ComboBox (you can choose it by beginning entering name of the tag when ComboBox focused) or use Button (...). Every object has **Flash, Rotation, Motion and Visibility** properties. Other properties depend on the object.

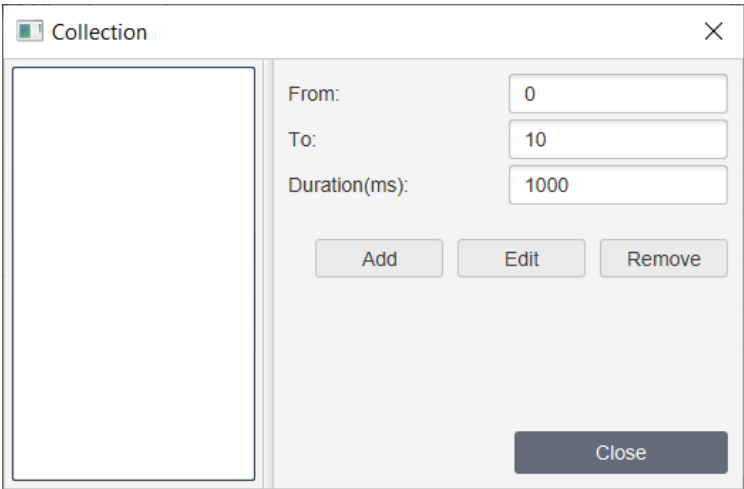


By clicking (...) when you bind tag to the property you'll get to the "Choose tag" dialog where you can choose tag from the hierarchy tree or enter its name in the ?eld "Tag name". In the ?eld you can use indirect names enclosed in curly braces {group}, {name} and {user-defined property} of the object. It's useful if you use many the same type objects and want to bind to the group of the same type tags.

6.2.5.1 Flash

The Flash property allows an object to ?ash when condition is TRUE or FALSE. To edit ?ash property click **Flash** tab on the object property window.



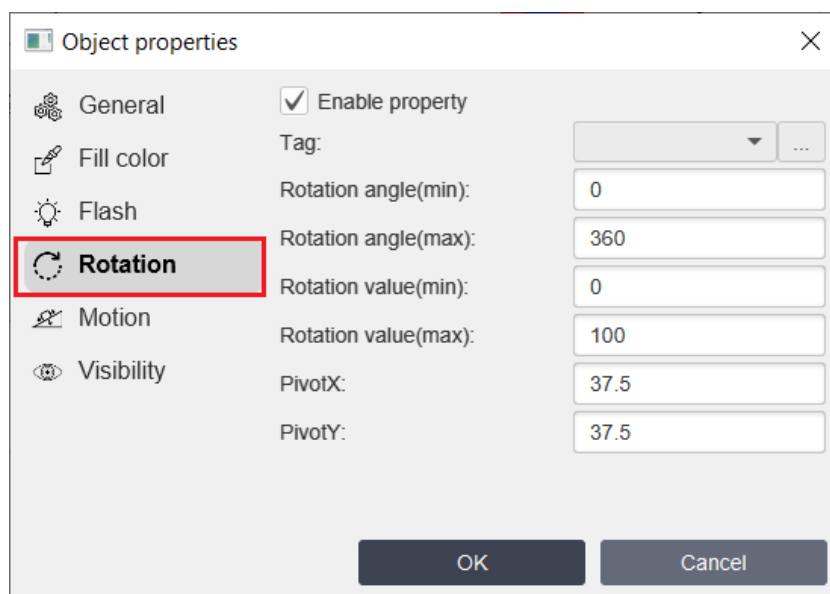
Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value. ▪ Tag.PV in the range - tag's value compares to the values in the ranges. To setup ranges click Collection button.
Duration TRUE(ms)*	Write period's time in milliseconds of objects ?asking if the comparison is true in the Duration TRUE(ms) ?eld. If you enter 0 the object will not ?asking.
Duration FALSE(ms)*	Write period's time in milliseconds of objects ?asking if the comparison is false in the Duration FALSE(ms) ?eld. If you enter 0 the object will not ?asking.
Ranges	<p>If you select Tag.PV in the range in the Type combobox and click Collection button. You'll see the window:</p> 

Property	Description
	<p>where:</p> <ul style="list-style-type: none"> • From - enter the value from which the object will flash with this periodicity in the ?eld. • To - enter the value to which the object will flash with this periodicity in the ?eld. • Duration(ms) - enter period of flashing in the ?eld. <p>You can Add, Edit or Remove collection element of flashing conditions.</p>

* This properties you can use in ST scripts by using **trueflashduration** and **falseflashduration** property keywords. For example: **Objects.Button.trueflashduration = 1000;**

6.2.5.2 Rotation

The Rotation property allows an object to rotate proportional to the value of the tag. To edit rotation property click Rotation tab on the object property window.

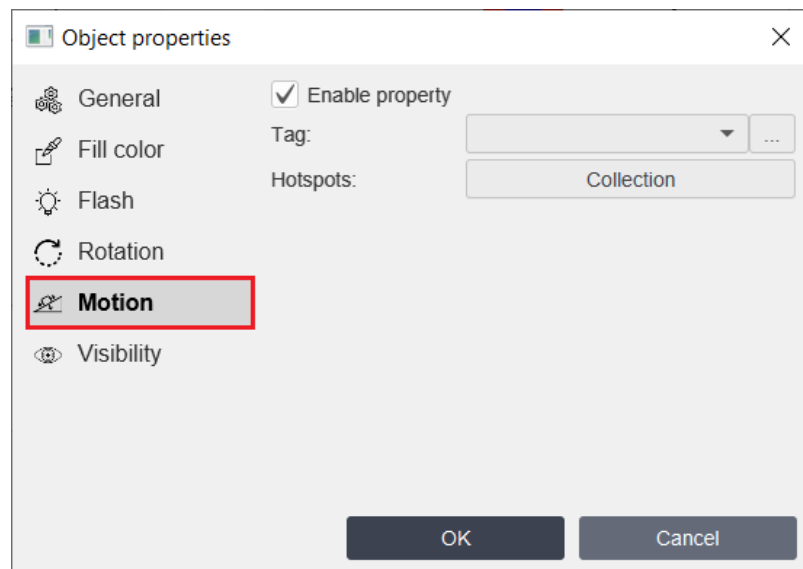


Property	Description
Tag	Select the tag which value will be compared.
Rotation angle(min)	Enter the minimum of rotation angle in the ?eld.
Rotation angle(max)	Enter the maximum of rotation angle in the ?eld.

Property	Description
Rotation value(min)	Write the minimum of the tag's value in the field.
Rotation value(max)	Write the maximum of the tag's value in the field.
PivotX	Enter X coordinate of the pivot in the field.
PivotY	Enter Y coordinate of the pivot in the field.

6.2.5.3 Motion

The Motion property allows an object to move depending on value of the tag. To configure the Motion property click **Motion** tab in the Object property window.

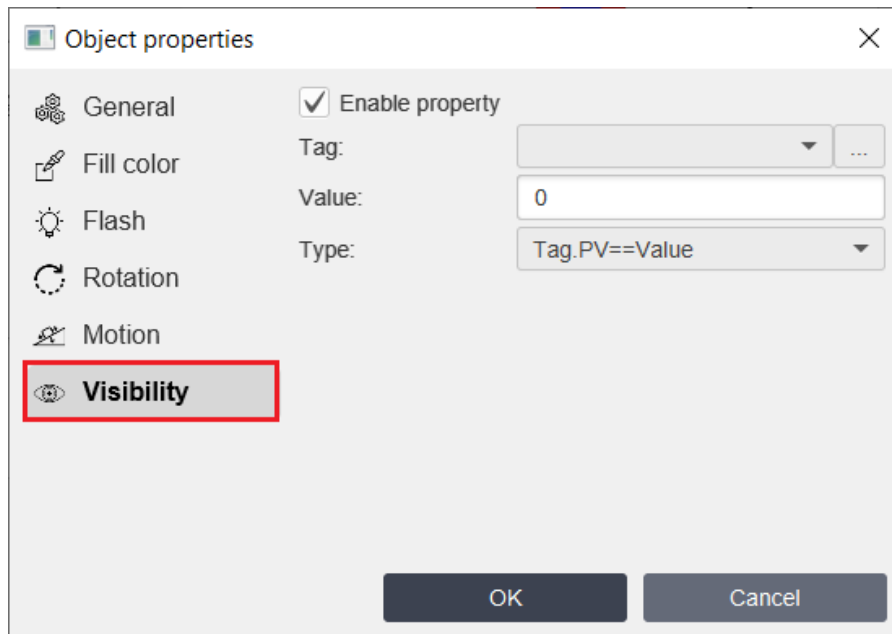


Property	Description
Tag	Select the tag depending on which value the object will change location coordinates.
Hotspots	Click Collection button to edit move conditions coordinates. After clicking you'll see the window:

Property	Description
	<div><div><div><div>Collection</div><div><div></div></div><div><div>From:0</div><div>To:10</div><div>TranslationX:0</div><div>TranslationY:0</div><div>AddEditRemove</div><div>Close</div></div></div></div><div><p>where:</p><ul style="list-style-type: none">▪ From - enter the value from which the object will change coordinates in the ?eld.▪ To - enter the value to which the object will change coordinates in the ?eld.▪ TranslationX - write X coordinate (X offset of the object position on the screen).▪ TranslationY - write Y coordinate (Y offset of the object position on the screen).</div></div>

6.2.5.4 Visibility

Visibility property allows an object to to make visible or not depending on the tag’s value. To configure the Visibility property click Visibility tab in the Object property window.



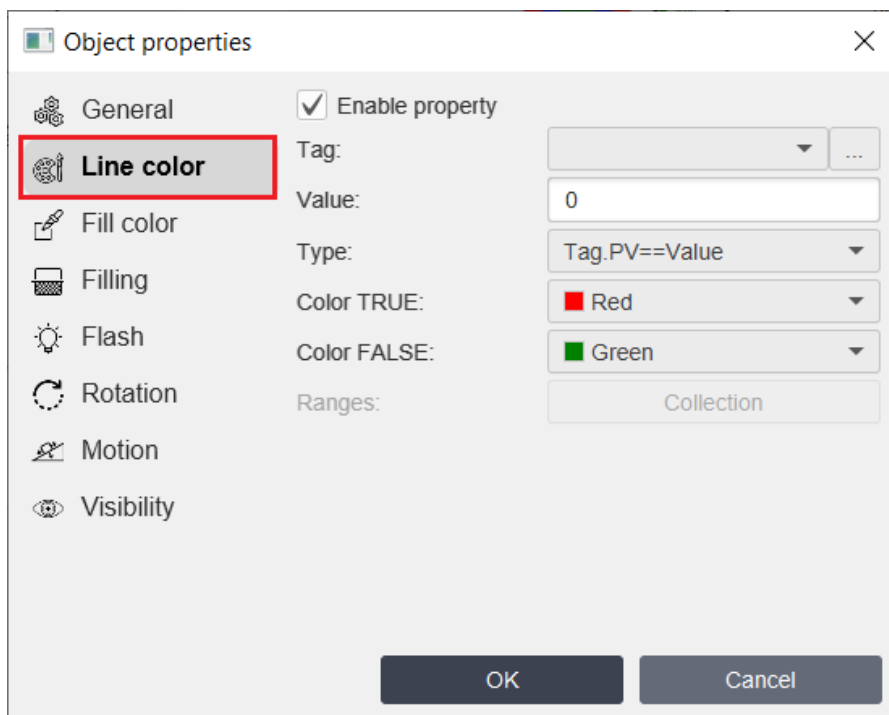
Property	Description
Tag	Select the tag which value will be compared.
Value*	Enter the comparison value.
Type	Select type of comparison: <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value.

* This property you can use in ST scripts by using **visibilityvalue** property keyword. For example: **Objects.Button.visibilityvalue = false;**

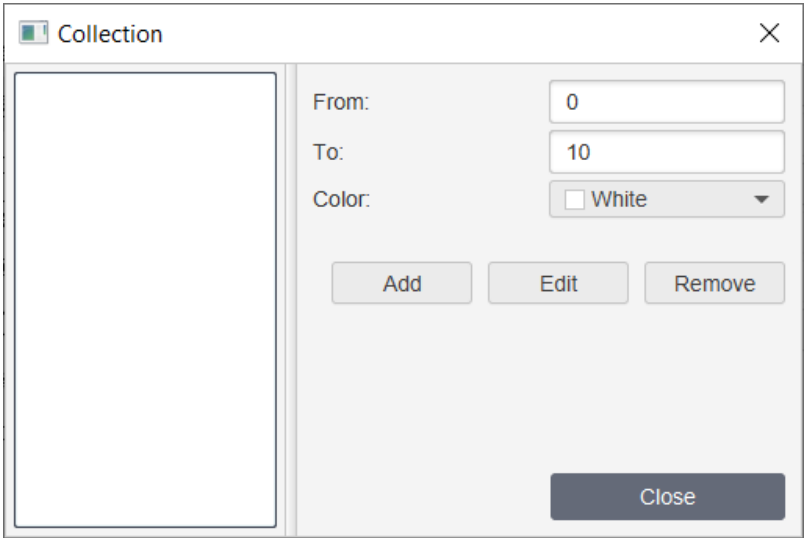
6.2.5.5 Line color

Not all objects have the Line color property!

The Line color property allows an object to change color of its line when condition is TRUE or FALSE. To configure Line color property click Line color tab in the Object property window.



Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value. ▪ Tag.PV in the range - tag's value compares to the values in the ranges. To setup ranges click Collection button.

Property	Description
Color TRUE	Choose a color that will result if the comparison is TRUE in this field.
Color FALSE	Choose a color that will result if the comparison is FALSE in this field.
Ranges	<p>If you select Tag.PV in the range in the Type combobox and click Collection button. You'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none"> • From - enter the value from which the object will change color in the field. • To - enter the value to which the object will change color in the field. • Color - choose color for this range. <p>You can Add, Edit or Remove collection element of line color conditions.</p>

6.2.5.6 Fill color

Not all objects have the Fill color property!

The Fill color property allows an object to change color of its filling when condition is TRUE or FALSE. To configure the Fill color property click **Fill color** tab in the Object property window.

Object properties

General ☒ Enable property

Line color Tag: [dropdown] ...

Fill color Value: 0

Filling Type: Tag.PV==Value

Flash Color TRUE: Red

Rotation Color FALSE: Green

Motion Ranges: Collection

Visibility

OK Cancel

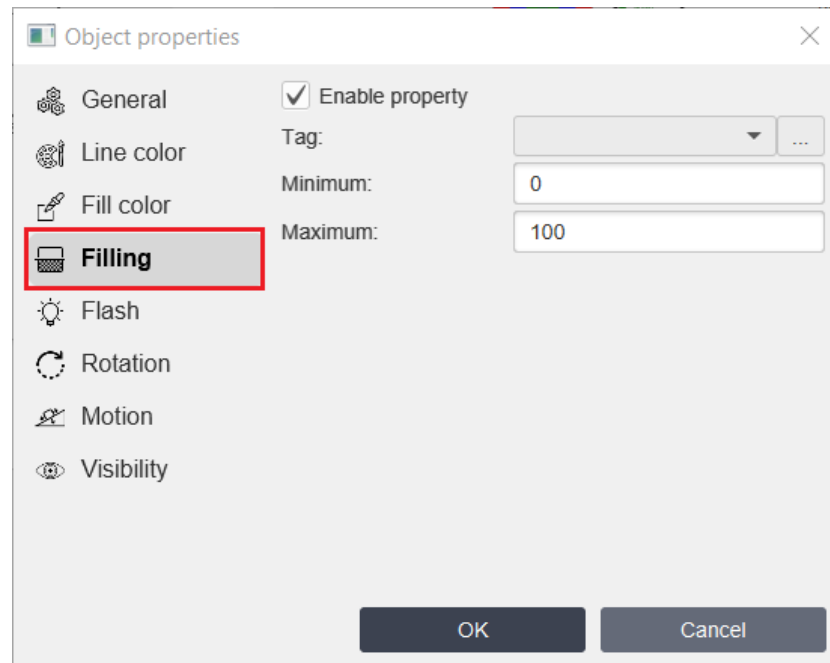
Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value. ▪ Tag.PV in the range - tag's value compares to the values in the ranges. To setup ranges click Collection button.
Color TRUE	Choose a color that will result if the comparison is TRUE in this field.

Property	Description
Color FALSE	Choose a color that will result if the comparison is FALSE in this field.
Ranges	<div><p>If you select Tag.PV in the range in the Type combobox and click Collection button. You'll see the window:</p><div><div>Collection</div><div><div></div><div><div>From:0</div><div>To:10</div><div>Color:White</div><div>AddEditRemove</div><div>Close</div></div></div></div><p>Where:</p><ul style="list-style-type: none">• From - enter the value from which the object will change color in the ?eld.• To - enter the value to which the object will change color in the ?eld.• Color - choose color for this range.<p>You can Add, Edit or Remove collection element of fill color conditions.</p></div>

6.2.5.7 Filling

Not all objects have the Filling property!

The Filling property allows an object to control ?lling of the object depending on tag's value. To confrigure theF?lling property click Filling tab in the Object property window.



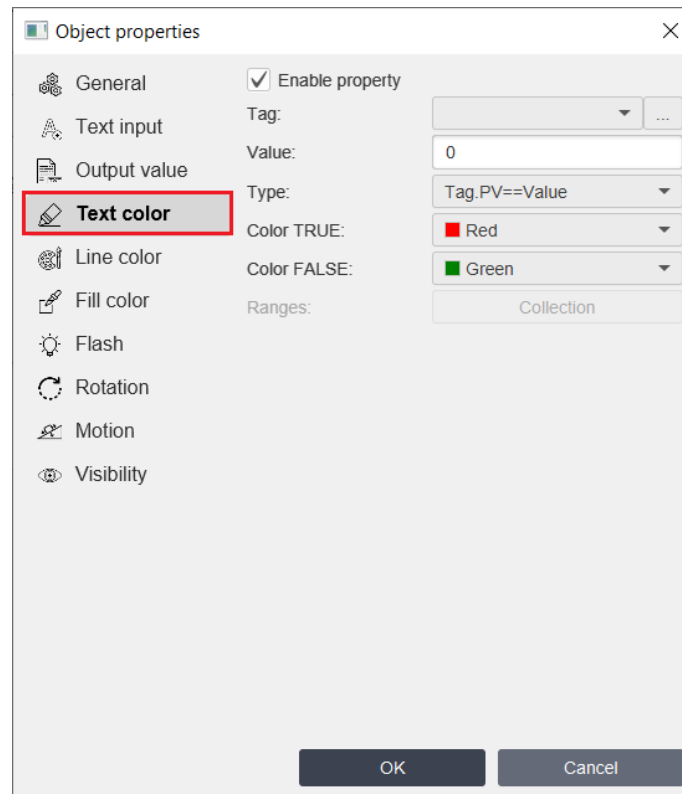
Property	Description
Tag	Select the tag which value will be used to control ?lling.
Minimum*	Enter minimum value of the object's ?lling in the ?eld.
Maximum*	Enter maximum value of the object's ?lling in the ?eld.

*** These properties you can use in ST scripts by using minimum or maximum properties keywords. For example, `Objects.Rectangle.maximum = 200`;**

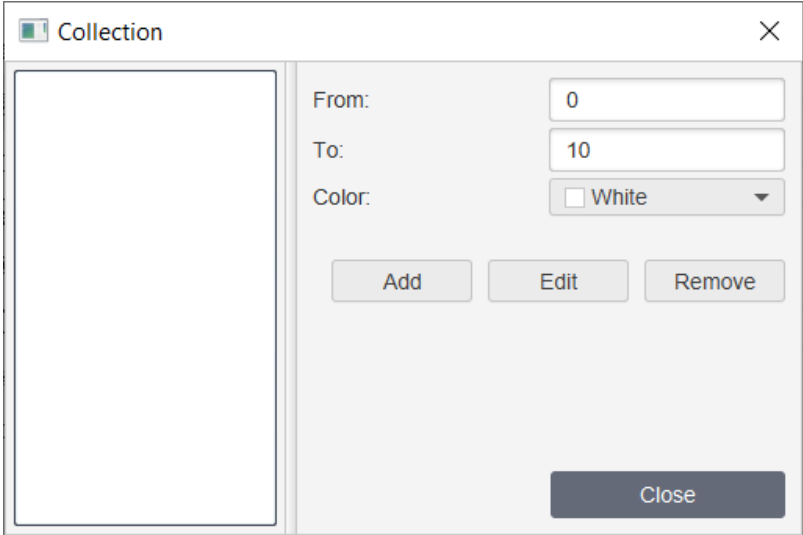
6.2.5.8 Text color

Not all objects have the Text color property!

The Text color property allows an object to change color of text when condition is TRUE or FALSE. To configure the Text color property click Text color tab in the Object property window.



Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value. ▪ Tag.PV in the range - tag's value compares to the values in the ranges. To setup ranges click Collection button.

Property	Description
Color TRUE	Choose a color that will result if the comparison is TRUE
Color FALSE	Choose a color that will result if the comparison is FALSE
Ranges	<p>If you select Tag.PV in the range in the Type combobox and click Collection button. You'll see the window:</p>  <p>Where:</p> <ul style="list-style-type: none"> • From - enter the value from which the object will change color in the ?eld. • To - enter the value to which the object will change color in the ?eld. • Color - choose color for this range. <p>You can Add, Edit or Remove collection element of text color conditions.</p>

6.2.5.9 Control (for buttons)

The Control property allows you to write a value to a tag, call/close a screen/pop-up window, and perform other functions listed in the table below. To configure the Control property click Control tab in the Object property window.

The screenshot shows the 'Object properties' dialog box. The 'Control' tab is selected and highlighted with a red rectangle. The 'Enable property' checkbox is checked. The 'Tag' dropdown is set to 'Tag1'. The 'Function' dropdown is set to 'Set'. The 'Value' field contains '0'. The 'Title' field contains 'Enter value'. The 'Screen' dropdown is empty. The 'Command and args' field is empty. The 'OK' and 'Cancel' buttons are at the bottom.

Property	Description
Tag	Select the tag which value will be written.
Function	<p>Select button's function:</p> <ul style="list-style-type: none"> ▪ Set - write TRUE(1) to the tag. ▪ Reset - write FALSE(0) to the tag. ▪ Toggle - if current tag's value TRUE(1) write FALSE(0), if current tag's value FALSE(0) write TRUE(1). ▪ Push - during pressing button write TRUE. ▪ Set value - write Value to the tag. ▪ Enter value - call dialog that lets you enter value to the tag. ▪ Call screen - call selected screen. ▪ Call popup - call selected popup screen. ▪ Close popup - close popup screen. ▪ Call external software - lets call external software by using command and arguments of OS. ▪ Close application - close application. ▪ Build report - build and show report of the project. ▪ Login - login user of the project.

Property	Description
	<ul style="list-style-type: none"> ▪ Logout - logout current user from the project. User with the less access level is login. ▪ Show/hide main menu - show/hide the main menu.
Value	When you select Set value function enter value that will be written to the tag.
Title	When you select Enter value function write title of the called dialog that lets you enter value.
Screen	When you select Call screen or Call popup function choose screen that will be called after clicking on the button. It's possible to bind button for calling Previous Screen.
Command and args	<p>This field is used in 2 ways:</p> <ol style="list-style-type: none"> 1. When you select Call screen, Call popup or Close popup function this field is used to enter global arguments separated by semicolons. Example: name=pump; description=pump 1 description; It's useful if you want to use some arguments in ST scripts. You can get them by using getglobalargument script command. Example: string name = getglobalargument("name", ""); string description = getglobalargument("description", ""); 2. When you select Call external software function this field is used to enter OS commands and arguments to call external software. Example: <ul style="list-style-type: none"> ▪ for MacOS: open /Applications/TextEdit.app ▪ for Windows: C:/Progra~1/somesoftware.exe ▪ for Android: opc.tesla.scada (name of the Android application package) ▪ for iOS: http://www.youtube.com/watch?v=VIDEO_IDENTIFIER (youtube scheme for calling in iOS).

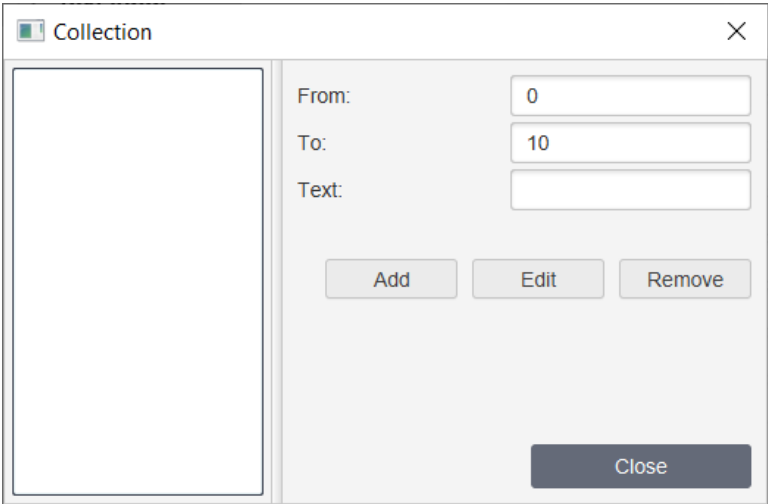
6.2.5.10 Text input

Not all objects have the Text input property!

The Text input property allows an object to display a tag value or text when condition is TRUE or FALSE. To configure text the Input property click Text input tab in the Object property window.

The screenshot shows the 'Object properties' dialog box. The 'Text input' property is selected and highlighted with a red rectangle. The 'Enable property' checkbox is checked. The 'Tag' dropdown is empty, 'Value' is 0, and 'Type' is 'Tag.PV'. Other properties like 'Text TRUE:', 'Text FALSE:', 'Ranges:', 'Text before:', 'Text after:', 'Before decimal position:', and 'Decimal position:' are also visible with their respective input fields.

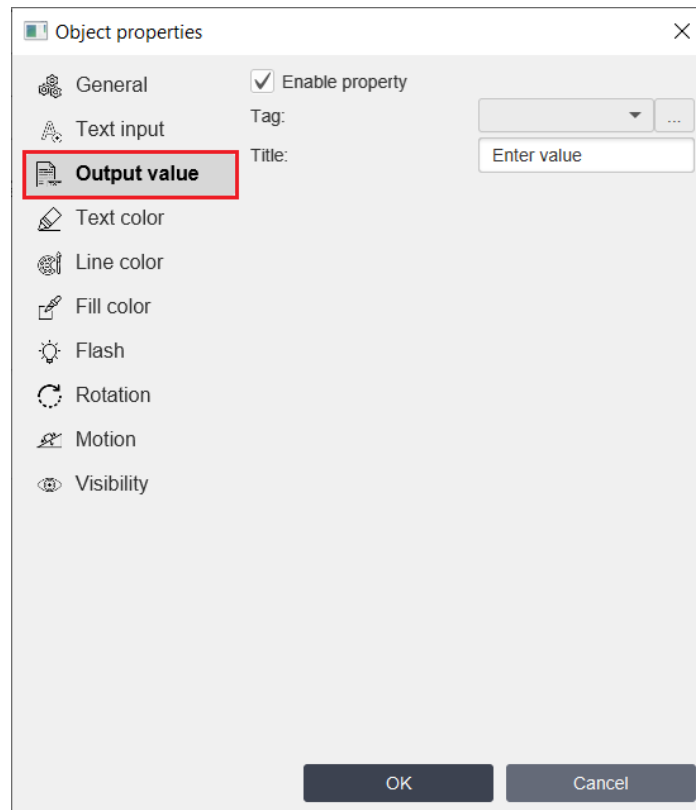
Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison or displaying:</p> <ul style="list-style-type: none"> ▪ Tag.PV - tag's value is displayed. ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value.

Property	Description
	<ul style="list-style-type: none"> ▪ Tag.PV in the range - tag's value compares to the values in the ranges. To setup ranges click Collection button.
Text TRUE	Enter text that will be written if the comparison is TRUE(1)
Text FALSE	Enter text that will be written if the comparison is FALSE(0)
Ranges	<p>If you select Tag.PV in the range in the Type combobox and click Collection button. You'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none"> • From - enter the value from which the object will change text in the ?eld. • To - enter the value to which the object will change text in the ?eld. • Text - enter text in the ?eld. <p>You can Add, Edit or Remove collection element of input text conditions.</p>
Text before	Write the text that will be displayed before the input text.
Text after	Write the text that will be displayed after the input text.
Before decimal position	If the input text is the numeric value of the tag enter number of digits before decimal position.
Decimal position	If the input text is the numeric value of the tag enter decimal position.

6.2.5.11 Output value

Not all objects have the Output value property!

The Output value property allows an object to write value to the tag. To configure the Output property click Output value tab in the Object property window.

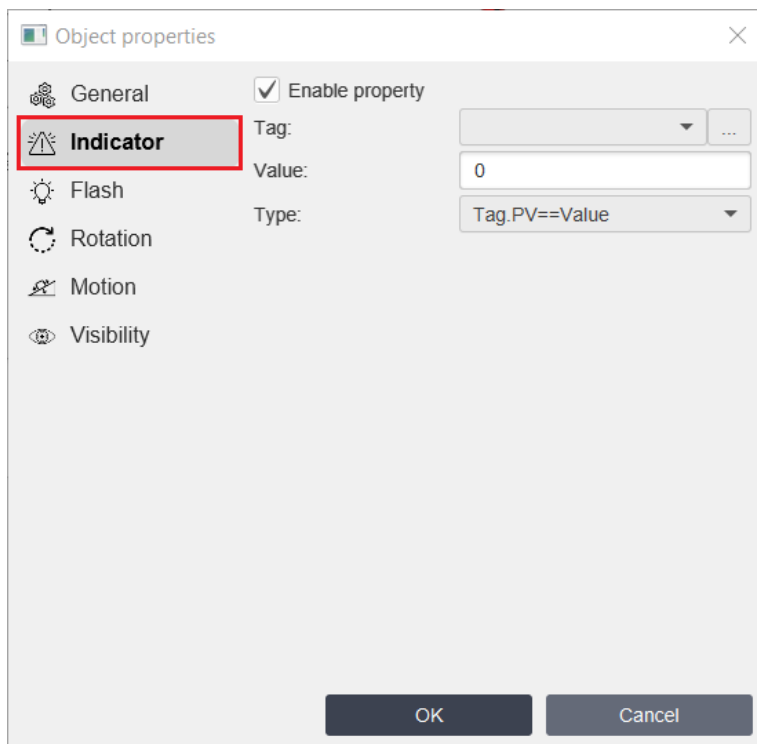


Property	Description
Tag	Select the tag which value will be written.
Title	Enter title of the dialog that will be used to write value to the tag.

6.2.5.12 Indicator

Not all objects have the Indicator property!

The Indicator property allows you to control the object indicator depending on the tag value. To configure this property, click on the Indicator tab in the Object properties Window.

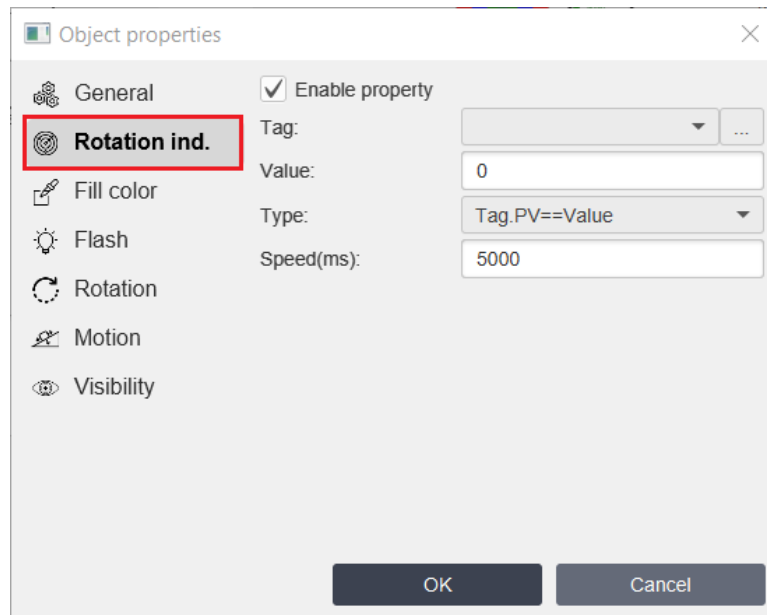


Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value.

6.2.5.13 Rotation indicator

Not all objects have the Rotation Indicator property!

The Rotation Indicator property allows an object to rotate around its center depending on value of the tag. To configure the indicator property click Rotation ind. tab in the Object property window.

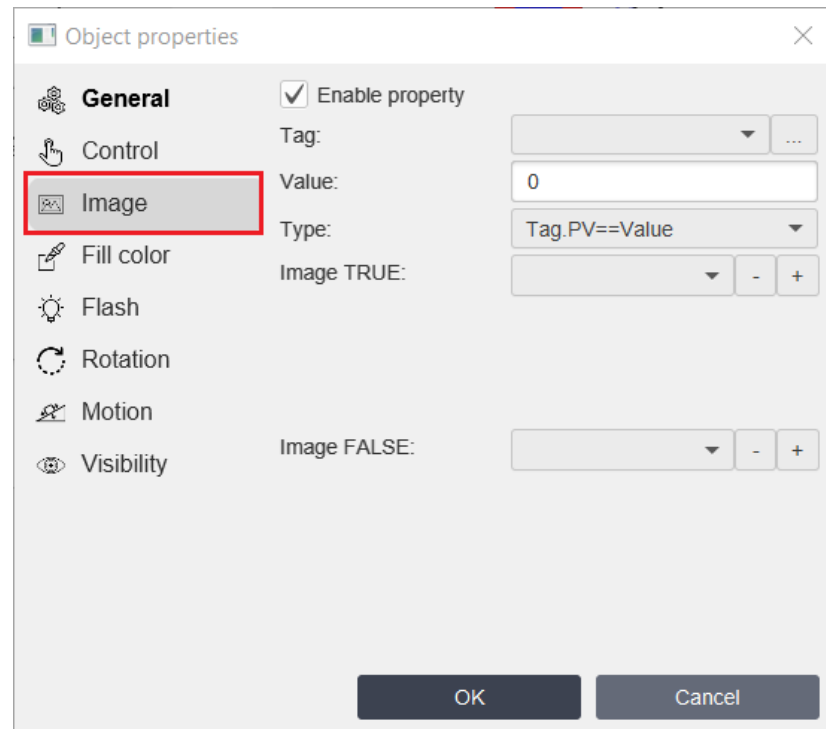


Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	Select type of comparison: <ul style="list-style-type: none"> ▪ Tag.PV == Value - tag's value is equal to the comparison value. ▪ Tag.PV >= Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV <= Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV > Value - tag's value is greater than the comparison value. ▪ Tag.PV < Value - tag's value is less than the comparison value. ▪ Tag.PV != Value - tag's value is not equal to the comparison value.
Speed(ms)	Enter rotation speed

6.2.5.14 Image

Not all objects have the Image property!

The Image property allows an object to change image when condition is TRUE or FALSE. To configure the Image property click Image tab in the Object property window.

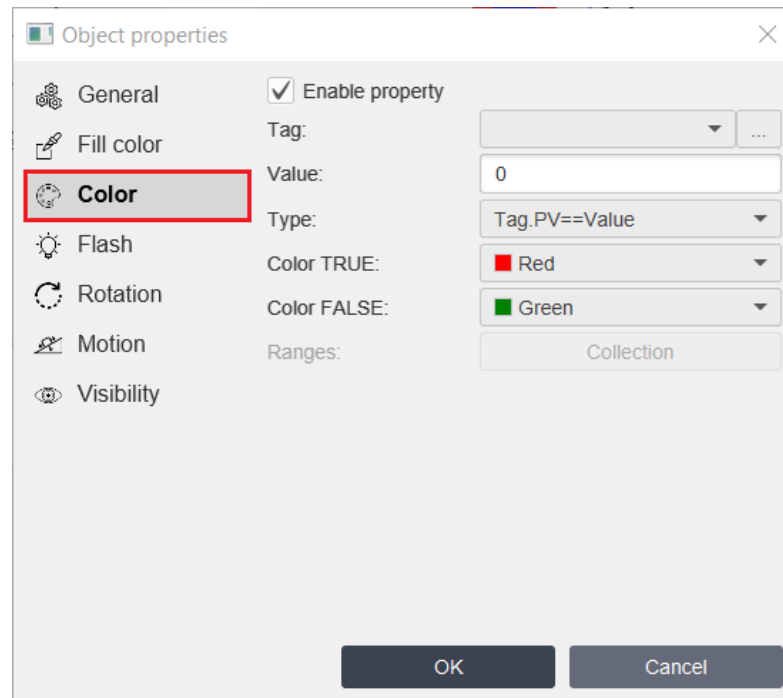


Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value.
Image TRUE	Choose image that will be shown if the comparison is TRUE
Image FALSE	Choose image that will be shown if the comparison is FALSE

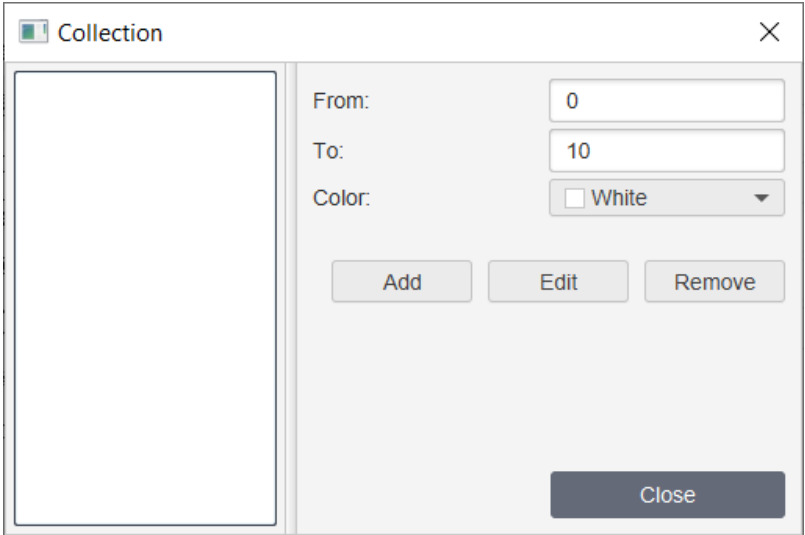
6.2.5.15 Color

Not all objects have the Color property!

The Color property allows an object to change its color when condition is TRUE or FALSE. To configure the Color property click Color tab in the Object property window.

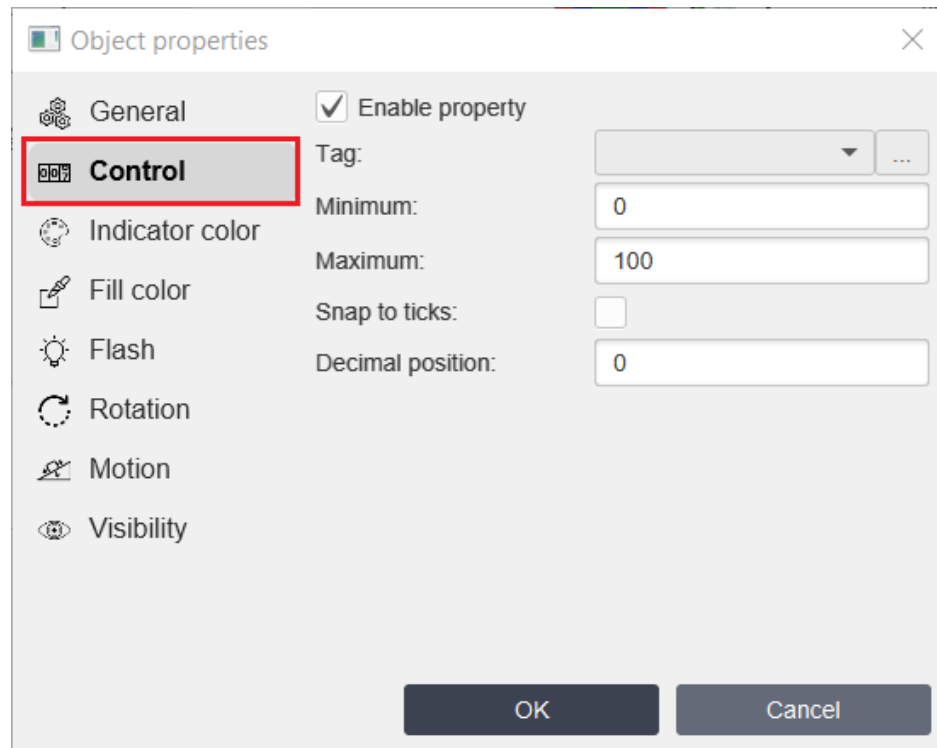


Property	Description
Tag	Select the tag which value will be compared.
Value	Enter the comparison value.
Type	<p>Select type of comparison:</p> <ul style="list-style-type: none"> ▪ Tag.PV==Value - tag's value is equal to the comparison value. ▪ Tag.PV>=Value - tag's value is equal to or greater than the comparison value. ▪ Tag.PV<=Value - tag's value is equal to or less than the comparison value. ▪ Tag.PV>Value - tag's value is greater than the comparison value. ▪ Tag.PV<Value - tag's value is less than the comparison value. ▪ Tag.PV!=Value - tag's value is not equal to the comparison value.

Property	Description
	<ul style="list-style-type: none"> ▪ Tag.PV in the range - tag's value compares to the values in the ranges. To setup ranges click Collection button.
Color TRUE	Choose a color that the object will have if the comparison is TRUE
Color FALSE	Choose a color that the object will have if the comparison is FALSE
Ranges	<p>If you select Tag.PV in the range in the Type combobox and click Collection button. You'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none"> • From - enter the value from which the object will have color of this range. • To - enter the value to which the object will have color of this range. • Color - choose color for this range. <p>You can Add, Edit or Remove collection element of color conditions.</p>

6.2.5.16 Control (for sliders)

The Control property allows to write value to the tag of the object. To configure the Control property click Control tab in the Object property window.



Property	Description
Tag	Select the tag which value will be changed.
Minimum*	Enter minimum value of the object's control.
Maximum*	Enter maximum value of the object's control.
Snap to ticks	Check it if you want to bind control's value to scale ticks.
Decimal position	Enter decimal position of displayed numeric text.

* These properties you can use in ST scripts by using minimum or maximum properties keywords. For example:

Objects.Slider.maximum = 200;

6.2.5.17 Control (for counters)

The Control property allows to write value to the tag. To configure the Control property click Control tab in the Object property window.

Object properties

General ☒ Enable property

Control Tag: [Dropdown] ...

Indicator color Minimum: 0

Text color Maximum: 100

Flash Delta: 1

Rotation Decimal position: 0

Motion

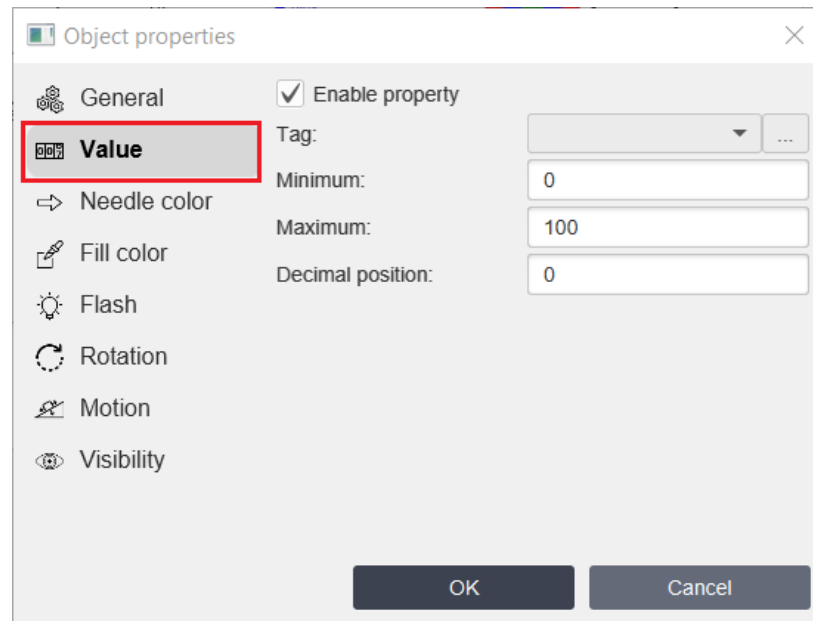
Visibility

OK Cancel

Property	Description
Tag	Select the tag which value will be changed.
Minimum*	Enter minimum value of the object's control
Maximum*	Enter maximum value of the object's control
Delta	This is the value by which the control value will change when the plus and minus buttons are pressed.
Decimal position	Enter decimal position of displayed numeric text in the field.

6.2.5.18 Value (for meters)

The Value property allows an object to control values of analog and digital meters depending on tag's value. To configure the Value property click Value tab in the Object property window.



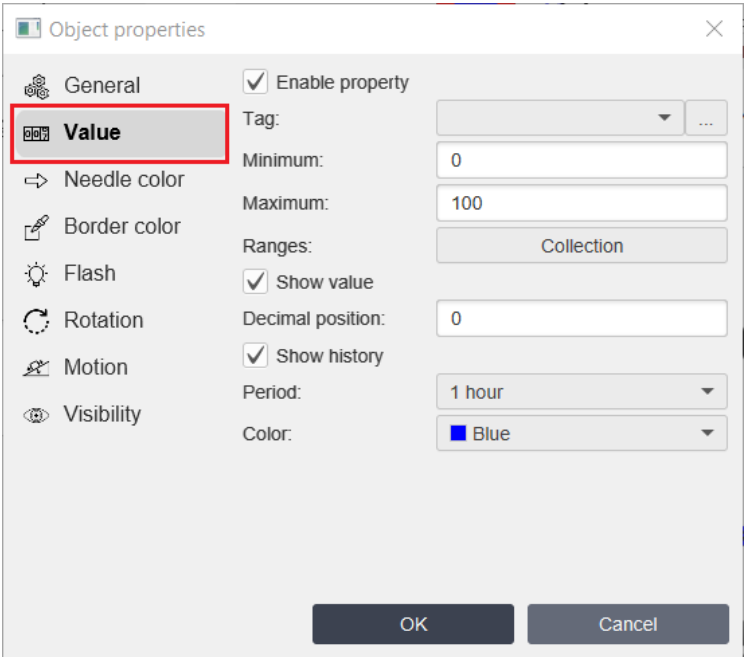
Property	Description
Tag	Select the tag which value will be changed.
Minimum*	Enter minimum value of the meter
Maximum*	Enter maximum value of the meter
Decimal position*	Enter decimal position of displayed numeric text in the ?eld.

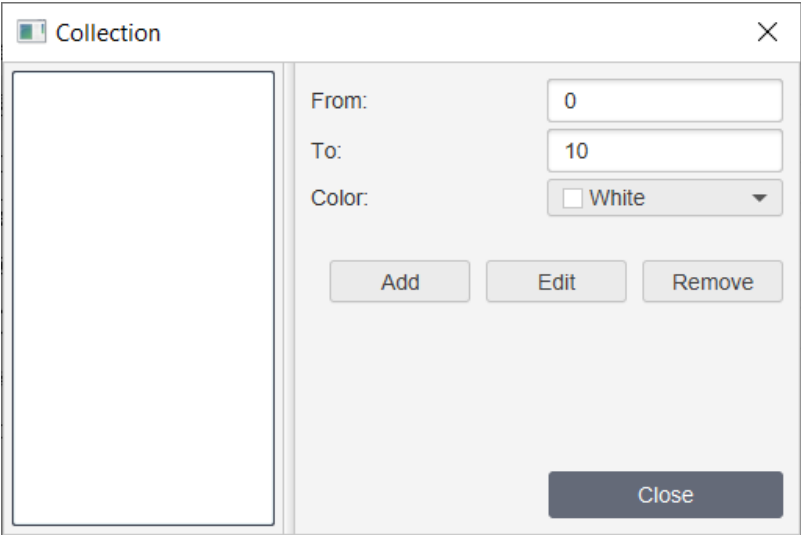
*** These properties you can use in ST scripts by using minimum, maximum and decimalpos properties keywords. For example:**

Objects.Meter.maximum = 200;

6.2.5.19 Value (for range indicators and gauges)

The Value property allows an object to display the value of a tag in an indicator. To configure the Value property click Value tab in the Object property window.



Property	Description
Tag	Select the Tag which value will be used to display on the indicator or gauge.
Minimum*	Enter the minimum value of the indicator or gauge.
Maximum*	Enter maximum value of the indicator or gauge
Ranges	<div>Click Collection button. You'll see the window:</div> <div></div> <div>where:</div> <ul style="list-style-type: none">• From - enter the value from which the object will have color of this range.

Property	Description
	<ul style="list-style-type: none"> • To - enter the value to which the object will have color of this range. • Color - choose color for this range. You can Add , Edit or Remove collection element of color conditions.
Show value	Check it if you want to make visible number representation.
Decimal position*	Enter decimal position of displayed numeric text in the ?eld.
Show history	Check if you want to make visible history information of the tag.
Period	Choose period of the history information.
Color	Choose color of the history information.

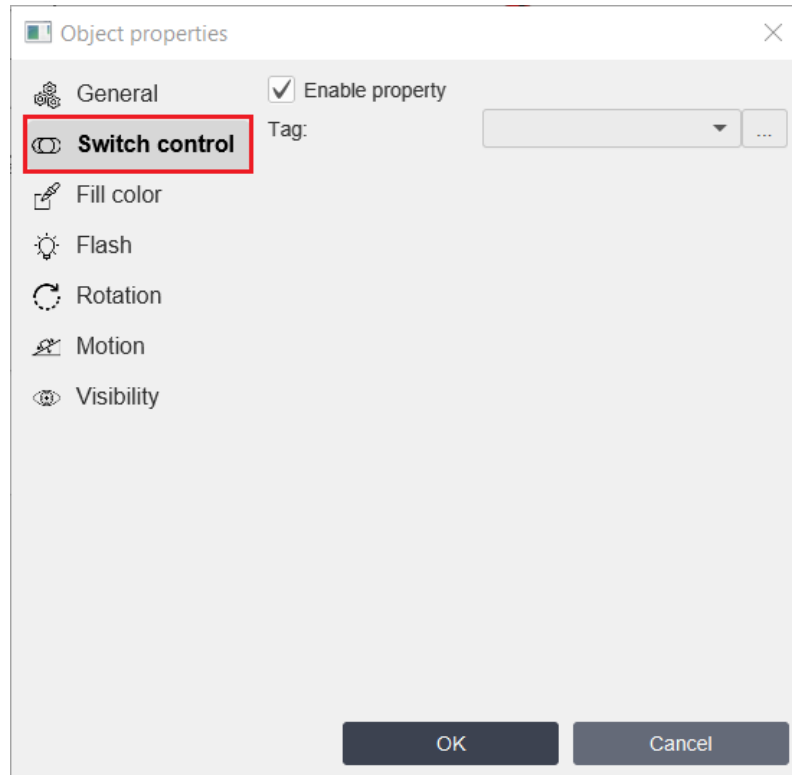
* These properties you can use in ST scripts by using **minimum**, **maximum** and **decimalpos** properties keywords. For example:

Objects.Gauge.maximum = 200;

6.2.5.20 Switch control

Not all objects have the Switch control property!

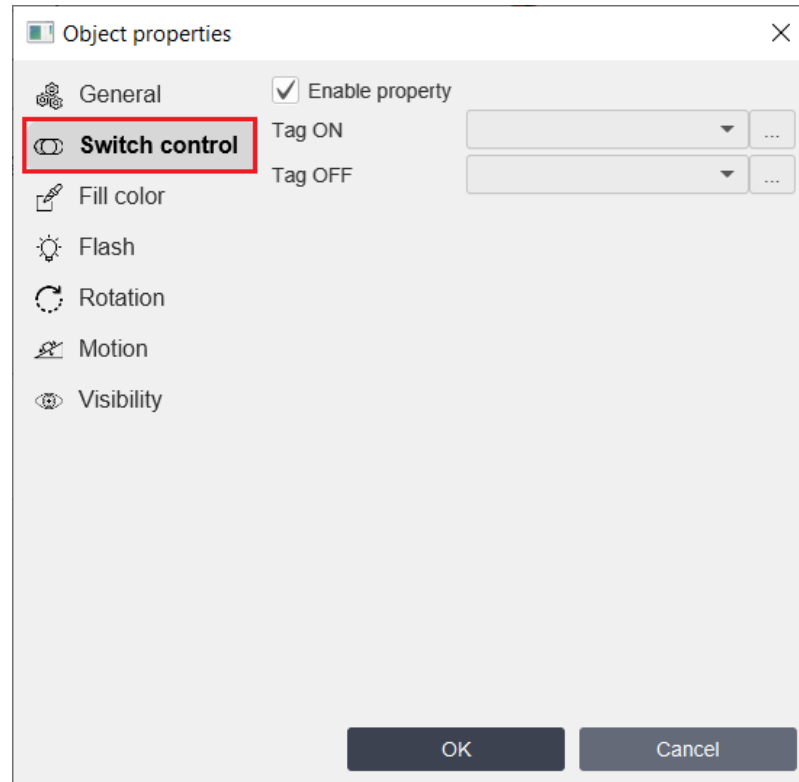
The Switch control property allows an object to switch boolean value of the tag. To configure the Switch control property click Switch control tab in the Object property window.



Property	Description
Tag	Select the tag which value will be controlled by the switch.

6.2.5.21 Switch control (for 3 position switch)

The Switch control property allows an object to switch boolean values of the tags. To configure the Switch control property click Switch control tab in the Object property window.



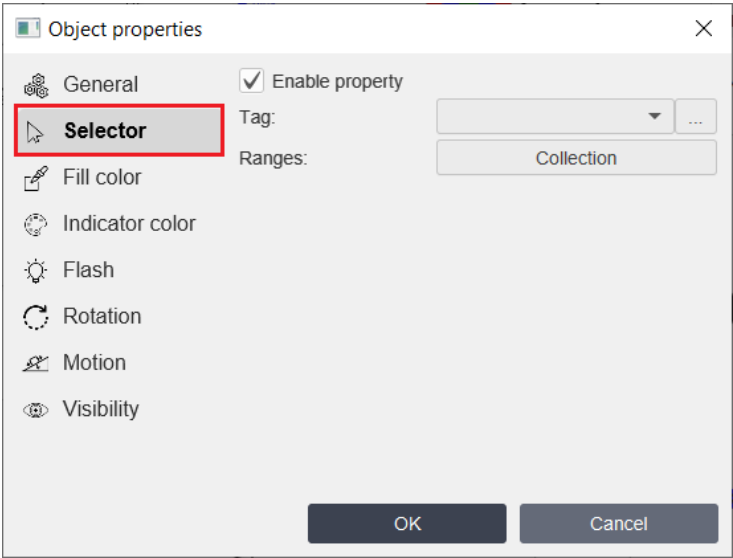
Property	Description
Tag ON	Select the Tag ON which value will be controlled by the switch.
Tag OFF	Select the Tag OFF which value will be controlled by the switch.

When the value of Tag ON is TRUE and the value of Tag OFF is FALSE the switch position will be ON. When the value of Tag ON is FALSE and the value of Tag OFF is TRUE the switch position will be OFF. In other situations the switch position will be Neutral. To switch click (or touch on mobile devices) on the position you want.

6.2.5.22 Selector

Not all objects have the Selector property!

The Selector property allows an object to enter values by clicking selector buttons. To configure the Selector property click Selector tab in the Object property window.

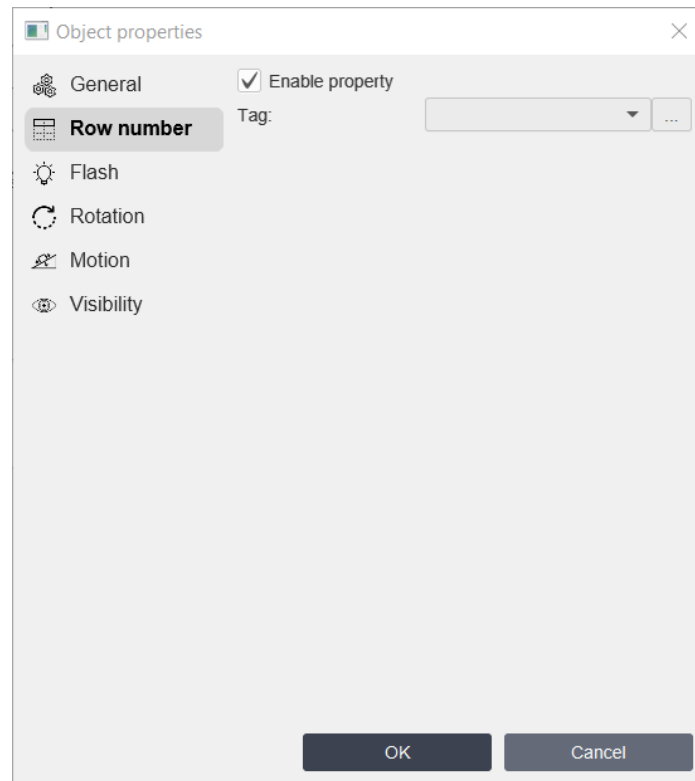


Property	Description
Tag	Select the tag which value will be controlled by the selector.
Ranges	<div>Click Collection button. You'll see the window:</div> <div> <p>The 'Collection' dialog box is shown. It has a list of items on the left: (0.0)-->Zero, (1.0)-->One, and (2.0)-->Two. To the right of the list are two input fields: 'Value:' with the value '0.0' and 'Text:' with the value 'Zero'. Below these fields are three buttons: 'Add', 'Edit', and 'Remove'. At the bottom right is a 'Close' button.</p></div> <div>where:</div> <ul style="list-style-type: none">• Value - enter the value which will be written after clicking the button of the selector.• Text - enter text of the selector's button. <div>You can Add, Edit or Remove collection element of the selector buttons.</div>

6.2.5.23 Row number

Not all objects have the Row number property!

The Row number property allows an object to choose row number of the [Recipe](#)^[483] database in [Parameter table](#)^[248] object. To configure this property click Row number tab in the Object property window.



Property	Description
Tag	Select the tag which value will choose row number of the Recipe ^[483] database.

6.3 Servers

Create server

To create a new server select the menu item [Project](#)^[67] -> [New Server](#)^[68] -> **Server** you want or choose [Servers](#)^[79] tab on the Project Window, click right button on it and choose [New Server](#)^[68] -> **Server** you want item. List of servers:

- [Modbus RTU](#)^[377] - create new Modbus RTU server and open window to edit its properties.
- [Modbus TCP\(UDP\)](#)^[379] - create new Modbus TCP(UDP) server and open window to edit its properties.

- [Siemens](#)^[381] - create new Siemens server and open window to edit its properties.
- [Allen Bradley](#)^[382] - create new Allen Bradley server and open window to edit its properties.
- [OPC UA](#)^[383] - create new OPC UA server and open window to edit its properties.
- [MQTT](#)^[385] - create new MQTT server and open window to edit its properties.
- [Omron](#)^[386] - create new Omron server and open window to edit its properties.
- [BACnet/IP](#)^[388] - create new BACnet server and open window to edit its properties.
- [Common RTU](#)^[389] - create new Common RTU server and open window to edit its properties.
- [Common TCP](#)^[390] - create new Common TCP server and open window to edit its properties.
- [Raspberry GPIO](#)^[391] - create new Raspberry GPIO server and open window to edit its properties.
- [Cloud](#)^[392] - create new Cloud server and open window to edit its properties

Open server properties

To open server properties on [Servers](#)^[79] tab:

- Double click on the server properties which you want to open.
- or
- Right click on the server properties which you want to open and choose Server properties item.

Copy server

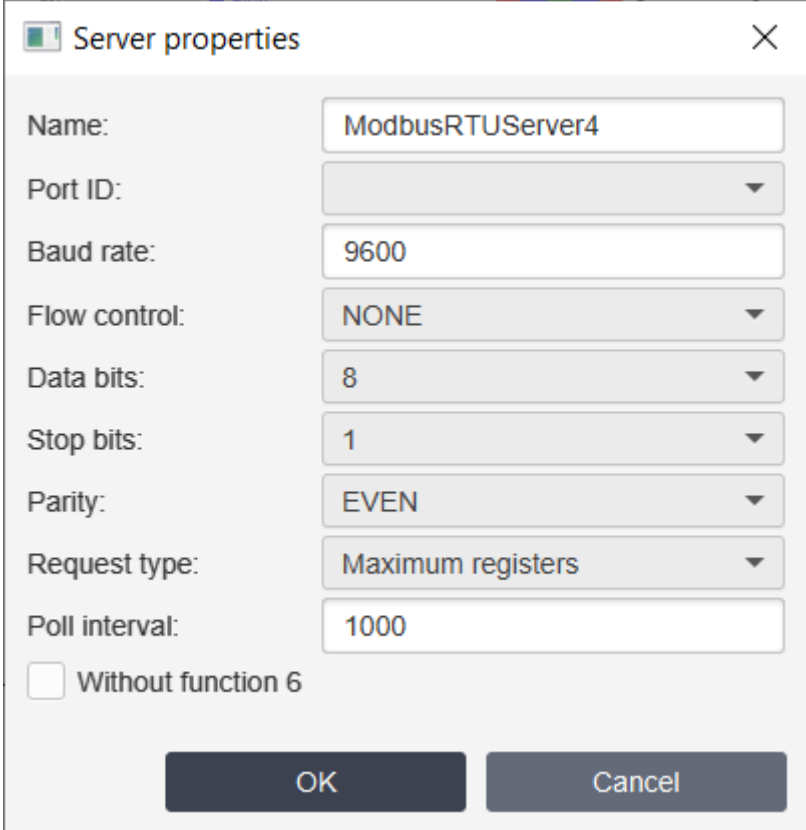
To copy server on [Servers](#)^[79] tab right click on the server you want to copy and choose Copy server item.

Delete server

To delete server on [Servers](#)^[79] tab right click on the server you want to delete and choose Delete server item.

6.3.1 Modbus RTU

To create a new Modbus RTU server select the menu item Modbus RTU. You'll see the following window:



Server properties

Name:

Port ID:

Baud rate:

Flow control:

Data bits:

Stop bits:

Parity:

Request type:

Poll interval:

☐ Without function 6

OK Cancel

List of properties:

Property	ST script field*	Description
Name		Name of the Modbus RTU server.
Port ID	portid	ID of the COM port. If this port can not be open in TeslaSCADA2 Runtime other port will be tried to ?nd and open.
Baud rate	baudrate	Baud rate of the Modbus RTU.
Flow control	flowcontrol	Flow control of the port. It can be NONE, RTSCTS and XONXOF.
Data bits	databits	Number of data bits. It can be 5, 6, 7 and 8.
Stop bits	stopbits	Number of stop bits. It can be 1, 1.5 and 2.
Parity	parity	Parity of the Modbus RTU. It can be NONE, EVEN, ODD, MARK and SPACE.
Request type	requesttype	Choose request type:

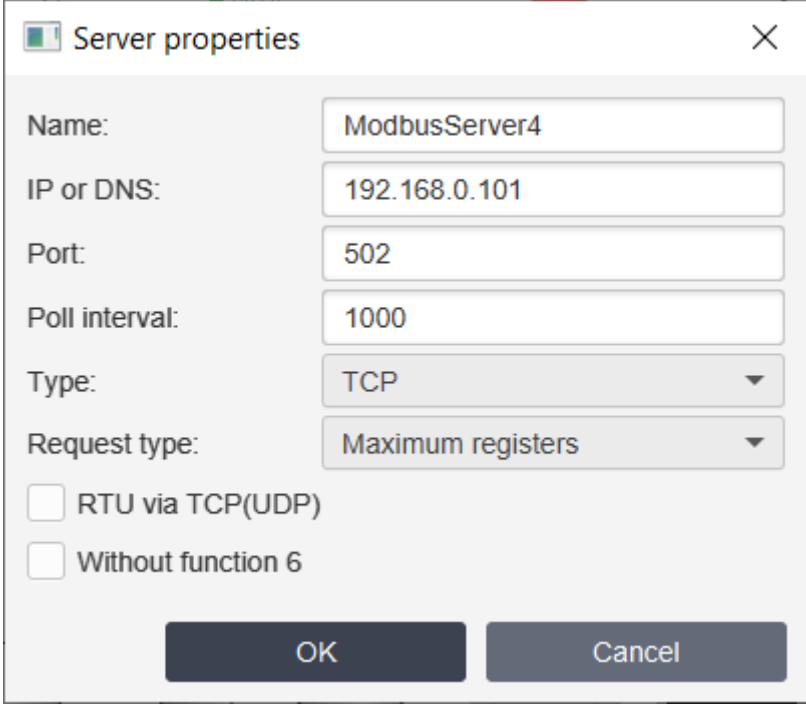
Property	ST script field*	Description
		<ul style="list-style-type: none"> - Maximum registers - if you choose this type the application during polling will send maximum modbus pointers in 1 polling request. - Consecutive registers - if you choose this type the application during polling will send only consecutive modbus pointers in 1 polling request. - 1 pointer registers - if you choose this type the application during polling will send only registers used by 1 pointer in 1 polling request.
Without function 6	withoutfun	Check if your controller doesn't support Modbus writing function 6.

* **This field is used in ST scripts.** For example, Servers.Server1.requesttype = 0. In this script command request type of the Server1 become Maximum registers. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection of the server.
- **lostconnection** - check lost or not connection of the server.

6.3.2 Modbus TCP

To create a new Modbus TCP(UDP) server select the menu item Modbus TCP(UDP). You'll see the following window:



Server properties

Name:

IP or DNS:

Port:

Poll interval:

Type:

Request type:

☐ RTU via TCP(UDP)

☐ Without function 6

OK Cancel

List or properties:

Property	ST script field*	Description
Name		Name of the Modbus TCP server.
IP or DNS	ipaddress	IP address or DNS of the Modbus TCP server.
Port	port	Port of the Modbus TCP server.
Poll interval	interval	Polling interval (period) of the server's requests.
Type	type	Communication protocol of Modbus server - TCP or UDP.
Request type	requesttype	Choose request type: - Maximum registers - if you choose this type the application during polling will send maximum modbus pointers in 1 polling request. - Consecutive registers - if you choose this type the application during polling will send only consecutive modbus pointers in 1 polling request. - 1 pointer registers - if you choose this type the application during polling will

Property	ST script field*	Description
		send only registers used by 1 pointer in 1 polling request.
RTU via TCP(UDP)	rtuviatcp	Check if you use Modbus converter from serial into TCP(UDP) protocol.
Without function 6	withoutfun	Check if your controller doesn't support Modbus writing function 6.

* **This field is used in ST scripts.** For example: Servers.Server1.requesttype = 0. In this script command request type of the Server1 become Maximum registers. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.3 Siemens

To create a new Siemens server select the menu item Siemens. You'll see the following window:

The screenshot shows a 'Server properties' dialog box with the following fields and values:

- Name: SiemensServer1
- IP or DNS: 192.168.0.101
- Port: 102
- Poll interval: 1000
- Controller type: User-defined (dropdown)
- Request type: Maximum registers (dropdown)
- Rack: 0
- Slot: 0

At the bottom are 'OK' and 'Cancel' buttons.

List of properties:

Property	ST script field*	Description
Name		Name of the Siemens server.

Property	ST script field*	Description
IP or DNS	ipaddress	IP address or DNS of the server.
Port	port	Port of the server.
Poll interval	interval	Polling interval (period) of the server's requests.
Controller type	plctype	Type of the Siemens PLC.
Request type	requesttype	Choose request type: - Maximum registers - if you choose this type the application during polling will send maximum modbus pointers in 1 polling request. - 1 pointer registers - if you choose this type the application during polling will send only registers used by 1 pointer in 1 polling request.
Rack	rack	Number of controller's rack
Slot	slot	Number of controller's slot

***This field is used in ST scripts.** For example, Servers.Server1.requesttype = 0. In this script command request type of the Server1 become Maximum registers. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.4 Allen Bradley

To create a new Allen Bradley server select the menu item Allen Bradley. You'll see the following window:

Server properties

Name:

IP or DNS:

Port:

Poll interval:

Controller type:

CPU slot:

Backplane:

List of properties:

Property	ST script field*	Description
Name		Name of the Allen Bradley server.
IP or DNS	ipaddress	IP address or DNS of the server.
Port	port	Port of the server.
Poll interval	interval	Polling interval (period) of the server's requests.
Controller type	plctype	Type of the Allen Bradley PLC.
CPU slot	cpuslot	PLC's cpu slot number.
Backplane	ethernetslot	PLC's backplane number.

***This field is used in ST scripts.** For example: Servers.Server1.interval = 2000. In this script command poll interval of the Server1 will be changed to 2000 ms. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.5 OPC UA

To create a new OPC UA server select the menu item OPC UA. You'll see the following window:

Server properties

Name: OPCUAServer5

URI: opc.tcp://192.168.0.102:4841

Poll interval: 1000

Security mode: None

Policy: None

☒ Anonymous

Username:

Password:

OK Cancel

List of properties:

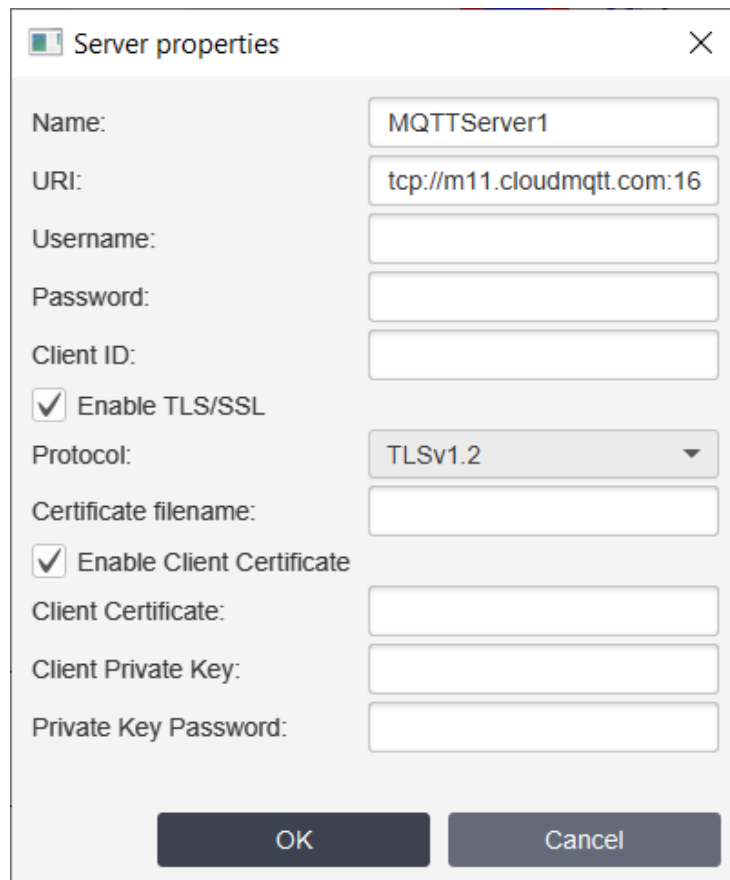
Property	ST script field*	Description
Name		Name of the OPC UA server.
URI	uri	OPC UA server address.
Poll interval	interval	Polling interval (period) of the server's requests.
Security mode	mode	Security mode of the OPC UA server - None, Sign, Sign and Encrypt.
Policy	policy	Security policy of the OPC UA server - Basic128RSA15, Basic256, Basic256SHA256
Anonymous	anonymous	Check if you don't want to use User's token.
Username	username	If you use user token enter username in this field.
Password	password	If you use user token enter password in this field.

***This field is used in ST scripts.** For example, Servers.Server1.interval = 2000. In this script command poll interval of the Server1 will be changed to 2000 ms. For OPC UA server you have to reconnect server. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.6 MQTT

To create a new MQTT server select the menu item MQTT. You'll see the following window:



The screenshot shows a 'Server properties' dialog box. The fields are as follows:

Field	Value
Name	MQTTSer1
URI	tcp://m11.cloudmqtt.com:16
Username	
Password	
Client ID	
Enable TLS/SSL	<input checked="" type="checkbox"/>
Protocol	TLSv1.2
Certificate filename	
Enable Client Certificate	<input checked="" type="checkbox"/>
Client Certificate	
Client Private Key	
Private Key Password	

List of properties:

Property	ST script field*	Description
Name		Name of the MQTT server.
URI	uri	MQTT server address.
Username	username	Username of the server.
Password	password	Password of the server.
Client ID		Client ID of the MQTT server. If you left this field empty server will generate it itself.

Property	ST script field*	Description
Enable TLS/SSL	enablessl	Check Enable TLS/SSL if you want to use server certificate for encryption messages.
Certificate filename	sslfilename	File should be placed in /private folder in the directory where TeslaSCADA2 installed.
Enable Client Certificate	enableclientcert	Check it if you want to use client certificate for encryption messages.
Client certificate**	clientcertname	File should be placed in /private folder in the directory where TeslaSCADA2 installed.
Client private key**	clientprivatekey	File should be placed in /private folder in the directory where TeslaSCADA2 installed.
Private key password**	privatekeypassword	Private key password.
PEM formatted**	pem	Check if your certificate and key files are PEM formatted

***This field is used in ST scripts.** For example: Servers.Server1.username= Admin. In this script command user name of the Server1 will be changed to Admin. For OPC UA server you have to reconnect server. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.
- **reconnect** - when field's value become TRUE server is reconnected.

**** If you use this project for iOS (iPhone or iPad) you should use .p12 format for the file of the certificate.** To create .p12 file you should in openssl utility use this type of command:

openssl pkcs12 -export -out [your file name].p12 -in [your file name].crt -inkey [your file name].key

For example,

openssl pkcs12 -export -out client.p12 -in client.crt -inkey client.key

The name of your .p12 you should place in the Client certificate field (client.p12 in our example). Client Private Key you can left empty. In the Private key password you should enter password of the .p12 file. PEM formatted you can left unchecked. All .p12 files are PEM formatted.

6.3.7 Omron

To create a new Omron server select the menu item Omron . You'll see the following window:

Server properties

Name: OmronServer1

IP or DNS: 192.168.0.101

Port: 9600

Poll interval: 1000

Type: UDP

Network address(DN...): 0

Node address(DA1): 0

Unit number(DA2): 0

OK Cancel

List of properties:

Property	ST script field*	Description
Name		Name of the Omron server.
IP or DNS	ipaddress	IP address or DNS of the server.
Port	port	Port of the server.
Poll interval	interval	Polling interval (period) of the server's requests.
Type	type	Communication protocol of the server - TCP or UDP.
Network address (DNA)	dna	Network address of the server.
Node address (DA1)	da1	Node address of the server. For TCP protocol it will be chosen automatically during communication.
Unit number (DA2)	da2	Unit number.

***This field is used in ST scripts.** For example: Servers.Server1.interval = 2000. In this script command poll interval of the Server1 will be changed to 2000 ms. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.8 BACnet/IP

To create a new Bacnet/IP server select the menu item Bacnet/IP . You'll see the following window:

The screenshot shows a 'Server properties' dialog box with the following fields and values:

Property	Value
Name	BacnetIPServer1
IP or DNS	192.168.1.1
Port	47808
Broadcast IP	255.255.255.255
Poll interval	1000
Device number	1338

List or properties:

Property	ST script field*	Description
Name		Name of the Bacnet server.
IP or DNS	ipaddress	IP address or DNS of the local device.
Port	port	Port of the server.
Broadcast IP	broadcastip	Broadcast IP address
Poll interval	interval	Polling interval (period) of the server's requests and discover devices.
Device number	devicenum	Device number in BACnet network.

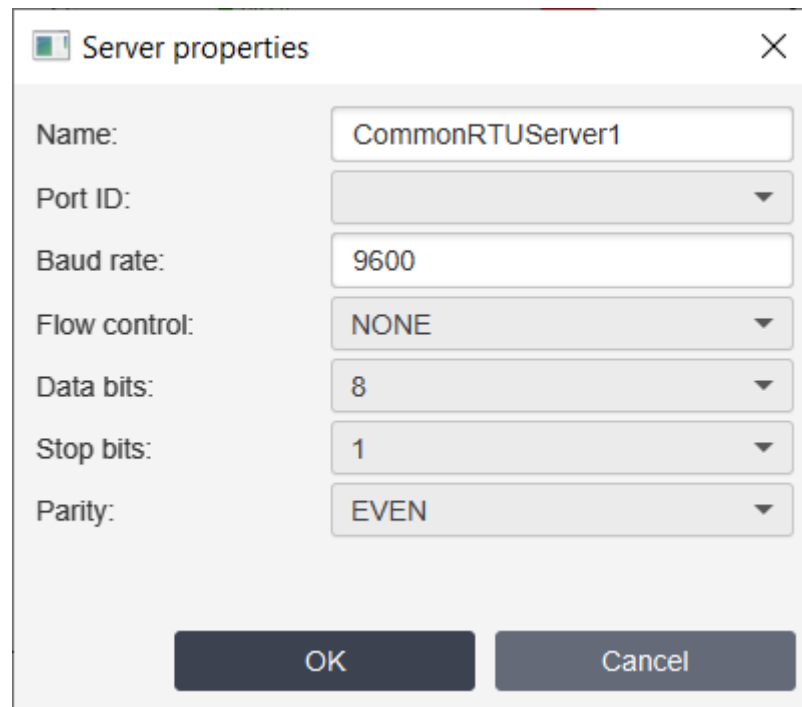
***This field is used in ST scripts.** For example: Servers.Server1.interval = 2000. In this script command poll interval of the Server1 will be changed to 2000 ms. Also for all servers you can use fields:

- **connect** - connect to the server.

- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.9 Common RTU Server

Common RTU server lets you implement user-defined protocol in your project. To create a new Common RTU server select the menu item Common RTU. You'll see the following window:



List of properties:

Property	ST script field*	Description
Name		Name of the Common RTU server.
Port ID	portid	ID of the COM port. If this port can not be open in TeslaSCADA2 Runtime other port will be tried to ?nd and open.
Baud rate	baudrate	Baud rate of the Common RTU server.
Flow control	flowcontrol	Flow control of the port. It can be NONE, RTSCTS and XONXOF.
Data bits	databits	Number of data bits. It can be 5, 6, 7 and 8.
Stop bits	stopbits	Number of stop bits. It can be 1, 1.5 and 2.

Property	ST script field*	Description
Parity	parity	Parity of the Common RTU. It can be NONE, EVEN, ODD, MARK and SPACE.

***This field is used in ST scripts.** For example, Servers.Server1.baudrate = 9600. In this script command server's baud rate is changed to 9600. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection of the server.
- **lostconnection** - check lost or not connection of the server.

6.3.10 Common TCP Server

Common TCP server lets you implement user-defined protocol in your project. To create a new Common TCP server select the menu item Common TCP. You'll see the following window:

List of properties:

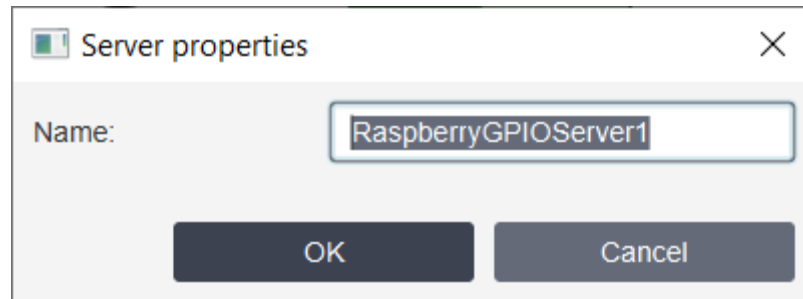
Property	ST script field*	Description
Name		Name of the Common TCP server.
IP or DNS	ipaddress	IP address or DNS of the Common TCP server.
Port	port	Port of the Common TCP server.

***This field is used in ST scripts.** For example, Servers.Server1.port = 502. In this script command server's port changed into 502. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection of the server.
- **lostconnection** - check lost or not connection of the server.

6.3.11 Raspberry GPIO

Raspberry GPIO server lets you implement connection to GPIO of Raspberry PI. To create a new Raspberry GPIO server select the menu item Raspberry GPIO. You'll see the following window:

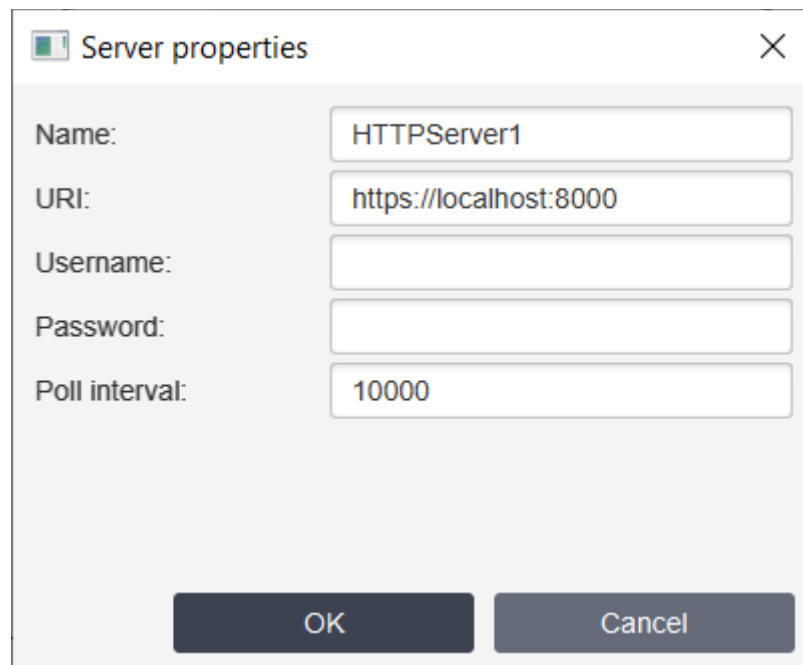


List of properties:

Property	ST script field*	Description
Name		Name of the Raspberry GPIO server.

6.3.12 HTTP-server

To create a new HTTP-server select the menu item HTTP-server . You'll see the following window:



List or properties:

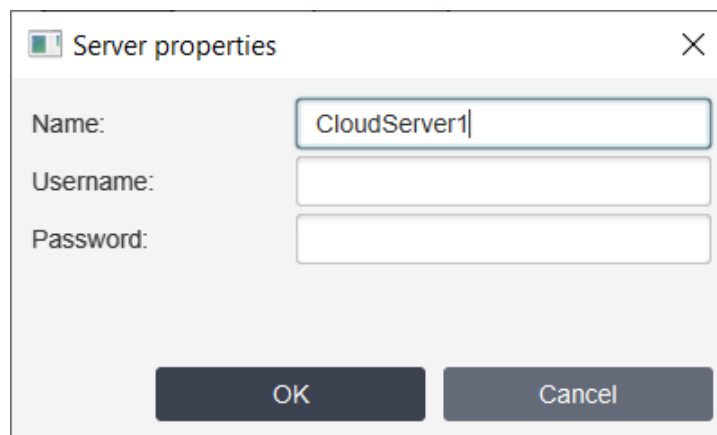
Property	ST script field*	Description
Name		Name of the HTTP server.
URI	uri	URI of the HTTP server.
Username	username	Username of the HTTP server.
Password	password	Password of the HTTP server
Poll interval	interval	Polling interval (period) of the server's requests .

* **This field is used in ST scripts.** For example: Servers.Server1.interval = 2000. In this script command poll interval of the Server1 will be changed to 2000 ms. Also for all servers you can use fields:

- **connect** - connect to the server.
- **connected** - check connection.
- **lostconnection** - check lost or not connection.

6.3.13 Cloud

To create a new Tesla Cloud client select the menu item Cloud . You'll see the following window:



List or properties:

Property	ST script field*	Description
Name		Name of the cloud server.
Username		Username of the Tesla Cloud user.
Password		Password of the Tesla Cloud user.

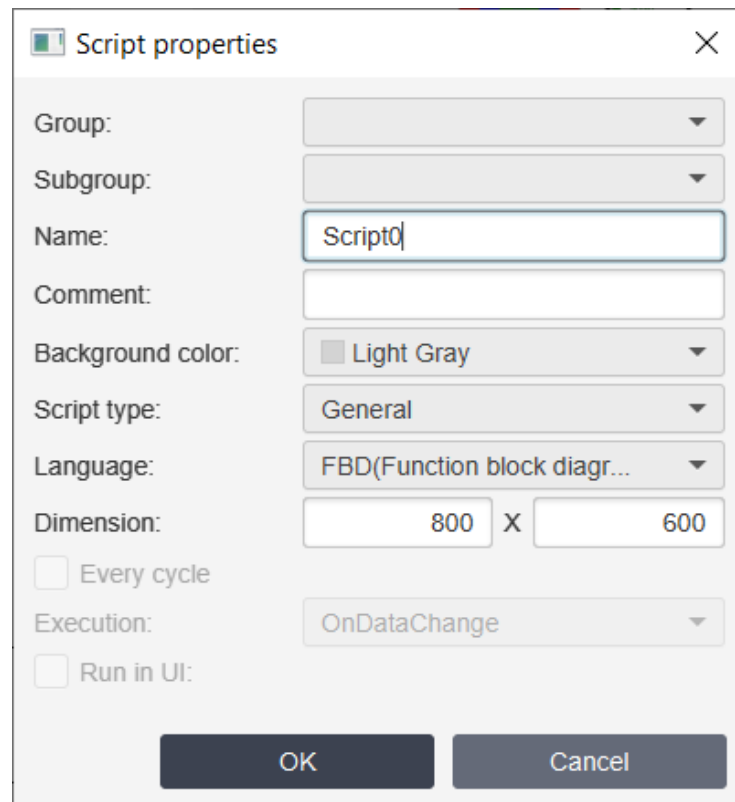
6.4 Scripts

At the moment in TeslaSCADA2 you can use two languages for writing scripts - [FBD](#)^[396] (Functional Block Diagram) and [ST](#)^[416] (Structured Text). They are similar to languages used in PLC programming. Depending on the task at hand, you can choose one or another language. For most tasks, it is better to use ST language as it is more functional. ST language script can be bound to an object or tag as opposed to FBD language and ST language scripts has more call options. FBD language script is called only when tag's values used in this script are changed. But FBD language is more descriptive and may be more familiar to PLC program developers. Also, FBD language has functions that are not available in ST language. These functions are mainly related to time management such as timers, multivibrators, etc. In any case, the choice of the language in which your scripts will be written is yours. Below will be described how to create a particular script and its properties.

Create script

To create a new script select the menu item [Project](#)^[67] -> **New Script** or choose [Scripts](#)^[76] in the Project Window, click right button on it and choose New Script item.

You'll see the [script properties](#)^[395] window:



Script properties

Group: [dropdown]

Subgroup: [dropdown]

Name: Script0

Comment: [text box]

Background color: Light Gray [dropdown]

Script type: General [dropdown]

Language: FBD(Function block diagr... [dropdown]

Dimension: 800 X 600

☐ Every cycle

Execution: OnDataChange [dropdown]

☐ Run in UI:

OK Cancel

Open script


To open script in [Scripts](#)^[76] tab of the Project window:

- Right click on the script you want to open and choose **Open** script item.
- or
- Double click on the script you want to open.


Copy script

To copy script on [Scripts](#)  tab of the Project window right click on the script you want to copy and choose **Copy** script item.


Delete script

To delete script on [Scripts](#)  tab of the Project window right click on the script you want to delete and choose **Delete** script item.

Edit script properties


To edit script properties on [Scripts](#)  tab of the Project window right click on the script you want to edit and choose **Script properties** item.

Export script


To export script on [Scripts](#)  tab of the Project window:

1. Right click on the script you want to export and choose **Export script** item.
2. Now select the location and click the button **Save** (TeslaSCADA2 screen extension .tsp2script).

Import script

To import script on [Scripts](#)  tab of the Project window:

1. Right click on the script window and choose **Import script** item.
2. Now select the script file and click **Open** (TeslaSCADA script extension .tsp2script).

See **Project Window->[Scripts](#)**  **tab** for more information about possible operation with scripts.

6.4.1 Script properties

Script properties

Group:

Subgroup:

Name:

Comment:

Background color:

Script type:

Language:

Dimension: X

☐ Every cycle

Execution:

☐ Run in UI:

OK Cancel

List of script properties:

Property	Description
Group	Select group for the script.
Subgroup	Select subgroup for the script.
Name	Name of the script.
Comment	Optionally specify a meaningful comment.
Background color	Background color of the screen for developing script using FBD language. It's not affect on script execution.
Script type	Select type of the script: <ul style="list-style-type: none"> ▪ General - is binded to the whole project. ▪ Screen - is binded to the screen. ▪ Tag - is executed depending on tag's value. ▪ Object - is binded to the object. ▪ Report - is binded to the report.
Language	Choose language for the script - FBD or ST. Description of the language you can find below in this tutorial.

Property	Description
Dimension	Screen's dimension for developing script using FBD language. It's not affect on script execution.
Every cycle	Check if you want this ST script to be executed every update period of the project. You can find out this period in Project properties (Update interval ^[101] (ms)).
Execution	<p>Choose if you want to use ST script and don't want it's executed every cycle:</p> <ul style="list-style-type: none"> • OnDataChange - script is executed when tag's values used in this script are changed. • OnStart (OnOpen, OnCreate) - script is executed when project is started (for general script type) or screen is opened (for screen script type) or object is created (for object script type). • OnStop (OnClose, OnDestroy) - script is executed when project is stopped (for general script type) or screen is closed (for screen script type) or object is destroyed (for object script type). • OnClick - script executed when screen is clicked (for general and screen script types) or object is clicked (for object script type)
Run in UI	Check if you want to run this script in UI thread. It's helpful if you want to update graphical objects after executing this script.

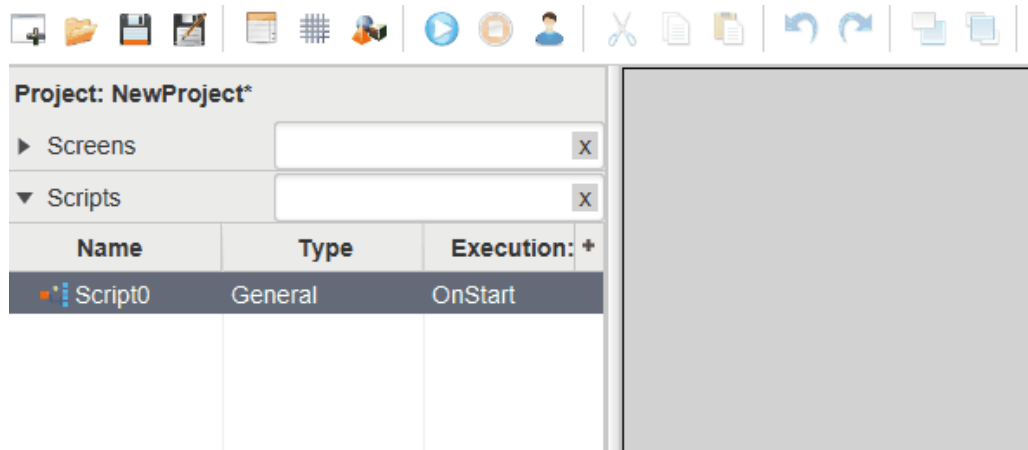
6.4.2 FBD language

Design FBD script

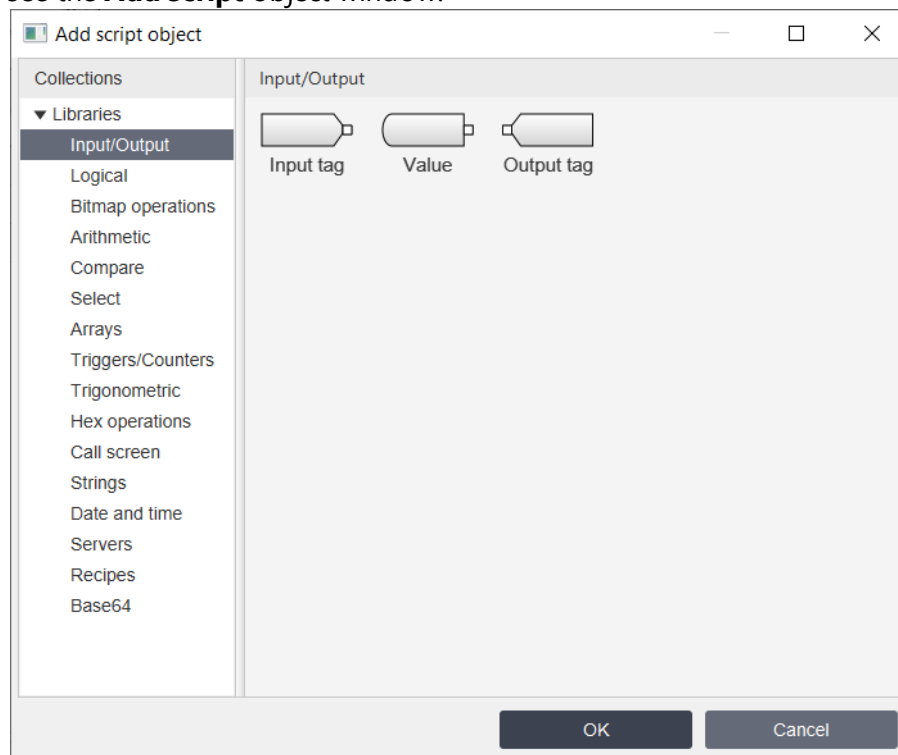
To start designing the script you want, you should double click on it or click right button on the [Project window](#)^[72] -> [Scripts](#)^[76] and choose **Open script** menu item. To develop a script in FBD language you should use FBD objects.

Create script object

Add new object to the screen you can in this way: click right button on the [Canvas](#)^[91] and choose New object menu item:



You'll see the **Add script** object window:



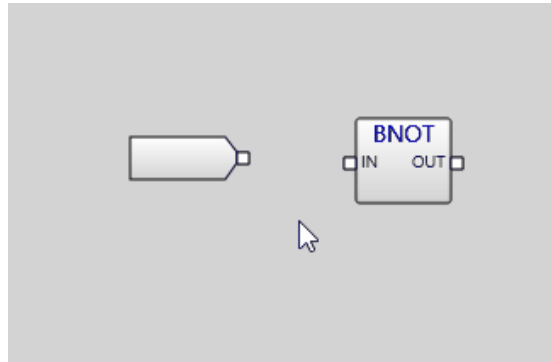
Select library which object you want to use in your project (all libraries and their objects described below). Object you can select in several ways:

1. By double clicking on the object.
2. By clicking on the object (select rectangle will appear) and then clicking OK button.
3. By clicking right button and choosing Select item.

Add script object window will disappear and you can select the location on the screen where you want to place an object.

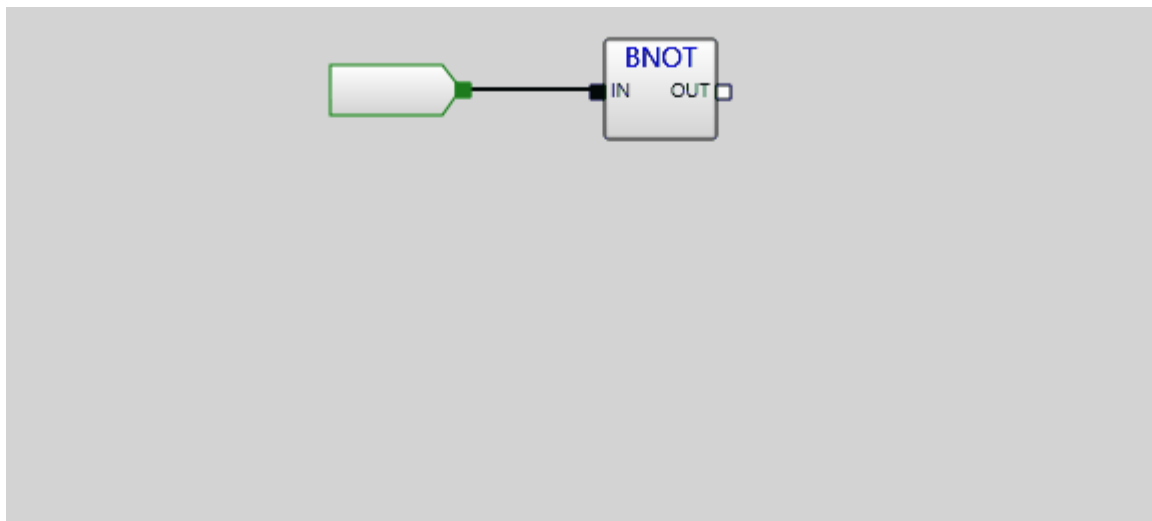
Connect script objects

To connect two objects, click the end of the first (the end to paint over) and click start the second. This will bring up a line connection.



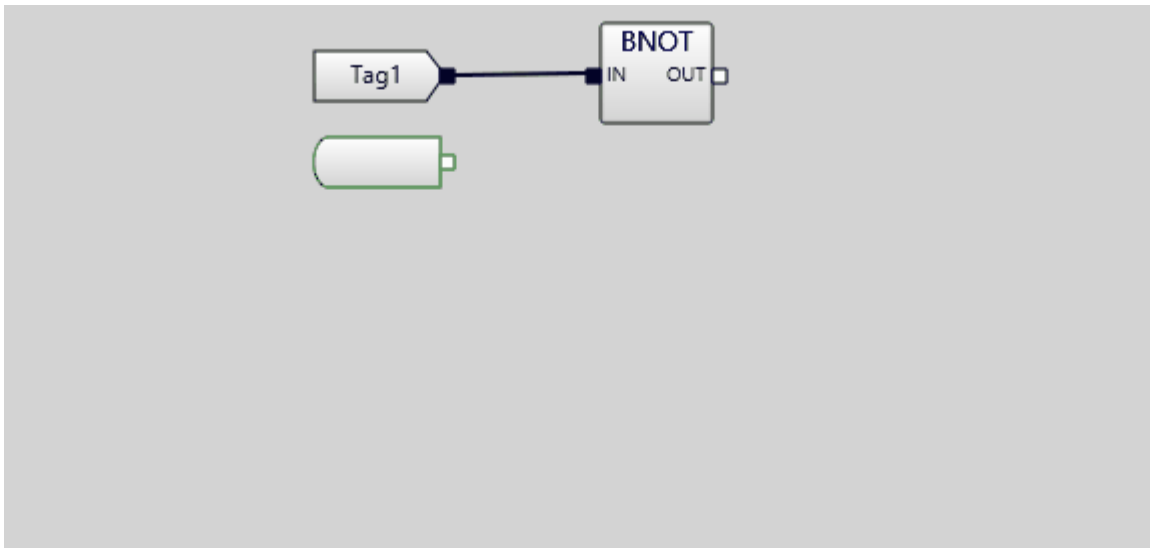
Bind script object to the tag

You can bind Input/Output script objects to the tag. To do this click on Input/Output script object, dialog will appear. Select tag you want to bind.



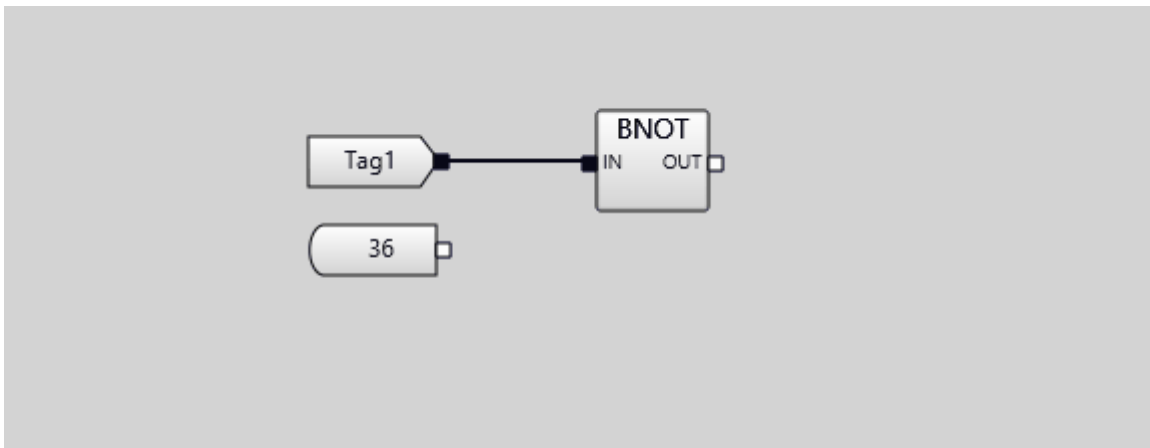
Enter value to the value script object

You can enter value to value script objects. To do this click on value script object, dialog will appear. Enter value you want to use with this object.



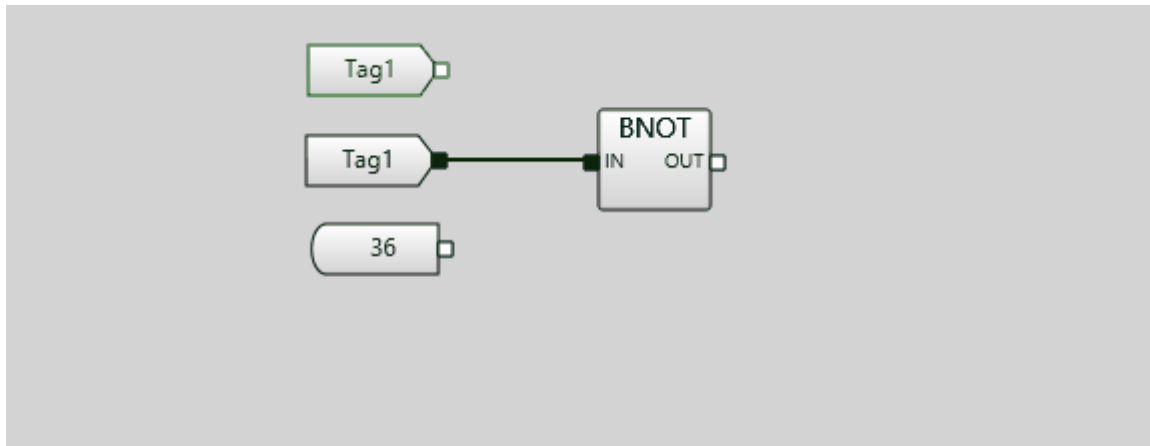
Duplicate script object

You can duplicate script object. Right click on the object you want to duplicate and select **Duplicate** menu item.



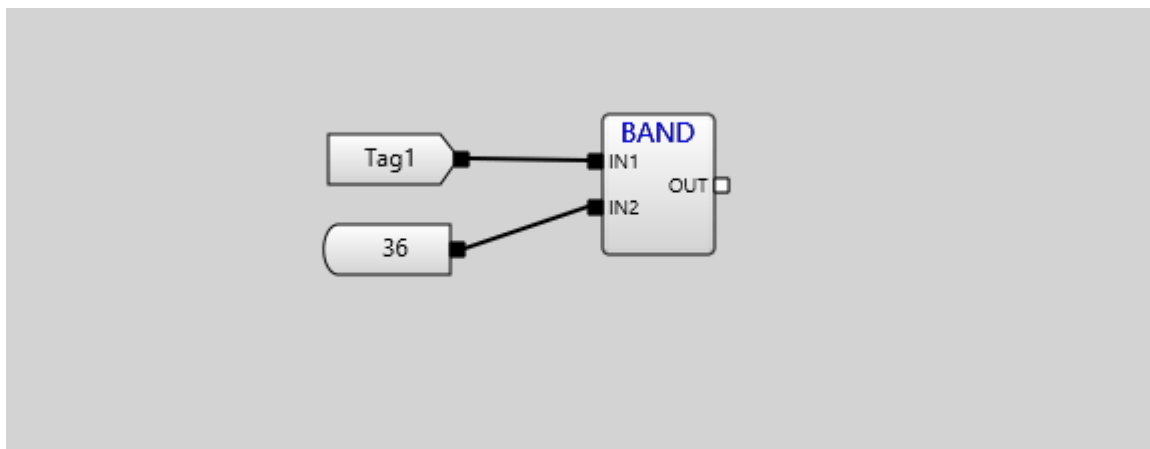
Erase script object

You can erase script object. Right click on the object you want to erase and select Erase menu item.



Erase connection line

You can erase connection line. Right click on the line you want to erase and select Erase menu item.



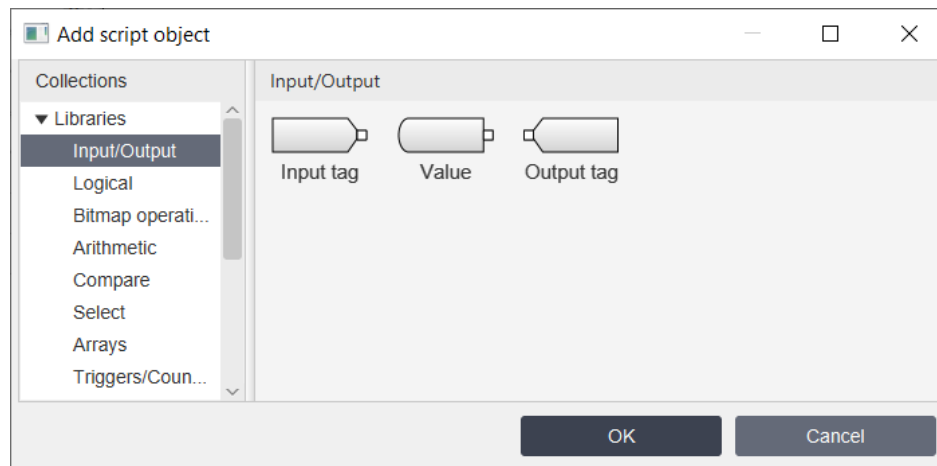
6.4.2.1 Script objects of FBD language

Below description of script libraries:

- [Input/Output library](#)^[401] - allows you to bind tags and constant values to the script.
- [Logical library](#)^[402] - contains objects for working with boolean logical operations.
- [Bitmap operations library](#)^[403] - contains objects for working with bits inside integer variables.
- [Arithmetic library](#)^[404] - contains objects for arithmetic operations.
- [Compare library](#)^[405] - contains objects for comparison operations.
- [Select library](#)^[406] - contains objects for selection operations.
- [Arrays library](#)^[407] - contains objects for working with arrays.
- [Triggers/Counters library](#)^[408] - contains objects for working with triggers and counters.
- [Trigonometric library](#)^[409] - contains objects for trigonometric mathematical operations.

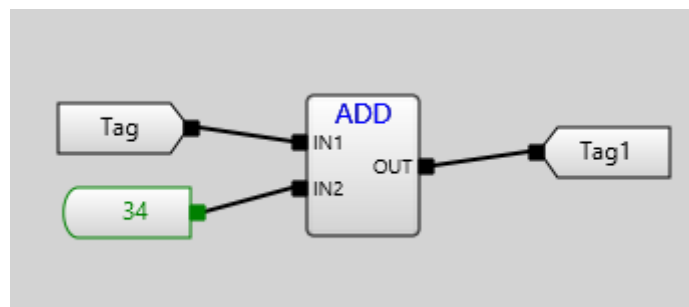
- [Hex operations library](#)^[410] - contains objects for converting decimal numbers to hexadecimal and back.
- [Call screen library](#)^[411] - contains objects for calling screens and popup screens.
- [Strings library](#)^[412] - contains objects for working with strings.
- [Date and time library](#)^[413] - contains object for getting date and time parts (year, day, hour, minute and etc).
- [Servers library](#)^[414] - contains objects for working with servers in the project.
- [Recipes library](#)^[245] - contains object for working with recipes.
- [Base64 library](#)^[415] - contains objects for converting array to base64 string and back.

6.4.2.1.1 Input/Output library



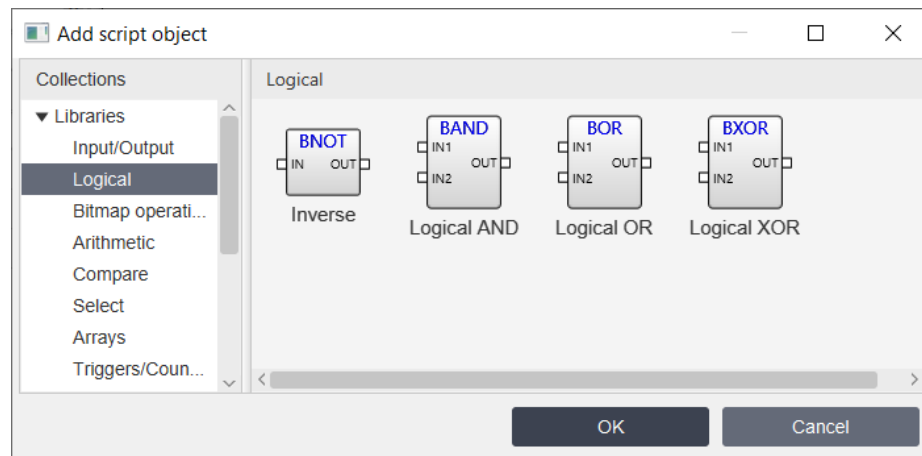
- **Input tag** - this script object used to bind input tag to the script.
- **Output tag** - this script object used to bind output tag to the script.
- **Value** - this script object used to bind input constant value to the script.

Example:



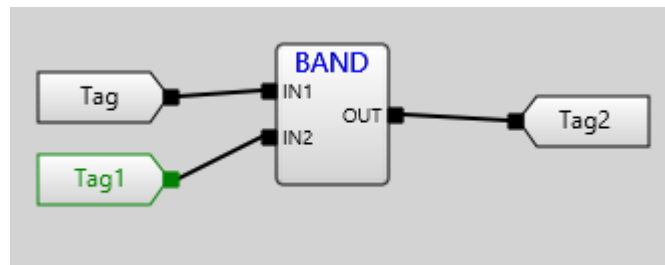
Tag1's value = Tag's value + 34;

6.4.2.1.2 Logical library



- **Inverse** - this script object used to inverse input boolean value (Output = ! Input).
- **Logical AND** - this script object used to logical operation AND for input boolean values (Output = Input & Input2).
- **Logical OR** - this script object used to logical operation OR for input boolean values (Output = Input || Input2).
- **Logical XOR** - this script object used to logical operation XOR for input boolean values (Output = Input XOR Input2).

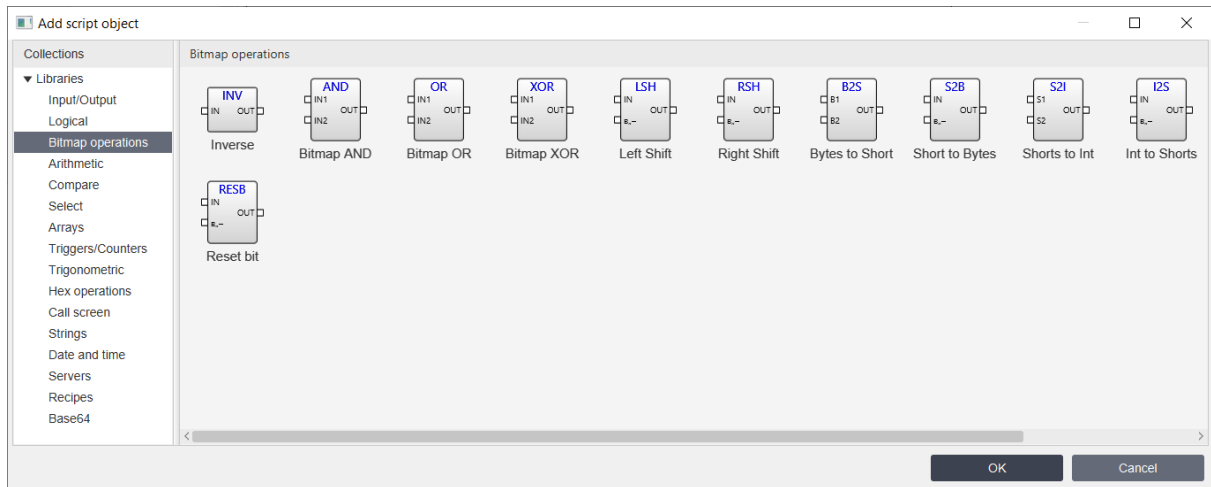
Example:



Tag2's value = Tag's value &(AND) Tag1's value;

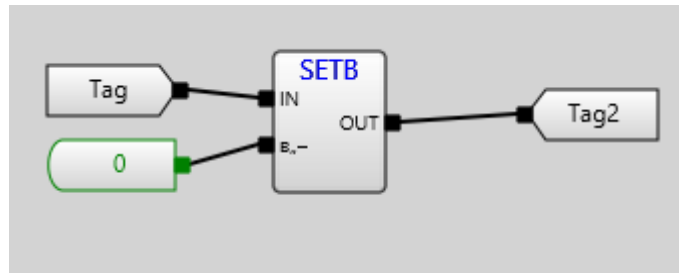
Tag	Tag1	Tag2
FALSE(0)	FALSE(0)	FALSE(0)
FALSE(0)	TRUE(1)	FALSE(0)
TRUE(1)	FALSE(0)	FALSE(0)
TRUE(1)	TRUE(1)	TRUE(1)

6.4.2.1.3 Bitmap operations library



- **Inverse** - this script object used to inverse input integer value (Output = \sim Input).
- **Bitmap AND** - this script object used to logical operation AND for input integer values (Output = Input & Input2).
- **Bitmap OR** - this script object used to logical operation OR for input integer values (Output = Input || Input2).
- **Bitmap XOR** - this script object used to logical operation XOR for input integer values (Output = Input XOR Input2).
- **Left Shift** - this script object used to left shift bits of input value (Output = Input << ? of bits).
- **Right Shift** - this script object used to right shift bits of input value (Output = Input >> ? of bits).
- **Bytes to Short** - this script object used to pack 2 bytes in the short (Output = Input << 8 + Input2).
- **Short to Bytes** - this script object used to unpack short value in 2 bytes (Output = Input[Input2]).
- **Shorts to Int** - this script object used to pack 2 shorts in the int (Output = Input << 16 + Input2).
- **Int to Shorts** - this script object used to unpack int value in 2 shorts (Output = Input[Input2]).
- **Read bit** - this script object used to read bit of the input value (Output = Input[Input2]).
- **Set bit** - this script object used to set bit of the input value (Output = Input | 1 << Input2).
- **Reset bit** - this script object used to reset bit of the input value (Output = Input & \sim (1 << Input2)).

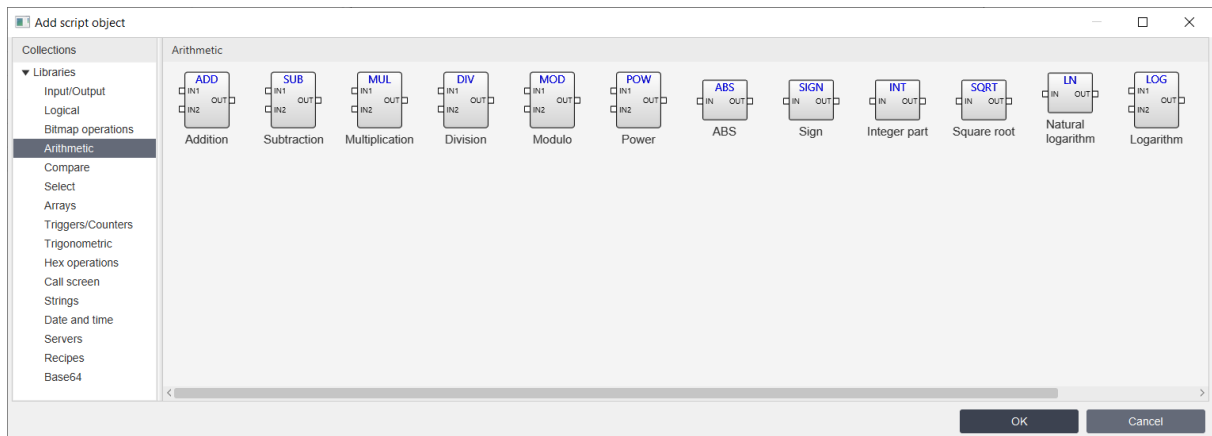
Example:



This operation set 0 bit of Tag's value and place result in Tag2's value.

Tag		Tag2
8	0	9

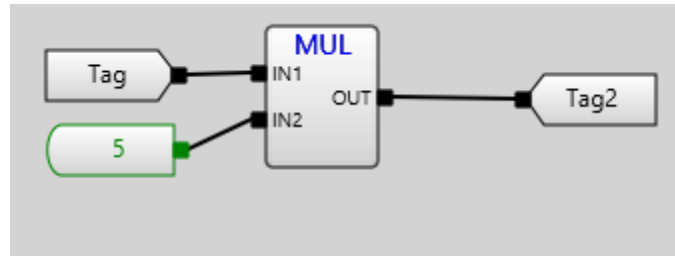
6.4.2.1.4 Arithmetic library



- **Addition** - this script object used to arithmetic operation addition for input values (Output = Input + Input2).
- **Subtraction** - this script object used to arithmetic operation subtraction for input values (Output = Input - Input2).
- **Multiplication** - this script object used to arithmetic operation multiplication for input values (Output = Input * Input2).
- **Division** - this script object used to arithmetic operation division for input values (Output = Input / Input2).
- **Modulo** - this script object used to arithmetic operation modulo for input values (Output = Input % Input2).
- **Power** - this script object used to arithmetic operation power for input values (Output = Input^Input2).
- **ABS** - this script object used to arithmetic operation absolute for input value (Output = |Input|).
- **Sign** - this script object used to arithmetic operation sign for input value (Output = -Input).
- **Int** - this script object used to arithmetic operation for getting integer part of the input value (Output = int(Input)).

- **Sqrt** - this script object used to arithmetic operation sqrt of the input value (Output = Math.Sqrt(Input)).
- **Ln**- this script object used to arithmetic operation ln (natural logarithm) of the input value (Output = Ln(Input)).
- **Log**- this script object used to arithmetic operation log (logarithm) of the input value (Output = LogInput2Input).

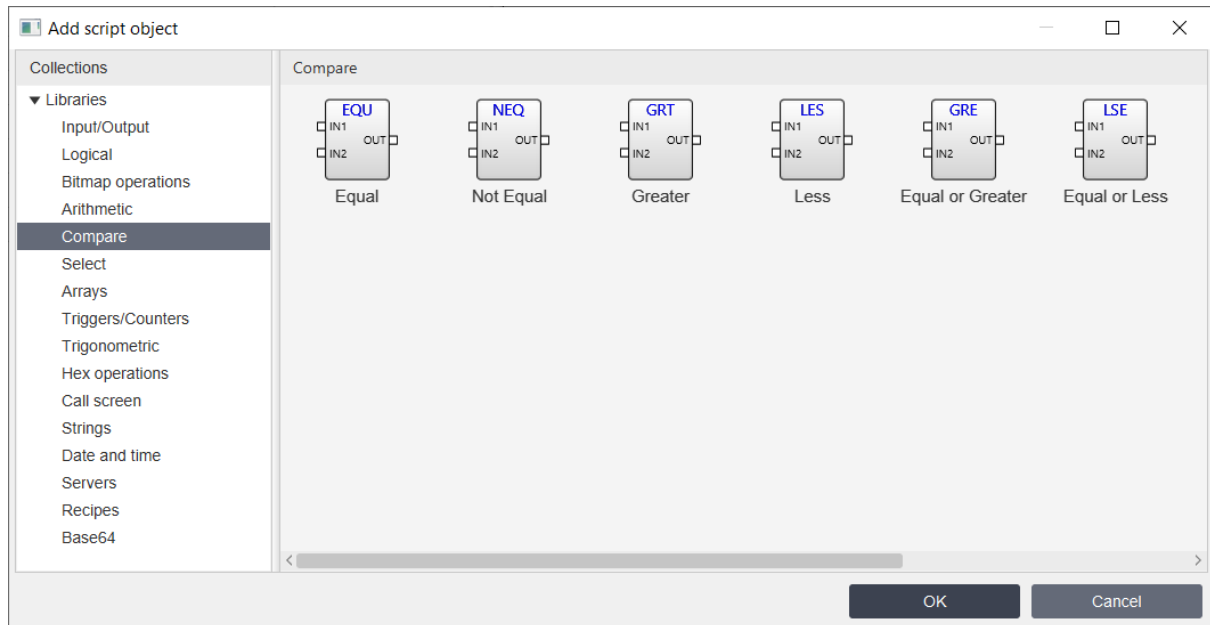
Example:



This operation multiply Tag's value by constant value 5 and place result in Tag2's value.

Tag		Tag2
2	5	10

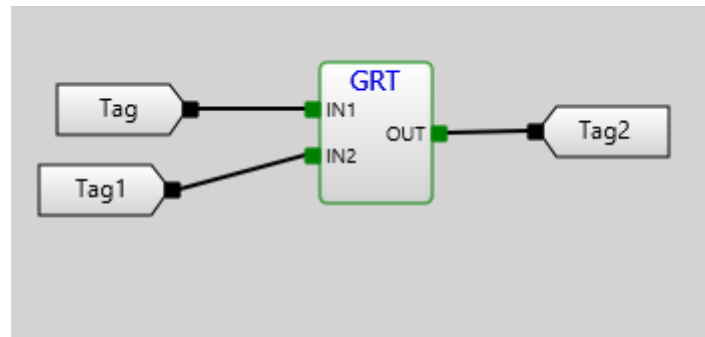
6.4.2.1.5 Compare library



- **Equal** - this script object used to comparison operation equal for input values (Output = Input == Input2).
- **Not Equal** - this script object used to comparison operation not equal for input values (Output = Input != Input2).

- **Greater** - this script object used to compare operation greater for input values (Output = Input > Input2).
- **Less** - this script object used to compare operation less for input values (Output = Input < Input2).
- **Equal or Greater** - this script object used to compare operation equal or greater for input values (Output = Input >= Input2).
- **Equal or Less** - this script object used to compare operation equal or less for input values (Output = Input <= Input2).

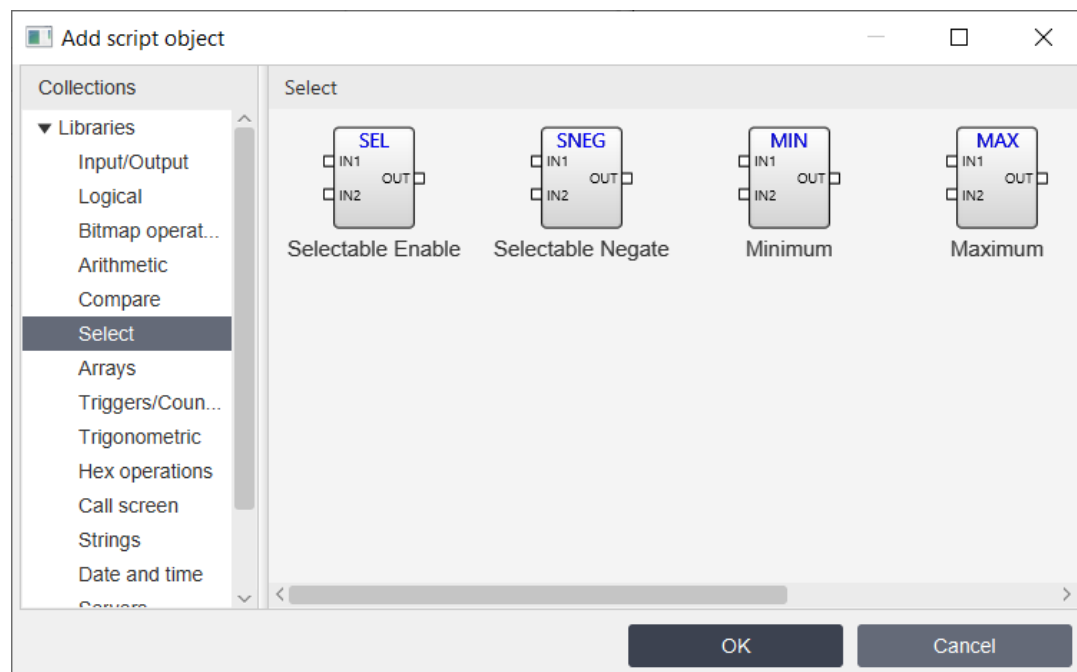
Example:



This operation compare Tag's value and Tag1's value and place result in Tag2's value. If Tag's value greater than Tag1's value Tag2's value equal TRUE(1).

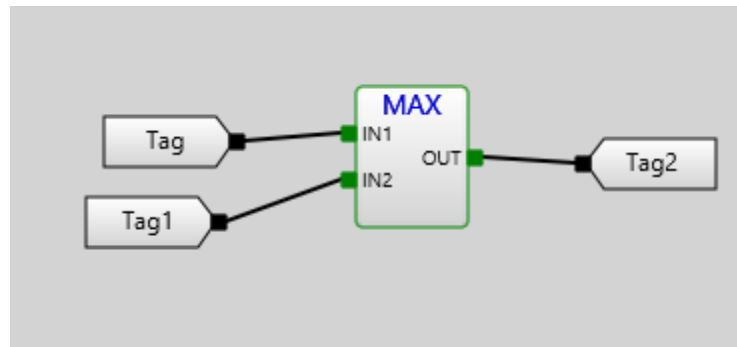
Tag	Tag1	Tag2
5	9	FALSE(0)
12	8	TRUE(1)

6.4.2.1.6 Select library



- **Selectable enable** - this script object used to select value form Input2 if Input1 is true (IF Input==true THEN Output=Input2).
- **Selectable negate** - this script object used to select value form Input2 if Input1 is false (IF Input==false THEN Output=Input2).
- **Minimum** - this script object used to select minimum value of Input2 and Input1 (Output=Min(Input, Input2)).
- **Maximum** - this script object used to select maximum value of Input2 and Input1 (Output=Max(Input, Input2)).

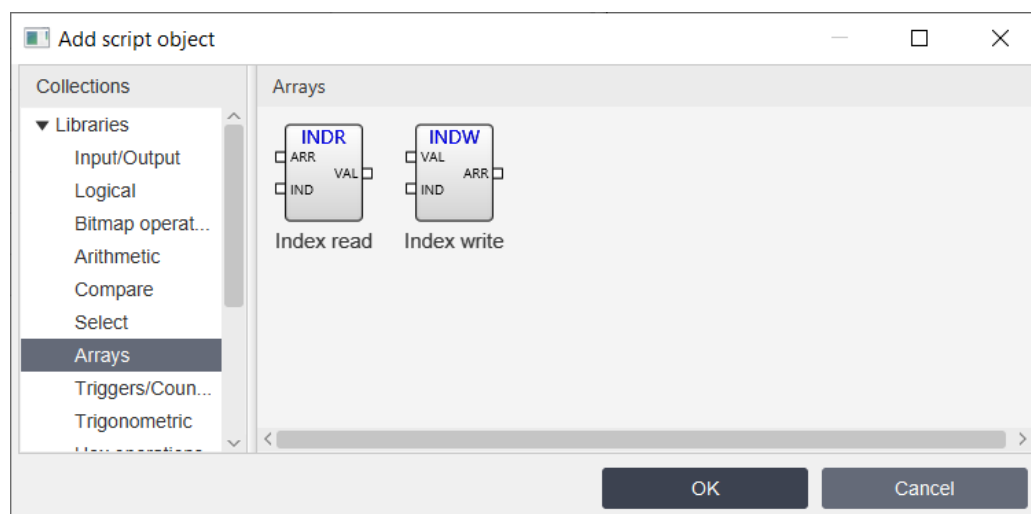
Example:



This operation compare Tag's value and Tag1's value and place result in Tag2's value. If Tag's value greater than Tag1's value, Tag2's value equal Tag's value.

Tag	Tag1	Tag2
5	9	9
12	8	12

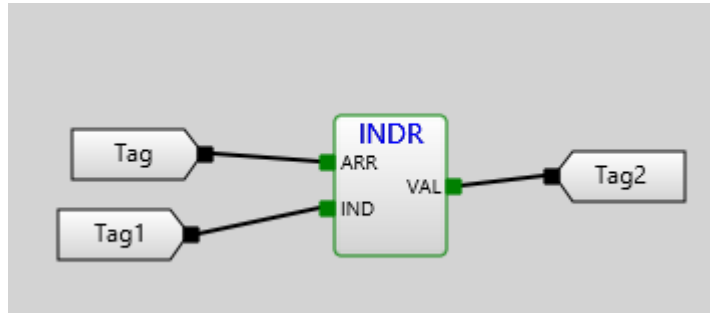
6.4.2.1.7 Arrays library



- **Index read** - this script object used to select array's element. Input1 is an array. Input2 is index of element (Output = Input1[Input2]).

- **Index write** - this script object used to change array's element. Input1 is an element. Input2 is index of element ($\text{Output}[\text{Input2}] = \text{Input1}$).

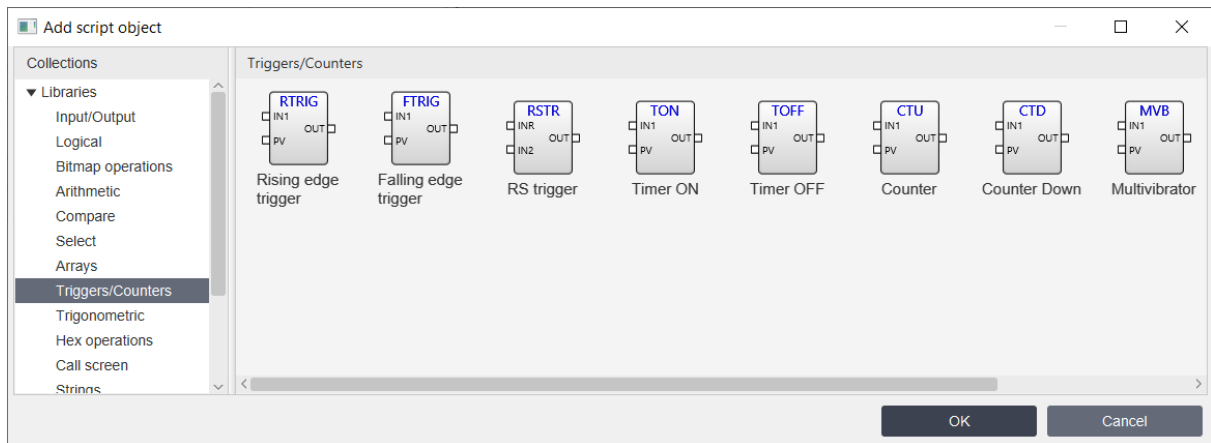
Example:



This operation get value from the array in Tag's value and index in Tag1's value and place result in Tag2's value.

Tag	Tag1	Tag2
[34, 23, 4, 7, 12]	0	34

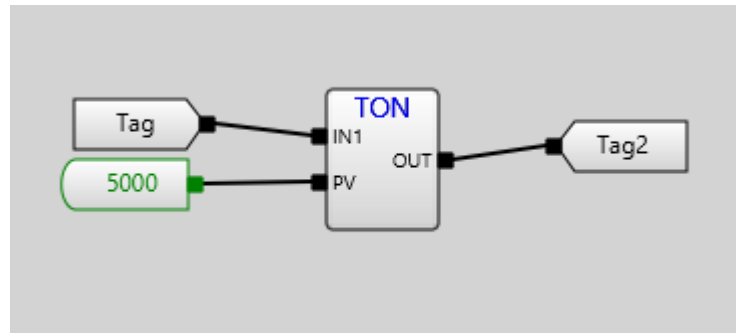
6.4.2.1.8 Triggers/Counters library



- **Rising edge trigger**- this script object used to generate rising impulse duration PV ms when Input1 get TRUE from FALSE.
- **Falling edge trigger**- this script object used to generate rising impulse duration PV ms when Input1 get FALSE from TRUE.
- **RS trigger**- this script object used to imitate RS trigger.
- **Timer ON**- this script object used for delay timer for the duration PV when Input1 get TRUE from FALSE.
- **Timer OFF**- this script object used for delay timer for the duration PV when Input1 get FALSE from TRUE.
- **Counter**- this script object used to count impulses of boolean value in Input1. Counter resets when Output become equal PV.

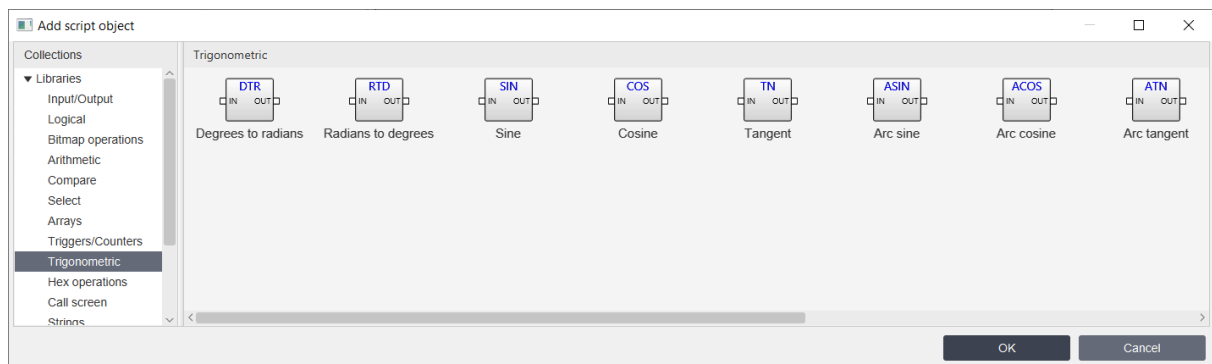
- **Counter Down**- this script object used to count impulses of boolean value in Input1. Counter starts from value PV. Counter resets when Output become equal 0.
- **Multivibrator** - this script imitates impulse generator with PV period. It starts when IN1 changed from FALSE to TRUE.

Example:



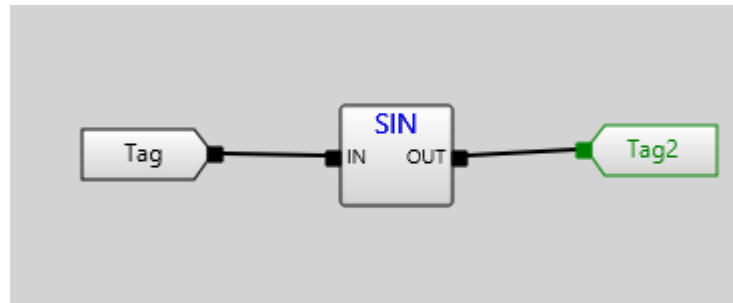
This operation set Tag2's value to TRUE(1) in 5000 ms when Tag's value become TRUE(1) from FALSE(0).

6.4.2.1.9 Trigonometric library



- **Degrees to radians** - this script object used to convert degrees to radians.
- **Radians to degrees** - this script object used to convert radians to degrees.
- **Sine** - this script object used to calculate sin of Input value. (Output = sin(Input)).
- **Cosine** - this script object used to calculate cos of Input value. (Output = cos(Input)).
- **Tangent** - this script object used to calculate tag of Input value. (Output = tag(Input)).
- **Arc Sine** - this script object used to calculate arc sin of Input value. (Output = arc sin(Input)).
- **Arc Cosine** - this script object used to calculate arc cos of Input value. (Output = arc cos(Input)).
- **Arc Tangent** - this script object used to calculate arc tag of Input value. (Output = arc tag(Input)).

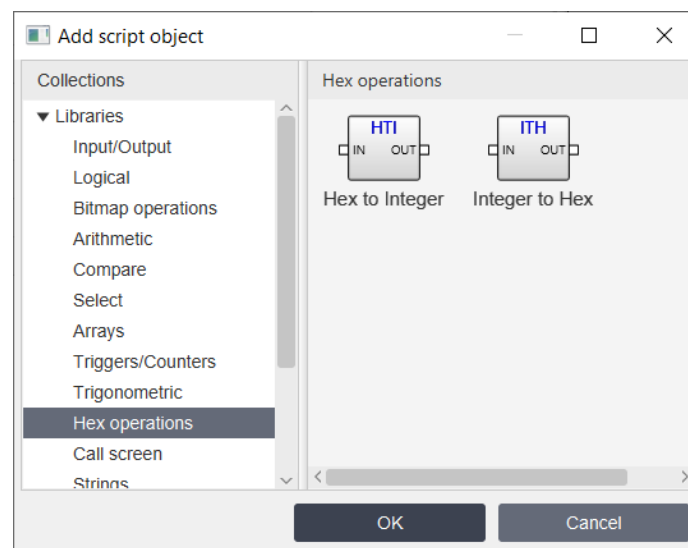
Example:



This operation counts sine of Tag's value and place result in Tag2's value.

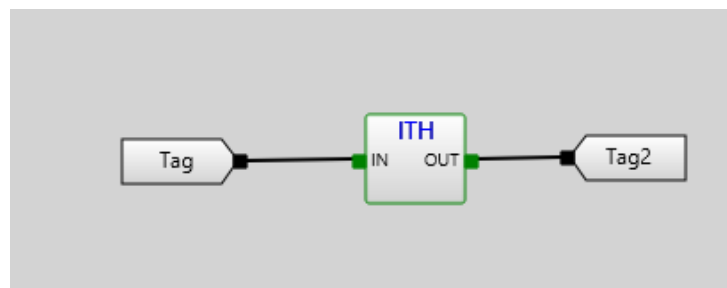
Tag	Tag2
1	0.8414709848078965066525023216303

6.4.2.1.10 Hex operations library



- **Hex to Integer** - this script object converts hex value into integer.
- **Integer to Hex** - this script object converts integer value into hex.

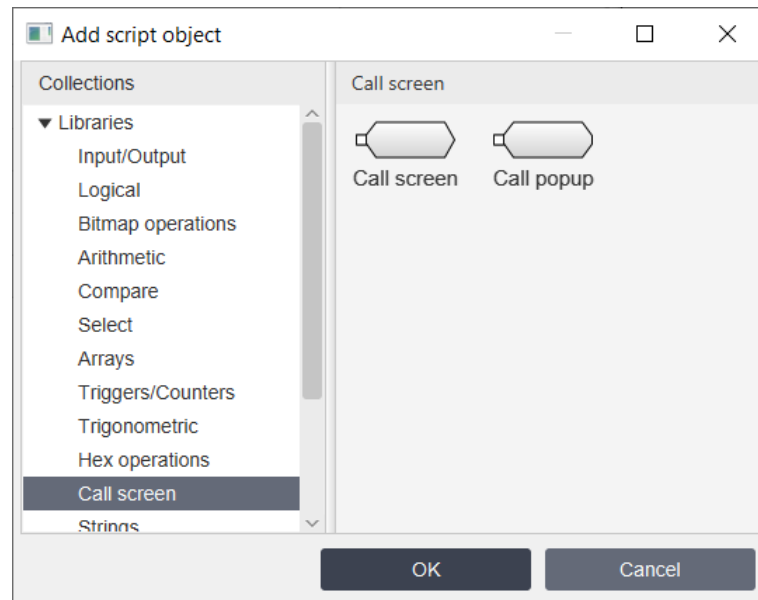
Example:



This operation converts Tag's value from decimal integer into hexadecimal and place result in Tag2's value.

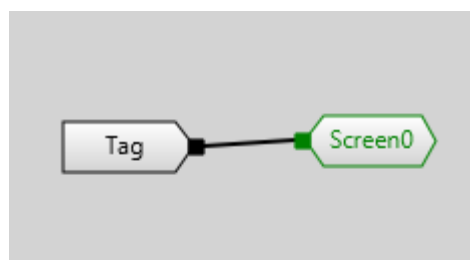
Tag	Tag2
255	0xFF

6.4.2.1.11 Call screen library



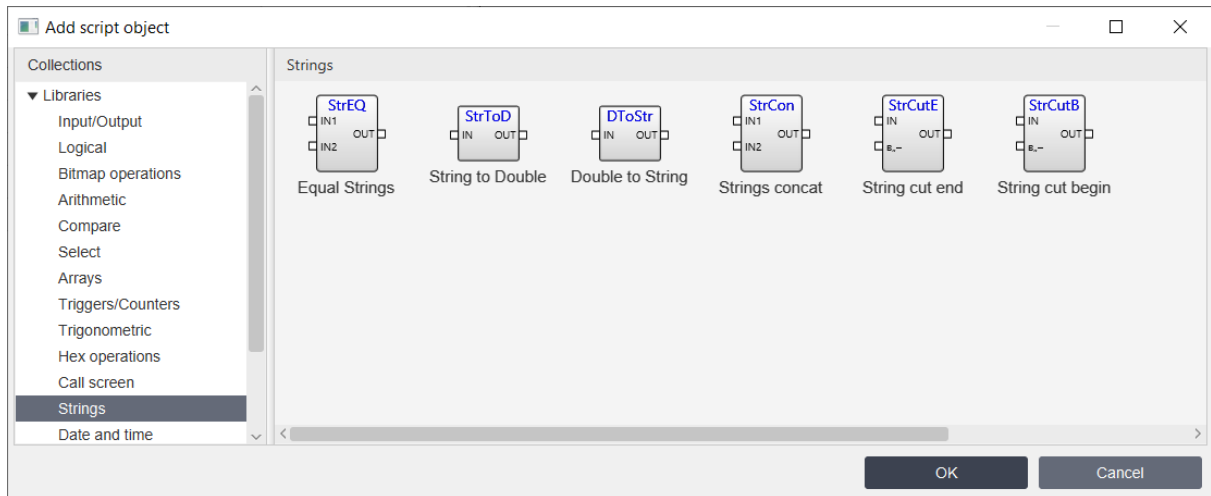
- **Call screen** - this script object used to call screen when Input's value turns from FALSE to TRUE.
- **Call popup** - this script object used to call popup screen when Input's value turns from FALSE to TRUE.

Example:



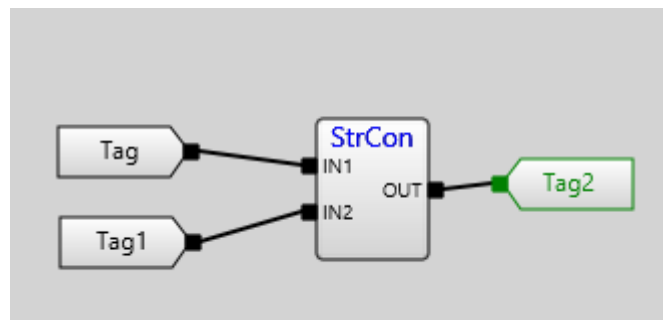
When Tag's value becomes TRUE from FALSE Screen0 will open.

6.4.2.1.12 Strings library



- **Equal Strings** - this script object compare two strings in Inputs and if their are equal it sets true into Output value.
- **String to Double** - this script object converts Input's string value into Output's double value.
- **Double to String** - this script object converts Input's double value into Output's string value.
- **Strings concat** - this script object concatenate Input's strings values into Output's string value. (Output = Input1+Input2).
- **String cut end** - this script object cuts end of Input's string value by the ? of characters and place result into Output's string value.
- **String cut begin** - this script object cuts begin of Input's string value by the ? of characters and place result into Output's string value.

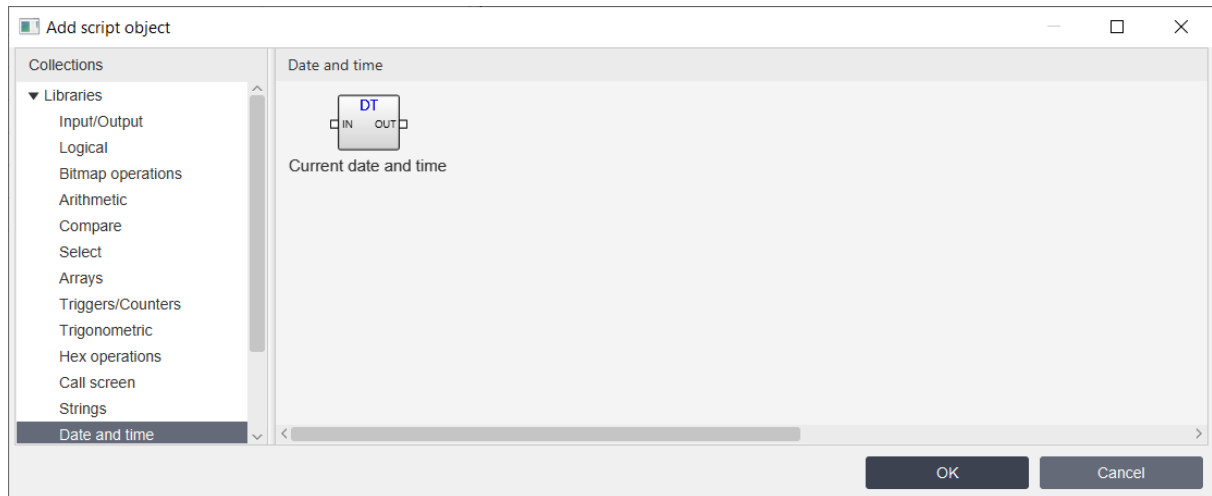
Example:



This operation concatenate Tag's value and Tag1's value and place result in Tag2's value.

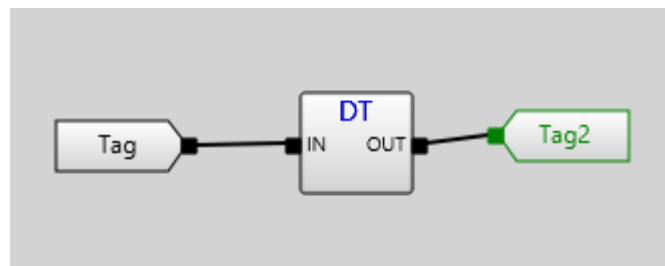
Tag	Tag1	Tag2
Hello	World	HelloWorld

6.4.2.1.13 Date and time library



- Current date and time - this script object used to get date and time components depending on Input value:
 - 0 - get seconds.
 - 1 - get minutes.
 - 2 - get hour of the day considering AM/PM.
 - 3 - get hour of the day.
 - 4 - get day of the week (1-Sunday, 2-Monday...).
 - 5 - get day of month.
 - 6 - get month (0 - January, 1 - February...).
 - 7 - get year.
 - 8 - get minutes of the day (hour*60 + minutes).

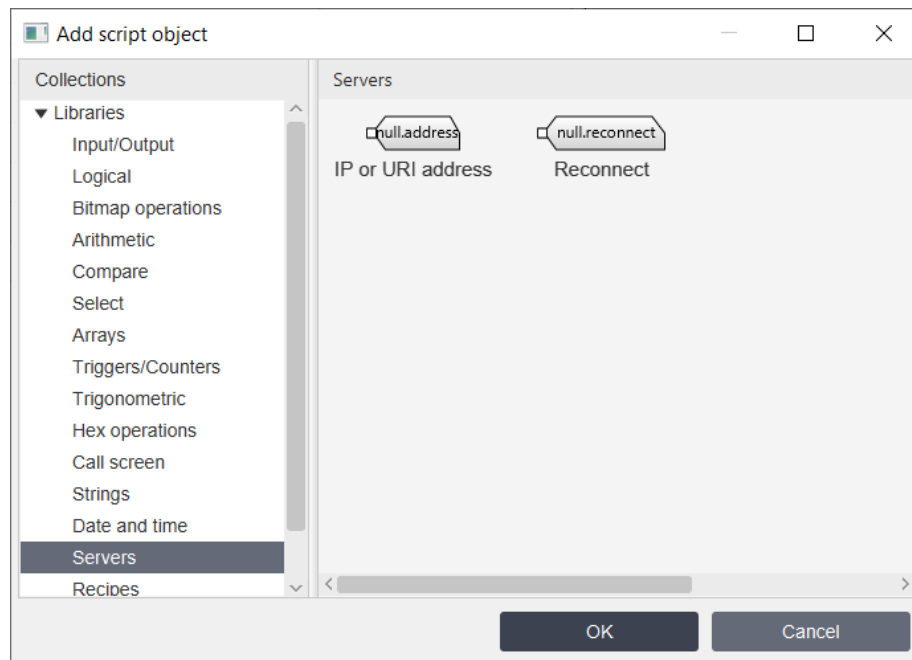
Example:



Depending on Tag's value place parameter of the current date and time.

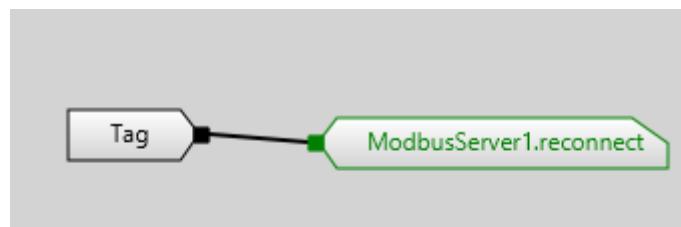
Tag	Tag2
7	2020

6.4.2.1.14 Servers library



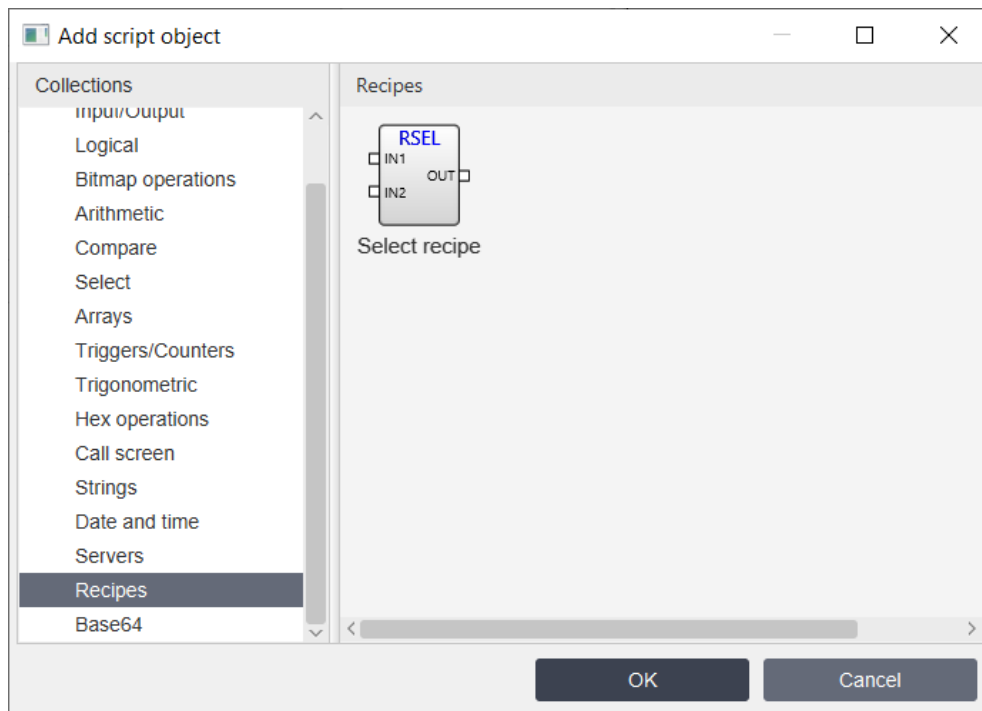
- **IP or URI address** - this script object used to change server's IP or URI address when Input's value changed.
- **Reconnect** - this script object used to reconnect server when Input's value turns from FALSE to TRUE.

Example:



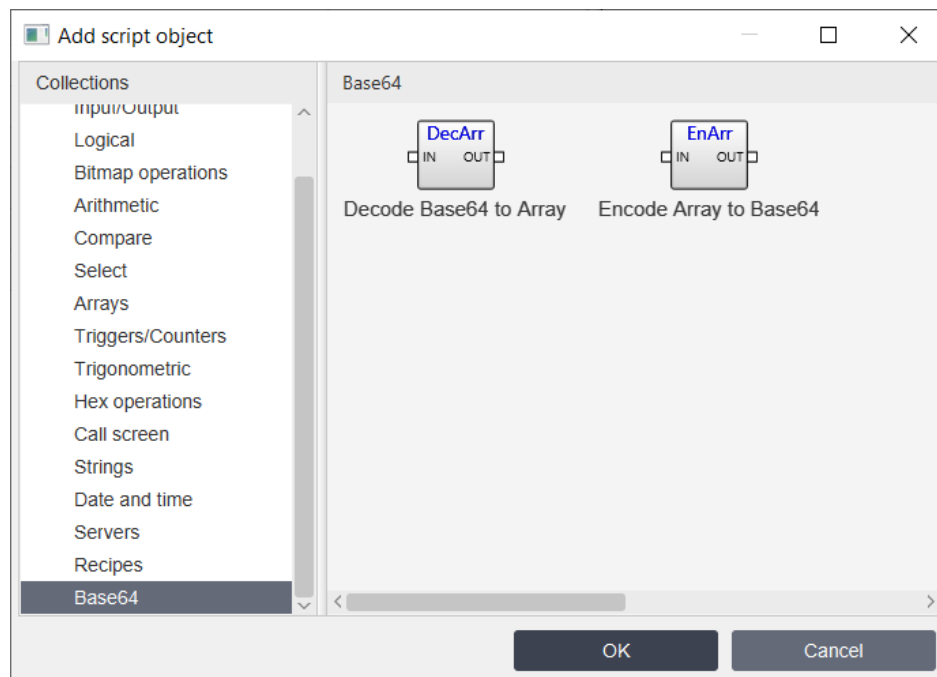
When Tag's value becomes TRUE from FALSE ModbusServer1 reconnect.

6.4.2.1.15 Recipes library



- **Select recipe** - this script object used to choose recipe row. Input2 is an input that contains name of the recipe. Input1 is number of the row (starting from 1). Output = true if recipe row is chosen.

6.4.2.1.16 Base64 library



- **Decode Base64 to Array** - this script object used to decode Base64 string to byte array. Input contains base64 encoded string. In Output will be decoded byte array.
- **Encode Array to Base64** - this script object used to encode byte array to Base64 string. Input contains byte array. In Output will be encoded Base64 string.

6.4.3 ST language

When you choose ST (Structured text) language in script properties and open this script you'll see two windows like in the picture:



Top window is a Code area and below window is a Debug(or log) area. You can enter your script program in the top window and compile this code by clicking [Run](#)⁷⁰ button on the [Toolbar](#)⁶⁹. All debug and log information you can see in the below window. Later in this chapter we will describe the rules of the ST language.

6.4.3.1 What is Structured Text Programming?

Structured Text for TeslaSCADA2 is different from PLC programming language defined by PLCOpen in IEC 61131-3. The programming language is text-based, compared to the graphicsbased Function Block Diagram. If you are already familiar with high-level programming languages like Java, PHP, Python and C, Structured Text will seem familiar to you. The syntax of Structured Text is developed to look like the syntax of a high-level programming language with loops, variables, conditions and operators. Before you read this tutorial I recommend that you take a brief look at this TeslaSCADA2 program written in Structured Text:

```
1 int a=5;
2 int b=7;
3 int c=0;
4 if (a>b){
5     c=a+b;
6 }
7 else{
8     c=a-b;
9 }
10 print(c);
```

Try to see if you can understand the function of this program. Does Structured Text look familiar to you?

6.4.3.2 Starting with the Syntax of Structured Text

The syntax of a programming language is the definition of how it is written. To be more precise, what symbols are used to give the language its form and meaning. As you can see in the example, Structured Text is full of colons, semicolons and other symbols. All these symbols have a meaning and are used to represent something. Some of them are operators, some are functions, statements or variables. All the details of the syntax will be explained as you move through this tutorial. But there are some general rules for the syntax of Structured Text you should know about. You don't have to memorize all the syntax rules for now, as you will when you get your hands into the programming:

All statements are divided by semicolons

Structured Text consists of statements and semicolons to separate them.

The language is case-sensitive

It is good practice to use upper- and lowercase for readability.

Spaces have no function

But they should be used for readability.

What's really important to understand here is that, when you write a TeslaSCADA2 program in IDE in Structured Text, your computer will translate that to a language the TeslaSCADA2 Runtime can understand. Before you use a project that contains the Structured Text TeslaSCADA2 program to your TeslaSCADA2 Runtime, the IDE will compile your program. This means that it will translate the code to a sort of machine code which can be executed by the TeslaSCADA2 Runtime. The compiler uses the syntax of the programming language to understand your program. For example: Each time the compiler sees a semicolon, it will know that the end of the current statement is reached. The compiler will read everything until it reaches a semicolon, and then execute that statement.

Comment Syntax

In textual programming languages you have the ability to write text that doesn't get executed. This feature is used to make comments in your code. Comments are good, and as a beginner you should always comment your code. It makes it easier to understand your

code later. In Structured Text you can make either one line comments or multiple line comments.

Single line comment:

```
//comment
```

Multiple line comment:

```
/* start comment
```

```
...
```

```
end comment */
```

6.4.3.3 Making Statements with Structured Text

So, Structured Text consists of statements. But what is statements? A statement tells the TeslaSCADA2 what to do. Let's take the first statement as an example:

bool x;

The compiler will read this as one statement, because when it reaches the semicolon, it knows that this is the end of that statement. Remember, statements are separated by semicolons. That's the main syntax rule of this language. In this statement you are telling the TeslaSCADA2 to create a variable called X and that variable should be a BOOL type. By default value of the variable is false.

6.4.3.4 Types in Structured Text

Data types of Structured Text are similar to data types of TeslaSCADA2:

Data Type	Format	Range
bool	Boolean	FALSE(0)/TRUE(1)
byte	Byte	-128 ... 127
short	Short	-32768 ... 32767
int	Integer	$-2^{31} \dots 2^{31}-1$
long	Long Integer	$-2^{63} \dots 2^{63}-1$
float	Float	$\pm 3.40282347\text{E}+38\text{F}$
double	Double	$\pm 1.79769313\text{E}+308$
string	Character string	"My string"
array	Array	byte[], short[], int[], float[]

Examples of variable initialisation:

```
bool x=false;
```

```
byte b = 2;
```

```
short s = 45;
```

```
int i = -4546;
```

```
long l = 394394832;
```

```
float f = 1.23;
```

```
double d = -545.64;
```

```
string str = "Hello";
```

```
byte bytes[10] = [1,2,3,4,5,6,7,8,9,10];
```

6.4.3.5 Operators and Expressions in STL

The next thing you should know about is operators. Operators are used to manipulate data and is a part of almost any programming language. This leads us to the second thing you should know about – expressions. Just like operators, expressions are a crucial part of programming languages. An expression is a construct that, when evaluated, yields a value. This means that when the compiler compiles an expression, it will evaluate the expression and replace the statement with the result. Take this example with the two variables A and B. A contains the value 10 and B contains 8.

A+B

The result of this expression is 18. So instead of A+B, the compiler will put in the value 18. An expression are composed of operators and operands. So what are operators and operands? Since, you just saw an example of an expression, you just saw both an operator and two operands. A and B are both operands and the + is an operator. Remember that operators are used to manipulate data. That is exactly what the + is doing. It is taking the value of the variable A and adding it to the value in B. The + is also called the addition operator because the operation is addition.

6.4.3.5.1 Operators

There are several operators available in Structured Text language:

Operation	Symbol	Precedence
Parenthesization	(expression)	Highest
Negation	-	
Complement	!	
Multiply	*	
Divide	/	
Modulo	%	
Add	+	
Subtract	-	
Left Shift	<<	
Right Shift	>>	
Comparison	<, >, <=, >=, ==, !=	
Boolean AND	&	
Boolean OR		
Boolean XOR	^	Lowest





All the operators in the table above are sorted after precedence. This is also called order of operations, and you may know about it from mathematics. The order of operations is the order in which the operations are executed or calculated. Just take a look at this expression:

A + B * C

How will this expression be evaluated by the compiler? There are two operations left: multiply and addition. But since multiply has a higher precedence, that will be the first to be evaluated. $B * C$ comes first and then the result is added to A . Every time an expression is evaluated, the evaluation follows the order of precedence as in the table above.

4 Types of Operators, 4 Types of Expressions

The operators used for expressions in Structured Text can be divided into four groups. Each group of operators will have its specific function and will yield a specific data type:

1. [Arithmetic Operators](#) 
2. [Relational Operators](#) 
3. [Logical Operators](#) 
4. [Bitwise Operators](#) 

6.4.3.5.1.1 Arithmetic Operators

All the arithmetic operators are often just called mathematical operators because they represent math. The result will always be the mathematical result of the expression.

- + (add)
- - (subtract/negate)
- * (multiply)
- / (divide)
- % (modulo divide)

Example:

15 % 4

Result:

3

6.4.3.5.1.2 Relational Operators

To compare or find a relation between two values you can use one of the relational operators. They are used for comparison and the result will be a boolean value (BOOL type), either TRUE(1) or FALSE(0).

- == (equal)
- < (less than)
- <= (less than or equal)
- > (greater than)
- >= (greater than or equal)
- != (not equal)

Example:

TEMPERATURE = 93.9;

TEMPERATURE >= 100.0;

Result:

false

6.4.3.5.1.3 Logical Operators

If you want to compare boolean values (BOOL) and make some logic out of it, you have to use logical operators. These operators also yields a boolean value of TRUE(1) or FALSE(0) as a result of the expression.

- &&
- ||
- ^
- !

Example:

```
limitswitch1 = true;  
limitswitch2 = false;  
limitswitch1 || limitswitch2
```

Result:

true

6.4.3.5.1.4 Bitwise Operators

The last group of operators are called bitwise operators because the operations are performed bitwise. It simply means that a logic operation is performed for each bit of two numbers. The result is a new number – the total result of the bitwise operations.

- &
- |
- ^
- <<
- >>

Example:

```
15 & 8
```

Result:

8

Since this operation is bitwise the calculation will be per bit. So to understand what's going on here, you have to convert the numbers to binary values:

15 = 1111 8 = 1000

Now each bit in the number 1111 (15) can be used in a logical operation with the other number 1000 (8): 1111 AND 1000

Bit number	1111 (15)	1000 (8)	Result
0	1	0	0
1	1	0	0
2	1	0	0
3	1	1	1

6.4.3.5.2 Operators and Statements

So, in the previous section you learned that expressions evaluate. Meaning that all expressions will yield the result and the compiler will replace the expression with the result. But what if you want the TeslaSCADA2 (compiler) not to evaluate something, but to DO something? Statements are the answer. Let's take a look at the actions or statements that you can make in Structured Text.

6.4.3.5.2.1 Assignment Statement and Operator

There are several statements available in Structured Text. All of them represent an action or a condition. Beginning with actions, the most fundamental statement in Structured Text is the assignment statement. Here's how an assignment statement looks like:

A = B;

What does this statement tell the compiler to do? To take the value of the variable B and put it in the variable A. The TeslaSCADA2 is assigning a value to a variable. Here's an even simpler example:

A = 10;

This statement will take the value 10 and put it into the variable A. Or said in another way – the variable A will be assigned the value 10. Since the value of A is now 10, we can make another statement, but this time with an expression:

B = A + 2;

When this line of code is compiled, the expression $A + 2$ will be evaluated to 12. The compiler will replace the expression with the result 12. The statement will now look like this to the compiler:

B = 12;

What will happen now, is that the compiler will assign the value 12 to the variable B. The last thing is that the = symbol is called the assignment operator. You can have all sorts of expressions in your assignment statements, from simple values like numbers to variables and functions. Because all expressions will be evaluated first, and then, the result of that evaluation will be used in the assignment statement.

6.4.3.5.2.2 Conditional Statements

The TeslaSCADA2 program is a piece of logic and therefore has to make some decisions. So in your TeslaSCADA2 program you need a way to make decisions. This brings us to conditional statements. Conditional statements are used for exactly that: To make decisions. There are one way of doing conditional statements in Structured Text: IF statement.

IF Statements

IF statements are decisions with conditions. There's a special syntax for IF statements. This means, that you have to write it in a certain way for the compiler to understand it. Because just like semicolons are used to end statements, there are special keywords to

make an IF statement. Here's how the syntax for IF statements looks like in STL for TeslaSCADA2:

```

if (boolean expression) {
  <statement>;
}
else if (boolean expression){
  <statement>;
} else {
  <statement>;
}

```

Statement starts with keyword IF. Then parentheses. Between those two brackets are the condition, which is an expression. But not just any expression. A boolean expression.

6.4.3.5.3 Boolean and Numeric Expressions

You can divide expressions into two groups depending on what they yield.

Boolean expressions evaluates to a BOOL type value, TRUE or FALSE.

Here's an example of a boolean expression:

```
1 == 1
```

This expression will evaluate to or yield TRUE(1). A boolean expression could also look like this:

```
1 > 2
```

But this time the boolean expression will evaluate to FALSE(0), since 1 is not larger than 2.

Numeric expressions evaluates to an integer or a floating point number.

A numeric expression could look as simple as this one:

```
13.2 + 19.8
```

This expression will evaluate to the floating point number 33.0, and therefore is a numeric expression.

Boolean expressions are used in IF statements as conditions. IF the boolean expression evaluates to TRUE, then the following statements will be executed. The TeslaSCADA2 will only execute the statements after the open bracket {, if the expression evaluates to TRUE. This is illustrated by the following example:

```

A = 0;
IF (A == 0) {
  B = 0;
}

```

Line number 3 will only be executed if A is equal to 0. In this case it will. A 0 is assigned to the variable A in a statement right before the IF statement. For now, you've seen a simple IF statement, where statements are only executed if an expression is TRUE. If that expression evaluates to FALSE the statements will simply not be executed. What to do if you want to

use multiple conditions? Just like most other programming languages you can use the ELSE IF and ELSE keywords for multiple conditions in the same IF statement. Both ELSE IF and ELSE are optional in IF statements, but this is how the syntax looks like:

```
if (boolean expression) {
    <statement>;
}
else if (boolean expression){
    <statement>;
} else {
    <statement>;
}
```

If the boolean expression on line 1 is FALSE, the statements below will simply not be executed. Instead the compiler will check the boolean expression after the ELSE IF keyword. Here it works just like with the IF keyword: If the boolean expression after the keyword is true, the following statements will be executed. At last is the ELSE keyword. It works as a default option for your IF statement. If all the IF and ELSE IF boolean expressions are evaluated to FALSE, the statements after the ELSE keyword will be executed.

Combining Operators for Advanced Conditions

Beside making multiple conditions you can also expand your conditions to include multiple variables. You can combine multiple expressions, typically done with a logical operator, to get a larger expression.



What if you want not just 1 but 2 inputs to be TRUE before an output is set. The expression would look like this:

```
if (INPUT1 & INPUT2) {
    OUTPUT1 = TRUE;
}
```

Now the expression will evaluate to TRUE, only if INPUT1 and INPUT2 is TRUE.

6.4.3.5.4 Iteration with Repeating Loops

Probably one of the most powerful features in Structured Text is the ability to make loops that repeat lines of code. In relation to TeslaSCADA2 programming loops can be used for many different purposes. You might have a function or a set of statements that you want to execute a certain amount of times or until something stops the loop. In Structured Text for TeslaSCADA2 you will find 2 different types of repeating loops:

- [FOR](#) 
- [WHILE](#) 

Common for all the types of loops is that they have a condition for either repeating or stopping the loop. The condition in FOR and WHILE loops decides whether the loop should repeat or not.

6.4.3.5.4.1 FOR Loops

The first loop is the FOR loop and is used to repeat a specific number of times. This is the syntax of FOR loops in Structured Text for TeslaSCADA2:

```
for (count = initial_value; condition; increment){  
  <statement>;  
}
```

Keyword that starts the **FOR** loop statement.

count = initial_value

This assignment operation is where you set the initial value you want to count from. Count is the variable name and initial_value is the value you want to start counting from.

;

Semicolon before condition statement.

condition of the loop's continuation.

;

Semicolon before incremental statement.

increment statement.

Usually used to increment initial value - count in this case. Then you place statements between {} that will execute during loops.

6.4.3.5.4.2 While Loops

The while loop is a little different from the FOR loop, because it is used to repeat the loop as long as some conditions are TRUE. A WHILE loop will repeat as long as a boolean expression evaluates to TRUE. Here's the syntax of WHILE loops:

```
while (boolean expression){  
  <statement>;  
}
```

Between the parentheses are the boolean expression. If that boolean expression evaluates to TRUE, all the statements between braces {} will be executed. When } is reached, the boolean expression will be evaluated again. This will happen over and over again until the expression doesn't evaluate to TRUE. But to make the loop stop at one point, you have to change a value in the boolean expression. Only in that way can the boolean expression go from TRUE to FALSE. Here's an example of a WHILE loop in Structured Text:

```
counter = 0;  
while (counter < 10){  
  counter = counter + 1;  
  machine_status = counter * 10;  
}
```

If you look at the third line you will see how the loop will eventually stop repeating. The boolean expression uses the counter variable and checks if its value is less than 10. But since the value of counter is set to 0 right before the WHILE loop, the boolean expression will be TRUE unless counter is changed. That is what's happening in line 3. This is the first statement in the WHILE loop, and with the other statements, are executed each time the loop repeats. In the third line the value of the counter variable is increased by 1. You can say

that the incremental value is 1. In the example above, the loop will repeat 10 times. When the value of count reaches 10, the boolean expression will be evaluated to FALSE (because 10 is not less than 10) and the loop will stop.

You can also use the BREAK keyword in the WHILE loop to stop repeating the loop before the boolean expression is FALSE. The syntax is an IF statement with the BREAK keyword in. Place it anywhere between braces {}.

```
if (boolean expression) {
    break;
}
```

6.4.3.6 User-defined functions

Also you can use user-defined functions in Structured Text language for TeslaSCADA2. You can find example below:

```
function fun(a,b){
int c;
if (a>b){
    c=a+b;
}
else{
    c=b-a;
}
return c;
}
int d = fun(13,17);
print(d);
```

In this example user function starts with key word **function**. Then name of the function. Then in parentheses arguments are listed. Inside braces {} statements of the function. User-defined function must be announced before main program. In this example program text of function **fun** is in the beginning. And only after statements of **fun** function, text of the main program. Results of this script will be **4** in the log window.

6.4.3.7 Using Tags in Structured Text

Of course for our purposes we need to use Tags in our scripts written in Structured Text language. How to do that? You can include Tags in your project's scripts by using keyword **Tags**. Then type dot (.) and name of your Tag. For possibility to compile this code the name of the tag should contain only English letters without white spaces and any signs.

Example:

```
int var = 10;
Tags.Tag1 = var;
```

In this example value of the variable **var** will be assigned to tag's value with name Tag1.

Other Example:

```
?oat f = Tags.Float1;
```

In this example value of the tag with name Float1 will be assigned to variable **f**.

Array Example:

```
byte bytes[10] = Tags.Array;
```

In this example value of the array tag with name Array will be assigned to the bytes array.

And you can use every element of the array for other operations. Like this:

```
for (int i=0;i<10;i++){
    print(bytes[i]);
}
```

6.4.3.8 Using Object property ?elds in Structured Text

You can include Object property ?elds in your project's scripts by using keyword **Objects**. Then type dot (.), name of your Object (for object type you can use keyword - **this**), again type dot (.) and name of property ?eld. For possibility to compile this code the name of the object and object property ?elds should contain only English letters without white spaces and any signs.

Example:

```
int width = 100;
Objects.Rectangle.width = var;
```

In this example value of the variable **var** will be assigned to Object with name **Rectangle** and ?eld property name **width**. Name of the property ?elds you can ?nd out in parentheses of object and property descriptions above.

Other Example:

```
Objects.this.?llcolor="0x66AA00FF";
```

Change color of the current object to which the script is attached. Color is represented in RGBA format. Where:

0x - Hex format of the color.

66 - Red color;

AA - Green color;

00 - Blue color;

FF - Transparency.

Also it's possible to use standard colours by using keyword **Color**.

Example:

```
Objects.Button.?llcolor=Color.BLUE;
```

List of colours:

Color	Code
Color.RED	"0xFF0000FF"
Color.BROWN	"0xA52A2AFF"
Color.GREEN	"0x00FF00FF"
Color.BLUEVIOLET	"0x8A2BE2FF"

Color	Code
Color.BLUE	"0x0000FFFF"
Color.CORAL	"0xFF7F50FF"
Color.AQUA	"0x00FFFFFF"
Color.CYAN	"0x00FFFFFF"
Color.AQUAMARINE	"0x7FFFD4FF"
Color.DARKBLUE	"0x00008BFF"
Color.AZURE	"0xF0FFFFFF"
Color.DARKCYAN	"0x008B8BFF"
Color.BLACK	"0x000000FF"
Color.DARKGREY	"0xA9A9A9FF"
Color.DARKGREEN	"0x006400FF"
Color.DARKORANGE	"0xFF8C00FF"
Color.DARKRED	"0x8B0000FF"
Color.DARKVIOLET	"0x9400D3FF"
Color.GOLD	"0xFFD700FF"
Color.GREY	"0x808080FF"
Color.INDIGO	"0x4B0082FF"
Color.IVORY	"0xFFFFF0FF"
Color.KHAKI	"0xF0E68CFF"
Color.LIGHTBLUE	"0xADD8E6FF"
Color.LIGHTCORAL	"0xF08080FF"
Color.LIGHTCYAN	"0xE0FFFFFF"
Color.LIGHTGREEN	"0x90EE90FF"
Color.LIGHTGREY	"0xD3D3D3FF"
Color.MAROON	"0x800000FF"
Color.NAVY	"0x000080FF"
Color.OLIVE	"0x808000FF"
Color.ORANGE	"0xFFA500FF"
Color.PINK	"0xFFC0CBFF"
Color.PURPLE	"0x800080FF"
Color.SILVER	"0xC0C0C0FF"
Color.VIOLET	"0xEE82EEFF"
Color.WHEAT	"0xF5DEB3FF"
Color.WHITE	"0xFFFFFFFF"
Color.YELLOW	"0xFFFF00FF"

6.4.3.9 Using Server parameter ?elds in Structured Text

You can include Server parameter ?elds in your project's scripts by using keyword Servers. Then type dot (.), name of your Server, again type dot (.) and name of parameter ?eld. For possibility to compile this code the name of the server and server parameter ?elds should contain only English letters without white spaces and any signs.

Example:

```
Servers.ModbusServer.ipaddress = "192.168.0.102";
```

In this example value "192.168.0.102" will be assigned to the server with name **ModbusServer** and ?eld property name **ipaddress**. Name of the property ?elds you can ?nd out in parentheses of server and parameter descriptions above. Also for parameters are written in descriptions you can use: **lostconnection**, **connect** and **connected**.

6.4.3.10 Using User parameter ?elds in Structured Text

You can include User parameter ?elds in your project's scripts by using keyword **Users**. Then type dot (.), name of your User or you can use key word **current** for choosing current user, again type dot (.) and name of parameter ?eld. For possibility to compile this code the name of the user and user parameter ?elds should contain only English letters without white spaces and any signs.





















Example:







```
Users.Operator.controlfunctions = true;
```

In this example value **true** will be assigned to the user with name **Operator** and ?eld property name **controlfunctions**. Name of the property ?elds you can ?nd out in parentheses of user and parameter descriptions above.

6.4.3.11 Embedded functions

In the Structured Text language for TeslaSCADA2 there are number of embedded functions. We grouped all functions in libraries:

- [Print](#) 
- [Arithmetic](#) 
- [Bitmap operations](#) 
- [Select](#) 
- [Trigonometric](#) 
- [Strings](#) 
- [Hex operations](#) 
- [Base64](#) 
- [Date and time](#) 
- [Server](#) 
- [Recipes](#) 
- [E-mail](#) 
- [Odoo ERP](#) 
- [Excel and screenshot](#) 
- [Database](#) 
- [HTTP](#) 
- [Global arguments](#) 
- [Tag properties](#) 
- [Dialog box](#) 
- [Trend's curve](#) 

- [Screen](#) 
- [Files](#) 
- [Common RTU](#) 
- [Call external software](#) 
- [User](#) 
- [Push](#) 

6.4.3.11.1 Print library

print(Input) - print *input* in the log.

Example:

```
print("Some message");
```

This function will print "Some message" in Debug window in TeslaSCADA IDE and in the log in TeslaSCADA2 Runtime.

6.4.3.11.2 Arithmetic library

sqrt(Input) - arithmetic operation square root of the input value.

pow(Input1, Input2) - arithmetic operation power for input values. output = $\text{Input1}^{\text{Input2}}$.

log(Input1, Input2) - arithmetic operation logarithm of the input value (Output = $\text{Log}_{\text{Input2}}\text{Input1}$).

ln(Input1) - arithmetic operation ln(natural logarithm) of the input value (Output = $\text{Ln}(\text{Input1})$).

abs(Input) - used to arithmetic operation absolute for input value (Output = $|\text{Input}|$).

sign(Input) - used to arithmetic operation sign for input value (Output = $-\text{Input}$).

int(Input) - used to arithmetic operation for getting integer part of the input value (Output = $\text{int}(\text{Input})$).

random(Input1, Input2) - arithmetic operation for generating random values in the range between Input1 and Input2.

Example:

```
int a = pow(5, 2);
```

```
print(a);
```

Response:

```
a = 25;
```

6.4.3.11.3 Bitmap operations library

bytestoshort(Input1, Input2) - used to pack 2 bytes into the short (Output = $\text{Input1} << 8 + \text{Input2}$).

bytestoint(Input1, Input2, Input3, Input4) - used to pack 4 bytes into the int (Output = $\text{Input1} << 24 + \text{Input2} << 16 + \text{Input3} << 8 + \text{Input4}$).

bytestofloat(Input1, Input2, Input3, Input4) - used to pack 4 bytes into the float (Output = $\text{IntToFloat}(\text{Input1} \ll 24 + \text{Input2} \ll 16 + \text{Input3} \ll 8 + \text{Input4})$).

bytestolong(Input1, Input2, Input3, Input4, Input5, Input6, Input7, Input8) - used to pack 8 bytes into the long (Output = $\text{Input1} \ll 56 + \text{Input2} \ll 48 + \text{Input3} \ll 40 + \text{Input4} \ll 32 + \text{Input5} \ll 24 + \text{Input6} \ll 16 + \text{Input7} \ll 8 + \text{Input8}$).

bytestodouble(Input1, Input2, Input3, Input4, Input5, Input6, Input7, Input8) - used to pack 8 bytes into the double (Output = $\text{LongToDouble}(\text{Input1} \ll 56 + \text{Input2} \ll 48 + \text{Input3} \ll 40 + \text{Input4} \ll 32 + \text{Input5} \ll 24 + \text{Input6} \ll 16 + \text{Input7} \ll 8 + \text{Input8})$).

shortstoint(Input1, Input2) - used to pack 2 shorts in the int (Output = $\text{Input} \ll 16 + \text{Input2}$).

inttoshort(Input1, Input2) - used to unpack int value into 2 shorts (Output = $\text{Input}[\text{Input2}]$).

inttobyte(Input1, Input2) - used to unpack int value into 4 bytes (Output = $\text{Input}[\text{Input2}]$).

floattobyte(Input1, Input2) - used to unpack float value into 4 bytes (Output = $\text{int}(\text{Input}[\text{Input2}])$).

longtobyte(Input1, Input2) - used to unpack long value into 8 bytes (Output = $\text{Input}[\text{Input2}]$).

doubletobyte(Input1, Input2) - used to unpack double value into 8 bytes (Output = $\text{long}(\text{Input}[\text{Input2}])$).

readbit(Input1, Input2) - used to read bit of the input value (Output = $\text{Input}[\text{Input2}]$).

setbit(Input1, Input2) - used to set bit of the input value (Output = $\text{Input} | 1 \ll \text{Input2}$).

resetbit(Input1, Input2) - used to reset bit of the input value (Output = $\text{Input} \& \sim(1 \ll \text{Input2})$).

Example:

```
int a = setbit(6, 0);  
print(a);
```

Response:

```
a = 7;
```

6.4.3.11.4 Select library

min(Input1, Input2) - used to select minimum value of Input2 and Input1 (Output=Min(Input, Input2)).

max(Input1, Input2) - used to select maximum value of Input2 and Input1 (Output=Max(Input, Input2)).

Example:

```
int a = max(6, 12);  
print(a);
```

Response:

```
a = 12;
```

6.4.3.11.5 Trigonometric library

toradians(Input) - used to convert degrees to radians.

todegrees(Input) - used to convert radians to degrees.

sin(Input) - used to calculate sin of Input value. (Output = sin(Input)).

cos(Input) - used to calculate cos of Input value. (Output = cos(Input)).

tan(Input) - used to calculate tag of Input value. (Output = tag(Input)).

asin(Input) - used to calculate arc sin of Input value. (Output = arc sin(Input)).

acos(Input) - used to calculate arc cos of Input value. (Output = arc cos(Input)).

atan(Input) - used to calculate arc tag of Input value. (Output = arc tag(Input)).

Example:

```
double angle = toradians(30);  
double a = sin(angle);  
print(a);
```

Response:

```
a = 0.5;
```

6.4.3.11.6 Strings library

stringequals(Input1, Input) - compare two strings in Inputs and if there are equals it returns true.

stringtodouble(Input) - converts Input's string value into double value.

doubletostring(Input) -converts Input's double value into string value.

stringtoint(Input) - converts Input's string value into integer value.

inttostring(Input) - converts Input's integer value into string value.

substring(Input1, Input2, Input3) - used to cut begin and end of Input1's string value by the ? of characters de?ned in Input2 and Input3.

cutbeginstring(Input1, Input2) - used to cut begin of Input1's string value by the ? of characters de?ned in Input2.

cutendstring(Input1, Input2) - used to cut end of Input1's string value by the ? of characters de?ned in Input2.

split(Input1, Input2, Input3) - used to split string in Input1 to string array. Input2 contains split regular expression; Input3 contains number of elements in array (if this number greater then number of elements that we get during operation, they will be ?lled by "")

Example:

```
split("hello;world", ";", 3);
```

Response:

```
string strarr[3] = ["hello", "world", ""];
```

Other Example:

```
string str = substring("Hello world", 2, 5);
```

```
print(str);
```

Response:

```
str = "llo";
```

6.4.3.11.7 Hex operations library

hextoint(Input) - converts hex value into integer.

inttohex(Input) - converts integer value into hex.

Example:

```
string a = inttohex(255);
```

```
print(a);
```

Response:

```
a = "ff";
```

6.4.3.11.8 Base64 library

base64decode(Input) - used to decode Base64 string to byte array. Input contains base64 encoded string. In Output will be decoded byte array.

base64encode(Input) - used to encode byte array to Base64 string. Input contains byte array. In Output will be encoded Base64 string.

Example:

```
int arr[5] = [1,2,3,4,5];
```

```
string a = base64encode(arr);
```

```
print(a);
```

Response:

```
a = "AQIDBAU=";
```

6.4.3.11.9 Date and time library

datetime(Input) - used to get date and time components depending on Input value:

- 0 - get seconds.
- 1 - get minutes.
- 2 - get hour of the day considering AM/PM.
- 3 - get hour of the day.
- 4 - get day of the week (1-Sunday, 2-Monday...).
- 5 - get day of month.
- 6 - get month (0 - January, 1 - February...).
- 7 - get year.
- 8 - get minutes of the day (hour*60 + minutes).

Example:

```
int a = datetime(7);  
print(a);
```

Response:

```
a = 2020;
```

currentdatetime(Input1) - used to get current date and time in string format. Input1 contains format of the date and time. Function returns formatted current date and time.

Example:

```
string date = currentdatetime("yyyy-MM-dd HH:mm:ss");
```

Response:

```
date = "2020-09-15 14:22:12"
```

currentdatetimeinmil() - used to get current date and time in milliseconds from 1 January 1970.

Example:

```
long date = currentdatetimeinmil();
```

Response:

```
date = 1627475044148
```

datetimefrom(Input1, Input2) - used to convert date time in milliseconds since 1 January 1970 into string format. Input1 contains format of the date and time. Input2 contains date time in milliseconds since 1 January 1970. Function returns formatted date and time in string.

Example:

```
string date = datetimefrom("yyyy-MM-dd HH:mm:ss", 1603713302140);
```

Response:

```
date = "2020-10-26 11:22:52"
```

datetimeto(Input1, Input2) - used to convert date time in string format into milliseconds since 1 January 1970. Input1 contains format of the date and time. Input2 contains date time in string format. Function returns time in milliseconds since 1 January 1970.

Example:

```
long date = datetimeto("yyyy-MM-dd HH:mm:ss", "2020-10-26 11:22:52");
```

Response:

```
date = 1603713302140
```

sleep(Input1) - used to make pause. Input1 contains time of the pause in milliseconds.

Example:

```
sleep(1000); //script sleeps 1000 ms.
```

6.4.3.11.10 Server library

reconnect(Input1, Input2) - used to reconnect to server with name from Input1 to IP address from Input2.

Example:

```
reconnect("ModbusServer1", "192.168.0.1");
```

Response:

Reconnect server with name **ModbusServer1** to IP address **192.168.0.1**.

opcua readattribute(Input1, Input2, Input3) - used to read attribute of the OPC UA server node. Input1 contains name of the server; Input2 contains name of the tag with de?ned Nodetd; Input3 contains number of the attribute. List of the attributes:

?	Attribute
1	Nodetd
2	NodeClass
3	BrowseName
4	DisplayName
5	Description
6	WriteMask
7	UserWriteMask

?	Attribute
8	IsAbstract
9	Symmetric
10	InverseName
11	ContainsNoLoops
12	EventNotifier
13	Value
14	DataType
15	ValueRank
16	ArrayDimensions
17	AccessLevel
18	UserAccessLevel
19	MinimumSamplingInterval
20	Historizing
21	Executable
22	UserExecutable

Example:

```
string description = opcuareadattribute("OPCUAServer", "tagname", 5);
```

connect(Input1) - used to connect to server with name from Input1.

Example:

```
connect("ModbusServer1");
```

disconnect(Input1) - used to disconnect to server with name from Input1.

Example:

```
disconnect("ModbusServer1");
```

6.4.3.11.11 Recipes library

selrecipe(Input1, Input2) - used to choose recipe row. Input2 is an input that contains name of the recipe. Input1 is number of the row (starting from 1). Output = true if recipe row is chosen.

Example:

```
selrecipe(2, "Recipe1");
```

Response:

Select row number **2** from recipe with name **Recipe1**.

6.4.3.11.12 E-mail library

sendemail(Input1, Input2) - send email (if it setup in [Project properties](#)^[111]) with subject from Input1 and message from Input2.

Example:

```
sendemail("Alarm", "Tag's alarm message");
```

Response:

Send E-mail to the addresses setup in project properties with subject "Alarm" and with body "Tag's alarm message".

setemailsubject(Input1) - set E-mail subject (if it setup in [Project properties](#)^[111]) from Input1.

Example:

```
setemailsubject("Alarm");
```

setnotificationpriority(Input1) - set notification priority from Input1. All event messages that have priority less then [Notifications\(Priority<\)](#)^[110] will be sent by E-mail, GSM modem, Telegram bot and arise alarm box.

Example:

```
setnotificationpriority(100);
```

setemailaddresses(Input1) - set E-mail addresses (if it setup in [Project properties](#)^[111]) from Input1. To which E-mail addresses the mail will be sent. Use commas to separate addresses.

Example:

```
setemailaddresses("email1@gmail.com , email2@gmail.com");
```

addemailrange(Input1, Input2, Input3, Input4) - add E-mail range for the client (if it setup in [Project properties](#)^[111]) and if "Depends on priority" is checked. Input1 contains name of the range. Input2 contains value of the range's start priority. Input3 contains value of the range's end priority. To which E-mail addresses the mail will be sent is placed in Input4. Use commas to separate addresses.

Example:

```
addemailrange("Emails" , 0, 100, "pochta@gmail.com");
```

removeemailrange(Input1) - remove E-mail range from the client (if it setup in [Project properties](#)^[111]) and if "Depends on priority" is checked. Input1 contains name of the range

Example:

```
removeemailrange("Emails" );
```

addemailtorange(Input1, Input2) - add E-mail address to the range of the client (if it setup in [Project properties](#)^[111]) and if "Depends on priority" is checked. Input1 contains name of the range. Input2 contains E-mail address.

Example:

```
addemailtorange("Emails" , "pochta@gmail.com");
```

removeemailfromrange(Input1, Input2) - remove E-mail address from the range of the client (if it setup in [Project properties](#)^[111]) and if "Depends on priority" is checked. Input1 contains name of the range. Input2 contains E-mail address.

Example:

```
removeemailfromrange("Emails" , "pochta@gmail.com");
```

6.4.3.11.13 Odoo ERP library

odoogetmodelcount(Input1, Input2, Input3) - used is to get number of rows that you get from Odoo ERP with name in Input1 (Example: "OdooERP0") and model with name in Input2 (Example: "mrp.workorder") that ?ts the ?lter in Input3. Filter is consisted with name of ?eld, comparison and value to compare separated by commas (Example:"state=,cancel" get rows where state == cancel).

Example:

```
Tags.orderscount = odoogetmodelcount("OdooERP0", "mrp.workorder", "state=,cancel");
```

odooreadmodel?eld(Input1, Input2, Input3, Input4, Input5) - used to read value of row's ?eld that you get from Odoo ERP with name in Input1 (Example: "OdooERP0") and model with name in Input2 (Example: "mrp.workorder") that ?ts the ?lter in Input3. Filter is consisted with name of the ?eld, comparison and value to compare separated by commas (Example:"state=,cancel" get rows where state == cancel). Name of the ?eld you have to enter in Input4 (Example:"production_id"). In Input5 you have to enter row position you want to read (Example:1).

Example:

```
Tags.Field = odooreadmodel?eld("OdooERP0", "mrp.workorder", "", "production_id", 1);
```

odoowritemodel?eld(Input1, Input2, Input3, Input4, Input5, Input6) - used to write value to the row's ?eld that you get from Odoo ERP with name in Input1 (Example: "OdooERP0") and model with name in Input2 (Example: "mrp.workorder") that ?ts the ?lter in Input3. Filter is consisted with name of the ?eld, comparison and value to compare separated by commas (Example: "state,=,cancel" get rows where state == cancel). Name of the ?eld you have to enter in Input4 (Example: "production_id"). In Input5 you have to enter row position you want to read (Example: 1). And in Input6 you have to enter value should be written (Example: "20"). If write is successful function return TRUE.

Example:

```
odoowritemodel?eld("OdooERP0","product.product", "id,=,2","list_price",0,Tags.Price);
```

odoocallfunction(Input1, Input2, Input3, Input4) - used to call function in Odoo ERP with name in Input1 (Example: "OdooERP0") and model with name in Input2 (Example: "mrp.workorder") with name in Input3 (Example: "action_toggle_is_locked"), and with parameter in Input4 (Example: 1). If call is successful function return TRUE.

Example:

```
odoocallfunction("OdooERP0","mrp.production","action_toggle_is_locked",Tags.ID);
```

6.4.3.11.14 Excel and screenshot library

saverecipeexcelreport(Input1, Input2, Input3, Input4) - used to save recipe report in Excel format bind to row. Input2 is an input that contains name of the recipe. Input1 is number of the row (starting from 1). Input3 contains ?le name of the report. Input4 contains title name. Output = true if recipe row is saved in Excel format. Report is saved in the folder you setup in Project properties->[Report folder](#)^[111].

Example:

```
saverecipeexcelreport(1,"RecExcel","streport","Title");
```

excelopenworkbook(Input1) - used to open excel workbook. Input1 contains name of the Excel file. Excel file is in the folder you setup in Project properties->[Report folder](#)^[111].

Example:

```
excelopenworkbook("reportfilename");
```

excelcreateworkbook() - this function create workbook for Excel ?le;

excelsaveworkbook(Input1) - used to save workbook in the Excel with name in Input1. Report is saved in the folder you setup in Project properties->[Report folder](#)^[111].

Example:

```
excelsaveworkbook("?lename");
```

excelcreatesheet(Input1) - create sheet in the workbook of Excel ?le with name in Input1.

Example:

```
excelcreatesheet("sheetname");
```

excelsetcolumnwidth(Input1, Input2, Input3) - set column width with name of the sheet in Input1, number of the column in Input2 and width in Input3.

Example:

```
excelsetcolumnwidth("sheetname", 0, 5000);
```

excelcreatestyle(Input1, Input2, Input3, Input4, Input5) - set cell style with name of the style in Input1, horizontal type in Input2 (can be "CENTER", "LEFT", "RIGHT"), vertical type in Input3 (can be "CENTER", "TOP", "BOTTOM"), font size in Input4 and bold or not in Input5.

Example:

```
excelcreatestyle("stylename", "CENTER", "CENTER", 14, false);
```

excelcreatecolorstyle(Input1, Input2, Input3, Input4, Input5, Input6) - set cell style with name of the style in Input1, horizontal type in Input2 (can be "CENTER", "LEFT", "RIGHT"), vertical type in Input3 (can be "CENTER", "TOP", "BOTTOM"), font size in Input4, bold or not in Input5 and color of the background in Input6 (can be "GREY", "GREEN", "RED", "BLUE", "YELLOW").

Example:

```
excelcreatecolorstyle("stylename", "CENTER", "CENTER", 14, false, "GREY");
```

excelcreatecell(Input1, Input2, Input3, Input4, Input5) - create cell with name of the sheet in Input1, number of the row in Input2 and position of the cell in Input3, style name in Input4 and text of the cell in Input5.

Example:

```
excelcreatecell("sheetname", 0, 0, "stylename", "Text");
```

excelreadcell(Input1, Input2, Input3) - read cell from the sheet with name in Input1, number of the row in Input2 and position of the cell in Input3.

Example:

```
String cellvalue = excelreadcel("sheetname", 0, 0);
```

excelcreatenumbercell(Input1, Input2, Input3, Input4, Input5, Input6) - create cell with name of the sheet in Input1, number of the row in Input2 and position of the cell in Input3, style name in Input4, numeric value in Input5 and decimal position for numeric value in Input6.

Example:

```
excelcreatenumbercell("sheetname", 0, 0, "stylename", Tags.Value, 2);
```

excelmergecells(Input1, Input2, Input3, Input4, Input5) - merge cells with name of the sheet in Input1, start row in Input2 and end row in Input3, start column in Input4 and end column in Input5.

Example:

```
excelmergecells("sheetname",0,1,0,1);
```

makescreenshot(Input1) - used to save screenshot with name in Input1. Screenshot is saved in the folder you setup in Project properties->[Report folder](#)^[111].

Example:

```
makescreenshot("?lename");
```

6.4.3.11.15 Database library

createdbsqliteconnection(Input1) - used to create connection to SQLite database with name in Input1. Database file is created in [DB](#)^[18] folder.

Example:

```
createdbsqliteconnection("?lename");
```

createdbconnection(Input1, Input2, Input3) - used to create connection to database with name in Input1, with username in Input2 and password in Input3.

Example:

createdbconnection("jdbc:mysql://192.168.0.76:3306/test", "username", "password"); in this example [MySQL](#)^[31] database is created. ("jdbc:mysql" in the beginning means that MySQL connection is created).

closedbconnection(Input1) - used to close database connection with name in Input1.

Example:

```
closedbconnection("?lename");
```

createdbtable(Input1, Input2, Input3) - used to create table in database with name of database in Input1, table name in Input2 and columns in Input3 (columns should be separated by commas, every table has auto incremented column "_id").

Example:

```
createdbtable("databasename", "tablename", "title, parameter1, parameter2");
```

insertvaluesintodb(Input1, Input2, Input3) - used to insert row into database with name of database in Input1, table name in Input2 and values in Input3 (values should be separated by commas).

Example:

```
insertvaluesintodb("databasename", "tablename", "Title, 10, 20");
```

readvaluefromdb(Input1, Input2, Input3, Input4) - used to read value from database with name of database in Input1, table name in Input2, name of the read column in Input3 and condition of read row in Input4 (if several rows ?t to condition ?rst row is read).

Example:

```
string parameter = readvaluefromdb("databasename", "tablename", "parameter1", "_id=1");
```

readvaluefromdbinpos(Input1, Input2, Input3, Input4, Input5) - used to read value from database with name of database in Input1, table name in Input2, name of the read column in Input3, condition of read row in Input4 and position of the row in Input5.

Example:

```
string parameter = readvaluefromdbinpos("databasename", "tablename", "parameter1", "title = Title", 1);
```

updatevalueindb(Input1, Input2, Input3, Input4, Input5) - used to update value in database with name of database in Input1, table name in Input2, name of the updated column in Input3, condition of the updated row in Input4 and updated value in Input5 (if several rows ?t to condition all rows values are changed)

Example:

```
updatevalueindb("databasename", "tablename", "parameter1", "title = Title", "10");
```

deleterowindb(Input1, Input2, Input3) - used to delete row(s) in database with name of database in Input1, table name in Input2 and condition that should ?t the row(s) in Input3.

Example:

```
deleterowindb("databasename", "tablename", "_id=1");
```

readvaluefromhistorydb(Input1, Input2, Input3, Input4, Input5) - used to read value from history database with name of history database in Input1, begin time in Input2, end time in Input3 (begin and end time in milliseconds since 1 January 1970 year, Input4 database name of the parameter to read, Input5 decimal position of the read value. If several rows ?t to time condition ?rst row is read.

Example:

```
string parameter = readvaluefromhistorydb("History DB0", 1636367879810, 1636367979810, "pressure", 2);
```

runsql(Input1, Input2) - used to execute SQL request with name of database in Input1 and SQL query in Input2.

Example:

```
runsql("databasename", "create table if not exists param (_id INTEGER PRIMARY KEY AUTOINCREMENT, temperature, pressure, humidity");
```

runsqlquery(Input1, Input2, Input3) - used to execute SQL request with name of database in Input1 and SQL query in Input2. Input3 contains name of the Result set (table). This Result set is place into global map where key is the name of the result set from the Input3.

Example:

```
runsqlquery("databasename", "select * from param", "resultname");
```

rsfirst(Input1) - used to move cursor of the result set (table) to the first row. Input1 contains name of result set. Return TRUE if the moving is successful. **This function doesn't work for SQL lite database.**

Example:

```
rsfirst("resultname");
```

rslast(Input1) - used to move cursor of the result set (table) to the last row. Input1 contains name of result set. Return TRUE if the moving is successful. **This function doesn't work for SQL lite database.**

Example:

```
rslast("resultname");
```

rsnext(Input1) - used to move cursor of the result set (table) to the next row. Input1 contains name of result set. Return TRUE if the moving is successful.

Example:

```
rsnext("resultname");
```

rsisempty(Input1) - used to check availability of the data in result set (table). Input1 contains name of result set. Return TRUE if the result set is empty. **This function doesn't work for SQL lite database.**

Example:

```
rsfempty("resultname");
```

rsmove(Input1, Input2) - used to move the cursor to position. Input1 contains name of result set. Input2 contains position value. Return TRUE if the moving is successful.

Example:

```
rsmove("resultname",3);
```

rsbeforefirst(Input1) - used to move cursor of the result set (table) to the position before the first row. Input1 contains name of result set. Return TRUE if the moving is successful. **This function doesn't work for SQL lite database.**

Example:

```
rsbeforefirst("resultname");
```

rsafterlast(Input1) - used to move cursor of the result set (table) to the position after last row. Input1 contains name of result set. Return TRUE if the moving is successful. **This function doesn't work for SQL lite database.**

Example:

```
rsafterlast("resultname");
```

rspos(Input1) - used to return the position of the cursor . Input1 contains name of result set. This function doesn't work for SQL lite database.

Example:

```
int pos = rspos("resultname");
```

rsreadstring(Input1, Input2) - used to read string value from the current cursor. Input1 contains name of result set. Input2 contains name of the column.

Example:

```
string name = rsreadstring("resultname","name");
```

rsreadstringnum(Input1, Input2) - used to read string value from the current cursor. Input1 contains name of result set. Input2 index of the column.

Example:

```
string name = rsreadstringnum("resultname",2);
```

rsreaddouble(Input1, Input2) - used to read double value from the current cursor. Input1 contains name of result set. Input2 contains name of the column.

Example:

```
double value = rsreaddouble("resultname","value");
```

rsreaddoublenum(Input1, Input2) - used to read double value from the current cursor. Input1 contains name of result set. Input2 index of the column.

Example:

```
double value = rsreaddoublenum("resultname",2);
```

rsreadint(Input1, Input2) - used to read int value from the current cursor. Input1 contains name of result set. Input2 contains name of the column.

Example:

```
int value = rsreadint("resultname","value");
```

rsreadintnum(Input1, Input2) - used to read int value from the current cursor. Input1 contains name of result set. Input2 index of the column.

Example:

```
int value = rsreadintnum("resultname",2);
```

rsreadbool(Input1, Input2) - used to read bool value from the current cursor. Input1 contains name of result set. Input2 contains name of the column.

Example:

```
bool value = rsreadbool("resultname","value");
```

rsreadboolnum(Input1, Input2) - used to read bool value from the current cursor. Input1 contains name of result set. Input2 index of the column.

Example:

```
bool value = rsreadboolnum("resultname",2);
```

rsgetcolnum(Input1) - used to get number of columns. Input1 contains name of result set.

Example:

```
int num = rsgetcolnum("resultname");
```

rsgetcol(Input1, Input2) - used to get column name from the result set. Input1 contains name of result set. Input2 index of the column. **This function doesn't work for SQL lite database.**

Example:

```
string name = rsgetcol("resultname",2);
```

rsremove(Input1) - used to remove result set from the global map memory. Input1 contains name of result set.

Example:

```
rsremove("resultname");
```

6.4.3.11.16 HTTP library

ifttttrigger(Input1, Input2, Input3, Input4, Input5) - used to send trigger event ifttt.com service. Input1 contains key; Input2 contains event trigger name; Input3, Input4, Input5 contain value1, value2 and value3 for ifttt.com service.

Example:

```
ifttttrigger("yourkey", "tag_trigger", "Tag is become true", Tags.Tag_2, "current value");
```

httppostcreate(Input1, Input2) - used to create HTTP post request. Input1 contains name of the request; Input2 contains url address.

Example:

```
httppostcreate("namehttppost",  
"https://hooks.zapier.com/hooks/catch/zapkey/otherzap/");
```

httppostaddvalue(Input1, Input2, Input3) - used to add value into HTTP post request. Input1 contains name of the request; Input2 contains name of the value; Input3 contains value.

Example:

```
httppostaddvalue("namehttppost", "valuenam", "value");
```

httppostexecute(Input1) - used to execute HTTP post request. Input1 contains name of the request. Function returns HTTP post response.

Example:

```
httppostexecute("namehttppost");
```

httppostgetvalue(Input1, Input2) - used to get value from the HTTP post response. Input1 contains response string; Input2 contains name of response value. Function returns value from the HTTP post response.

Example:

```
string value = httppostgetvalue("{valuenam: value}", "valuenam");
```

6.4.3.11.17 Global arguments library

TeslaSCADA IDE project has storage is RAM of the device with global arguments. You can add and get arguments by using [control property](#)³⁵⁹ of the button and functions described below:

getglobalargument(Input1, Input2) - used to get value from the global storage of the software. Input1 contains name of the value; Input2 contains default value, if the value is not available in the storage.

Example:

```
getglobalargument("value", "1");
```

putglobalargument(Input1, Input2) - used to put value into the global storage of the software. Input1 contains name of the value; Input2 contains value that will be written in the storage.

Example:

```
putglobalargument("value", "1");
```

6.4.3.11.18 Tag properties library

gettagvalue(Input1, Input2) - used to get value of the tag. Input1 contains name of the tag; Input2 contains default value, if the tag is not exist.

Example:

```
string value = gettagvalue("value", "1");
```

gettagvalueerror(Input1, Input2) - used to get value of the tag. Input1 contains name of the tag; Input2 contains name of the error tag. If tag with name in Input1 doesn't exist TRUE is placed in the tag with name in Input2.

Example:

```
string value = gettagvalueerror("value", "errortag");
```

settagvalue(Input1, Input2) - used to set value of the tag. Input1 contains name of the tag; Input2 contains value.

Example:

```
settagvalue("value", "1");
```

gettagdescription(Input1, Input2) - used to get description of the tag. Input1 contains name of the tag; Input2 contains default description, if the tag is not exist.

Example:

```
string description = gettagdescription("value", "description");
```

settagdescription(Input1, Input2) - used to set description of the tag. Input1 contains name of the tag; Input2 contains description.

Example:

```
settagdescription("value", "1");
```

gettagenablealarms(Input1) - used to get tag information about enable or not alarms. Input1 contains name of the tag.

Example:

```
bool enablealarm = gettagenablealarms("tagname");
```

settagenablealarms(Input1, Input2) - used to enable or disable alarms for the tag. Input1 contains name of the tag; Input2 contains value (true for enable or false for disable).

Example:

```
settagenablealarms("tagname", "true");
```

settagalarm(Input1, Input2, Input3) - used to enable or disable alarm for the tag. Input1 contains name of the tag; Input2 contains alarm's type("hihi", "hi", "lolo", "lo", "normal"); Input3 contains value (true for enable or false for disable).

Example:

```
settagalarm("tagname", "hihi", "true");
```

settagalarmlimit(Input1, Input2, Input3) - used to set alarm limit of the tag. Input1 contains name of the tag; Input2 contains alarm's type("hihi", "hi", "lolo", "lo"); Input3 contains limit's value.

Example:

```
settagalarmlimit("tagname", "hihi", 500);
```

settagalarmpriority(Input1, Input2, Input3) - used to set alarm priority of the tag. Input1 contains name of the tag; Input2 contains alarm's type("hihi", "hi", "lolo", "lo", "normal"); Input3 contains priority's value.

Example:

```
settagalarmpriority("tagname", "hihi", 500);
```

settagalarmmessage(Input1, Input2, Input3) - used to set tag's alarm message. Input1 contains name of the tag; Input2 contains alarm's type("hihi", "hi", "lolo", "lo", "normal"); Input3 contains message value.

Example:

```
settagalarmmessage("tagname", "hihi", "Value is to high");
```

settagalarmdeadband(Input1, Input2) - used to set tag's alarm deadband. Input1 contains name of the tag; Input2 contains deadband's value.

Example:

```
settagalarmdeadband("tagname", 0.5);
```

settagenablehistory(Input1, Input2) - used to enable or disable tag's history. Input1 contains name of the tag; Input2 contains value (true for enable or false for disable).

Example:

```
settagenablehistory("tagname", "true");
```

settagstorageperiod(Input1, Input2) - used to set tag's storage period. Input1 contains name of the tag; Input2 contains storage period's value.

Example:

```
settagstorageperiod("tagname", 1000);
```

settagstoreindb(Input1, Input2) - used to enable or disable tag's storage value in DB. Input1 contains name of the tag; Input2 contains value (true for enable or false for disable).

Example:

```
settagstoreindb("tagname", "true");
```

settaghistorydeadband(Input1, Input2) - used to set tag's history deadband. Input1 contains name of the tag; Input2 contains history deadband's value.

Example:

```
settaghistorydeadband("tagname", 1.0);
```

6.4.3.11.19 Dialog box library

infodialogbox(Input1, Input2) - used to call information dialog. Input1 contains title of the dialog box; Input2 contains message.

Example:

```
infodialogbox("Title", "Some message here");
```

setdialogbox(Input1, Input2, Input3, Input4) - used to call set tag's value dialog box. Input1 contains title of the dialog box; Input2 contains message, Input3 tag's name, Input4 contains value to set.

Example:

```
setdialogbox("Value set", "Set value", "FanStartRotation", "true");
```

6.4.3.11.20 Trend's curve library

addcurve(Input1, Input2, Input3, Input4, Input5, Input6, Input7, Input8) - used to add curve in the trend. Input1 contains name of the trend; Input2 contains name of the curve; Input3 contains name of the tag; Input4 contains line width of the curve; Input5 red part of the curve's color (0-255); Input6 green part of the curve's color (0-255); Input7 blue part of the curve's color (0-255); Input8 contain curve's type (0-3).

Example:

```
addcurve("Trend", "curve", "tagname", 2, 255, 255, 0, 1);
```

removecurve(Input1, Input2) - used to remove curve from the trend. Input1 contains name of the trend; Input2 contains name of the curve;

Example:

```
removecurve("Trend", "curve");
```

hidecurve(Input1, Input2, Input3) - used to hide or show curve on the trend. Input1 contains name of the trend; Input2 contains name of the curve; Input3 contains information about hide or not the curve in the trend.

Example:

```
hidecurve("Trend", "curve", true);
```

6.4.3.11.21 Screen library

callpopup(Input1) - used to call popup screen. Input1 contains name of the popup screen.

Example:

```
callpopup("Screen1").
```

callscreen(Input1) - used to call screen. Input1 contains name of the screen.

Example:

```
callscreen("Screen1").
```

closepopup() - used to close popup screen.

Example:

```
closepopup().
```

currentscreenname() - used to get current screen name.

Example:

```
string screenname = currentscreenname();
```

previouscreenname() - used to get previous screen name.

Example:

```
string screenname = previouscreenname();
```

6.4.3.11.22 Files library

createfile(Input1) - used to create file. Input1 contains path to the file. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB¹⁸](#) folder of the application. The function returns TRUE if the file is created.

Example:

```
bool created = createfile("filename.txt");
```

Response:

File is created in the [DB¹⁸](#) folder of the application.

Example:

```
bool created = createfile("D:/filename.txt");
```

Response:

File is created in the root of storage D.

deletefile(Input1) - used to delete file. Input1 contains path to the file. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB¹⁸](#) folder of the application. The function returns TRUE if the file is deleted.

Example:

```
bool created = deletefile("filename.txt");
```

Response:

File is deleted from the [DB¹⁸](#) folder of the application.

fileexists(Input1) - used to check file exist or not. Input1 contains path to the file. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB¹⁸](#) folder of the application. The function returns TRUE if the file is exist.

Example:

```
bool exist = fileexist("filename.txt");
```

Response:

Check the file with name "filename.txt" exist or not in the [DB](#)¹⁸ folder of the application.

filedatetime(Input1) - used to get time of the file creation. Input1 contains path to the file. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB](#)¹⁸ folder of the application. The function returns time of the file creation in milliseconds since 1 January 1970.

Example:

```
Tags.datetime = datetimefrom("yyyy-MM-dd HH:mm:ss",filedatetime(Tags.filename));
```

Response:

In the tag with name **datetime** we'll get date time of the file creation with name in the tag with name **filename**. (For example: "2020-10-26 12:12:34").

renamefile(Input1, Input2) - used to rename file. Input1 contains path to the file you want to rename. Input2 contains new path with new name of the file. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB](#)¹⁸ folder of the application. The function returns TRUE if the file is renamed successfully.

Example:

```
renamefile("filename.txt","D:/newfilename.txt");
```

copyfile(Input1, Input2) - used to copy file. Input1 contains path to the file you want to copy. Input2 contains path where you want to copy file. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB](#)¹⁸ folder of the application. The function returns TRUE if the file is copied successfully.

Example:

```
renamefile("filename.txt","D:/filename.txt");
```

openfile(Input1) - used to open file. Input1 contains path to the file you want to open. If path contains "/" it means we use the full path. If path doesn't contain "/" the file will be created in [DB](#)¹⁸ folder of the application. The function returns TRUE if the file is opened successfully.

Example:

```
openfile("filename.txt");
```

closefile() - used to close file. File opened by **openfile** command is closed.

Example:

```
closefile();
```

checkeof() - used to check end of file. File opened by **openfile** command is checked. Check the cursor at the end of file or not.

Example:

```
checkeof();
```

writeline(Input1) - used to write line into the file opened by **openfile** command. Input1 contains line is going to be written.

Example:

```
writeline("The line is written");
```

readline() - used to read line from the file opened by **openfile** command. The function returns line in string format.

Example:

```
string line = readline();
```

writebool(Input1) - used to write boolean value into the file opened by **openfile** command. Input1 contains boolean value is going to be written.

Example:

```
writebool(true);
```

readbool() - used to read boolean value from the file opened by **openfile** command. The function returns boolean value.

Example:

```
bool b = readbool();
```

writebyte(Input1) - used to write byte value into the file opened by **openfile** command. Input1 contains byte value is going to be written.

Example:

```
writebyte(-34);
```

readbyte() - used to read byte value from the file opened by **openfile** command. The function returns byte value.

Example:

```
byte b = readbyte();
```

writeshort(Input1) - used to write short value into the file opened by **openfile** command. Input1 contains short value is going to be written.

Example:

```
writeshort(934);
```

readshort() - used to read short value from the file opened by **openfile** command. The function returns short value.

Example:

```
short b = readshort();
```

writeint(Input1) - used to write int value into the file opened by **openfile** command. Input1 contains int value is going to be written.

Example:

```
writeint(-45934);
```

readint() - used to read int value from the file opened by **openfile** command. The function returns int value.

Example:

```
int b = readint();
```

writelong(Input1) - used to write long value into the file opened by **openfile** command. Input1 contains long value is going to be written.

Example:

```
writelong(8745934);
```

readlong() - used to read long value from the file opened by **openfile** command. The function returns long value.

Example:

```
long b = readlong();
```

writelfloat(Input1) - used to write float value into the file opened by **openfile** command. Input1 contains float value is going to be written.

Example:

```
writelfloat(8.34);
```

readfloat() - used to read float value from the file opened by **openfile** command. The function returns float value.

Example:

```
float b = readfloat();
```

writedouble(Input1) - used to write double value into the file opened by **openfile** command. Input1 contains double value is going to be written.

Example:

```
writedouble(9.14);
```

readdouble() - used to read double value from the file opened by **openfile** command. The function returns double value.

Example:

```
double b = readdouble();
```

writestring(Input1) - used to write string value into the file opened by **openfile** command. Input1 contains string value is going to be written.

Example:

```
writestring("Hello world");
```

readstring() - used to read string value from the file opened by **openfile** command. The function returns string value.

Example:

```
string str = readstring();
```

seek(Input1) - used to move cursor's position in the file opened by **openfile** command. Input1 contains offset of the cursor from the beginning.

Example:

```
seek(10);
```

getfilepos() - used to get cursor's position in the file opened by **openfile** command. The function returns cursor's position.

Example:

```
long pos = getfilepos();
```

filelength() - used to get length of the file opened by **openfile** command. The function returns length of the file in bytes.

Example:


```
long len = filelength();
```

saveproject(Input1) - used to save project to the file. Input1 contains name of the file (works only on desktop versions).

Example:


```
saveproject("filename.tsp2");
```

6.4.3.11.23 Report library

reporttopdf(Input1, Input2) - used to save report to PDF format file. Input1 contains name of the report. Input2 contains name of the pdf file. Report is saved in the folder you setup in Project properties->[Report folder](#) .

Example:

```
reporttopdf("Report1", "reportfile");
```

reporttoxls(Input1, Input2) - used to save report to Excel format file. Input1 contains name of the report. Input2 contains name of the Excel file. Report is saved in the folder you setup in Project properties->[Report folder](#) .

Example:

```
reporttoxls("Report1", "reportfile");
```

reporttofile(Input1, Input2) - used to save report to any format file. Input1 contains name of the report. Input2 contains name of the file. Report is saved in the folder you setup in Project properties->[Report folder](#)^[111]. Possible formats: pdf, xls, html, docx, csv, jpg, png, gif, rtf, pptx, ods, odt

Example:

```
reporttofile("Report1","reportfile.jpg");
```

reportsendbyemail(Input1, Input2, Input3, Input4) - used to send report by email. Input1 contains name of the report. Input2 contains name of the file saved and then send by e-mail. Report is saved in the folder you setup in Project properties->[Report folder](#)^[111]. Possible formats: pdf, xls, html, docx, csv, jpg, png, gif, rtf, pptx, ods, odt. Input3 contain subject of the E-mail message. Input4 body of the E-mail message. You [E-mail client](#)^[111] should be setup correctly.

Example:

```
reportsendbyemail("Report1","reportfile.jpg", "Report title", "Here's report from TeslaSCADA");
```

mergepdffiles(Input1, Input2, Input3, Input4, Input5) - used to merge several pdf files. Input1 contains name of the destination file. Input2-Input5 contain name of files to merge. Left "" if you need to merge less then 4 files.

Example:

```
mergepdffiles("Report","Report1", "Report2", "Report3", "Report4");
```

mergexlsfiles(Input1, Input2, Input3, Input4, Input5) - used to merge several xls files. Input1 contains name of the destination file. Input2-Input5 contain name of files to merge. Left "" if you need to merge less then 4 files.

Example:

```
mergexlsfiles("Report","Report1", "Report2", "Report3", "Report4");
```

6.4.3.11.24 Common RTU and TCP library

commonserverwrite(Input1, Input2) - used to write byte to the common server. Input1 contains name of the server. Input2 contains value to write.

Example:

```
commonserverwrite("CommonServer", 1);
```

commonserverwritearray(Input1, Input2) - used to write byte array to the common server. Input1 contains name of the server. Input2 contains array to write.

Example:

```
byte bytes[8] = [01,04,00,01,00,02,32,11];  
commonserverwritearray("CommonServer", bytes);
```

commonserverwritestring(Input1, Input2) - used to write string to the common server. Input1 contains name of the server. Input2 contains string to write.

Example:

```
commonserverwritestring("CommonServer", "Hello");
```

commonserverread(Input1) - used to read byte from the common server. Input1 contains name of the server.

Example:

```
int value = commonserverread("CommonServer");
```

commonserverreadarray(Input1) - used to read byte array from the common server. Input1 contains name of the server.

Example:

```
byte bytes[8] = [00,00,00,00,00,00,00,00];  
bytes = commonserverreadarray("CommonServer");
```

commonserverreadstring(Input1, Input2) - used to read string from the common server. Input1 contains name of the server.

If Input2 is true ENTER (/r/n) value is excluded.

Example:

```
string text = commonserverreadstring("CommonServer", true);
```

6.4.3.11.25 Call external software

callexternalsoftware(Input1) - used to call external software. Input1 contains command for calling external software. It depends on OS.

Examples:

- for MacOS: **callexternalsoftware("open /Applications/TextEdit.app");**
- for Windows: **callexternalsoftware("C:/Progra~1/somesoftware.exe");**

- for Android: **callexternalsoftware**("opc.tesla.scada"); (name of the Android application package)
- for iOS: **callexternalsoftware**("http://www.youtube.com/watch?v=VIDEO_IDENTIFIER"); (youtube scheme for calling in iOS)

callexternalsoftware2(Input1,Input2) - used to call external software. Input1 contains command for calling external software. Input2 separator for commands. It depends on OS.

Example:

- for Windows: **callexternalsoftware2**("C:/Progra~1/somesoftware.exe",",");

6.4.3.11.26 User library

adduser(Input1, Input2, Input3, Input4, Input5) - used to add User to the project. Input1 contains name of the user. Input2 contains password of the user. Input3 contains priority of the user. Input4 contains access level of the use. Input5 contains other (boolean) user properties. Input5 represented in Integer format, every bit of which is bound to property:

- 0 - Control functions.
- 1 - Acknowledge events.
- 2 - Delete events.
- 3 - Insert events.
- 4 - Insert history.
- 5 - Settings.
- 6 - Edit recipes.
- 7 - Save control operations.
- 8 - Can close.
- 9 - Can stop.

Example:

adduser("Operator", "111", 950, 200, 1023);

removeuser(Input1) - used to remove user from the project.

Example:

removuser("Operator");

6.4.3.11.27 Push library

sendpush(Input1, Input2) - send push notifications ([Push notifications](#)¹¹⁸) should be enabled and topic should be setup). Input1 contains title of the notification, Input2 contains message of the notification.

Example:

```
sendpush("Alarm", "Temperature is too high");
```

6.5 Tags

Create tag

To create a new tag select the menu item [Project](#)^[67] -> **New tag** or choose [Tags](#)^[81] tab on the Project Window, click right button on it and choose New tag item.

You'll see the tag properties window on tabs:

- [General](#)^[462] - general properties of the tag.
- [Scaling](#)^[472] - properties to setup scaling parameters.
- [Alarms](#)^[473] - properties to setup tag's alarms.
- [History](#)^[474] - properties to setup history parameters for collecting tag's value.
- [Script](#)^[477] - properties if you want to bind script to this tag.
- [Cloud](#)^[478] - properties for TeslaCloud tag representation.

Copy tag

To copy tag on [Tags](#)^[81] tab right click on the tag you want to copy and choose **Copy** tag item.

Delete tag

To delete tag on [Tags](#)^[81] tab right click on the tag you want to delete and choose **Delete** tag item.

Open tag properties

To open tag properties on [Tags](#)^[81] tab:

1. Double click on the tag properties which you want to open.
- or
2. Right click on the tag properties which you want to open and choose **Tag properties** item.

See **Project Window**->[Tags](#)^[81] tab for more information about possible operation with tags.

6.5.1 General tab

Tag properties

General

Scaling

Alarms

History

Script

Cloud

Group:

Subgroup:

Name:

Data type:

Number of elements:

1 element:

Access mode:

Initial PV:

Access level:

Input/Output

PV Input server:

PV Input tag:

☐ Output differs from Input:

PV Output server:

PV Output tag:

Description:

OK

Cancel





List of properties:

Property	Description
Group	Select group for the tag.
Subgroup	Select subgroup for the tag.
Name	Name of the tag. The name should be unique for the project. You can use indirect name by using group and subgroup names. To do this use curve braces {}. For example if group's name is "group" and subgroup's name is "1" you can enter {group}{subgroup}name and you'll get name of the tag is "group1name".
Data type	The user tells the program in what form to store information. When declaring a new variable, you must specify its type

Property	Description																																				
	<p>depending on the range of possible values that it can take. It is especially important to specify the correct data types in very large projects, as this will have a significant impact on performance. For example, for a variable that stores integer values from 0 to 100, correctly specify the Byte type instead of Integer. Although the program will work with both types, in the case of using the Byte type the variable will occupy 8 bits in memory, instead of 32 bits when using the Integer type.</p> <table><tr><th>Data type</th><th>Memory</th><th>Description</th><th>Range</th></tr><tr><td>Boolean</td><td>1 bit</td><td>Boolean True (1) or False (0) values</td><td>0...1</td></tr><tr><td>Byte</td><td>8 bit</td><td>Signed integers</td><td>-128...127</td></tr><tr><td>Short</td><td>16 bit</td><td>Signed integers</td><td>-32768...32767</td></tr><tr><td>Int</td><td>32 bit</td><td>Signed integers</td><td>-2147483648...2147483647</td></tr><tr><td>Long</td><td>64 bit</td><td>Signed integers</td><td>-9223372036854775808...9223372036854775807</td></tr><tr><td>Float</td><td>32 bit</td><td>Floating point numbers</td><td>1.18 x 10E-38...3.4 x 10E38</td></tr><tr><td>Double</td><td>64 bit</td><td>Floating point numbers</td><td>2.23 x 10E-308 ... 1.79 x 10E308</td></tr><tr><td>String</td><td>-</td><td>String</td><td></td></tr></table>	Data type	Memory	Description	Range	Boolean	1 bit	Boolean True (1) or False (0) values	0...1	Byte	8 bit	Signed integers	-128...127	Short	16 bit	Signed integers	-32768...32767	Int	32 bit	Signed integers	-2147483648...2147483647	Long	64 bit	Signed integers	-9223372036854775808...9223372036854775807	Float	32 bit	Floating point numbers	1.18 x 10E-38...3.4 x 10E38	Double	64 bit	Floating point numbers	2.23 x 10E-308 ... 1.79 x 10E308	String	-	String	
Data type	Memory	Description	Range																																		
Boolean	1 bit	Boolean True (1) or False (0) values	0...1																																		
Byte	8 bit	Signed integers	-128...127																																		
Short	16 bit	Signed integers	-32768...32767																																		
Int	32 bit	Signed integers	-2147483648...2147483647																																		
Long	64 bit	Signed integers	-9223372036854775808...9223372036854775807																																		
Float	32 bit	Floating point numbers	1.18 x 10E-38...3.4 x 10E38																																		
Double	64 bit	Floating point numbers	2.23 x 10E-308 ... 1.79 x 10E308																																		
String	-	String																																			

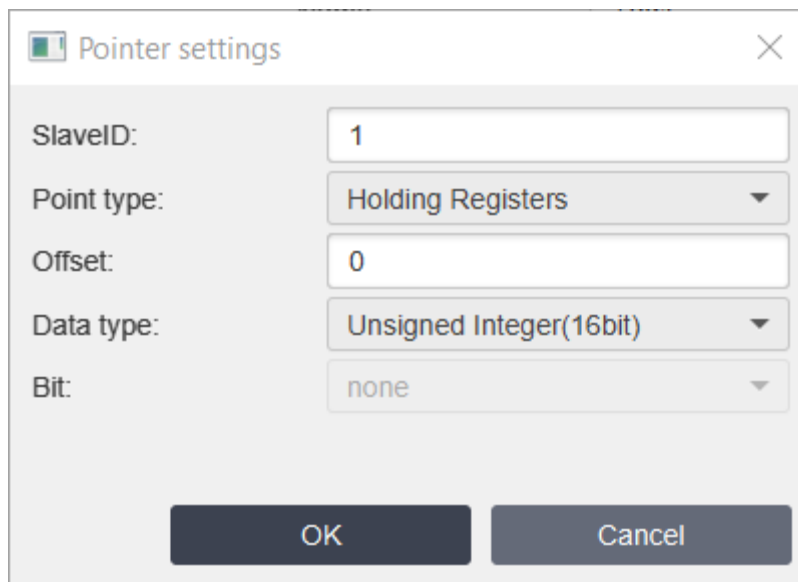
Property	Description								
	<table><tr><th>Data type</th><th>Memory</th><th>Description</th><th>Range</th></tr><tr><td>Array</td><td>-</td><td>Array of elements (Byte, Short, Int, Float).</td><td></td></tr></table>	Data type	Memory	Description	Range	Array	-	Array of elements (Byte, Short, Int, Float).	
Data type	Memory	Description	Range						
Array	-	Array of elements (Byte, Short, Int, Float).							
Number of elements	If you select String or Array data types enter number of elements (letters).								
1 element	If you select String or Array data types choose data type of 1 element (letter).								
Access mode	Select access mode for the tag: Read, Write or ReadWrite.								
Initial PV	Enter default tag's value into Initial PV. In the Initial PV field you can also use indirect values:{group}, {subgroup} and {name}.								
Access level	If tag's access level greater then access level of the current user the value couldn't be written to the current tag by this user.								
Input/Output	In the Input/Output section bind tag to the server's tag. In the PV Input server choose server you want to bind. Then click «...» button to set up server's tag settings or enter it into the PV Input tag. In the PV input tag you can use indirect values {group}, {subgroup} and {name}.								
Output differs from input	If the output server's tag differs from the input server's tag check Output differs from input and select PV Output server and enter PV Output tag. In the PV output tag you can use indirect values {group}, {subgroup} and {name}. When you check this property, you can force data to be written to the tag even when it does not differ from the previous one.								
Description	Description of the tag. In the description you can use indirect values {group}, {subgroup} and {name}.								

Depending on the type of PV Input server or PV Output server you'll see different server's tag (pointer) settings window:

- [Modbus tag settings](#) 
- [Siemens tag settings](#) 
- [Allen Bradley tag settings](#) 
- [Micrologix tag settings](#) 

- [OPC UA tag settings.](#) ⁴⁶⁸
- [MQTT tag settings](#) ⁴⁶⁹
- [Omron tag settings.](#) ⁴⁷⁰
- [BACnet tag settings.](#) ⁴⁷⁰
- [Raspberry GPIO settings](#) ⁴⁷¹

6.5.1.1 Modbus tag settings



List of properties:

Property	Description
SlaveID	SlaveID of Modbus device.
Point type	Point type of the register.
Offset	Offset of the Modbus register.
Data type	Data type of the Modbus pointer. The tag's data type overrides the data type of Modbus pointer during using in project.
Bit	Choose number of bit if the data type of the pointer is binary.

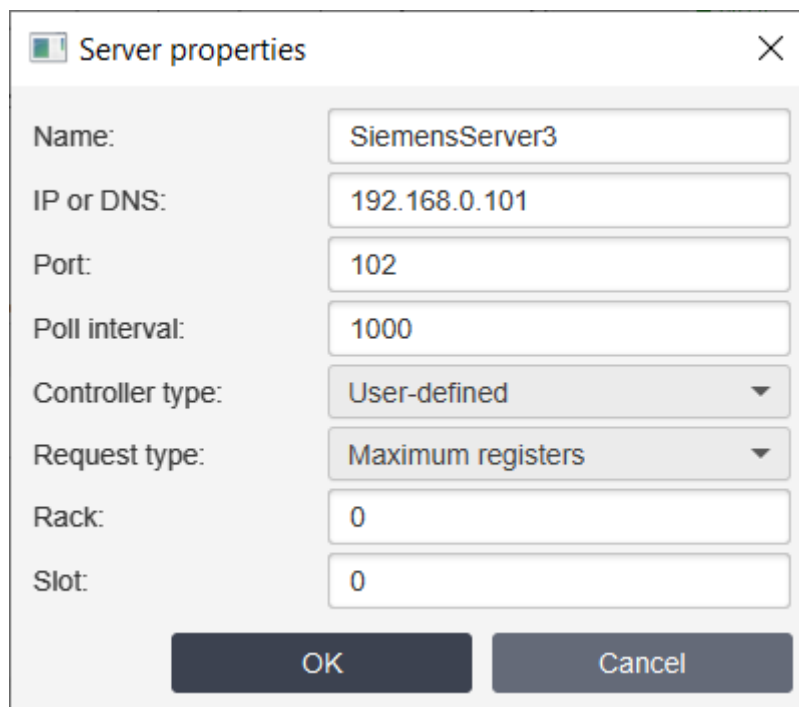
After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:

```
s=1;pt=3;o=0;dt=2;
```

where:

- **s** - SlaveID.
- **pt** - Point type.
- **o** - Offset.
- **dt** - Data type.

6.5.1.2 Siemens tag settings



Server properties

Name: SiemensServer3

IP or DNS: 192.168.0.101

Port: 102

Poll interval: 1000

Controller type: User-defined

Request type: Maximum registers

Rack: 0

Slot: 0

OK Cancel

List of properties:

Property	Description
Storage area	Choose storage area of the siemens tag: I,Q,M or DB.
DB?	Write DB number in the DB? ?eld if you choose DB storage area.
Data type	Data type of the Siemens pointer. The tag's data type overrides the data type of Siemens pointer during using in project.
Byte?	Enter byte number of the area into Byte? ?eld.
Bit	Choose number of bit if the data type of the pointer is Bit.

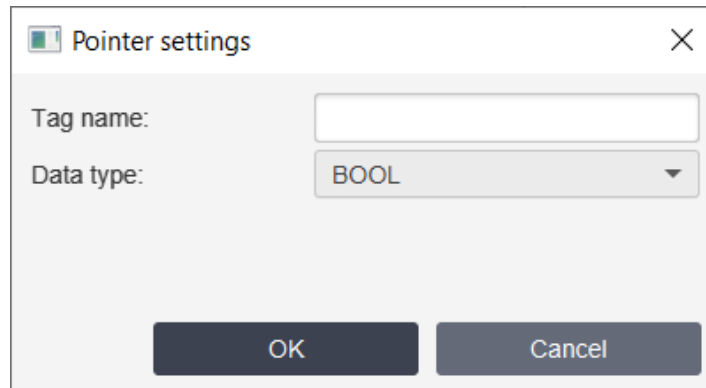
After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:
I0.0 [a=0;db=0;dt=0;bn=0;b=0;]

where:

- **a** - Storage area.
- **db** - DB? .
- **dt** - Data type.
- **bn** - Byte? .
- **b** - Bit

(I0.0 - its just for Siemens users and it's not used in encoding)

6.5.1.3 Allen Bradley tag settings



List of properties:

Property	Description
Tag name	Enter tag name.
Data type	Data type of the Allen Bradley pointer. The tag's data type overrides the data type of AB pointer during using in project.

After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:
type=0;name=Tag
where:

- **type** - Data type.
- **name** - Tag name.

6.5.1.3.1 Micrologix tag settings

If you choose Micrologix or SLC500 controller type in the Allen Bradley server settings you'll see the following window:

Pointer settings

File type: Output(O)

File number: 0

Element: 0

Word:

Bit: none

OK Cancel

List of properties:

Property	Description
File type	Choose file type of the server's tag.
File number	Write file number in the ?eld.
Element	Enter element of the servers tag.
Word	Choose word for some ?le types.
Bit	Choose number of bit.

After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:

00:0

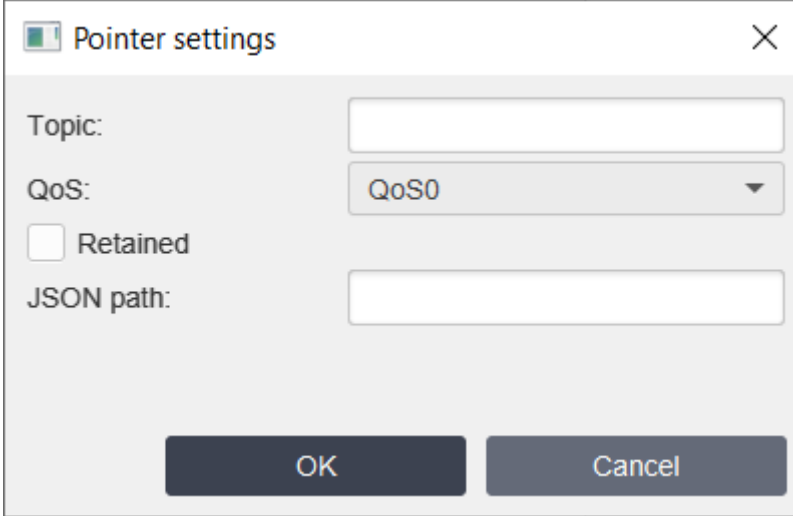
where:

- **0** - File type.
- **0** - File number.
- **0** - Element.

6.5.1.4 OPC UA tag settings

After clicking «...» button when you choose OPC UA server you'll get into the Address Space window. Browse through the address space by double clicking on the nodes and choose the tag(node) you need by clicking right button on it and choosing Select menu item on the popup window. You'll get NodeID in PV Input Tag.

6.5.1.5 MQTT tag settings



The screenshot shows a 'Pointer settings' dialog box with the following fields and controls:

- Topic:** A text input field.
- QoS:** A dropdown menu currently showing 'QoS0'.
- Retained:** An unchecked checkbox.
- JSON path:** A text input field.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

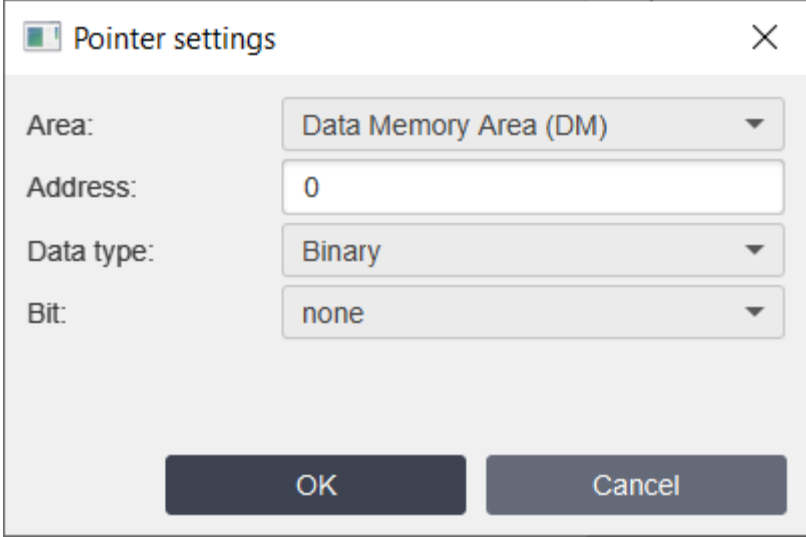
List of properties:

Property	Description
Topic	Topic of the MQTT server.
QoS	Choose QoS of the MQTT tag.
Retained	Check retained if you want to use this property.
JSON path	If MQTT response contains JSON array enter JSON path to parse the value. For example if response is: «{foo: bar, lat: 0.23443, long: 12.3453245}» to get long value enter «long» in the ?eld. If response is not JSON format left ?eld empty. If response contains multi dimension JSON format, separate keys by commas without blank spaces.

After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:
t=temperature;qos=0;r=1;json=
where:

- **t** - Topic.
- **qos** - QoS.
- **r** - Retained.
- **json** - JSON path.

6.5.1.6 Omron tag settings



List of properties:

Property	Description
Area	Choose address area.
Address	Address of the tag.
Data type	Data type of the Omron pointer. The tag's data type overrides the data type of Omron pointer during using in project.
Bit	Choose number of bit if the data type of the pointer is binary.

After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:
D00000 [a=0;ad=0;dt=16;]

where:

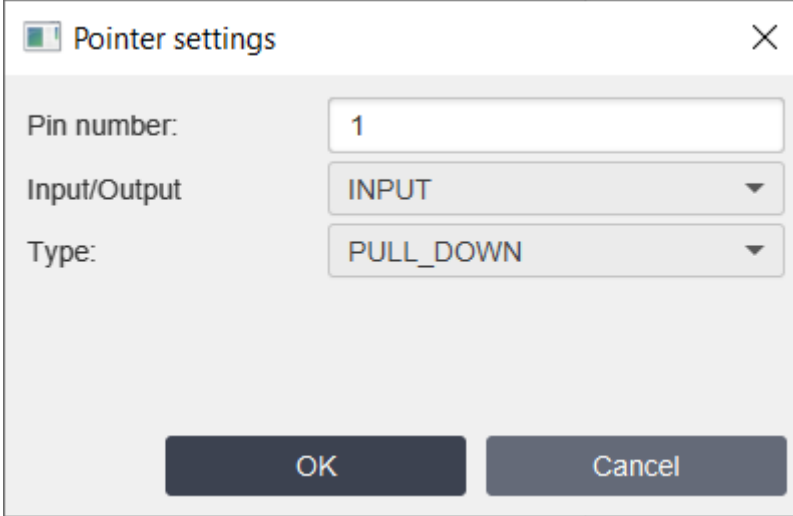
- **a** - Area.
- **ad** - Address.
- **dt** - Data type.
- **b** - Bit.

(D00000 - its just for Omron users and it's not used in encoding)

6.5.1.7 BACnet tag settings

After clicking «...» button when you choose BACnet server you'll get into the Address Space window. Browse through the address space by clicking on the remote devices and choose the object you need by clicking right button on it and choosing Select menu item on the popup window. You'll get object identifier in PV Input Tag.

6.5.1.8 Raspberry GPIO tag settings



Pointer settings

Pin number: 1

Input/Output: INPUT

Type: PULL_DOWN

OK Cancel

List of properties:

Property	Description
Pin number	Pin number of Raspberry PI GPIO.
Input/Output	Use contact as Input or Output.
Type	Type of the Input.

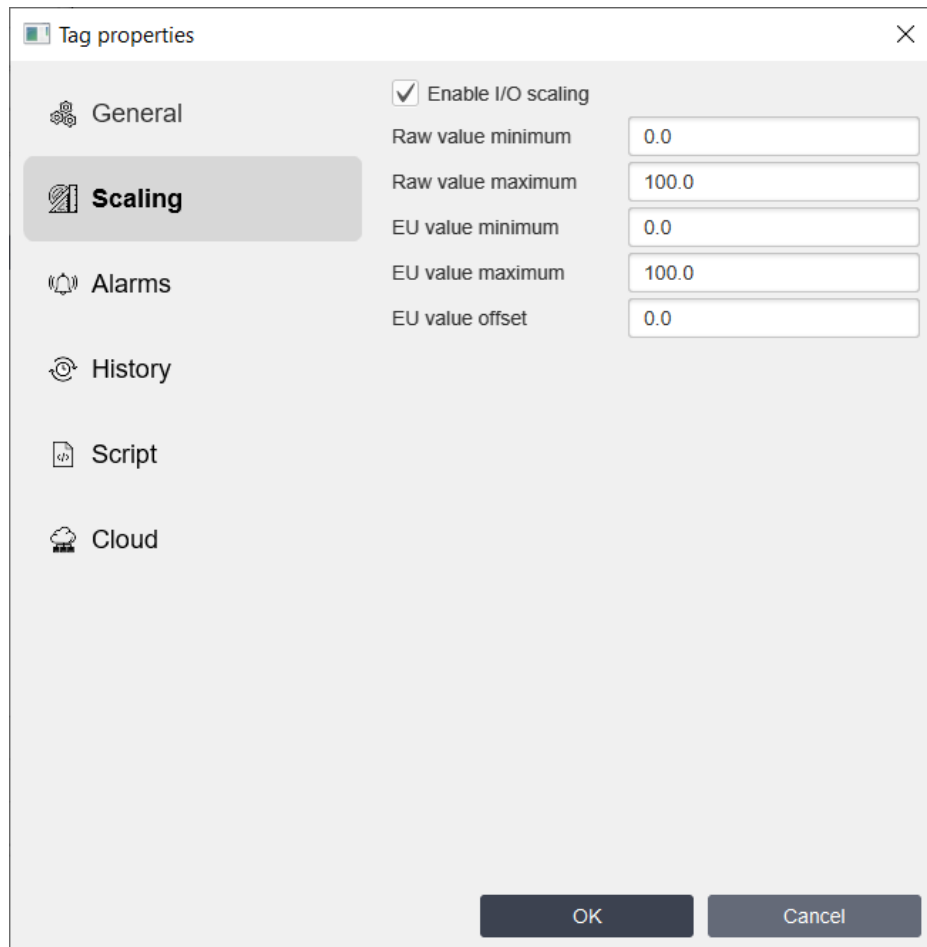
After clicking OK you'll get pointer settings in **PV Input tag** encoded in String like:

pin=3;o=0;t=1;

Where:

- **pin** - Pin number.
- **o** - Output or Input.
- **t** - Type.

6.5.2 Scaling tab



Tag properties

General

Scaling

Alarms

History

Script

Cloud

☒ Enable I/O scaling

Raw value minimum: 0.0

Raw value maximum: 100.0

EU value minimum: 0.0

EU value maximum: 100.0

EU value offset: 0.0

OK Cancel

List of properties:

Property	Description
Enable I/O scaling	Check it if you want to scale a value get from the server field.
Raw value minimum	Enter minimum server tag's value into this property field.
Raw value maximum	Enter maximum server tag's value into this property field.
EU value minimum	Enter minimum tag's value in engineer units into this property field.
EU value maximum	Enter maximum tag's value in engineer units into this property field.
EU value offset	Write tag's value offset in this property field.

When you get some value from the server application use this formula:

$$\text{value} = (\text{value} - \text{rawmin}) * (\text{eumax} - \text{eumin}) / (\text{rawmax} - \text{rawmin}) + \text{eumin} + \text{offset}$$

6.5.3 Alarms tab

Tag properties

General

Scaling

Alarms

History

Script

Cloud

☒ Enable alarms

☒ HiHi Limit 1.0 Priority 50
Message

☒ Hi Limit 0.0 Priority 500
Message

☒ Lo Limit 0.0 Priority 500
Message

☒ LoLo Limit 0.0 Priority 50
Message

☒ Normal Priority 900
Message

Deadband 0.0

☒ Enable OPC UA event

OK Cancel

List of properties:

Property	Description
Enable alarms	Check this property if you want to use alarms for this tag.
HiHi, Hi, Lo, LoLo, Normal	Check HiHi, Hi, Lo, LoLo or Normal if you want to use the correspondent alarm(event).
Limit	Write this property for the correspondent alarm(event). If the value of the tag plus Deadband will be more than HiHi or Hi limit the correspondent alarm will be called and be written into Events database ¹⁰⁷ . If the value of the tag minus Deadband will be less than LoLo or Lo limit the correspondent alarm will be raised and be written into Events database ¹⁰⁷ .
Priority	Enter this property for the correspondent alarm(event). If the priority of the alarm(event) is less than value of

Property	Description
	Notifications(Priority<) ¹¹⁰ you set in the project properties the notification dialog will be called.
Message	Enter this property for the correspondent alarm(event). In the message you can use indirect values {group}, {subgroup}, {name} and {description}. Also you can use keyword {value} for displaying current value.
Deadband	Hysteresis to avoid triggering an alarms when the tag value fluctuates slightly.
Enable OPC UA event	Check this property if you bind this tag to the OPC UA server tag(node) and you want to use EventNotifier of this tag(node).

6.5.4 History tab

Tag properties

General

Scaling

Alarms

History

Script

Cloud

☒ Enable history

Save period(ms) 10000

☒ Store in DB

☒ Use deadband

Deadband 0.0

OK Cancel

List of properties:

Property	Description
Enable history	Check this property if you want to use history for this tag.
Storage period(ms)	Enter period of saving values in operating memory or in general database that you can setup in Project properties- > Events/History tab ^[107] . For History DB ^[485] that are configured in Databases ^[88] tab it doesn't work. For History DB ^[485] you setup period of storage in its properties.
Store in DB	Check this property if you want to store data in general history database that you can setup in Project properties- > Events/History tab ^[107] . For History DB ^[485] that are configured in Databases ^[88] tab you have to add this tag in the Collection. To have possibility to add tag in the Collection of History DB you no need to check "Store in DB" property.
Use deadband	Check this property if you want to use

Property	Description
	hysteresis for storage history information. If the tag's value minus the last saved tag's value less than value set in Deadband property the tag's value will not be saved in the general database. This property works only for general database that you can setup in Project properties-> Events/History tab ¹⁰⁷ . For History DB it doesn't work.
Deadband	This property contains deadband (hysteresis) value.

6.5.5 Script tab

Tag properties

General

Scaling

Alarms

History

Script

Cloud

☒ Enable script

Script

Value

Type

Deadband

0.0

0.0

Tag.PV==Value

0.0

OK Cancel

List of properties:

Property	Description
Enable script	Check this property if you want to use script bind to this tag's value.
Script	Choose script you want to bind to this tag's value.
Value	Enter value you want to compare with current tag's value.
Type	Choose type of the compare operation. Script is executed when condition becomes TRUE from FALSE.
Deadband	Hysteresis for compare operation. If tag's value plus/minus deadband greater/less Value (depends on type of the compare operation) script will be executed.

6.5.6 Cloud

Tag properties

General

Scaling

Alarms

History

Script

Cloud

☒ Enable cloud

Type:

Unit:

Image:

Description:

Group:

Minimum:

Maximum:

Decimal position:

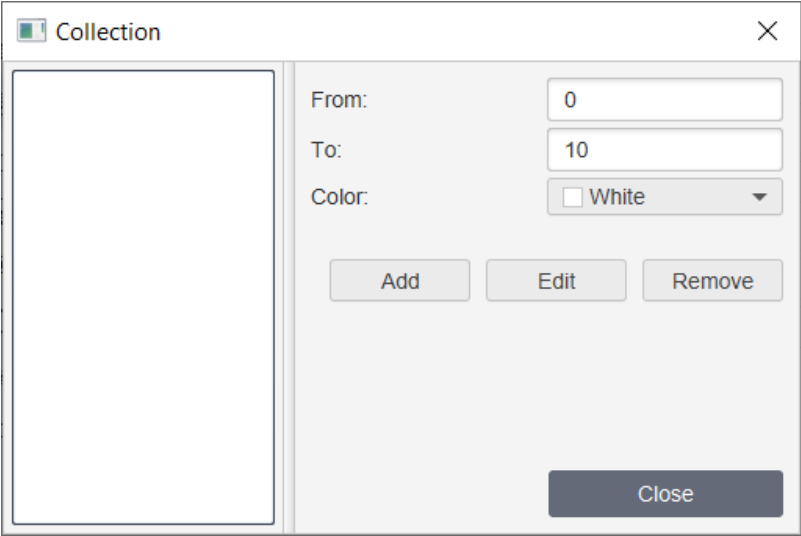
Color: Collection

Background color: Collection

OK Cancel

List of properties:

Property	Description
Enable cloud	Check this property if you want to use this tag on the cloud for web-browser.
Type	Type of the tag's card to represent this tag's value.
Unit	Unit of the tag's value.
Image	Icon image for the tag's card. You can choose it from the list or enter name from Material icons list .
Description	Description of the tag's card.
Group	Group of the tags. You can sort tags by these groups on the dashboard.

Property	Description
Minimum	Minimum of the tag's value. It's useful for some tag's cards.
Maximum	Maximum of the tag's value. It's useful for some tag's cards.
Decimal position	Decimal position of the tag's value.
Color	<p>Color of the tag's card elements on the dashboard: If you click Collection button. You'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none">• From - enter the value from which the object will have the color of the range.• To - enter the value to which the object will have the color of the range.• Color - choose color for this range. <p>You can Add, Edit or Remove collection element of color conditions.</p>
Background color	<p>Color of the tag's card background on the dashboard: If you click Collection button. You'll see the window:</p>

Property	Description
	<div><div><div>Collection</div><div><div></div><div>From: 0 To: 10 Color: <input type="checkbox"/> White</div><div>Add Edit Remove</div><div>Close</div></div></div><p>where:</p><ul style="list-style-type: none">• From - enter the value from which the object will have the color of the range.• To - enter the value to which the object will have the color of the range.• Color - choose color for this range.<p>You can Add, Edit or Remove collection element of color conditions.</p></div>

6.6 Users

Create user

User is not a mandatory element of the project. You can use or not use in it. To create a new user select the menu item [Project](#)⁶⁷ -> **New User** or choose Users in the Project Window -> [Users](#)⁸⁶, click right button on it and choose New User item.

You'll see the following window:

User properties

Name:

Password:

Access level:

☒ Control functions

☒ Acknowledge events

☒ Delete events

☒ Insert events

☒ Insert history

☒ Settings

☒ Edit recipes

☒ Can close

☒ Can stop

☒ Save control operation

Priority:

OK Cancel

List of properties:

Property	ST script field*	Description
Name	name	Name of the user.
Password	password	Write password for the current user.
Access level	accesslevel	Access level of the current user. Depending of this user can be restricted on writing values in some tag and opening some screens.
Control functions	controlfunctions	Check if you want the user can write values into the server's tags.
Acknowledge events	acknowledge events	Check if you want the user can acknowledge events in events database ^[107] .
Delete events	deleteevents	Check if you want the user can delete events from events database ^[107] .
Insert events	insertevents	Check if you want that during running application events are inserted into events database ^[107] when the user is logged in.
Insert history	inserthistory	Check if you want that during running application history information is inserted into

Property	ST script field*	Description
		history database ^[107] when the user is logged in.
Settings	settings	Check if you want the user can enter Settings menu of TeslaSCADA2 Runtime application.
Edit recipes	editrecipes	Check if you want the user can Add, Edit and Delete recipes ?elds.
Save control operation	savecontroloperations	Check if you want to save this user control operations in events database ^[107] . (it will be saved if you check Enable alarms in Tag properties)
Can stop	canstop	Check if you want to let this user to stop execution of the project.
Can close	canclose	Check if you want to let this user to close application - TeslaSCADA2 IDE or TeslaSCADA2 Runtime
Priority	priority	Priority of the user control operations events that will be save in event database ^[107] .

* This field is used in ST scripts. For example: **Users.Operator0.controlfunctions = 0**. After this script command is executed user with name **Operator0** can't write values in the tag.

Open user properties

To open user properties on [Users](#)^[86] tab:

1. Double click on the user properties which you want to open.
or
2. Right click on the user properties which you want to open and choose **User properties** item.

Copy user

To copy user on [Users](#)^[86] tab right click on the user you want to copy and choose **Copy user** item.

Delete user

To delete user on [Users](#)^[86] tab right click on the user you want to delete and choose **Delete user** item.

6.7 Databases

Create database

Database is not a mandatory element of the project. You can use or not use it in the project. Database consists of 3 types:

- [Recipe](#)^[483]
- [History](#)^[485]
- [Odoo ERP](#)^[488]

Open database properties

To open database properties on [Databases](#)^[88] tab:

1. Double click on the database properties which you want to open.
or
2. Right click on the database properties which you want to open and choose **Database properties** item.

Copy database


To copy database on [Databases](#)^[88] tab right click on the database you want to copy and choose **Copy database** item.

Delete database

To delete database on [Databases](#)^[88] tab right click on the database you want to delete and choose **Delete database** item.

6.7.1 Recipe

To create a new recipe select the menu item [Project](#)^[67] and [New Database](#)^[69] - >**Recipe** or choose [Databases](#)^[88] on the Project Window, click right button on it and choose **New Database>Recipe** item. You'll see the following window:

 Database properties

Name:

Recipe3

DB name:

recipes

Table name:

recipes3

Username:

Password:

Ingredients

Collection

OK

Cancel

List of properties:


Property	Description
Name	Name of the recipe.
DB name	Write name of the database for the current recipe. If you enter the simple name like recipes for example you will connect to the SQLite database. The SQLite database ?le .db will be created in /DB/ ^[18] folder. If you choose names beginning with jdbc:mysql: like jdbc:mysql://192.168.0.104:3306/test the application will connect to MySQL* ^[31] database. if you choose names beginning with jdbc:sqlserver: like jdbc:sqlserver://192.168.1.17:1433;databaseName=test where test name of the database you want to connect. The application will connect to MSSQL* ^[55] database. If you choose names beginning with jdbc:postgresql: like jdbc:postgresql://192.168.1.17:5432/test where test name of the database you want to connect. The application will connect to PostgreSQL* ^[58] database.
Table name	Write table name of the database for the recipe.
Username	Username if needed for MySQL databases.
Password	Password if needed for MySQL database.
Ingredients	Click Collection to ?ll up ingredients of the recipe. After clicking Collection button you'll see the following window:

Property	Description
	<div> <div>Collection</div> <div> <div></div> <div> <div>Tag: Tag1</div> <div>Name: Ingredient</div> <div>DB column name: ingredient0</div> <div>Unit:</div> <div> <div>Add</div> <div>Edit</div> <div>Remove</div> </div> <div>Close</div> </div> </div> </div> <p>where:</p> <ol style="list-style-type: none"> 1. Choose Tag you want to bind to the ingredient. 2. Enter Name of the ingredient. 3. Enter DB column name for the database. 4. Enter Unit of the DB ingredient.

* for mobile device is possible to use only SQLite databases.

6.7.2 History DB

To create a new history db select the menu item [Project](#)^[67] and [New Database](#)^[69] - >**History** or choose [Databases](#)^[88] on the Project Window, click right button on it and choose **New Database>History** item. You'll see the following window:

 History database properties

Name:

HistoryDB1

DB name:

historydatabase

Table name:

histories1

Username:

Password:

Storage type:

Time

Save period(ms)

10000

Tag:

Storage DB period:

Week

Archive since:

Never

Ingredients

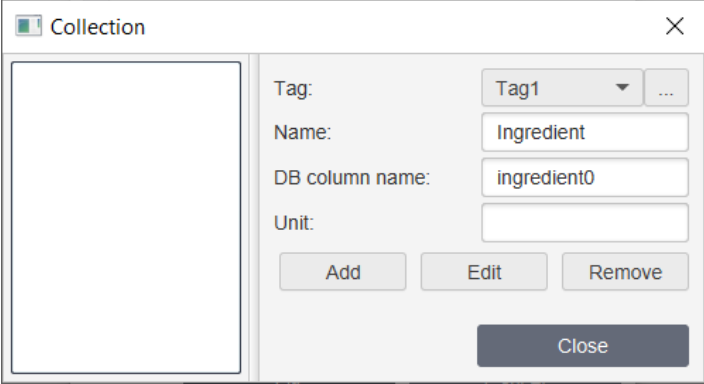
Collection

OK

Cancel

List of properties:

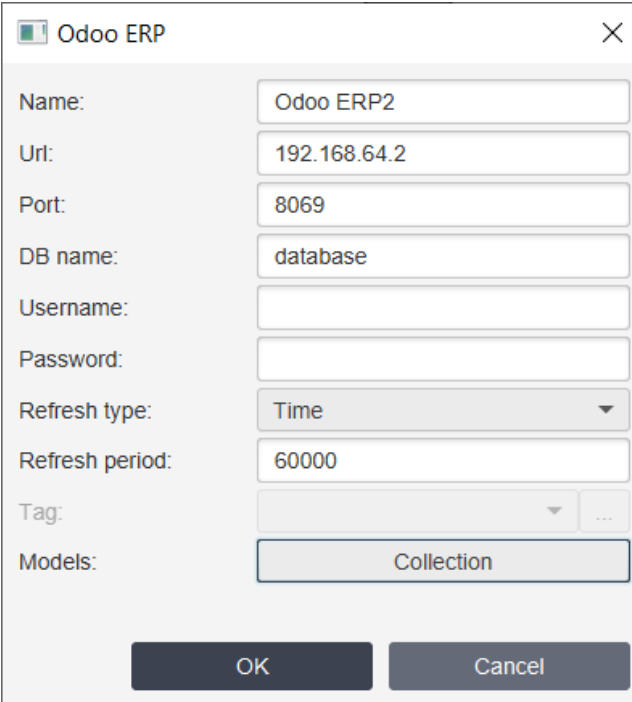
Property	Description
Name	Name of the history database.
DB name	Write name of the database for the current history. If you enter the simple name like hisstory for example you will connect to the SQLite database. The SQLite database ?le .db will be created in /DB/ folder. If you choose names beginning with jdbc:mysql: like jdbc:mysql://192.168.0.104:3306/test the application will connect to MySQL* database.if you choose names beginning with jdbc:sqlserver: like jdbc:sqlserver://192.168.1.17:1433;databaseName=test where test name of the database you want to connect. The application will connect to MSSQL* database. If you choose names beginning with jdbc:postgresql: like jdbc:postgresql://192.168.1.17:5432/test where test name of the database you want to connect. The application will connect to PostgreSQL* database.

Property	Description
Table name	Write table name of the database for the recipe.
Username	Username if needed for MySQL databases.
Password	Password if needed for MySQL database.
Storage type	Choose storage type - Time or Tag. If you chose Time every Save period values of tags included in Ingredients will be saved into history database. If you choose Tag values of tags will be saved when Tag's value become True(1).
Archive since	Select an archive period. The data collected before the archive period is stored in the archive database. The data collected for the selected period is stored in the main database. This improves performance when querying the underlying database.
Save Period(ms)	Time interval of saving Ingredients tag values into history database. This property used when you choose Time Storage type.
Tag	Choose Tag dependent on which value (when value become True(1)) Ingredients tag values will be saved in history database.
Ingredients	<p>Click Collection to ?ll up ingredients of the history. After clicking Collection button you'll see the following window:</p>  <p>Where:</p> <ol style="list-style-type: none"> 1. Choose Tag you want to bind to the ingredient. 2. Enter Name of the ingredient. 3. Enter DB column name for the database. 4. Enter Unit of the DB ingredient.

*** for mobile device is possible to use only SQLite databases.**

6.7.3 Odoo ERP

To create a new Odoo ERP connection (we've tested it only with Odoo 12 and Odoo14 version. To work with new versions Odoo (13, 14) you have to use TeslaSCADA2 starting from version 2.45.1) select the menu item [Project](#)^[67] and [New Database](#)^[69] ->Odoo ERP or choose [Databases](#)^[88] on the Project Window, click right button on it and choose New Database>Odoo ERP item. You'll see the following window:

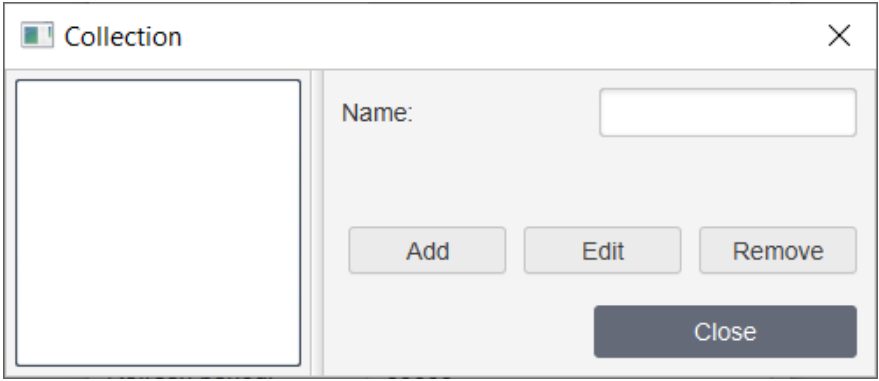


The screenshot shows a dialog box titled "Odoo ERP" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Name:** Text input field containing "Odoo ERP2".
- Url:** Text input field containing "192.168.64.2".
- Port:** Text input field containing "8069".
- DB name:** Text input field containing "database".
- Username:** Text input field (empty).
- Password:** Text input field (empty).
- Refresh type:** Dropdown menu with "Time" selected.
- Refresh period:** Text input field containing "60000".
- Tag:** Dropdown menu (empty) with a "..." button next to it.
- Models:** Text input field containing "Collection".
- Buttons:** "OK" and "Cancel" buttons at the bottom.

List of properties:

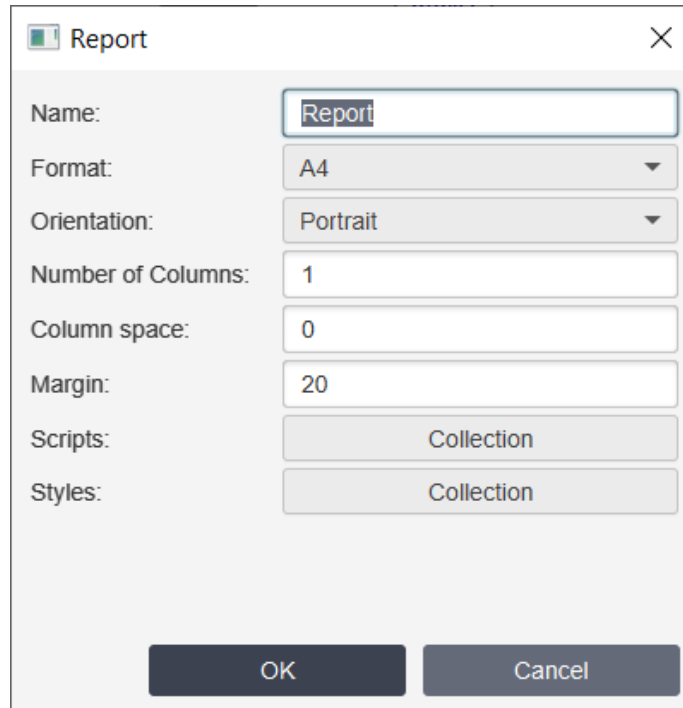
Property	Description
Name	Name of the Odoo ERP connection.
Url	Url of the Odoo ERP.
Port	Port of the Odoo ERP.
DB name	Name of the Odoo ERP database.
Username	Username for connecting to the Odoo ERP databases.
Password	Password for connecting to the Odoo ERP database.
Refresh type	Choose Refresh type to renew data information.
Refresh period(ms)	Refresh period of Odoo ERP information.

Property	Description
Tag	Choose Tag dependent on which value (when value become True(1)) Odoo ERP information is refreshed.
Models	<p>Click Collection to ?ll up model names of the Odoo ERP. After clicking Collection button you'll see the following window:</p>  <p>where:</p> <ol style="list-style-type: none"> 1. Name of the model.

6.8 Reports

Create report

To create a new report select the menu item [Project](#)^[67] -> **New Report** or choose [Reports](#)^[89] on the Project Window, click right button on it and choose **New Report** item. You'll see the [report properties](#)^[491] window:

A screenshot of a 'Report' dialog box. The dialog has a title bar with a green icon and the text 'Report' and a close button. It contains several fields: 'Name' with a text box containing 'Report'; 'Format' with a dropdown menu showing 'A4'; 'Orientation' with a dropdown menu showing 'Portrait'; 'Number of Columns' with a text box containing '1'; 'Column space' with a text box containing '0'; 'Margin' with a text box containing '20'; 'Scripts' with a button labeled 'Collection'; and 'Styles' with a button labeled 'Collection'. At the bottom are 'OK' and 'Cancel' buttons.

Name:	Report
Format:	A4
Orientation:	Portrait
Number of Columns:	1
Column space:	0
Margin:	20
Scripts:	Collection
Styles:	Collection

OK Cancel

Open report

To open report on [Reports](#)^[89] tab of the Project window:

- Right click on the report you want to open and choose **Open** item.
- or
- Double click on the report you want to open.

Copy report

To copy report on [Reports](#)^[89] tab of the Project window right click on the report you want to copy and choose **Copy** item.

Delete report

To delete report on [Reports](#)^[89] tab of the Project window right click on the report you want to delete and choose **Delete** item.

Open report properties

To open [report properties](#)^[491] on [Reports](#)^[89] tab of the Project window right click on the report you want to open and choose **Report properties** item.

Export report

To export report on [Reports](#)^[89] tab of the Project window:

1. Right click on the report you want to export and choose **Export report** item.

2. Now select the location and click the button **Save** (TeslaSCADA2 screen extension .tsp2report).

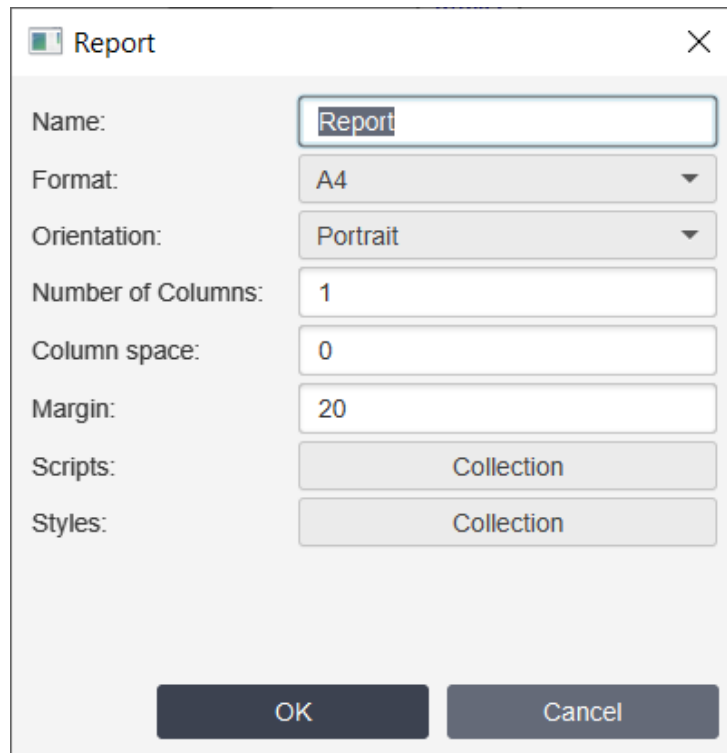
Import report

To import report on [Reports](#) ⁸⁹ tab of the Project window:

1. Right click on the report window and choose **Import report** item.
2. Now select the report file and click **Open** (TeslaSCADA script extension .tsp2report).

See **Project Window**->[Reports](#) ⁸⁹ tab for more information about possible operation with reports.

6.8.1 Report properties

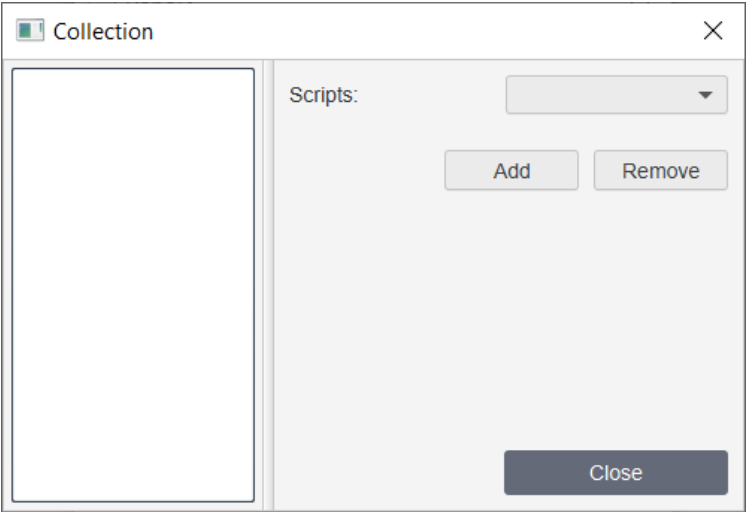







The screenshot shows a 'Report' dialog box with the following fields and options:

- Name:** A text input field containing 'Report'.
- Format:** A dropdown menu set to 'A4'.
- Orientation:** A dropdown menu set to 'Portrait'.
- Number of Columns:** A text input field containing '1'.
- Column space:** A text input field containing '0'.
- Margin:** A text input field containing '20'.
- Scripts:** A button labeled 'Collection'.
- Styles:** A button labeled 'Collection'.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.


List of report properties:

Property	Description
Name	Enter name of the report. It should be unique.
Format	Select format of the report's pages (A5, A4, A3, A2, A1).
Orientation	Orientation of the page - Landscape or Portrait.
Number of Columns	Number of columns of the report's table.
Column space	Space between columns of the report's table.

Property	Description
Margin	Page margins of the report.
Scripts	<p>Click Collection to set up report's scripts . After clicking you'll see the window:</p> <div></div> <p>where:</p> <ul style="list-style-type: none">• Scripts - list of available report type scripts in the project.• Add - add script to the collection.• Remove - remove script from the collection.
Styles	<p>Click Collection to set up report's styles. After clicking you'll see the window:</p>

Title	
Page header	
Table	
Page footer	
Summary	

Create report object

You can add new report object on the page by clicking  button. Depending on the page's zone you'll see Add report object. In the table zone you'll see window with tables:

Add report object

Collections

▼ Libraries

Tables

Tables

No columns in table

General history

No columns in table

History database

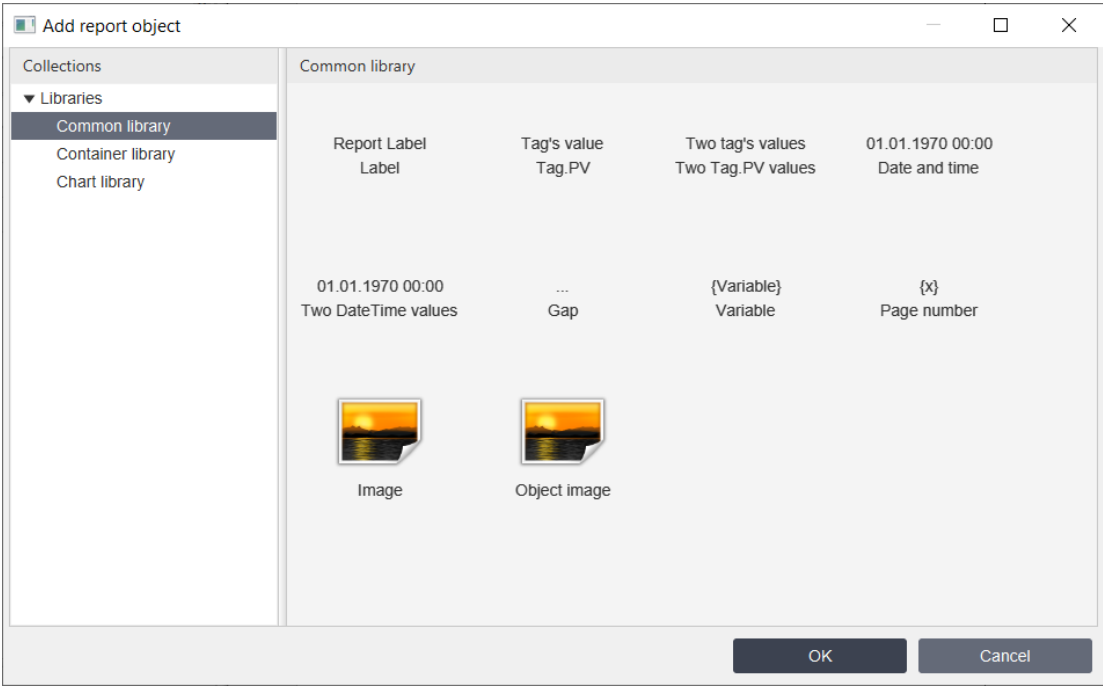
Name	Time	Ty...	Pr...	Message	Value
No content in table					

General events

OK

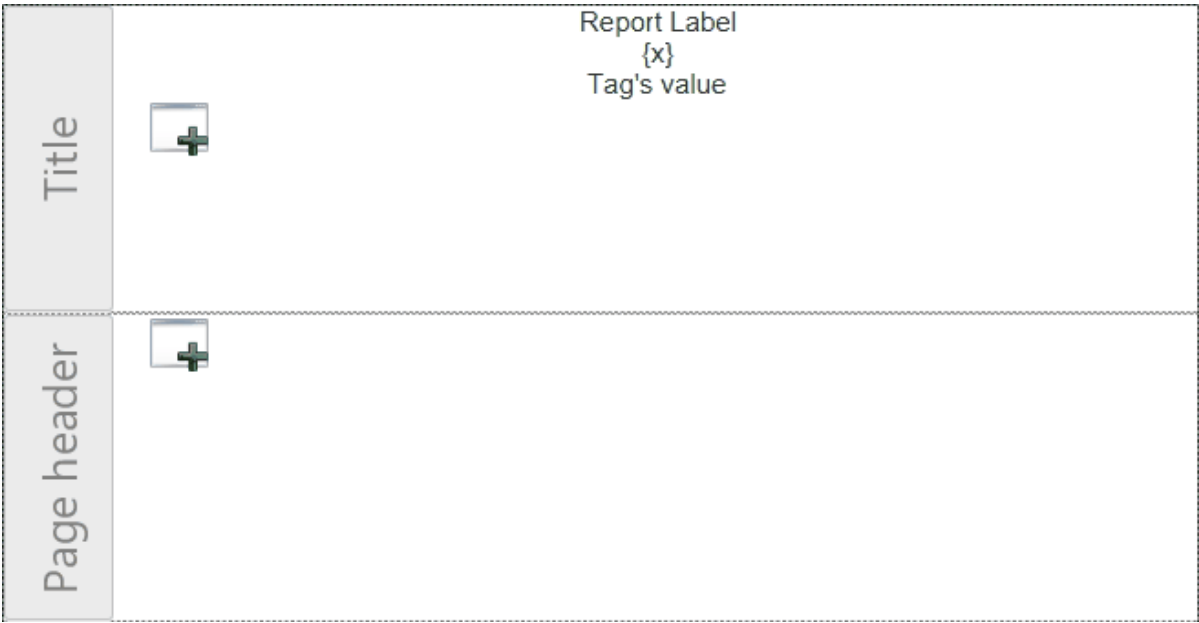
Cancel

In other zones you'll window:



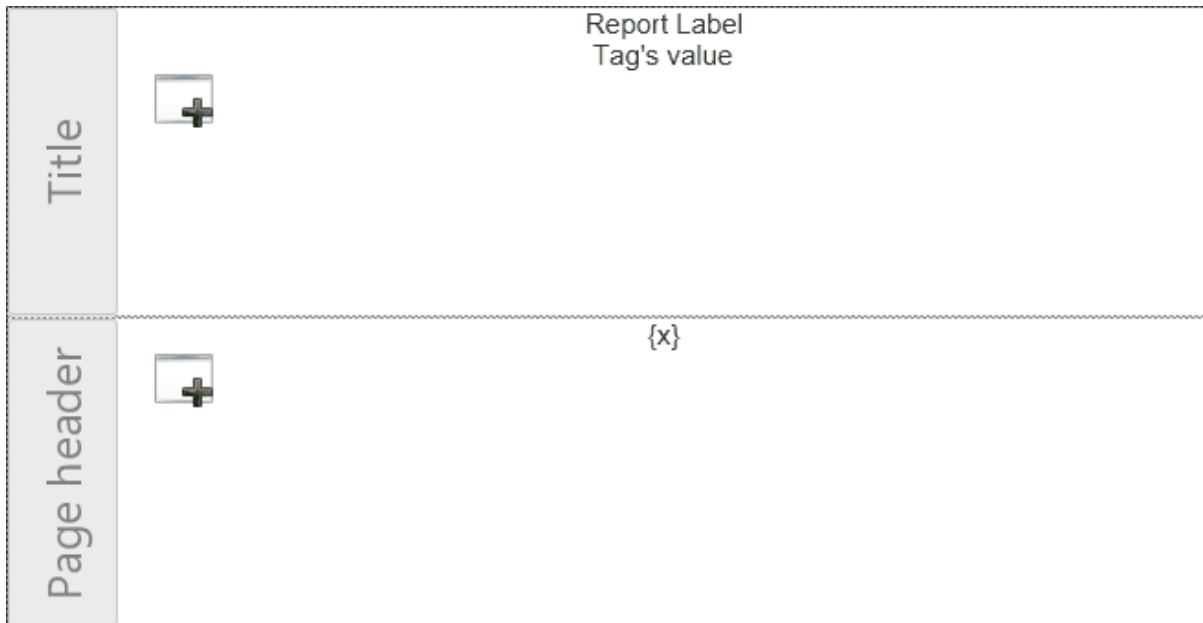
Move report object

You can move report objects by using Drag and Drop technology. You can also move objects by using context menu and choose direction.



Erase report object

You can erase report objects by using context menu and choose Erase menu item.

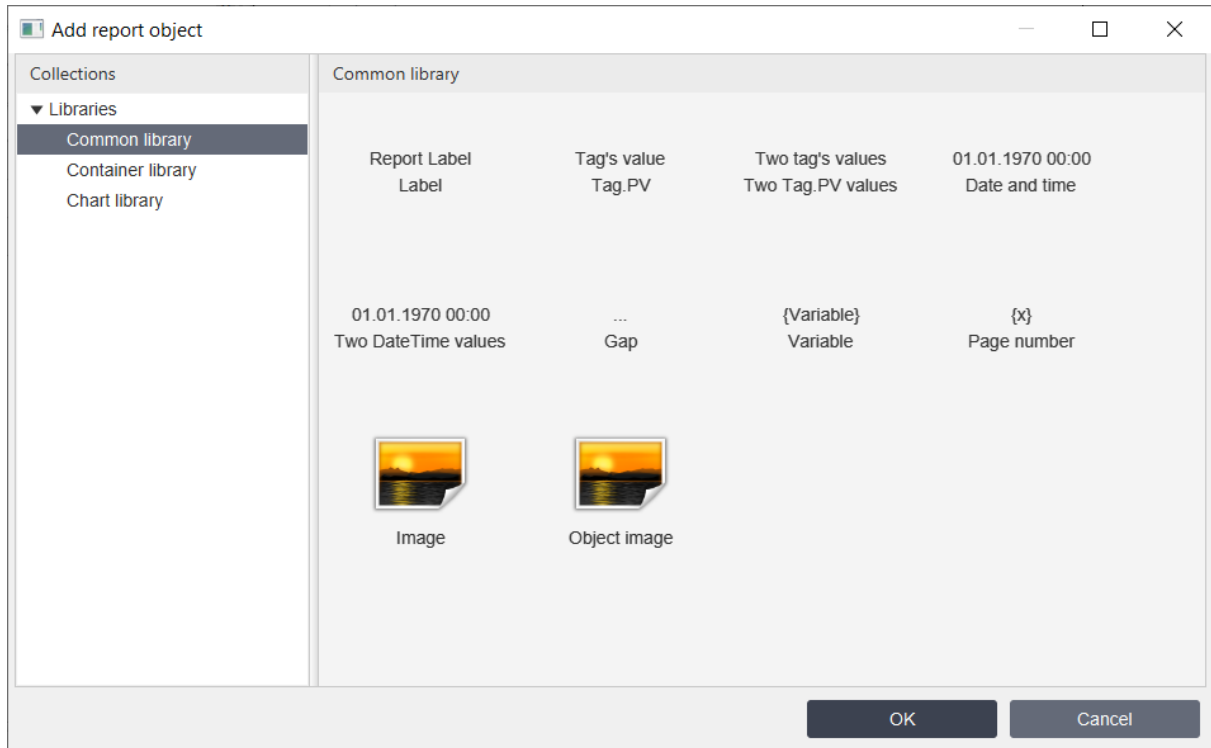


6.8.3 Other report objects

You can add new report objects on the other (not table) zones of the page by clicking



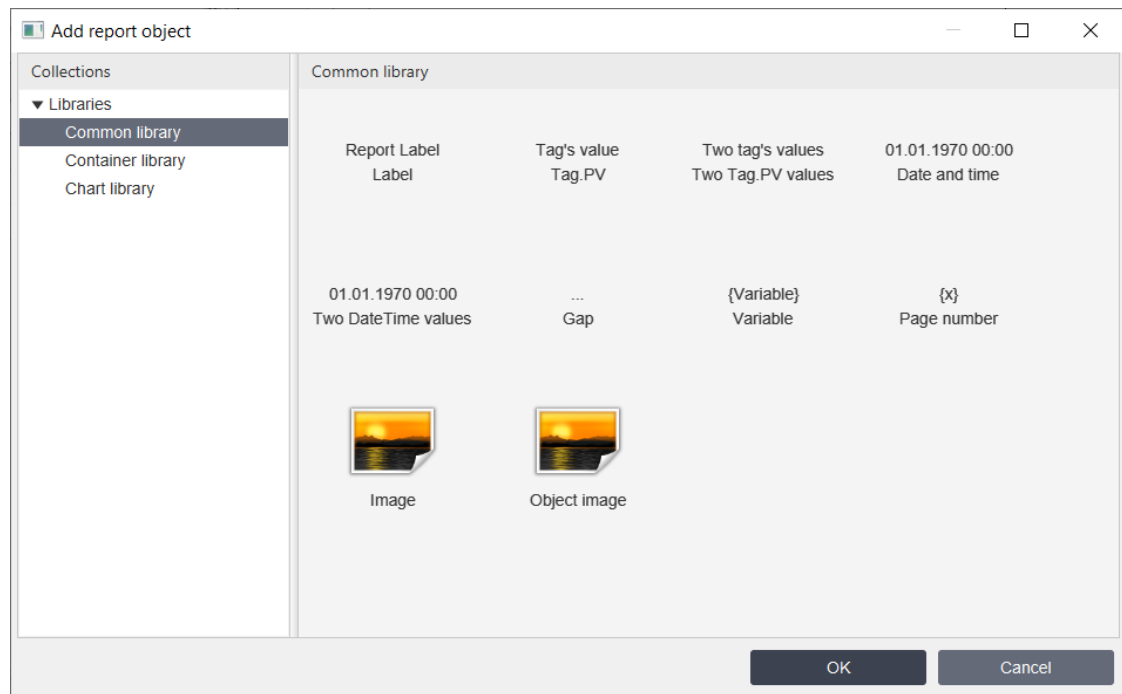
button. You'll see window:



Every report object has the following properties:

Property	Description
Name	Name of the report object.
Style	Style of the report object.


6.8.3.1 Common report library



Report common library contains:

- [Label](#) ⁴⁹⁹
- [Tag.PV](#) ⁴⁹⁹
- [Two Tag.PV values](#) ⁵⁰⁰
- [Date and time](#) ⁵⁰¹
- [Two DateTime values](#) ⁵⁰²
- [Gap](#) ⁵⁰³
- [Variable](#) ⁵⁰³
- [Page number](#) ⁵⁰⁴
- [Image](#) ⁵⁰⁵
- [Object image](#) ⁵⁰⁵

6.8.3.1.1 Label

 Object properties

Name:

Label

Style:

Text:

Report Label

Width:


0

OK

Cancel

Property	Description
Text	Text of the label.
Width	Width of the label.

6.8.3.1.2 Tag.PV

 Object properties

Name:

Tag.PV

Style:

Tag:

...

Width:

0

Format:

###

Text before:

Text after:


OK

Cancel

Property	Description
Tag	Choose tag you want to bind to the object.
Width	Width of the object.
Format	Format of tag's value.

Property	Description
Text before	Text before tag's value.
Text after	Text after tag's value.

6.8.3.1.3 Two Tag.PV values

 Object properties

×

Name:

Two Tag.PV values

Style:

Tag:

...

Tag:

...

Width:

0

Format:

###

Text before:

Text between:


Text after:

OK

Cancel

Property	Description
Tag	Choose tag you want to bind to the object.
Tag	Choose second tag you want to bind to the object.
Width	Width of the object.
Format	Format of tag's values.
Text before	Text before tag's values.
Text between	Text between tag's values
Text after	Text after tag's values.

6.8.3.1.4 Date and time

 Object properties

Name:

Date and time

Style:

Tag:

...

Width:

0

Format:

dd MMMM YYYY HH:mm:ss

Text before:


Text after:

OK

Cancel

Property	Description
Tag	Choose date time tag you want to bind to the object.
Width	Width of the object.
Format	Format of tag's value.
Text before	Text before tag's value.
Text after	Text after tag's value.

6.8.3.1.5 Two DateTime values

 Object properties

Name:

Two DateTime values

Style:

Tag:

...

Tag:

...

Width:

0

Format:

dd MMMM YYYY HH:mm:ss

Text before:

Text between:


Text after:

OK

Cancel

Property	Description
Tag	Choose datetime tag you want to bind to the object.
Tag	Choose second datetime tag you want to bind to the object.
Width	Width of the object.
Format	Format of tag's values.
Text before	Text before tag's values.
Text between	Text between tag's values
Text after	Text after tag's values.

6.8.3.1.6 Gap

 Object properties

Name:

Gap

Style:

Height:

10

Width:


0

OK

Cancel

Property	Description
Height	Height of the gap.
Width	Width of the gap.

6.8.3.1.7 Variable

 Object properties

Name:

Variable

Style:

Variable:

Width:

0

Format:

###

Text before:

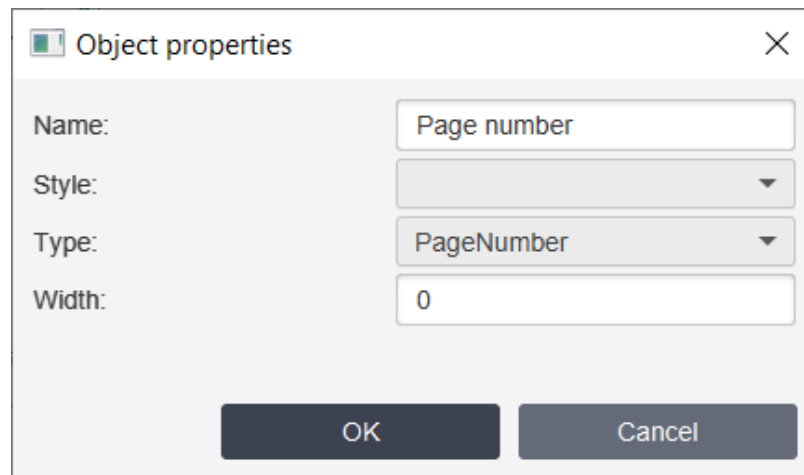
Text after:

OK

Cancel

Property	Description
Variable	Choose variable you want to bind to the object.
Width	Width of the object.
Format	Format of variable's value.
Text before	Text before variable's value.
Text after	Text after variable's value.

6.8.3.1.8 Page number



The screenshot shows a dialog box titled "Object properties" with a close button (X) in the top right corner. The dialog contains four labeled input fields: "Name:" with the text "Page number", "Style:" with a dropdown arrow, "Type:" with a dropdown menu showing "PageNumber", and "Width:" with the text "0". At the bottom of the dialog are two buttons: "OK" and "Cancel".

Property	Description
Type	Type of the page number.
Width	Width of the object.

6.8.3.1.9 Image

Object properties

×

Name:

Style:

Height:

Width:

-

+

Image

OK

Cancel

Property	Description
Height	Height of the image.
Width	Width of the image.
Image	Choose image of the report object.

6.8.3.1.10 Object image

Object properties

×

Name:

Style:

Height:

Width:

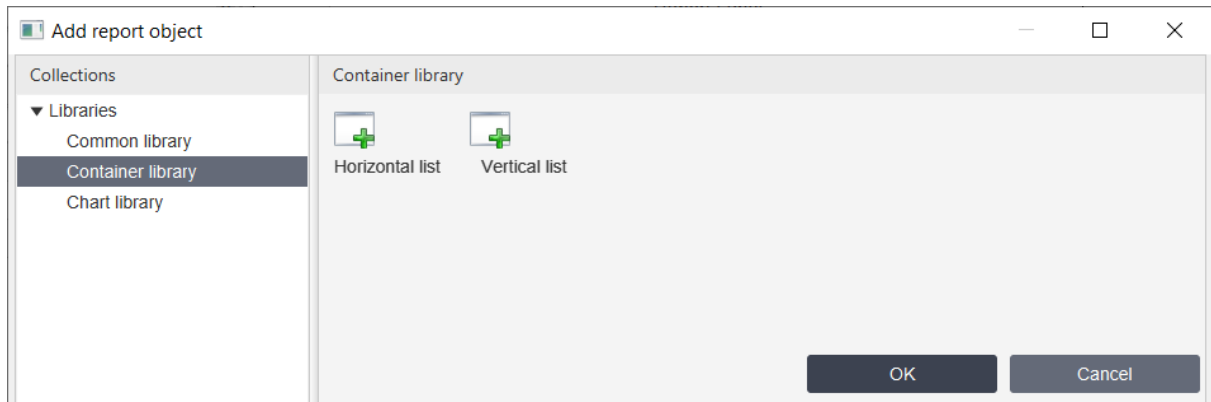
Object:

OK

Cancel

Property	Description
Height	Height of the image.
Width	Width of the image.
Object	Choose object you want to display in the report. Useful for trends.

6.8.3.2 Container library



Container library contains two objects that lets you add other report objects in Vertical and Horizontal lists.

6.8.3.3 Chart library

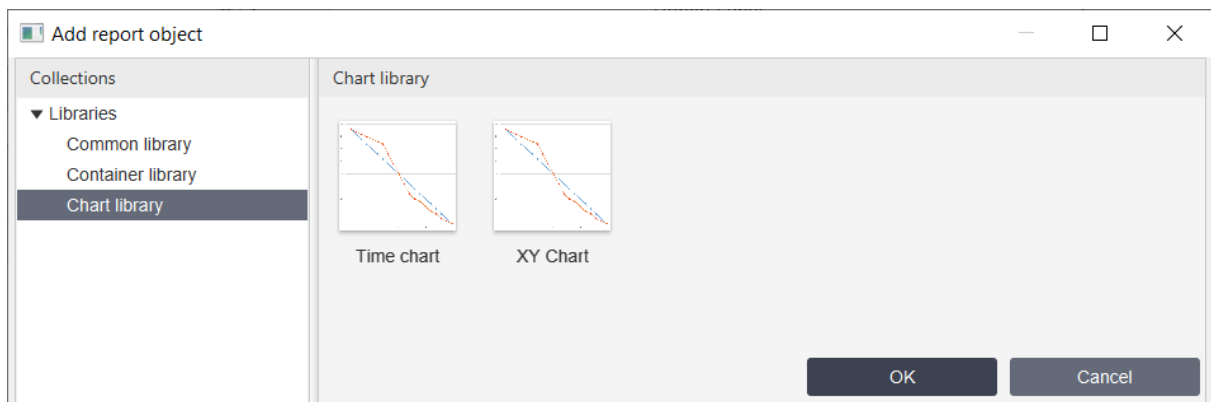



Chart library objects contains objects:

- [Time chart](#)⁵⁰⁷
- [XY chart](#)⁵⁰⁸

6.8.3.3.1 Time chart

 Object properties

Name:

Time chart

Style:

Title:

Chart Title

Title font size:

20

Time period:

Time period type:

Second

Width:

0

Height:

0

Curves:

Collection

OK

Cancel

Property	Description
Title	Title of the chart.
Title font size	Font size of the title's text.
Time period	Choose column for time axis.
Time period type	Choose period of the time. (Second, Minute, Hour....).
Height	Height of the chart.
Width	Width of the chart.
Curves	Click Collection to set up chart's curves. After clicking you'll see the window:

Property	Description
	<div><div><div>Collection</div><div><div></div><div>Column name: <div></div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div><div>where:<ul style="list-style-type: none">▪ Column name - name of the column bind to the curve.</div></div>

6.8.3.3.2 XY chart

Object properties

Name:

XY Chart

Style:

Title:

Chart Title

Title font size:

20

X Value:

X label:

X

Y label:

Y

Width:

0

Height:

0

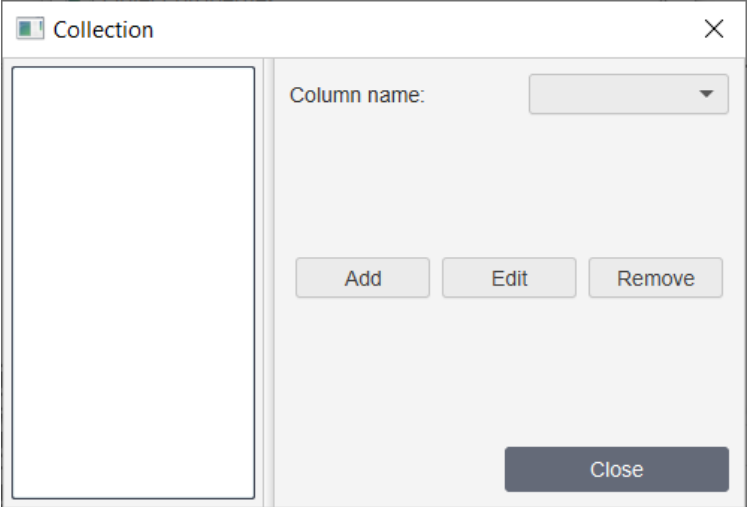
Curves:

Collection


OK

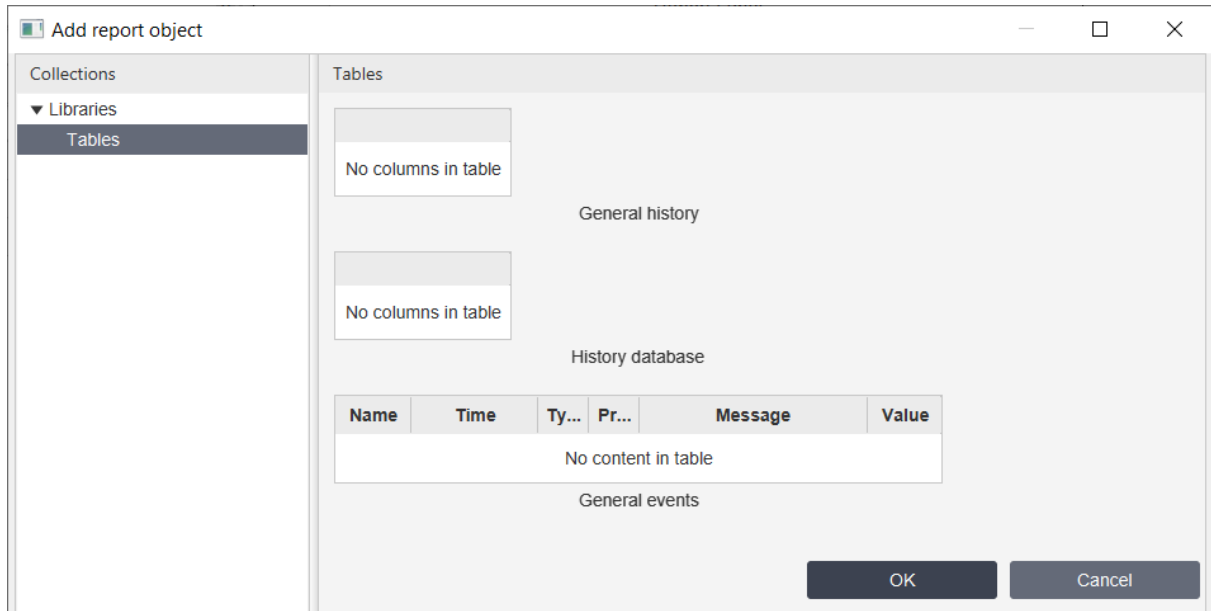
Cancel

Property	Description
Title	Title of the chart.
Title font size	Font size of the title's text.
X value	Choose column for X axis.

Property	Description
X label	Enter label for X axis.
Y label	Enter label for Y axis.
Height	Height of the chart.
Width	Width of the chart.
Curves	<p>Click Collection to set up chart's curves. After clicking you'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none">▪ Column name - name of the column bind to the curve.

6.8.4 Table report objects

You can add new table report object on the table's zone of the page by clicking  button. You'll see window:



Every table object has the following properties:

Property	Description
Name	Name of the report table object.
Column title style	Style of the column titles.
Start Date and Time	Initial time of data taken from the database.
End Date and Time	End time of data taken from the database.
Highlight Even Row	Highlight even rows of the table.

6.8.4.1 General history table

Object properties

Name:

General history

Column title style:

Start Date And Time:

...

End Date And Time:

...

Time interval:

...

Highlight Even Row:

false

Columns

Collection

Condition styles

Collection

Variables

Collection

OK

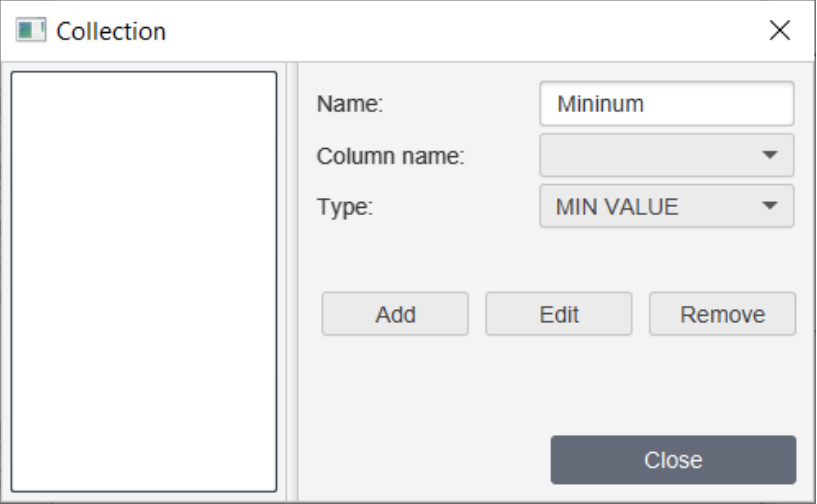
Cancel

General history report table get data from the [general history database](#)

Property	Description
Time interval	Time interval with which data is taken from the database.
Columns	Click Collection to set up report's columns . After clicking you'll see the window:

Property	Description
	<div><div>Collection</div><div><div></div><div><div><div>Name:</div><div>Column</div></div><div><div>Title:</div><div>Column</div></div><div><div>Type:</div><div>Tag</div></div><div><div>Tag:</div><div>Tag1</div><div>...</div></div><div><div>Processing type:</div><div>FIRST VALUE</div></div><div><div>Width:</div><div>80</div></div><div><div>Format:</div><div></div></div><div><div>Styles:</div><div>Title</div></div><div><div>Condition styles</div><div>Collection</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div> <div>where:</div> <div><ul style="list-style-type: none">• Name - name of the column.• Title - title of the column.• Type - type of the column (Tag, DateTime, Row number).• Tag - choose tag you want to bind to this column.• Processing type - processing tag's columns values in interval.<ul style="list-style-type: none">✓ FIRST VALUE - take first value from the interval.✓ LAST VALUE - take last value from the interval.✓ MIN VALUE - take minimum value from the interval.✓ MAX VALUE - take maximum value from the interval.✓ AVG VALUE - take average value from the interval.• Width - width of the column.• Format - how to format value in the column.• Styles - choose style for the column.• Condition styles - condition styles. You can setup it by clicking Collection:</div>

Property	Description
	<div><div><div><div>Collection</div><div><div></div></div><div><div><div>Name:</div><div>Condition</div></div><div><div>Column name:</div><div></div></div><div><div>Type:</div><div>==</div></div><div><div>Value:</div><div>0</div></div><div><div>Color:</div><div><input type="checkbox"/> White</div></div><div><div>Font style:</div><div>Normal</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div></div> <p>where:</p> <ul style="list-style-type: none">• Name - name of the condition style.• Column name - name of the column.• Type - type of the comparison.• Value - value to the comparison.• Color - color of the cell when the condition is right.• Font style - text's font style of the cell when the condition is right.
Condition styles	<p>Click Collection to set up condition styles . After clicking you'll see the window:</p> <div><div><div><div>Collection</div><div><div></div></div><div><div><div>Name:</div><div>Condition</div></div><div><div>Column name:</div><div></div></div><div><div>Type:</div><div>==</div></div><div><div>Value:</div><div>0</div></div><div><div>Color:</div><div><input type="checkbox"/> White</div></div><div><div>Font style:</div><div>Normal</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div></div> <p>where:</p> <ul style="list-style-type: none">• Name - name of the condition style.• Column name - name of the column.

Property	Description
	<ul style="list-style-type: none">• Type - type of the comparison.• Value - value to the comparison.• Color - color of the row when the condition is right.• Font style - text's font style of the row's cells when the condition is right.
Variables	<p>Click Collection to set up variables. After clicking you'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none">• Name - name of the variable.• Column name - name of the column.• Type - type of the variable.<ul style="list-style-type: none">✓ MIN VALUE - minimum value in the column.✓ MAX VALUE - maximum value in the column.✓ AVG VALUE - average value in the column.✓ SUM VALUE - summary value in the column.

6.8.4.2 General events table

Object properties

General

Columns

Name:

General events

Column title style:

Rows style:

Start Date And Time:

...

End Date And Time:

...

From priority:

...

To priority

...

Highlight Even Row:

false

Time format:

dd:MM:YYYY HH:mm:ss

Value format:

###

Condition styles

Collection

OK

Cancel

General events report table get data from the [general events database](#)

Property	Description
Rows style	Style of the table's rows.
From priority	The tag's value is used to determine the initial priority.
To priority	The tag's value is used to determine the end priority.
Time format	Format of the time displayed in the column.
Value format	Format of the value displayed in the column.
Condition styles	Click Collection to set up condition styles . After clicking you'll see the window:

Property	Description
	<div><div><div>Collection</div><div><div></div><div><div>Name:Condition</div><div>Column name:<div></div></div><div>Type:==</div><div>Value:0</div><div>Color:<div><div></div>White</div></div><div>Font style:Normal</div><div><div>Add</div><div>Edit</div><div>Remove</div><div>Close</div></div></div></div></div><div><p>where:</p><ul style="list-style-type: none">• Name - name of the condition style.• Column name - name of the column.• Type - type of the comparison.• Value - value to the comparison.• Color - color of the row when the condition is right.• Font style - text's font style of the row's cells when the condition is right.</div></div>

6.8.4.2.1 Columns

Object properties

General

Columns

☒ Name

Title:

Name

Width:

60

☒ Time

Title:

Time

Width:

100

☒ Type

Title:

Type

Width:

40

☒ Priority

Title:

Priority

Width:

40

☒ Message

Title:

Message

Width:

180

☒ Value

Title:

Value

Width:

60

☐ Ack.time

Title:

Ack.Time

Width:


80

OK

Cancel

Property	Description
Enable (not shown)	Enable or disable correspondent column: <ul style="list-style-type: none">NameTimeTypePriorityMessageValueAck.time
Title	Title of the corresponding column.
Width	Width of the corresponding column.

6.8.4.3 History database table

 Object properties

Name:

History database

History DB:

Column title style:

Start Date And Time:

...

End Date And Time:

...

Time interval:

...

Highlight Even Row:

false

Columns

Collection

Condition styles

Collection

Variables

Collection

OK

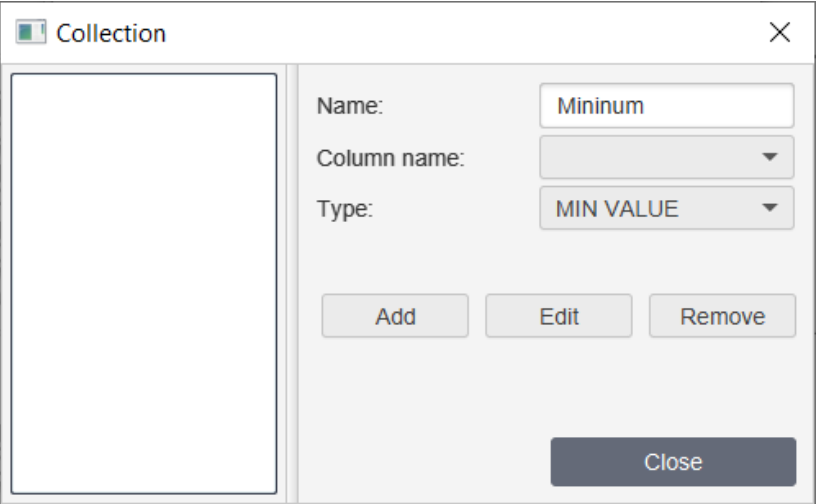
Cancel

History database report table get data from the [history database](#)⁴⁸⁵.

Property	Description
History DB	Choose History DB you want to bind this history report table.
Time interval	Time interval with which data is taken from the database.
Columns	Click Collection to set up report's columns . After clicking you'll see the window:

Property	Description
	<div><div>Collection</div><div><div></div><div><div><div>Name:</div><div>Column</div></div><div><div>Title:</div><div>Column</div></div><div><div>Type:</div><div>Tag</div></div><div><div>Tag:</div><div>Tag1</div><div>...</div></div><div><div>Processing type:</div><div>FIRST VALUE</div></div><div><div>Width:</div><div>80</div></div><div><div>Format:</div><div></div></div><div><div>Styles:</div><div>Title</div></div><div><div>Condition styles</div><div>Collection</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div> <div><div>where:</div><div><ul style="list-style-type: none">• Name - name of the column.• Title - title of the column.• Type - type of the column (Tag, DateTime, Row number).• Tag - choose tag you want to bind to this column.• Processing type - processing tag's columns values in interval.<ul style="list-style-type: none">✓ FIRST VALUE - take first value from the interval.✓ LAST VALUE - take last value from the interval.✓ MIN VALUE - take minimum value from the interval.✓ MAX VALUE - take maximum value from the interval.✓ AVG VALUE - take average value from the interval.• Width - width of the column.• Format - how to format value in the column.• Styles - choose style for the column.• Condition styles - condition styles. You can setup it by clicking Collection:</div></div>

Property	Description
	<div><div><div><div>Collection</div><div><div></div></div><div><div><div>Name:</div><div>Condition</div></div><div><div>Column name:</div><div></div></div><div><div>Type:</div><div>==</div></div><div><div>Value:</div><div>0</div></div><div><div>Color:</div><div><input type="checkbox"/> White</div></div><div><div>Font style:</div><div>Normal</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div><div><p>where:</p><ul style="list-style-type: none">• Name - name of the condition style.• Column name - name of the column.• Type - type of the comparison.• Value - value to the comparison.• Color - color of the cell when the condition is right.• Font style - text's font style of the cell when the condition is right.</div></div>
Condition styles	<div><p>Click Collection to set up condition styles . After clicking you'll see the window:</p><div><div><div><div>Collection</div><div><div></div></div><div><div><div>Name:</div><div>Condition</div></div><div><div>Column name:</div><div></div></div><div><div>Type:</div><div>==</div></div><div><div>Value:</div><div>0</div></div><div><div>Color:</div><div><input type="checkbox"/> White</div></div><div><div>Font style:</div><div>Normal</div></div><div><div>Add</div><div>Edit</div><div>Remove</div></div><div><div>Close</div></div></div></div></div><div><p>where:</p><ul style="list-style-type: none">• Name - name of the condition style.• Column name - name of the column.</div></div></div>

Property	Description
	<ul style="list-style-type: none"> • Type - type of the comparison. • Value - value to the comparison. • Color - color of the row when the condition is right. • Font style - text's font style of the row's cells when the condition is right.
Variables	<p>Click Collection to set up variables. After clicking you'll see the window:</p>  <p>where:</p> <ul style="list-style-type: none"> • Name - name of the variable. • Column name - name of the column. • Type - type of the variable. <ul style="list-style-type: none"> ✓ MIN VALUE - minimum value in the column. ✓ MAX VALUE - maximum value in the column. ✓ AVG VALUE - average value in the column. ✓ SUM VALUE - summary value in the column.

6.8.5 Reports from trend's and event's dialog boxes

For some graphical objects like [Events log](#)²³⁸, [History trends](#)²²⁸, [Recipe table](#)²⁴⁷ and others you can create Reports during running project. You can create 2 types of Reports - Excel reports and report for printing. See example window:

To get Excel report you have to click **Save report...** . Then choose file to save Excel report and make some other settings like Title.

To get report for printing you have to click Print button. You'll see Report settings window:

In Report settings you can setup some parameters of the report:

Paper, where:

- Format of the paper.

- Orientation of the paper.
- Paper width and Paper height.
- Set Pagination if you want to show page numbers.

Banner, where:

- Choose Image of the banner.
- Setup Width and Height of the banner.
- Setup Horizontal Alignment of the banner.
- Use banner For All Pages or not.

Report title, where:

- Title caption of the report.
- Font of the caption.
- Color of the caption.
- Horizontal Alignment of the caption.

Report subtitle, has the same Font, Color and Horizontal Alignment parameters like Report title. Column headers, has the same Font, Color and Horizontal Alignment parameters like Report title.

And has some other parameters, where:

- Background color of the caption.
- Border of the caption.
- Vertical Alignment of the caption.
- Number of columns using in report.
- Group by tag if you want to use report's grouping.

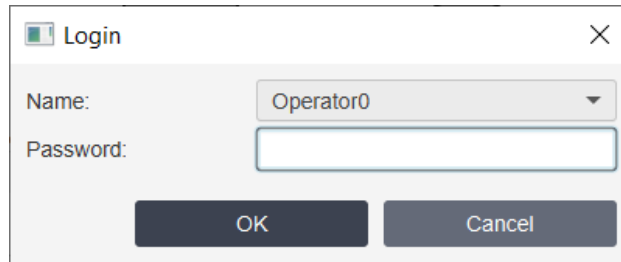
Cell properties, has the same Font, Color and Horizontal Alignment parameters like Report title and Background color, Border and Vertical Alignment parameters like Column headers. And has some other parameters, where:

- Check Highlight Even Row if you want to do it.
- Choose Even Row Background.
- Set up Save every (...) sec for trends reports for choosing save period.

You can Save this report settings template for this graphical object and then Open... it. To create report by using these settings click Print. You'll see Preparing report window. After some time you'll see your Report. You can print directly by choosing your printer or you can save this report in some format: pdf, html, csv and others.

6.9 Simulation

You can simulate behavior of you project. To start simulation select the menu item **Project->Run simulation**^[67] or click button on the **Toolbar**^[70]. If you use users in your project Login dialog will appear:



A login dialog box titled "Login" with a close button (X) in the top right corner. It contains two input fields: "Name:" with a dropdown menu showing "Operator0" and "Password:" with a text input field. At the bottom, there are two buttons: "OK" and "Cancel".

Select user and enter password in the field. Now you can simulate your project.

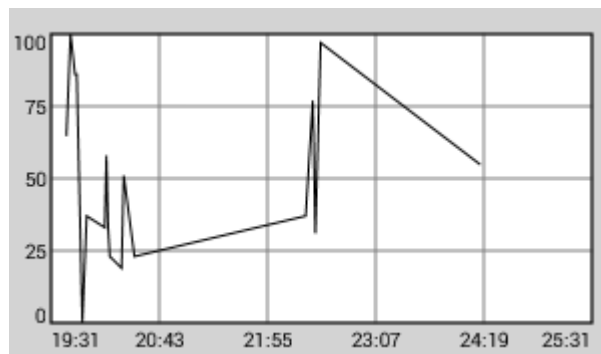
You can change value of the tag by double clicking on it in the [Project window](#)^[72] - [Tags](#)^[81] or you can click by right button on the tag and select [Simulate->Set value](#)^[83] menu item. Also you can simulate behavior of the tag:

1. Random value - periodically change the value of the tag randomly from 1 to 100.
2. Ramp value - periodically change the tag value from 1 to 100 by adding 1.

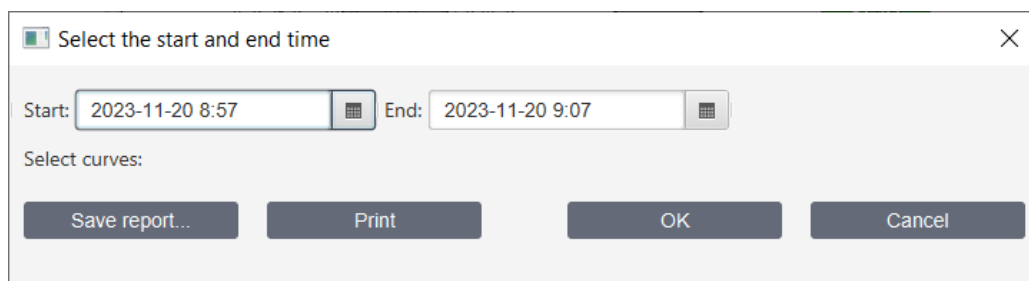
By selecting [Simulate->Cancel](#)^[83] menu item you annul the task.

Also it's possible to change value of the tag using control graphical objects of your project like [text](#)^[159], [buttons](#)^[180], [slider](#)^[214], [counter](#)^[216] and etc. For example if you use Text object enable output property and bind to the tag you want to use. During simulation click on it and enter value you want.

Also you can simulate behavior of [Trend](#)^[228] and [Events log](#)^[238] objects. Place these objects on the [Canvas](#)^[91]. Set properties of the object as describe in previous chapters. During simulation trend will be look like this:



To select start and end time click on it. You'll see the following dialog. Select times and click OK.



A dialog box titled "Select the start and end time" with a close button (X) in the top right corner. It contains two input fields: "Start:" with the value "2023-11-20 8:57" and "End:" with the value "2023-11-20 9:07". Below these fields is a label "Select curves:". At the bottom, there are four buttons: "Save report...", "Print", "OK", and "Cancel".

During simulation Events log will be look like this:

Events (All)						
Name	Time	Type	Prio...	Message	Value	
Value	16/09 09:24:16	Normal	900	Value is normal	55	
Value	16/09 09:22:31	HiHi	50	Value is too high	97	
Value	16/09 09:22:26	Normal	900	Value is normal	38	
Value	16/09 09:22:25	Hi	500	Value is high	77	
Value	16/09 09:20:28	Normal	900	Value is normal	54	

1. To View message in the separate dialog double click on it or click right button on it and select View menu item.
2. To acknowledge record click by right button on it and select Acknowledge menu item.
3. To acknowledge all records on the table click by right button on the table and select Acknowledge All menu item.
4. To delete record click by right button on it and select Delete menu item.
5. To delete all records on the table click by right button on the table and select Delete All menu item.

You can select records that you want to see in the table. Click on the table's title. You'll see Select time and priority conditions dialog. Select start and end times of records displayed in the table. You can also set records with what priorities will be displayed.

Select time and priority conditions

☒ From time

☒ To time

☒ From priority

☒ To priority

Save report...

Print

OK

Cancel

7 Load on Device

When project is created (screens, servers, tags, scripts and users), the project can be loaded on the mobile device or other PC. First, the corresponding TeslaSCADA Runtime mobile app on the Android device or PC apps on the Windows, Linux or MAC OS should be installed and started.

If the app has been installed on the mobile device or PC, there are 2 ways to load the project to the device:

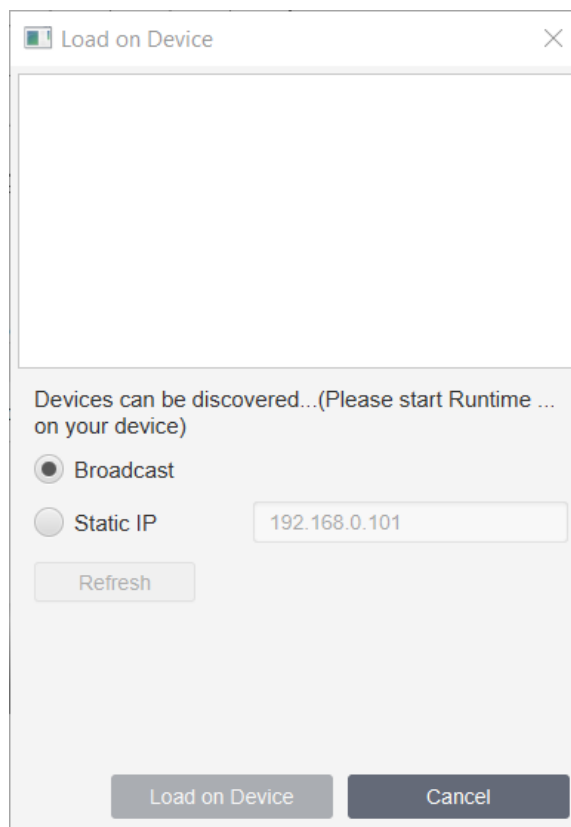
1. Network method.
2. Manual method.

Network method

Start the PC on which TeslaSCADA IDE is installed, and also start the mobile device or PC on which TeslaSCADA Runtime is installed, the devices must be on the same Wi-Fi network.

Procedure:

1. Enable WiFi on your mobile device or PC where TeslaSCADA Runtime is installed .
2. Start the TeslaSCADA2 Runtime app.
3. Open in the editor TeslaSCADA2 IDE the desired project to be transferred and select the menu item **File->Load on Device**.
4. The dialog "Load on Device" opens and it will search for mobile devices with the active TeslaSCADA2 Runtime. You can start a broadcast search and browse the entire network. However, since some routers do not forward broadcasts, there is also the possibility of a specific device search on the IP address. This search takes normally 5-10s. In individual cases it may happen that this search can take to 3 minutes. If you can't find a device you can try to restart "Load on Device" dialog and TeslaSCADA2 Runtime application:



5. After a successful search in this dialog box all found mobile devices with active TeslaSCADA Runtime app will be shown.
6. Now select the desired target device and press the **Load on Device** button.
7. After a successful transfer, the target device with TeslaSCADA2 Runtime loads new project.

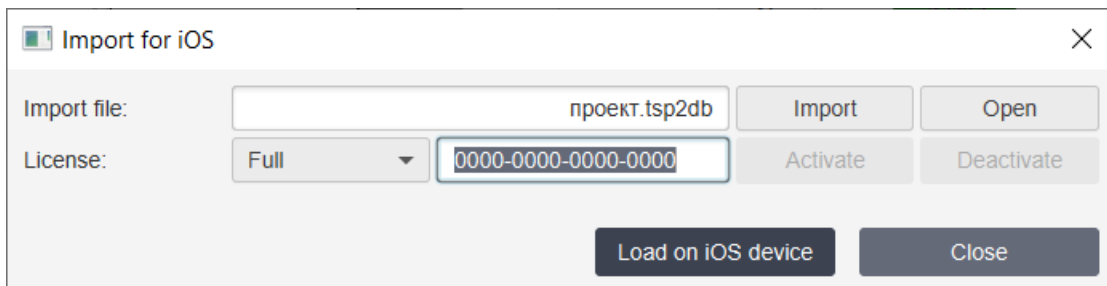
Manual method

Another way to load a project on the mobile device is a file explorer such as: the Android File Transfer. Once the TeslaSCADA Runtime installed mobile app and once started on the sd card, a folder called **Android/dat/tesla.scada2.android/files/Projects** is created.

Now the project, which is stored in a file with the .tsp2 extension from Windows, Linux or MacOS, can be manually copied to the SD card folder of the mobile device on which TeslaSCADA Runtime is installed. Now the project can be loaded manually by clicking the Open button on the TeslaSCADA Runtime main menu. Similarly, you can perform the above steps on a PC where TeslaSCADA Runtime is installed by copying the project file. You can use a local network, a flash drive, or any other portable storage device.

8 Import for iOS

When project is created, it can be imported for iOS mobile devices. To do import for iOS devices you should enter **File-> Import for iOS**. When you do it Import for iOS window will appear:



To do import, click the Import button, a file dialog will appear, enter a file name and click OK. The import file has the extension *.tsp2db. The file is based on a database in SQL format. You can open and check the data by opening it in any program that works with SQL databases. You can also open the imported file by clicking the Open button. The imported or opened file appears in the text field. To activate the project:

1. Choose license type.
2. Enter license number.
3. Click Activate button (it will change background color to green and «License available for activation» message will appear).
4. If you want to deactivate license click Deactivate button (it will change background color to green).
5. Load project on iOS device.
6. When loading of the project is completed on iOS device «Activation completed» message will appear (device should have an Internet access).

If TeslaSCADA2 Runtime is installed on your iOS device (iPhone or iPad), there are 2 ways to download the imported project to the device:

1. Network method.
2. Manual method.

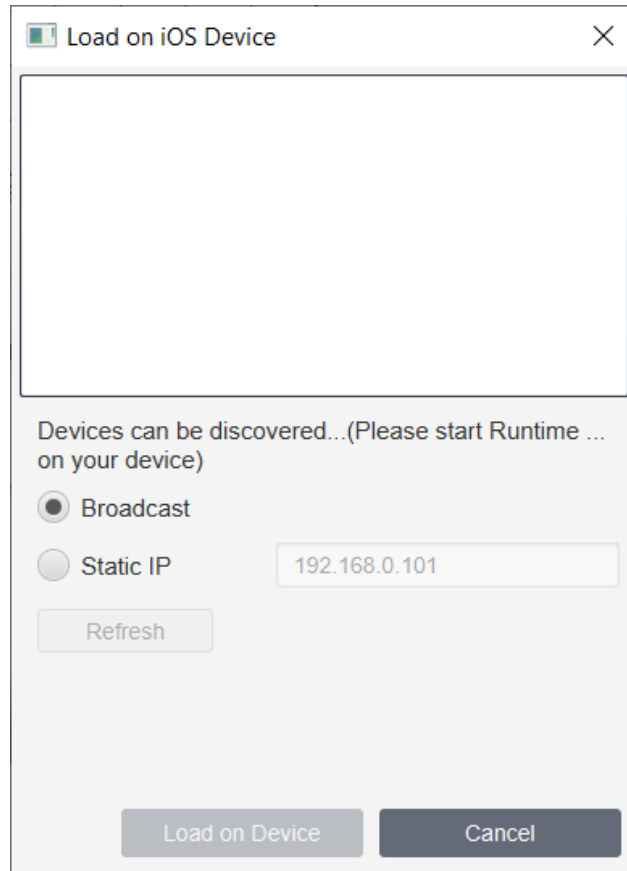
Click **Load on iOS device** to use Network method to load imported file on your iOS device.

Network method

In order to use this method, the PC on which TeslaSCADA IDE is installed must be turned on, and the iOS device on which TeslaSCADA2 Runtime is installed must be running, the devices must be on the same Wi-Fi network.

Perform the following steps in sequence:

1. Turn on WiFi on the mobile device on which TeslaSCADA2 Runtime is installed.
2. Launch TeslaSCADA2 Runtime.
3. Select the menu item File->Import for iOS into TeslaSCADA2 IDE.
4. Open the desired project to import. Click the Download button on your iOS device.
5. After this, a dialog box will open and the search for devices with active TeslaSCADA2 Runtime will begin. You can start searching for broadcast and explore the entire network. However, since some routers do not support broadcast, it is also possible to search for a specific device by IP address. Usually the search takes 5-10s. In some cases this can last up to 3 minutes. If you cannot find the device, you can re-launch the Download to iOS device and TeslaSCADA2 Runtime dialog box. After a successful search, all found devices with running TeslaSCADA2 Runtime applications will appear in the dialog box:



6. Now select the device you want to download the project to and click the **Download on Device** button.

7. After successful data transfer, TeslaSCADA2 Runtime will load a new project.

Manual method

Another way to download a project to an iOS mobile device is iTunes -> File Sharing.

Important! For newer versions of MacOS, you can download the project to your device using Finder.

1. Open iTunes on your Mac or PC.
2. Connect your iPhone or iPad to your computer using the USB cable that comes with the device.
3. Click on your device in iTunes.
4. In the side menu, click Apps. Then scroll to the File Sharing section at the bottom of the page.
5. Find the "TeslaSCADA2 Runtime" folder, copy the project file (*.tsp2db) to this folder.

9 Examples

This chapter provides examples of the most commonly used tasks.

Important! For all examples below we'll change properties in Object properties window, but you can do it in [Property sheet](#)^[91] if you want.

9.1 Change the color of an object


Let's consider the most common cases when you want to change the color of an object when the value of its associated variable changes. All of the examples below can be applied to different colors - fills, borders, text, etc. Below you can find out several examples from common to more complex with scripts:

- [Common color change](#)^[530]
- [Common multiple color change](#)^[531]
- [Common multiple color change with scripts](#)^[533]
- [Complex color change](#)^[535]
- [Complex color change with scripts](#)^[539]

9.1.1 Simple color change

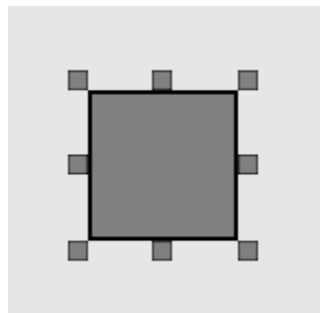
Let's assume that in our project there is a certain object that has two states: on, off. The object's state data is passed to the tag. We want the object's fill color to differ on the screen depending on the state of the object.

1. Let's create a tag named State, which is responsible for data about object's state (set the data type of the tag to Boolean and the default value to false):

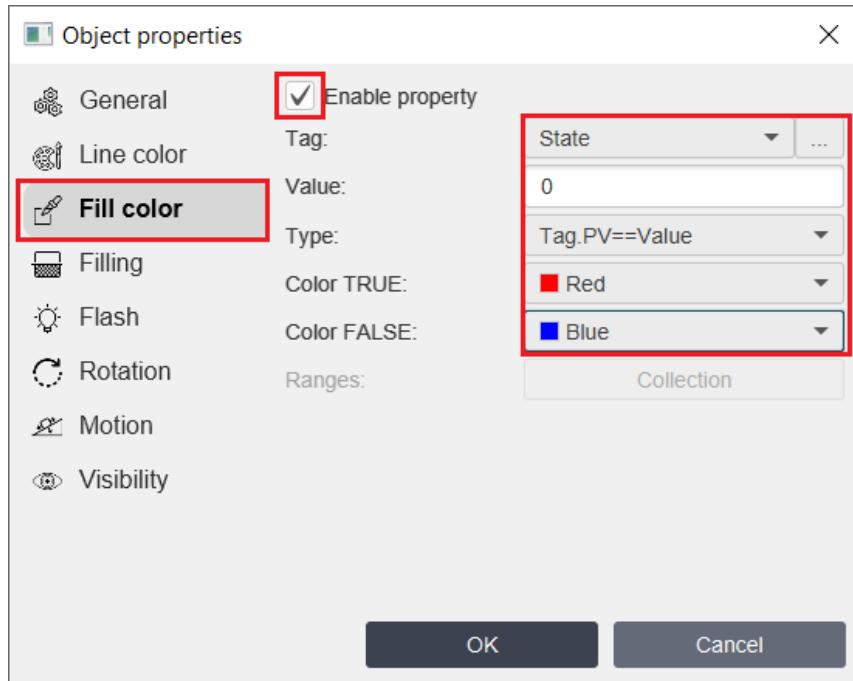


Name	Value
State	false

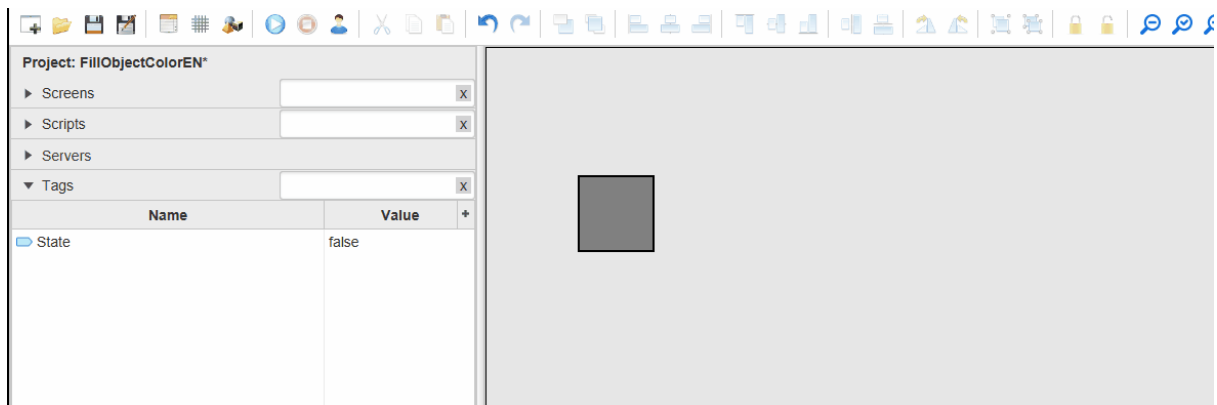
2. Now let's create a Rectangle object (choose the object that suits your specific case) and bind it to the State tag.



3. Let's set the "Fill Color" property. For example, we want to have red fill color if the object is turned off, and blue fill color if the object is turned on:



4. Let's run the simulation to check the settings:



You can download this project [here](#).

9.1.2 Simple multiple color change

Suppose we have a certain object (let there be a valve) that has several operating modes (open, closed, mode1, mode2). We want to display an object on the screen with a different color depending on the operating mode.

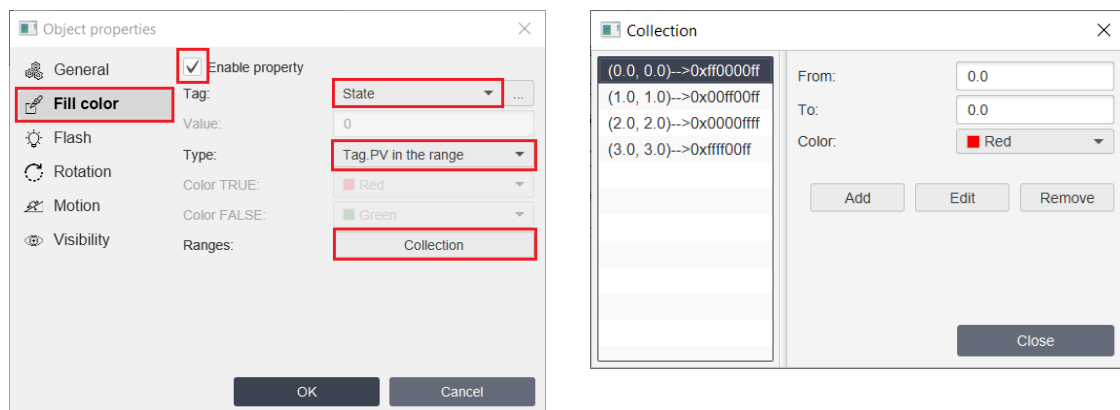
1. Let's create a State tag, which is responsible for the operating mode of the valve (select the data type - Byte (8bit), and the default value is 0):



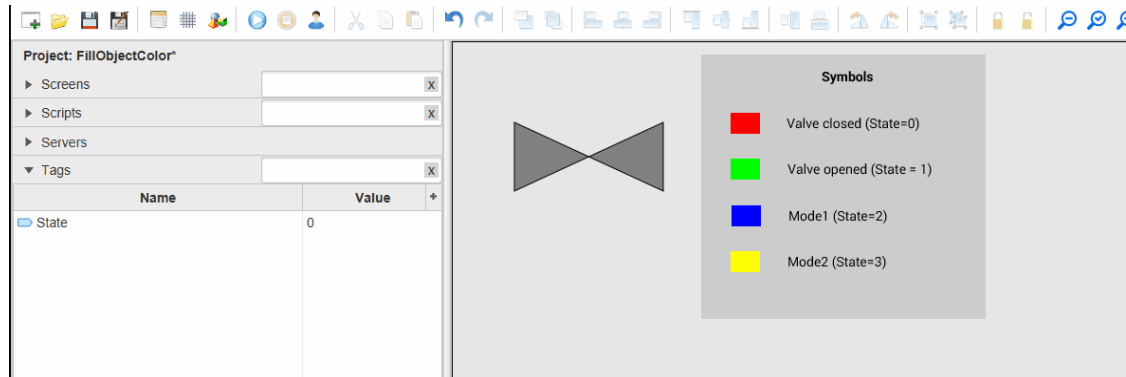
2. Let's create an object - [Valve ISA](#)¹⁹⁴ and set the "Fill Color" property depending on the tag value:

State	Color
0	RED
1	GREEN
2	BLUE
3	YELLOW

To do this, in the "Fill Color" tab, check the "Enable Property" and select the **"Tag.PV in Range"** type, and then set the colors for each mode:



3. Let's [Run simulation](#)⁷⁰ to check the settings:



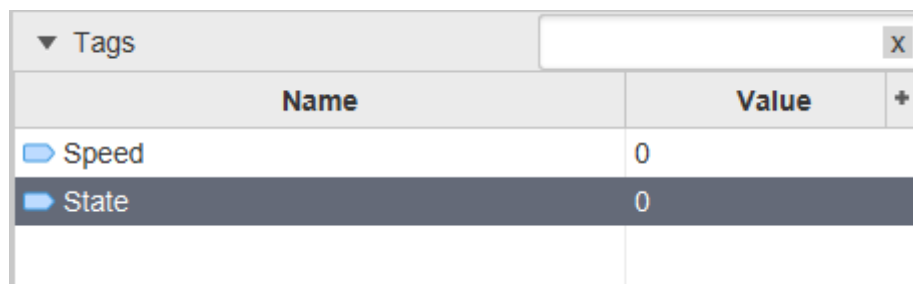
You can download this project [here](#).

9.1.3 Simple multiple color change with scripts

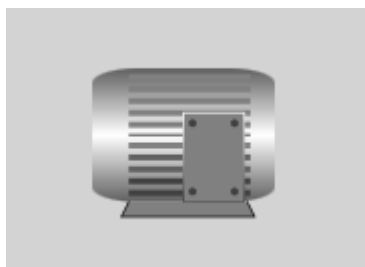
If you need to change the color depending on several tags, you need to use scripts. For example, you have a Motor object that has 2 parameters - State and Speed, and you want to use fill color depending on the State and Speed values:

State	Speed	Fill color
0	Any	RED
1	0...500	GREEN
1	500...1000	YELLOW
1	>1000	BLUE

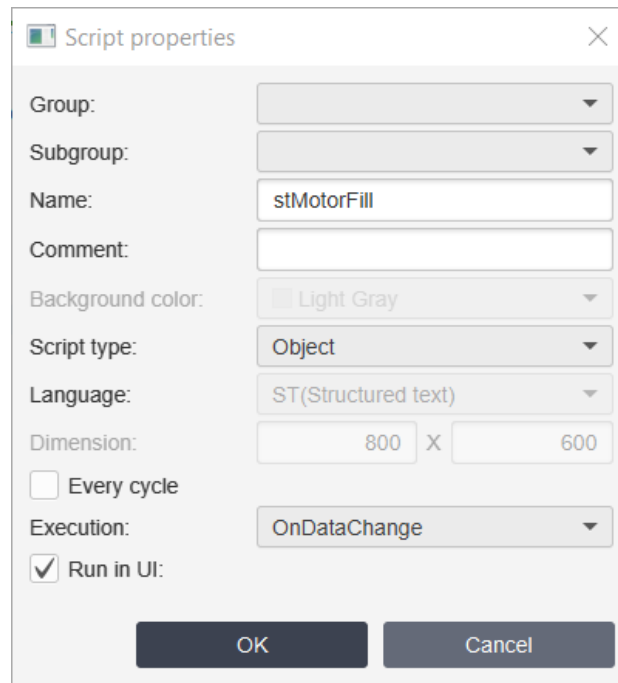
1. Create tags: Speed (set the data type - Short(16bit), initial PV - 0) and State (set the data type - Byte(8bit), initial PV - 0):



2. Let's create a graphical object - Motor for our example:



3. Create a script with the name stMotorFill, type - Object and execution type - OnDataChange:




The 'Script properties' dialog box is shown with the following settings:

- Group: (empty dropdown)
- Subgroup: (empty dropdown)
- Name: stMotorFill
- Comment: (empty text box)
- Background color: Light Gray
- Script type: Object
- Language: ST(Structured text)
- Dimension: 800 X 600
- ☐ Every cycle
- Execution: OnDataChange
- ☒ Run in UI

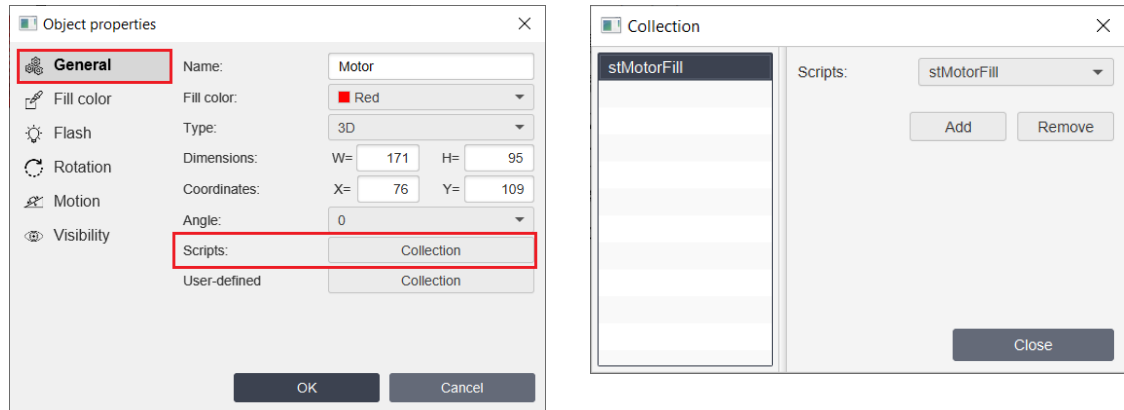
Buttons: OK, Cancel

4. Let's write the script::

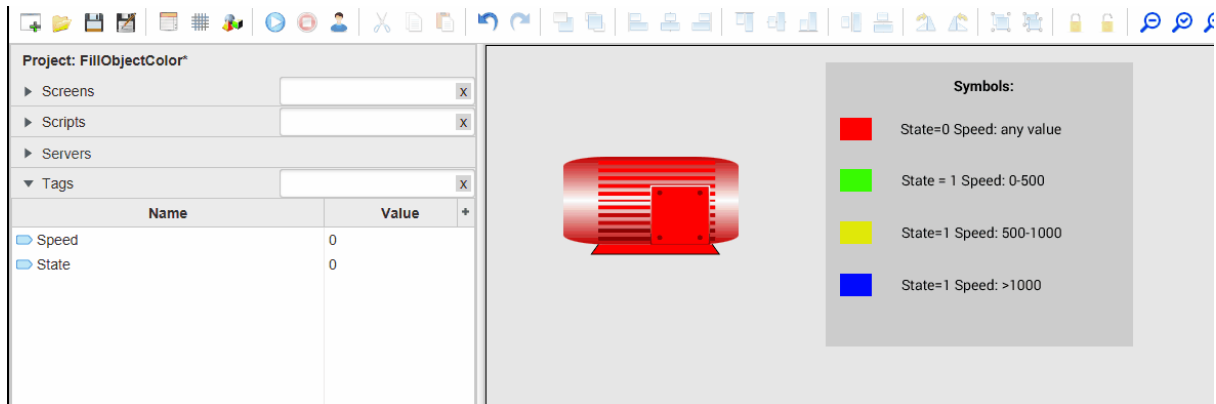
```
1 if (Tags.State==0) // if State tag's value equal 0 fill color of the Motor is RED
2 {
3     Objects.Motor.fillcolor = Color.RED;
4 }
5 else{
6     // else
7     if (Tags.Speed>=0 && Tags.Speed<=500){ // if Speed tag's value between 0 and 500 fill color is Green
8         Objects.Motor.fillcolor = Color.GREEN;
9     }
10    else if (Tags.Speed>500 && Tags.Speed<=1000){
11        Objects.Motor.fillcolor = Color.YELLOW; //if Speed tag's value between 500 and 1000 fill color is Yellow
12    }
13    else
14    {
15        Objects.Motor.fillcolor = "0x0000FFFF"; // else fill color is Blue
16    }
17 }
```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

5. Now let's bind the script to our Motor object, go to the object's properties (General tab) and add our script to the "Scripts" field:



6. Let's [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

9.1.4 Complex color change

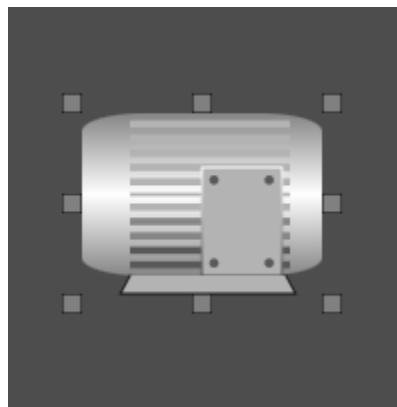
Consider the following example: you have large number of objects of the same type (motor), which have several operating modes (State), and you need to display the motor on the screen with color depending on the set operating mode.

Because we have many objects of the same type, we will use indirect names to bind tags based on user-defined properties.

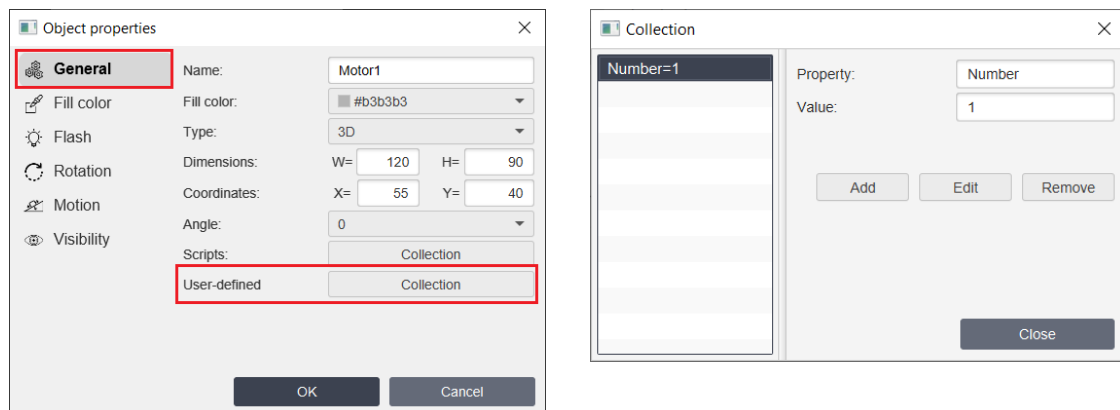
1. First, let's create tags (by the number of objects).

Tags		
		X
Name	Value	+
State1	0	
State2	0	
State3	0	
State4	0	
State5	0	

2. Let's create a graphical Motor object for our example:

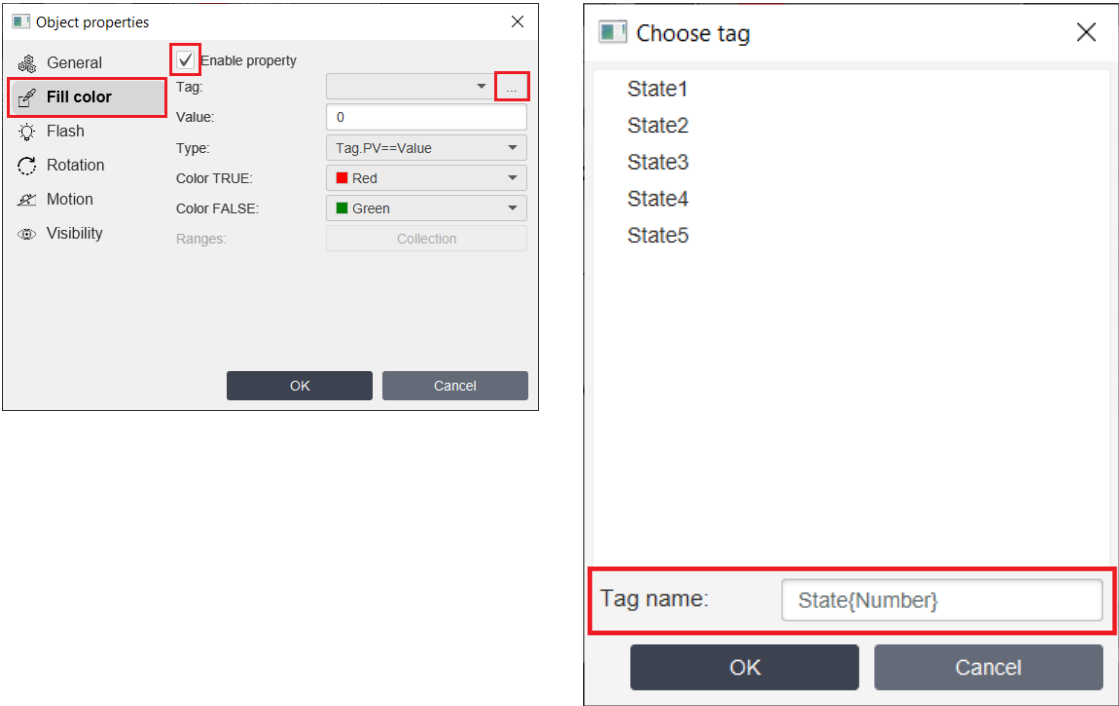


3. In the Object properties, set up the user-defined property "Number:" and set its value "1", because We will bind Motor1 to State1:



Click "OK" to save this user-defined property.

4. Next, bind the object to State1. Open the Object properties window again and select the "Fill Color" tab. Then in the "Tag" field (click on the "..." button) and in the window that opens in the "Tag Name" field we set State{Number} , where "Number" is our user-defined property (the value of which we set to "1" for the first object):

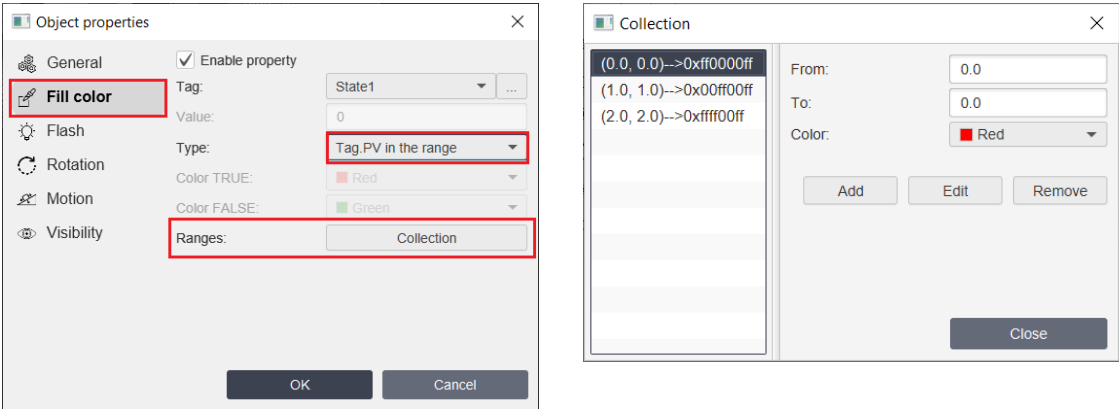


5. Let's make sure that our object is bound to State1 (save the Object Properties by clicking "OK") and open the "Object Properties" window again.

6. So, the "Fill Color" property is bound to the State1 tag. Now let's set the Color of the object depending on the value of this tag:

State1	Color
0	RED
1	GREEN
2	YELLOW

Select the Type **"TagPV in range"** and set the colors for the tag values:



Now we have a Motor -object with the Fill Color property set.

7. Now we need to create the same objects with the same settings. Because We used indirect names based on user-defined properties to bind tags, we do not need to set the Fill Color property for each new object (there is no need to set ranges for each object). We just need to duplicate the Motor ("Duplicate") and bind it to the tag by specifying the value of the Number user-defined property that corresponds to the tag. The fastest way to do this is in the Property Sheet

Confirm the changes and close the object properties window by clicking OK. To copy this motor and bind the fill color property to the tags - State2 and State3 you don't need to configure the fill color property for each Motor, you only need to duplicate the Motor:



And change in the Property Sheet: the value of the user-defined property "Number" depending on which tag you want to bind the object to:

Screen: Scre... **Object: Motor2**

Search

▼ 01.General

Name: Motor2

Fill color: Gray

Type: 3D

Width: 91.0

Height: 74.0

Position X: 147.0

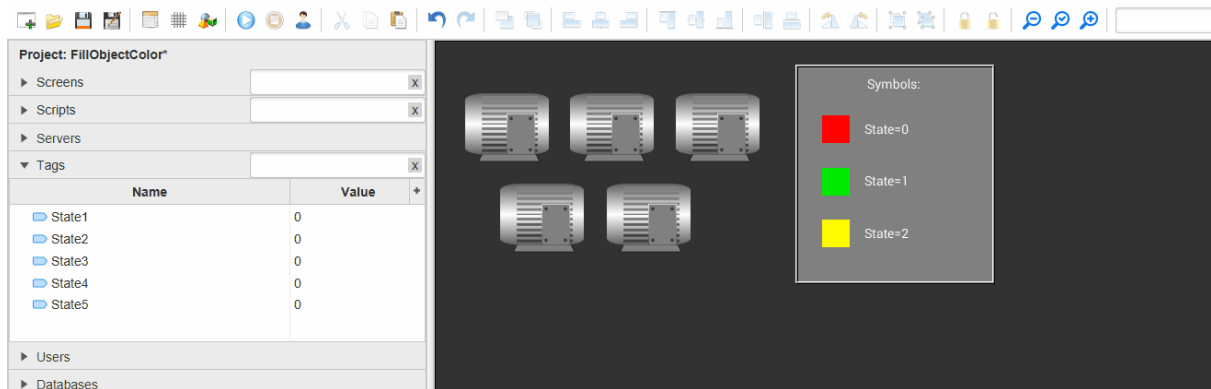
Position Y: 57.0

Angle: 0

Scripts: Collection

Number: 2

8. Let's [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

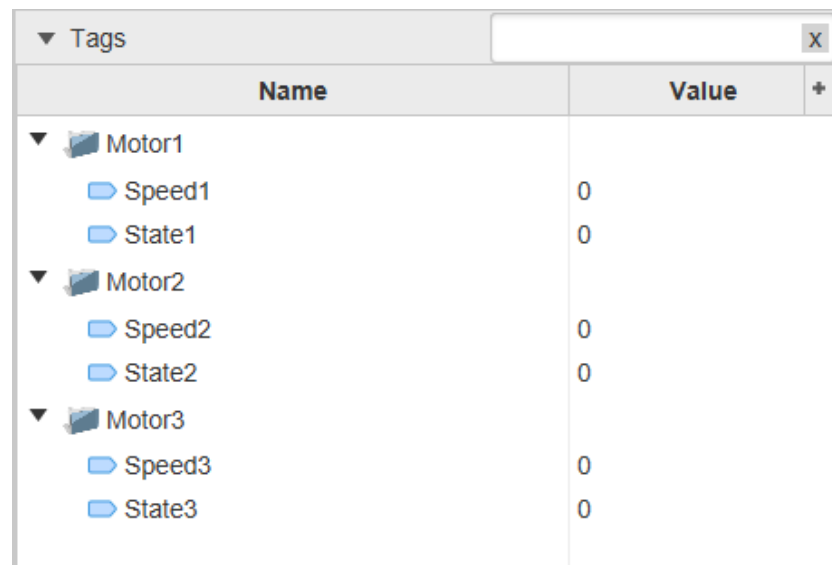
9.1.5 Complex color change with scripts

Suppose you have many objects (motors) of the same type, each of which has 2 parameters (state and speed). You need to change the color of an object depending on its state and speed.

We already know that if the color changes depending on the values of several tags, we need to use scripts; and if we use objects of the same type, in order to simplify the binding of duplicated objects to tags, we need to use indirect names based on user-defined properties.

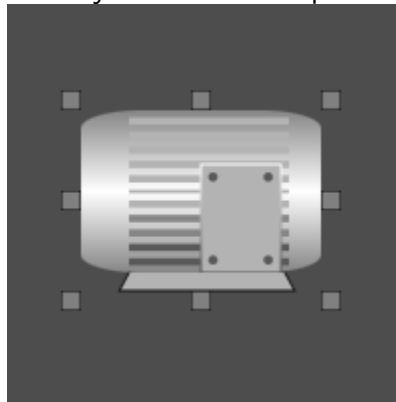
Let's look at an example.

1. Create 2 tags (State and Speed) for each Motor object. For convenience, we'll do this as a group, and then copy the group by the number of objects in the project:

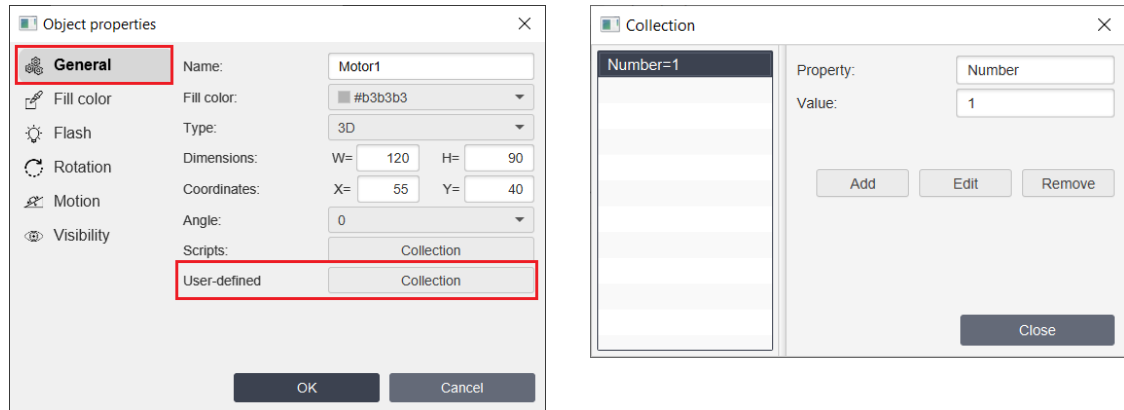


▼ Tags	
Name	Value
▼ Motor1	
Speed1	0
State1	0
▼ Motor2	
Speed2	0
State2	0
▼ Motor3	
Speed3	0
State3	0

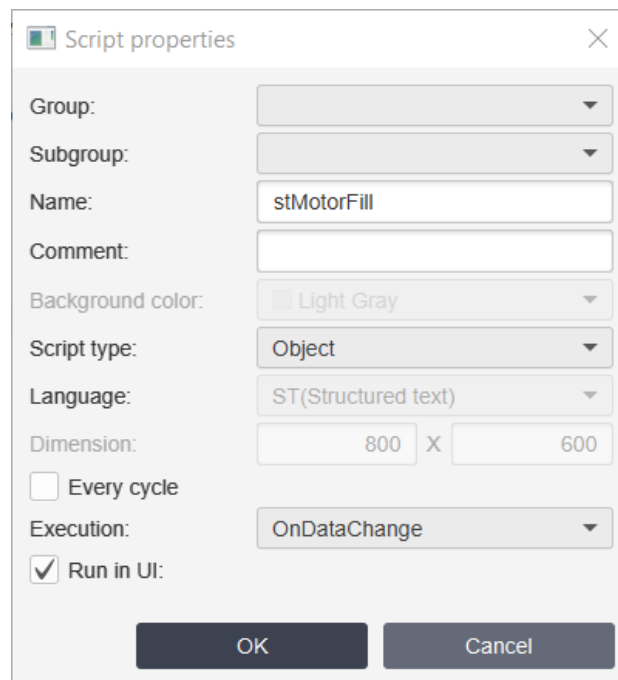
2. Let's create a graphical Motor object for our example:



Let's open the Object properties window, create user-defined property - "Number" with the value "1", because We will bind this object to the State1 Speed1 tags:



3. Now we need to create a script for an object in the ST language with an execution type - onDataChange:



Depending on tag's values for every Motor object use fill color:


State	Speed	Color
0	Any	RED
1	0...500	GREEN
1	500...1000	YELLOW
1	>1000	BLUE

Let's write our script:

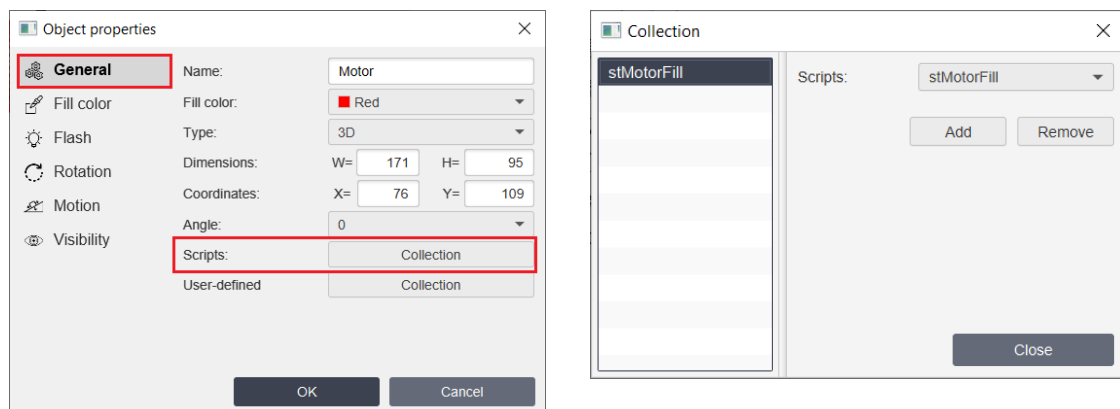
```

1 string statetagname = "State"+Objects.this.Number;// get tag's name by using State + Number user-defined property
2 string speedtagname = "Speed"+Objects.this.Number;// we use indirect name to have possibility to use the same script for other objects
3
4 byte state = gettagvalue(statetagname, 0); //get tag's values
5 int speed = gettagvalue(speedtagname, 0);
6
7 if (state==0){
8     Objects.this.fillcolor = Color.RED; // if state=0 change fill color of the object to RED
9 }
10 else if (state==1){
11     //if state=1 fill color will change depending on speed
12     //if speed = 0...500 fill color equal green
13     if (speed>=0 && speed<=500){
14         Objects.this.fillcolor = Color.GREEN;
15     }
16     else if (speed>500 && speed<=1000){
17         // speed between 500 and 1000 color equal yellow
18         Objects.this.fillcolor = "0xFFFF00FF";
19     }
20     else{
21         Objects.this.fillcolor = Color.BLUE; // when speed is other color equal blue
22     }
23 }


```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

4. Let's bind our script to the object: open the properties of the Motor object, then - Scripts/Collection and bind our script:





Now we have a Motor with the Fill Color property configured in the script.


5. Now let's duplicate the object as many times as needed for the project. Because we used a custom property, we don't need to customize the script for each Motor. We only need to duplicate the Motor and change the value of the user-defined property in the [Property sheet](#) :




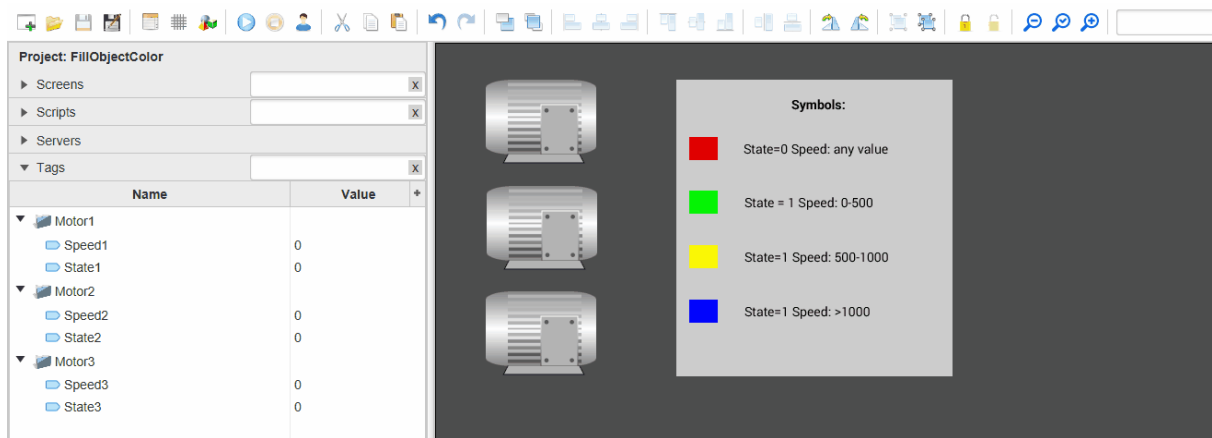
Screen: Scre... **Object: Motor2**

  Search

▼ 01.General

Name:	Motor2
Fill color:	 Gray
Type:	3D
Width:	91.0
Height:	74.0
Position X:	147.0
Position Y:	57.0
Angle:	0
Scripts:	Collection
Number:	2

6. Let's [Run simulation](#)  to check the settings:



You can download this project [here](#).

9.2 Object flashing

Let's look at the most common case, where you want an object to flash and change its frequency when the value of its associated variable changes. Below you can find several examples from simple to more complex with scripts:

- [Simple flashing](#) ⁵⁴⁴
- [Simple multiple flashing](#) ⁵⁴⁶
- [Complex flashing with scripts](#) ⁵⁴⁷

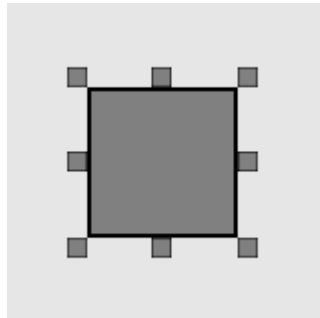
9.2.1 Simple flashing

Let's assume that in our project there is a certain object with a certain parameter (tag). We want the object to flash at 1000ms if the tag value is not "0".

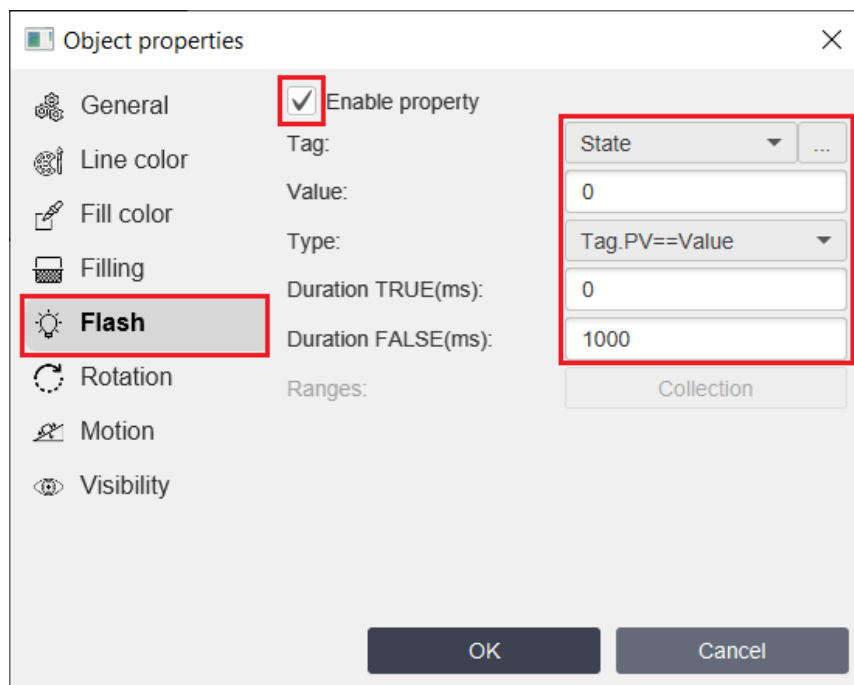
1. Create a tag named State:



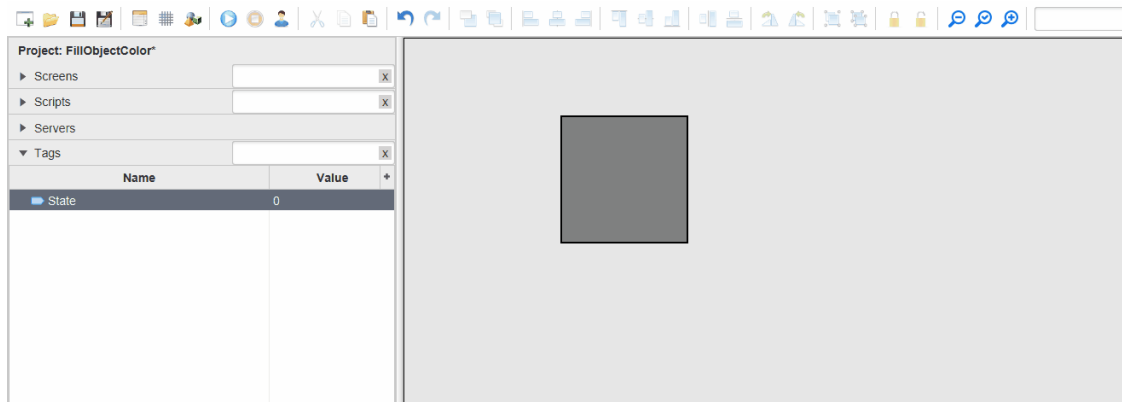
2. Let's create an object, let it be a rectangle (instead of a Rectangle there may be another object that is more suitable for your project):



3. Now let's set the Flash property. Let's open the Object properties, go to the "Flash" tab and configure it as we planned above (bind it to State1, set the tag value = 0, comparison type "Tag.PV==Value"). Now, if the State tag value ==0, the result comparison TRUE, set the flash duration =0 (the object does not flash). If the value of the State tag is !=0, the comparison result is FALSE, set the flash duration to 1000ms (the object flashes with a frequency of 1000ms):



4. Let's [Run simulation](#) ⁷⁰ to check the settings:

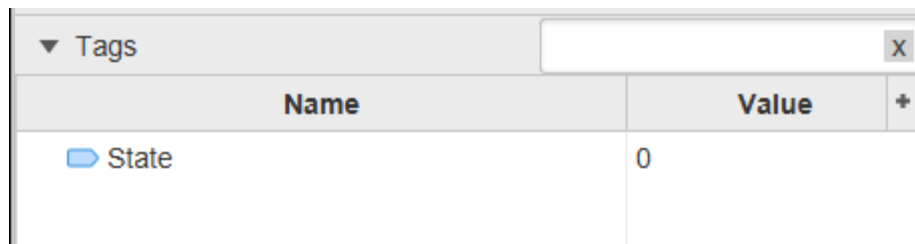


You can download this project [here](#).

9.2.2 Simple multiple flashing

Let's assume we want an object to flash at multiple tag values and with different flash duration. In this case, we need to use the comparison type **"Tag.PV in range"**. Let's look at an example.

1. Create a tag - State, which is responsible for the operating mode of the valve:



2. Create an ISA Valve object:

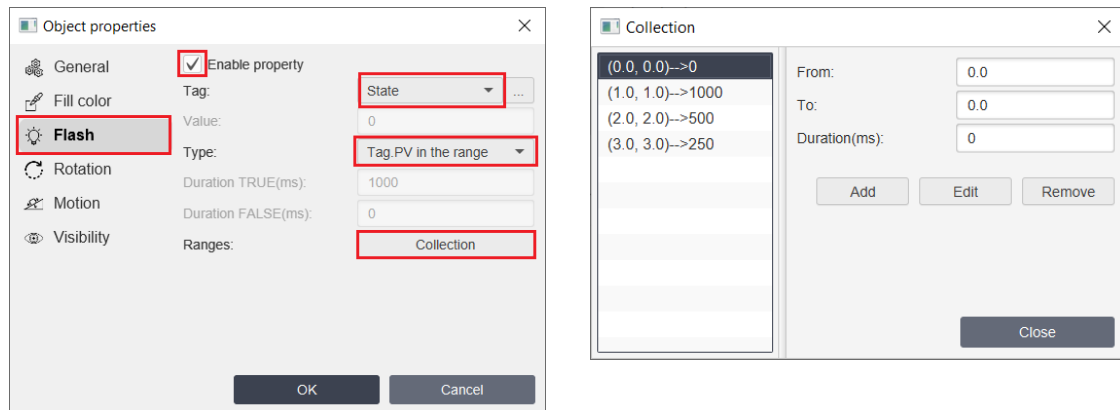


3. Set the Flash property as follows:

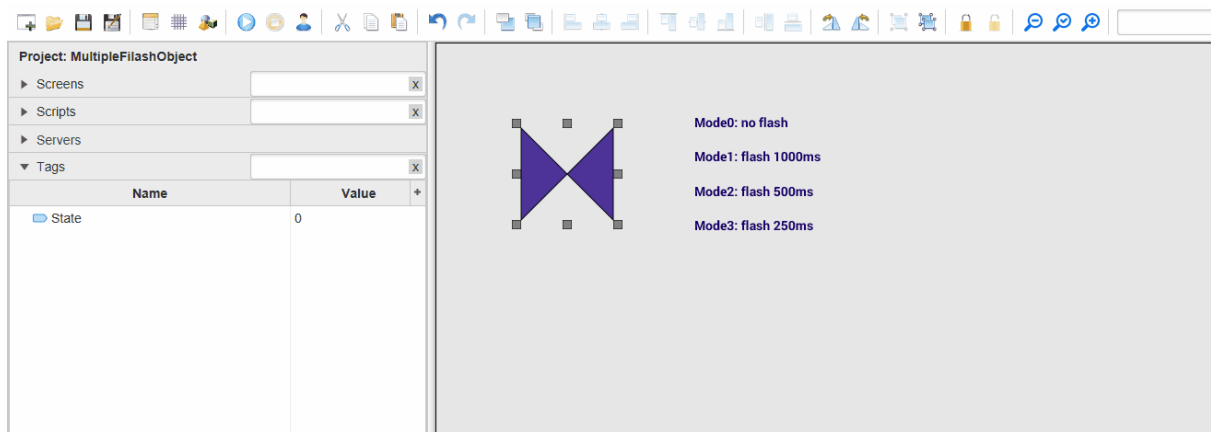
State	Flash
0	0

State	Flash
1	1000
2	500
3	250

To do this, open Object properties on the Flash tab and set the flash ranges:



4. Let's [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

9.2.3 Complex flashing with scripts

Let's assume that in our project we have many Motor objects of the same type, each of which has several State and Speed parameters, and depending on their values, we want the objects (Motor) to flash/not flash on the screen. Since in this case there is a dependence of flashing on several tags, it is necessary to use scripts. And since we have several objects of the same type in our project, it is more convenient to use indirect names to bind tags to an object.

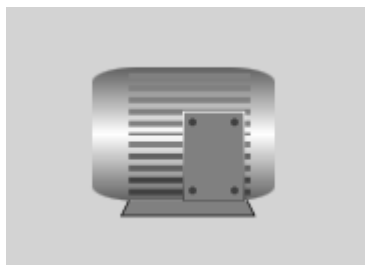
1. First, let's create tags for each object - State and Speed (we'll arrange them as a group):

Tags	
Name	Value
▼ Motor1	
Speed1	0
State1	0
▼ Motor2	
Speed2	0
State2	0
▼ Motor3	
Speed3	0
State3	0

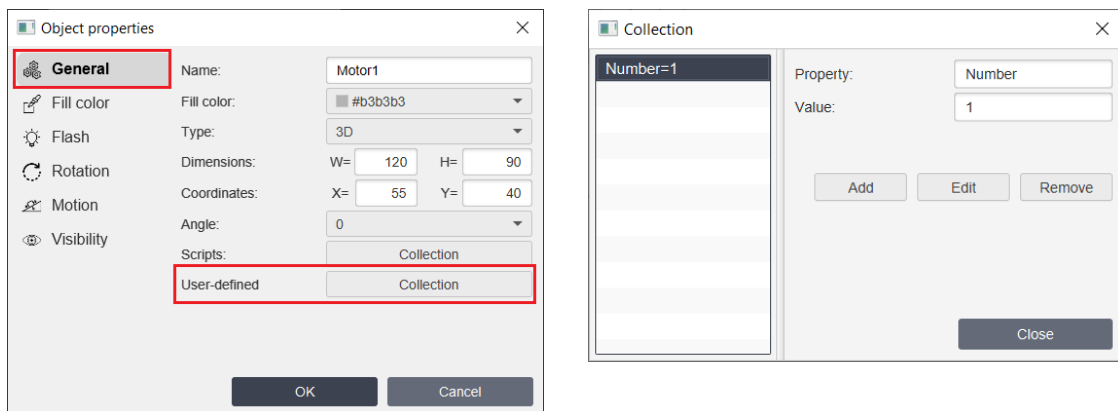
2. In this case, we need intermediate tags to enable or disable the flashing of an object.
- Flash1, Flash2 and Flash3, let's create them:

Tags	
Name	Value
▼ Motor1	
Flash1	false
Speed1	0
State1	0
▼ Motor2	
Flash2	false
Speed2	0
State2	0
▼ Motor3	
Flash3	false
Speed3	0
State3	0

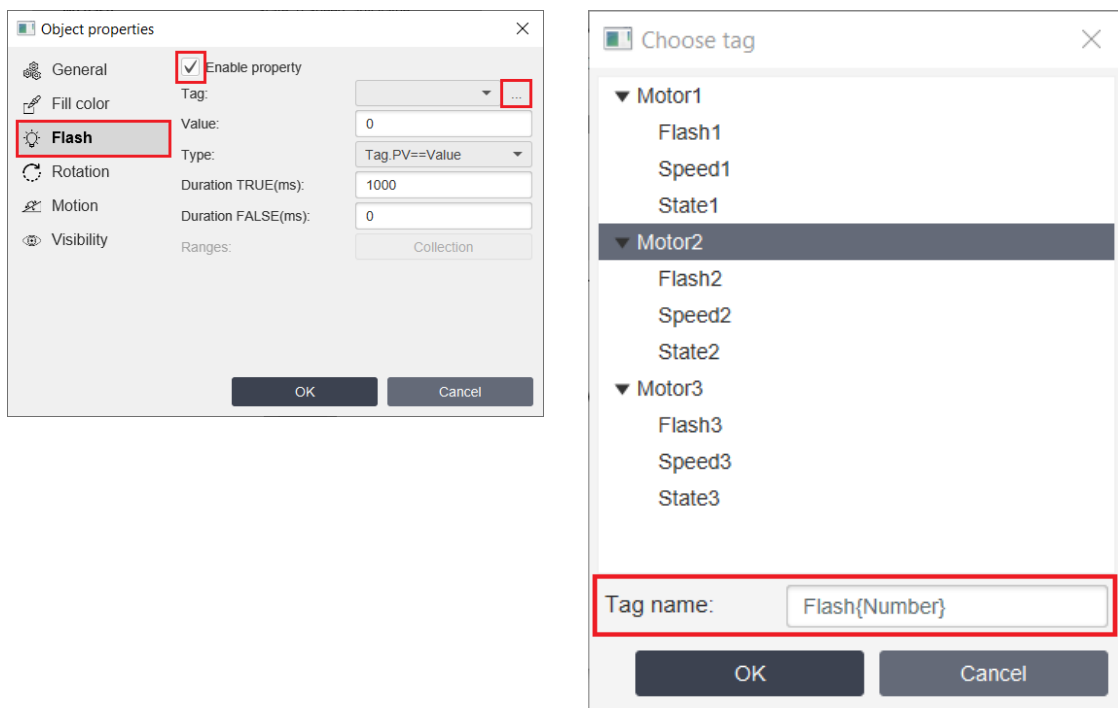
3. Let's create a graphical Motor object for our example:



In the Object properties in the "General" tab, create a user-defined property "Number" and set its value "1", because we will bind this object to the Flash1 tag:



4. Close the "Collections" window and click "OK" in the Object Properties Window to save the changes. Now let's open the Object Properties again and configure the Flash property: bind the tag by specifying the indirect name: Flash{Number}:



5. Depending on the tag values for each Motor object, we will use the frequency:

State	Speed	Flash frequency
0	Any	Not flashing
1	0...500	1000

State	Speed	Flash frequency
1	500...1000	500
1	>1000	250

6. Let's create a script for an object in the ST language with the execution type - onDataChange:

Script properties

Group:

Subgroup:

Name:

Comment:

Background color:

Script type:

Language:

Dimension: X

☐ Every cycle

Execution:

☒ Run in UI:


OK Cancel

Let's write a script:

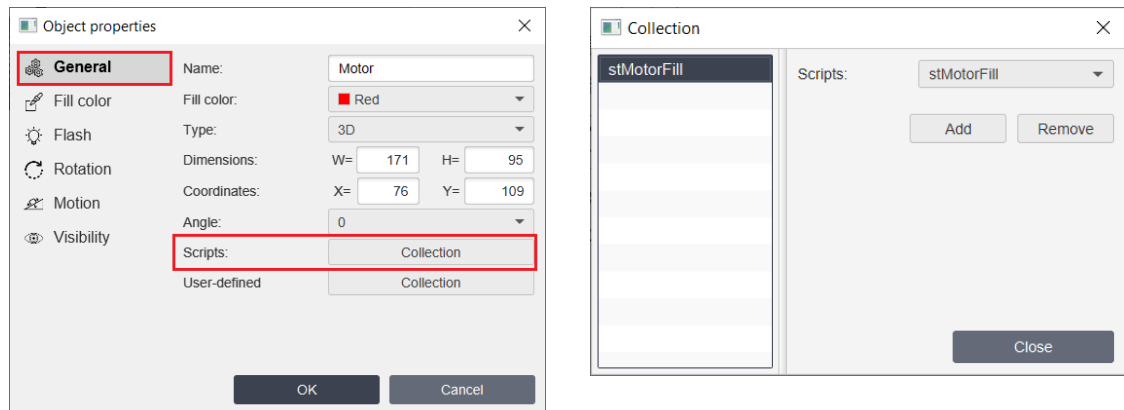
```

1 string statetagnam = "State"+Objects.this.Number; // get tag's name by using State + Number user-defined property
2 string speedtagname = "Speed"+Objects.this.Number; // we use indirect name to have possibility to use the same script for other objects
3 string flashtagname = "Flash"+Objects.this.Number;
4
5 byte state = gettagvalue(statetagnam, 0); //get tag's values
6 int speed = gettagvalue(speedtagname, 0);
7
8 if (state==0){
9     settagvalue(flashtagname, false); // if state=0 change flashing to false
10 }
11 else if (state==1){
12     settagvalue(flashtagname, true); //if state=1 change flashing to true and depending on speed frequency of flash
13     if (speed>0 && speed<=500){ //if speed = 0...500 frequency=1000
14         Objects.this.trueflashduration = 1000;
15     }
16     else if (speed>500 && speed<=1000){ // speed between 500 and 1000 frequency 500
17         Objects.this.trueflashduration = 500;
18     }
19     else{
20         Objects.this.trueflashduration = 250; // when speed is other frequency is 250
21     }
22 }

```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

7. Now let's bind the script to the object - open the properties of the object, select the "Main" tab and fill in the "Collection" in the "Scripts" field:



So, we have a Motor with the Flash property set according to the script.

8. Now we need to duplicate the created Motor object as many times as needed in the project, and in each newly created object correct the value of the user-defined property "Number" ((most quickly, on the Property Sheet):



Screen: Scre... **Object: Motor2**

Search

▼ 01.General

Name: Motor2

Fill color: Gray

Type: 3D

Width: 91.0

Height: 74.0

Position X: 147.0

Position Y: 57.0

Angle: 0

Scripts: Collection

Number: 2

9. Let's [Run simulation](#) ⁷⁰ to check the settings:

Project: FillObjectColor

Screens

Scripts

Servers

Tags

Name	Value
Motor1	
Flash1	false
Speed1	0
State1	0
Motor2	
Flash2	false
Speed2	0
State2	0
Motor3	
Flash3	false
Speed3	0
State3	0

Symbols:

No flash State=0 Speed: any value

Flash with duration 1000ms State = 1 Speed: 0-500

Flash with duration 500ms State=1 Speed: 500-1000

Flash with duration 250ms State=1 Speed: >1000

You can download this project [here](#).

9.3 Object visibility

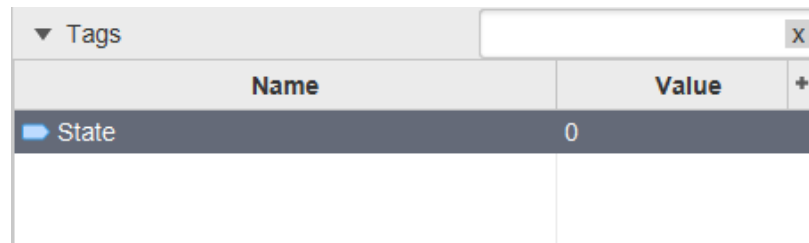
Let's look at the most common cases when you need to adjust the visibility of an object (make an object visible/invisible) if the value of a variable associated with it changes. Below you can find several examples from simple to more complex with scripts:

- [Simple visibility](#) ⁵⁵³
- [Complex visibility with scripts](#) ⁵⁵⁴

9.3.1 Simple visibility

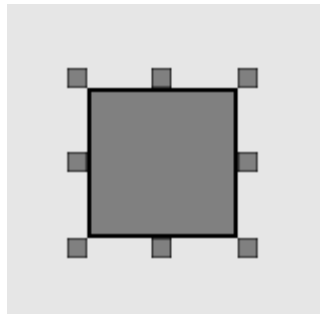
Let's assume that in our project there is a certain object with a certain parameter (tag). We want the object to be invisible unless the tag value is "0".

1. Create a tag named State:

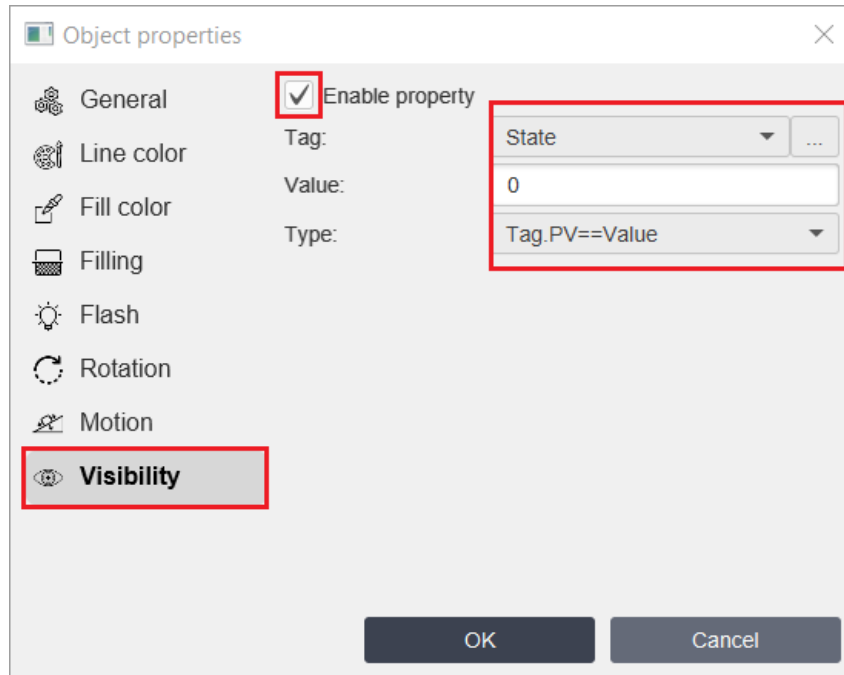


▼ Tags		
Name	Value	+
State	0	

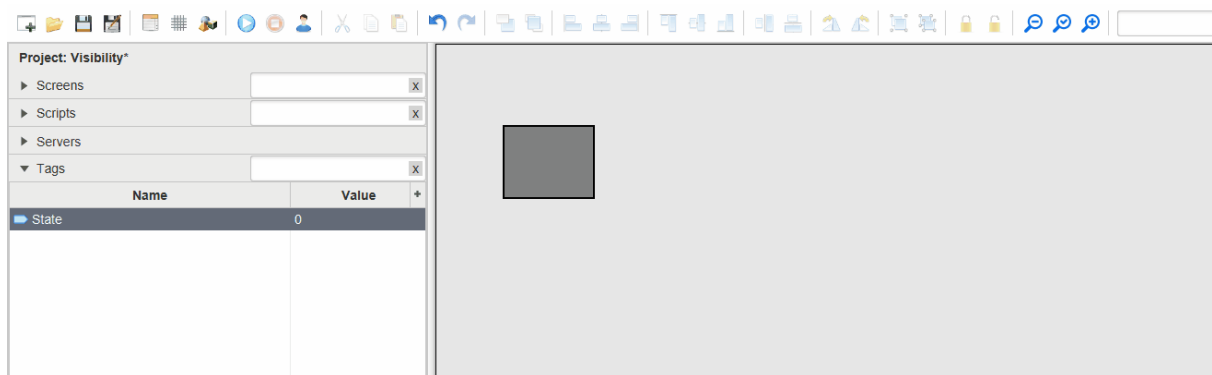
2. Create a Rectangle object (there may be other objects instead of a Rectangle):



3. Set up the "Visibility" property (enable the property, bind the State tag, the value of which determines the visibility of the object, set the tag value to "0" and the comparison type "Tag.PV==Value"). The object will be visible if the value of our State tag is 0, and invisible if the tag value is not 0:



4. Let's [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

9.3.2 Complex visibility with scripts

Let's assume that in our project we have many Motor objects of the same type, each of which has several State and Speed parameters, and depending on their values, we want the objects to be visible/not visible on the screen. Since in this case there is a dependence of Visibility on several tags, it is necessary to use scripts. And since we have several objects of the same type in our project, it is more convenient to use indirect names to bind tags to an object.

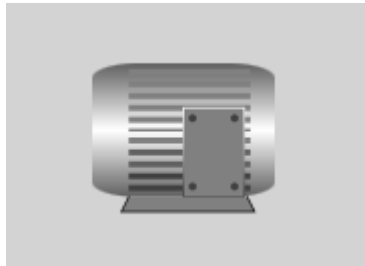
1. Let's create tags for each object - State and Speed (use grouping):

▼ Tags		
Name		Value
▼ Motor1		
Speed1		0
State1		0
▼ Motor2		
Speed2		0
State2		0
▼ Motor3		
Speed3		0
State3		0

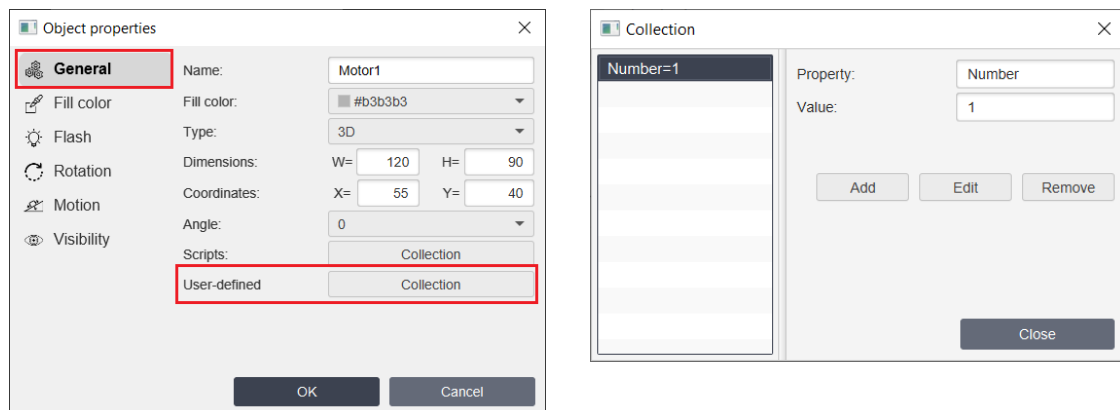
2. In this case, we need intermediate tags to enable or disable the visibility of an object - Visible1, Visible2 and Visible3, let's create them:

▼ Tags		
Name		Value
▼ Motor1		
Speed1		0
State1		0
Visible1		false
▼ Motor2		
Speed2		0
State2		0
Visible2		false
▼ Motor3		
Speed3		0
State3		0
Visible3		false
▼ Motor4		
Speed4		0
State4		0
Visible4		false

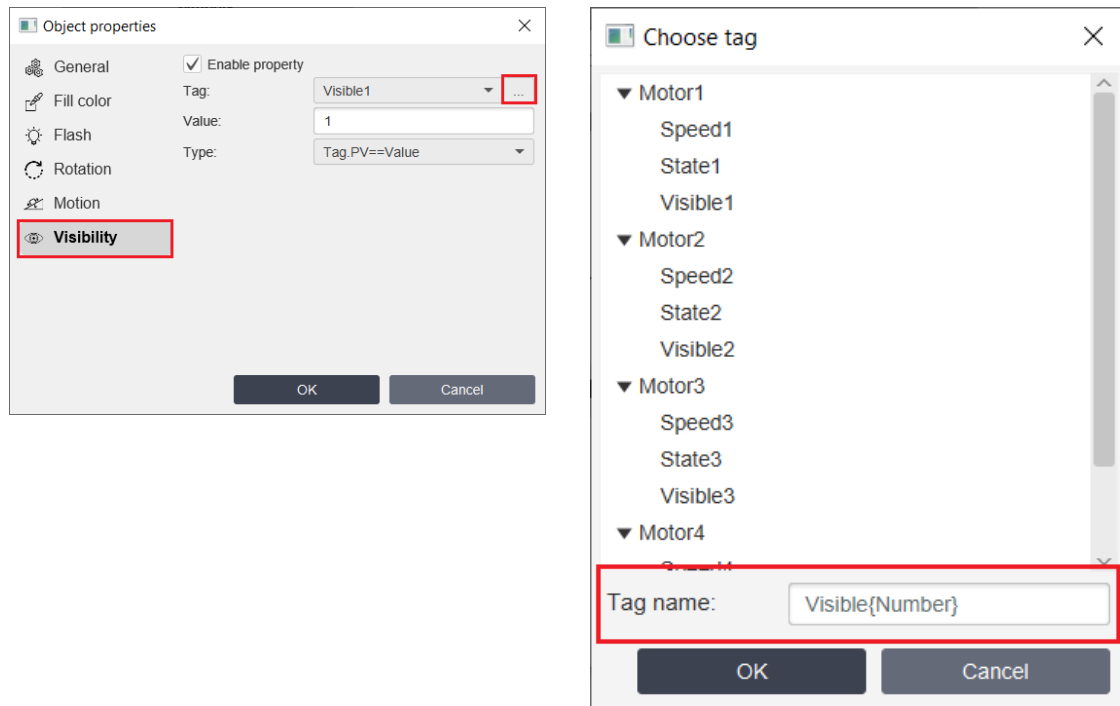
3. Create a Motor object:



Let's create a user-defined property - Number and set its value to "1", because We will bind the first object to the Visibility1 tag:



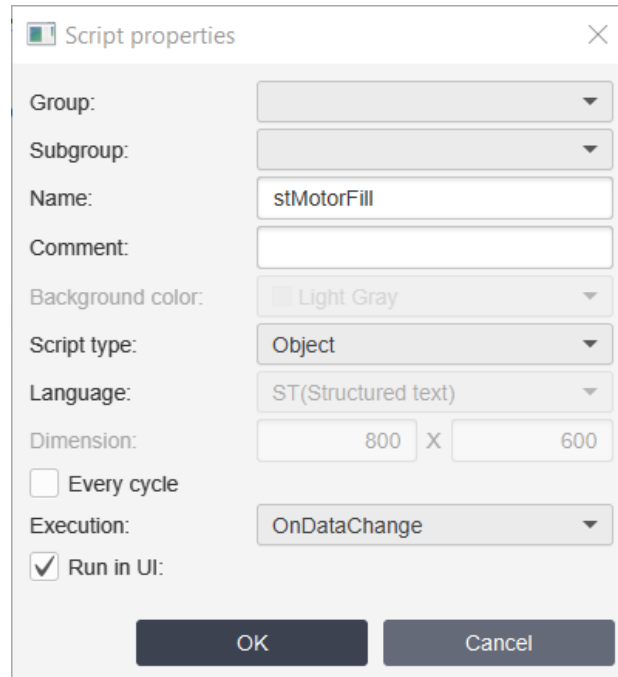
4. Close the "Collections" window and click "OK" in the Object Properties Window to save the changes. Now let's open the Object Properties again and set up the Visibility property: enable the property, set the value = 1, type "Tag.PV==value", bind the tag by specifying an indirect name: Visible{Number}:



5. We will set visibility depending on the tag values for each Motor object:

State	Speed	Visibility
0	Any	Not visible
1	0...500	Visible
1	500...1000	Visible
1	>1000	Not visible

6. Now let's create a script with type "object" in the ST language with the execution type
- onDataChange:



The 'Script properties' dialog box is shown with the following settings:

- Group: (empty dropdown)
- Subgroup: (empty dropdown)
- Name: `stMotorFill`
- Comment: (empty text box)
- Background color: `Light Gray`
- Script type: `Object`
- Language: `ST(Structured text)`
- Dimension: `800` X `600`
- ☐ Every cycle
- Execution: `OnDataChange`
- ☒ Run in UI


Buttons: OK, Cancel

Let's write a script:

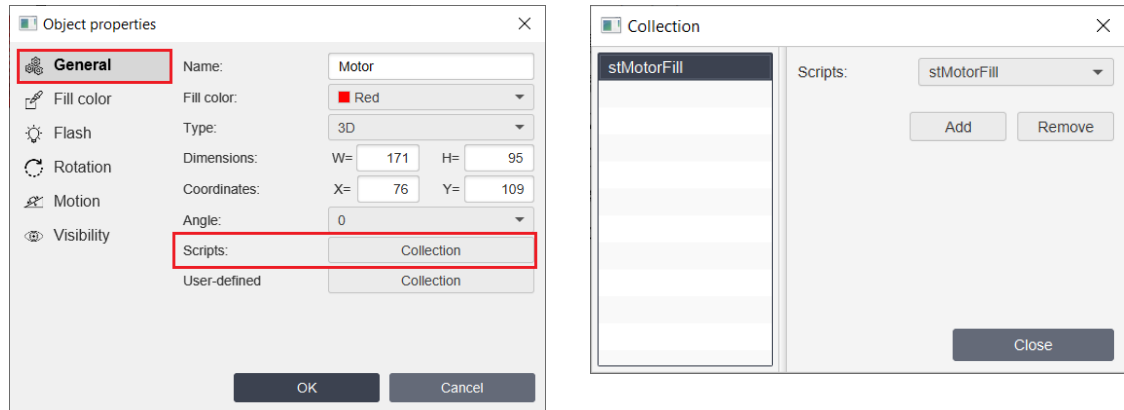
```

1 string statetagnam = "State"+Objects.this.Number;// get tag's name by using State + Number user-defined property
2 string speedtagnam = "Speed"+Objects.this.Number;// we use indirect name to have possibility to use the same script for other objects
3 string visibletagnam = "Visible"+Objects.this.Number;
4
5 byte state = gettagvalue(statetagnam, 0); //get tag's values
6 int speed = gettagvalue(speedtagnam, 0);
7
8 if (state==0){
9     settagvalue(visibletagnam, false); // if state=0 make object invisible
10 }
11 else if (state==1){
12     if (speed>=0 && speed<=500){ //if speed = 0...500 make object visible
13         settagvalue(visibletagnam, true);
14     }
15     else if (speed>500 && speed<=1000){ // speed between 500 and 1000 make object visible
16         settagvalue(visibletagnam, true);
17     }
18     else{
19         settagvalue(visibletagnam, false); // when speed is other make object invisible
20     }
21 }
22 }

```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

7. Now let's bind the script to the object - open the properties of the object, select the "General" tab and fill in the "Collection" in the "Scripts" field:



So, we have a Motor with the visibility property set by script.

8. Now we need to duplicate the created Motor object as many times as needed in the project, and in each newly created object correct the value of the "Number" user-defined property in the Property Sheet:



Screen: Scre... **Object: Motor2**

Search

▼ 01.General

Name: Motor2

Fill color: Gray

Type: 3D

Width: 91.0

Height: 74.0

Position X: 147.0

Position Y: 57.0

Angle: 0

Scripts: Collection

Number: 2

9. Let's [Run simulation](#) to check the settings:

Project: FillObjectColor

Screens

Scripts

Servers

Tags

Name	Value
Motor1	
Speed1	0
State1	0
Visible1	false
Motor2	
Speed2	0
State2	0
Visible2	false
Motor3	
Speed3	0
State3	0
Visible3	false
Motor4	
Speed4	0
State4	0
Visible4	false

Symbols:

Invisible	State=0 Speed: any value
Visible	State = 1 Speed: 0-500
Visible	State=1 Speed: 500-1000
Invisible	State=1 Speed: >1000

You can download this project [here](#).

9.4 Change the text of an object

Let's look at the most common cases when you need to change the text of an object, depending on the value of the variable associated with it. We will use the properties

"Output text", "Input value"). Below you can find several examples from simple to more complex with scripts:

- [Simple text change](#)^[561] (based on the Text Input property);
- [Simple multiple text change](#)^[562] (based on the Text Input property);
- [Display tag's value](#)^[564] (based on the Text Input property);
- [Enter tag's value](#)^[566] (based on the Output value property);
- [Complex text change with scripts](#)^[568]

9.4.1 Simple text change

Let's assume we have an object containing some text, and we want the text to change depending on the value of the tag. In this way we can display inscriptions about the operating mode or state of the tag.

1. Create a tag named State:

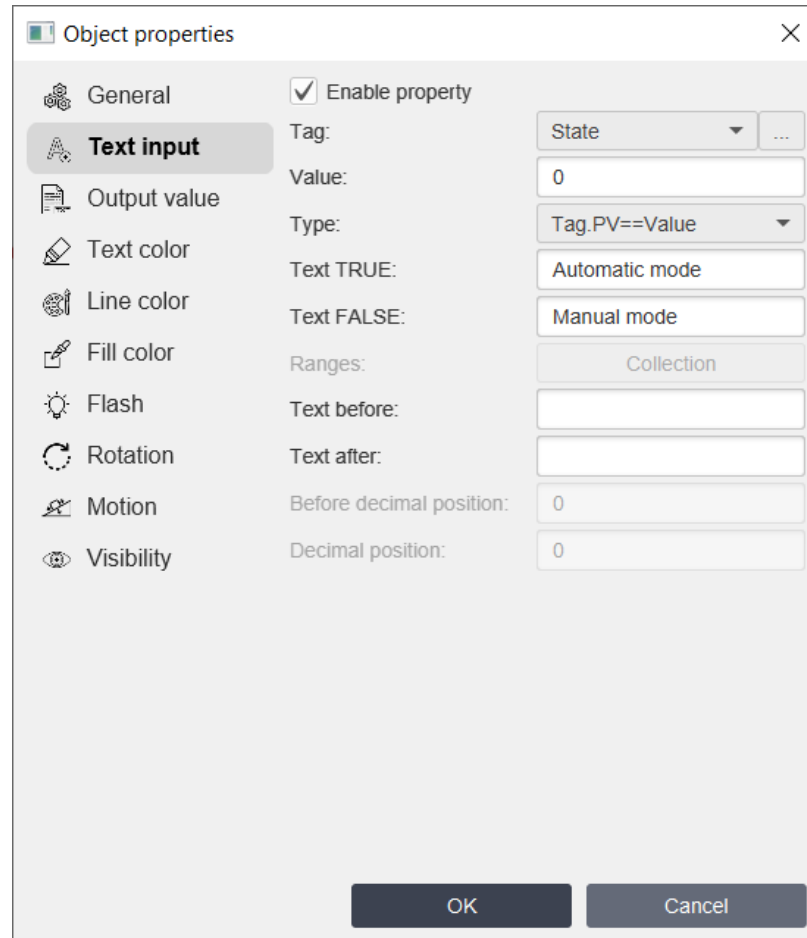


Name	Value
State	0

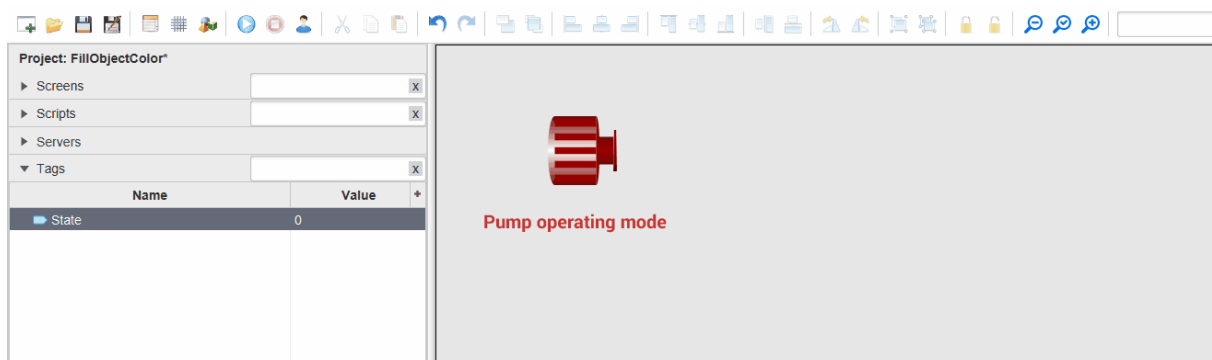
2. Create a Text/EditField object (other objects with the Text Input property can be used):



3. Set up the "Text Input" property. For example, we want to have the text "Automatic Mode" if the tag value (State) is 0, and the text "Manual Mode" if the tag value is not 0. In the Object Properties window, go to the "Text Input" tab, enable the property, bind the tag, set the value to "0", comparison type "TagPV==Value", fill the text with TRUE and FALSE:



4. Let's [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

9.4.2 Simple multiple text change

Suppose we have some text that, depending on the value of the tag, should have different content. In this case, you need to select the "**Tag.PV in range**" type in the "Text Input" property.

1. Let's create a tag - State, which is responsible for the state of a certain device:



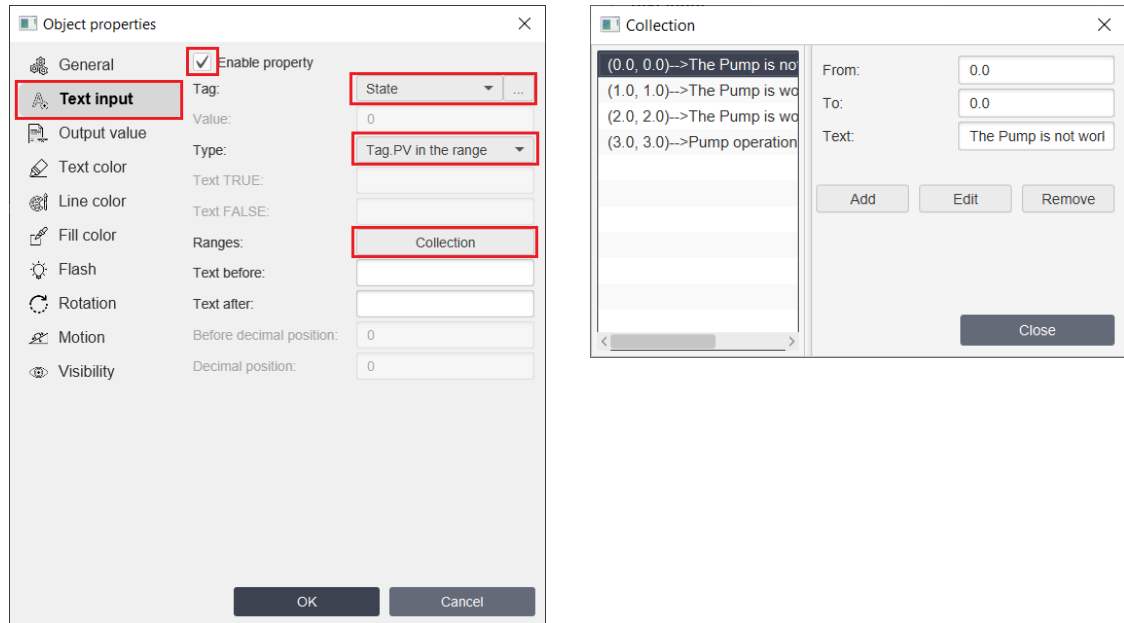
2. Create a Text/EditField object (other objects with the Text Input property can be used):



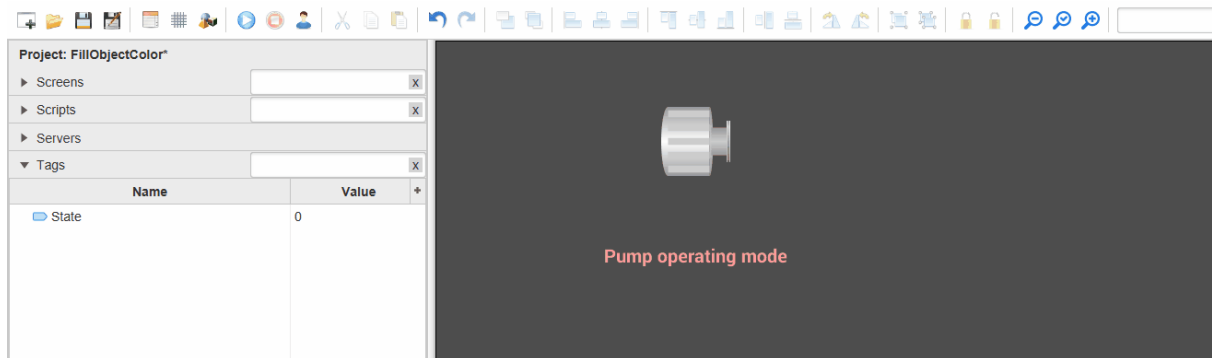
3. Let the "Text Input" property have the following conditions::

State	Text
0	The Pump is not working
1	The Pump is working
2	The Pump is working uncorrectly
3	Pump operation accident

In the Object properties, go to the "Text Input" tab, enable the property, bind a tag, select the comparison type "**Tag.PV in a range**" and fill the ranges in the "Collections":



4. Let's [Run simulation](#) ⁷⁰ to check the settings:

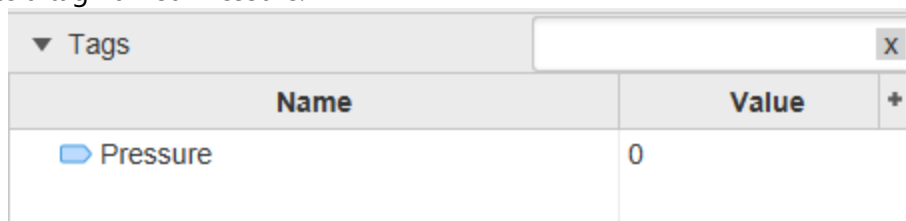


You can download this project [here](#).

9.4.3 Display tag's value

Let's assume we want to display the value of some tag on the screen (the pressure level in the tank).

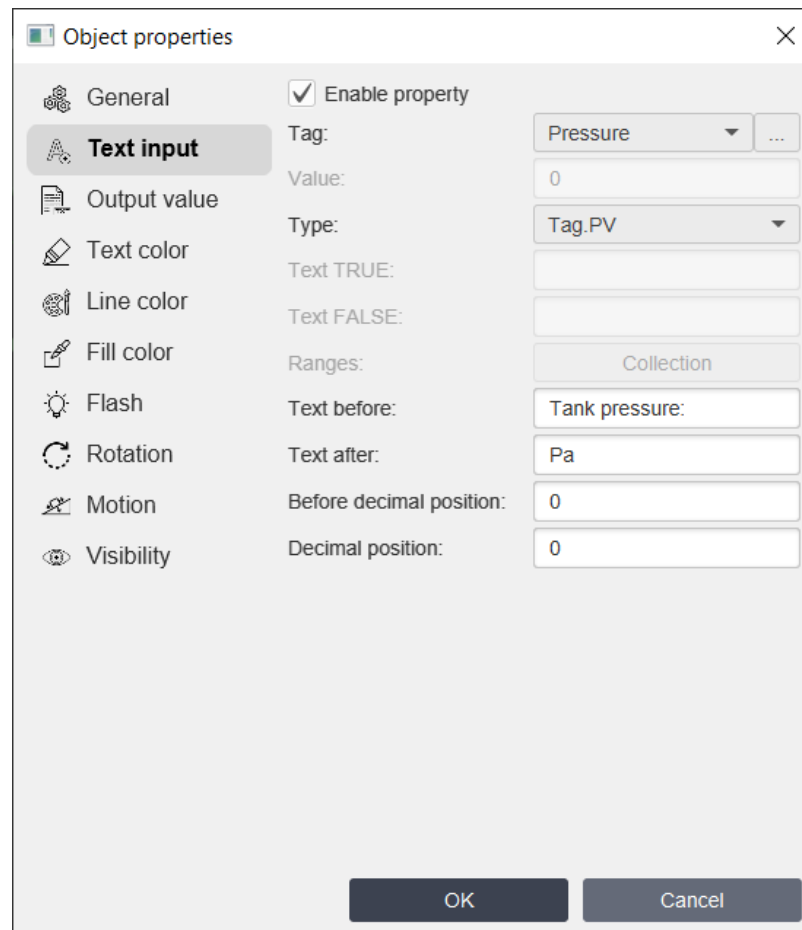
1. Create a tag named Pressure:




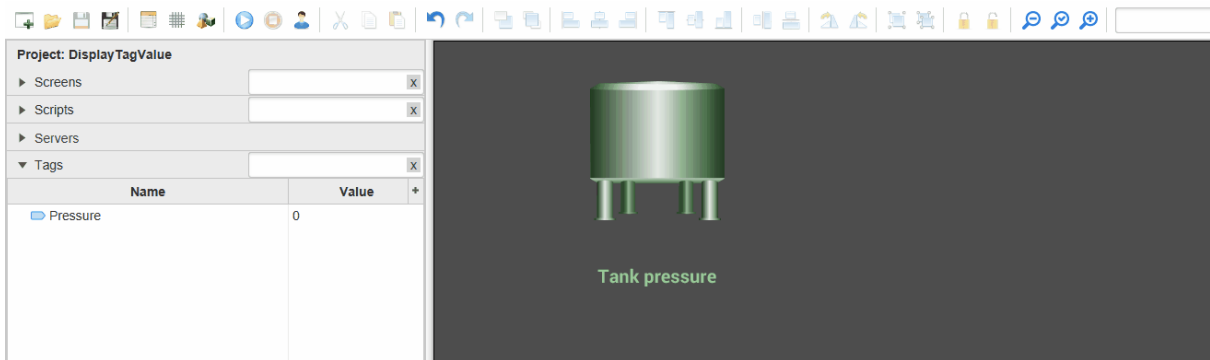
2. Create a Text/EditField object (other objects with the Text Input property can be used):



3. Set up the "Text Input" property, bind the Pressure tag, select the **"Tag Value"** type, fill in the "Text before" and "Text after" fields::



4. Let's [Run simulation](#)  to check the settings:



You can download this project [here](#).

9.4.4 Enter tag's value

In this example, we want to show how we can set the value of a tag using the Output Value property, and display the specified value on the screen using the Text Input property.

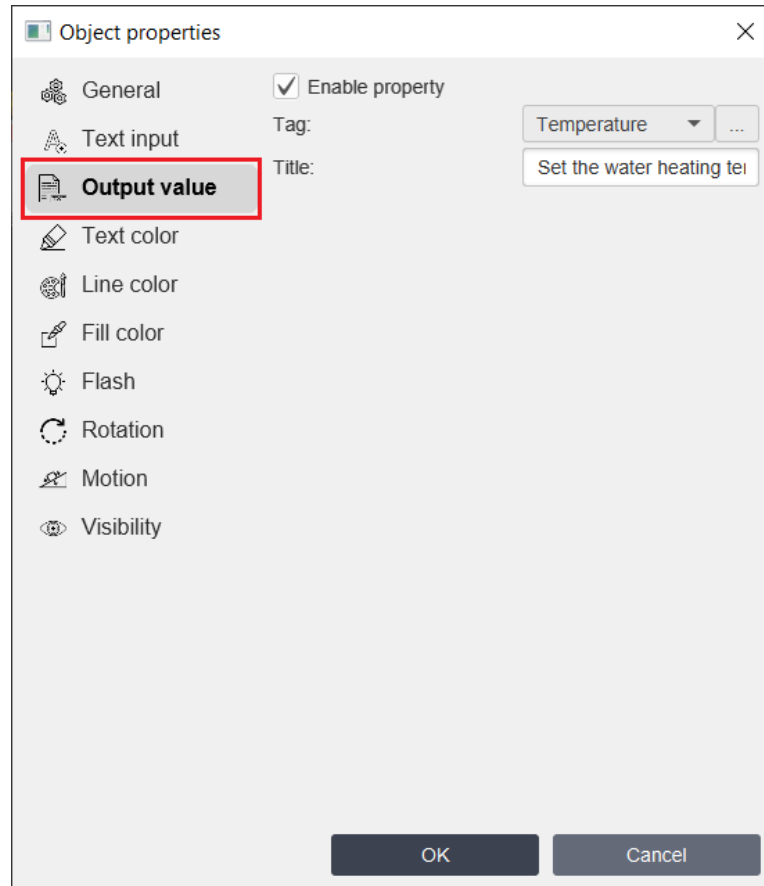
1. Create a tag named Temperature:



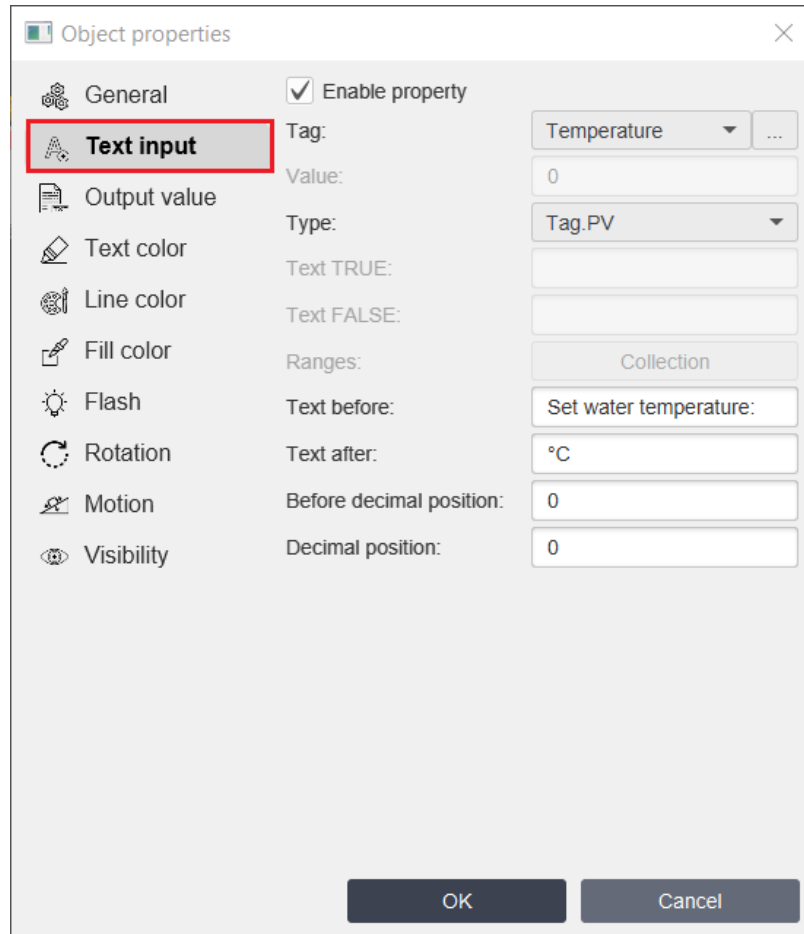
2. Create a Text/EditField object (you can use other objects that have the "Output Value" property):



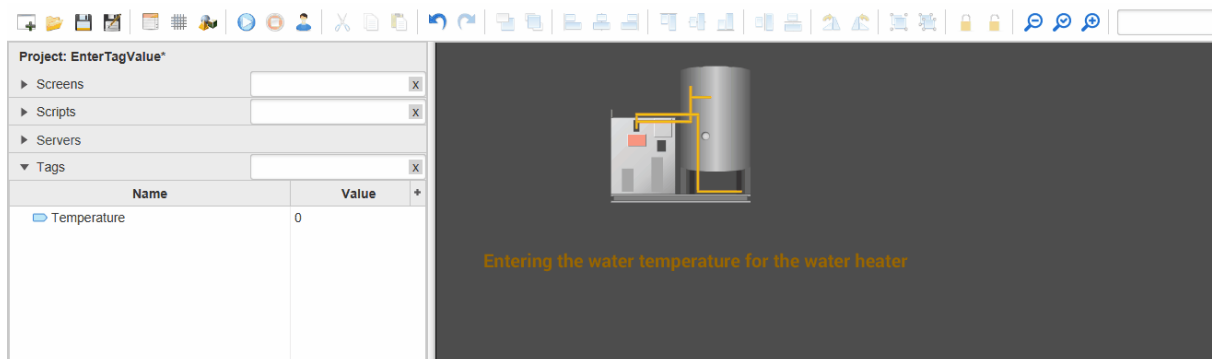
3. Set up the Output value property. Open the Object properties, go to the "Output value" tab, bind the tag to which we will set the value, and enter a Title for the dialog box:



4. Using this object, we will immediately display the specified value for the tag; to do this, we will configure the "Text Input" property:



5. Let's [Run simulation](#) to check the settings:

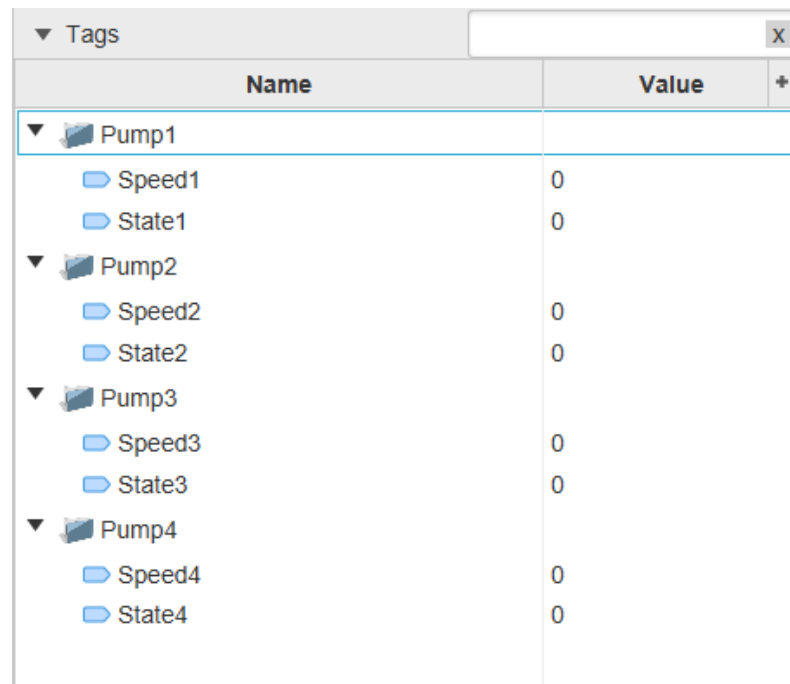


You can download this project [here](#).

9.4.5 Complex text change with scripts

If you need to change text depending on multiple tags, you need to use scripts. Assume we have several objects of the same type (Pump), each of which has two parameters (tags) - State and Speed, and we want to display text about the operation of the pump depending on tags' values.

1. Create tags for each object - State and Speed:

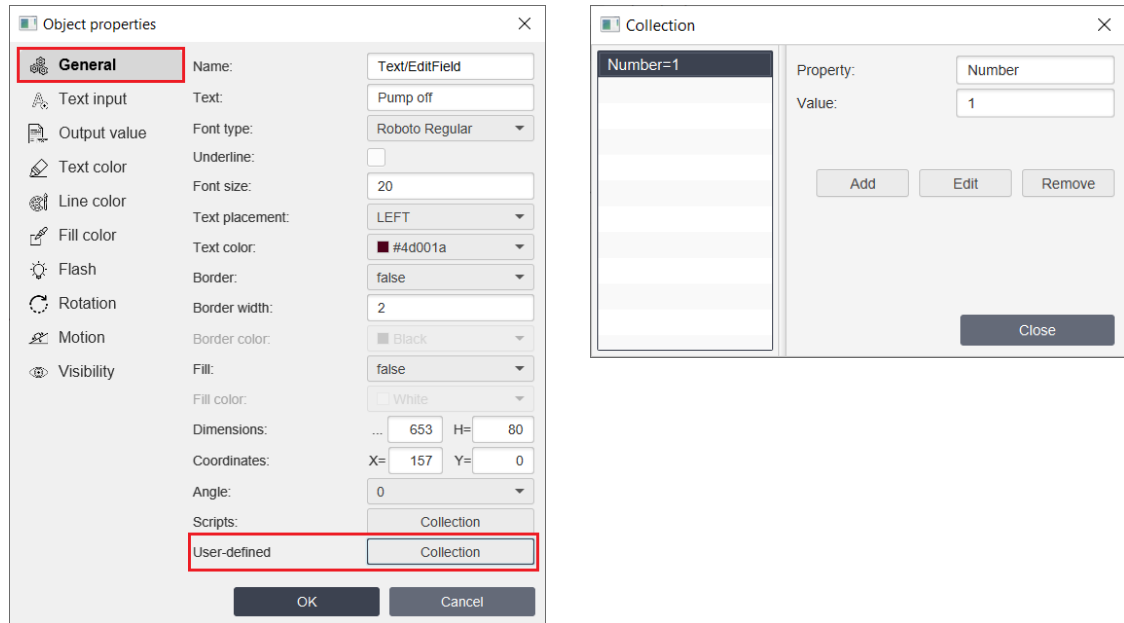


Name	Value
▼ Pump1	
Speed1	0
State1	0
▼ Pump2	
Speed2	0
State2	0
▼ Pump3	
Speed3	0
State3	0
▼ Pump4	
Speed4	0
State4	0

2. Create a Text/EditField object:



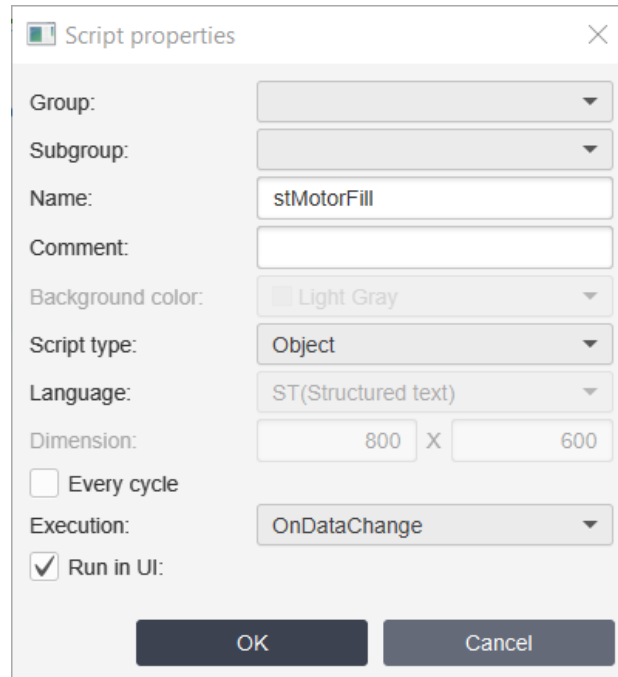
3. Create a user-defined property - "Number" and set its value to "1", because we will bind to the State1 and Speed1 tags:



4. Let the Text/EditField object display texts depending on the tag values::

State	Speed	Text
0	Any	Pump off
1	0...500	Pump speed within normal limits: PV
1	500...1000	Pump speed is high: PV
1	> 1000	Attention! Pump speed too high!

Next, we will create a script with type "object" in the ST language with the execution type - onDataChange:



The 'Script properties' dialog box contains the following fields and controls:


- Group:** A dropdown menu.
- Subgroup:** A dropdown menu.
- Name:** A text field containing 'stMotorFill'.
- Comment:** An empty text field.
- Background color:** A color selection dropdown showing 'Light Gray'.
- Script type:** A dropdown menu showing 'Object'.
- Language:** A dropdown menu showing 'ST(Structured text)'.
- Dimension:** Two text fields for width and height, both set to '800' and '600' respectively, separated by an 'X'.
- Every cycle:** An unchecked checkbox.
- Execution:** A dropdown menu showing 'OnDataChange'.
- Run in UI:** A checked checkbox.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

And let's write a script::

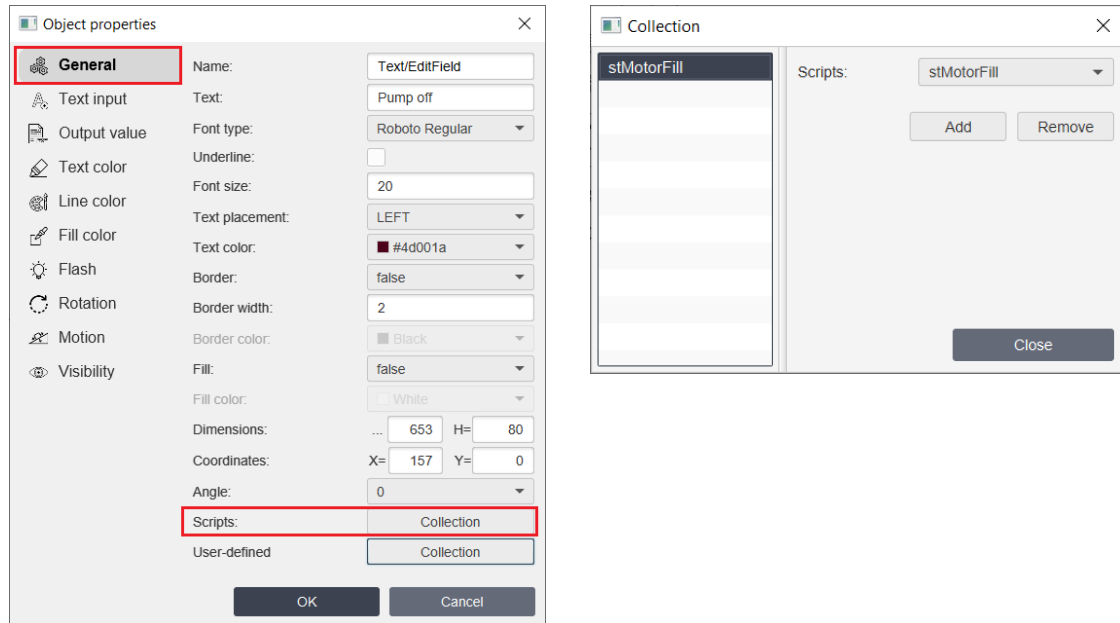
```

1 string statetagname = "State"+Objects.this.Number;// get tag's name by using State + Number user-defined property
2 string speedtagname = "Speed"+Objects.this.Number;// we use indirect name to have possibility to use the same script for other objects
3
4 byte state = gettagvalue(statetagname, 0); //get tag's values
5 int speed = gettagvalue(speedtagname, 0);
6
7 if (state==0){
8     Objects.this.text = "Pump off"; //if state=0 pump is stopped
9 }
10 else if (state==1){
11     //if state=1 text is dependent on speed
12     if (speed>=0 && speed<=500){ //if speed = 0...500 pump speed is normal
13         Objects.this.text = "Pump speed within normal limits: "+speed;
14     }
15     else if(speed>500 && speed<=1000){ // speed between 500 and 1000 warning pump is high
16         Objects.this.text = "Pump speed is high: " + speed;
17     }
18     else{
19         Objects.this.text = "Attention! Pump speed too high!"; // when speed is other pump is too high
20 }

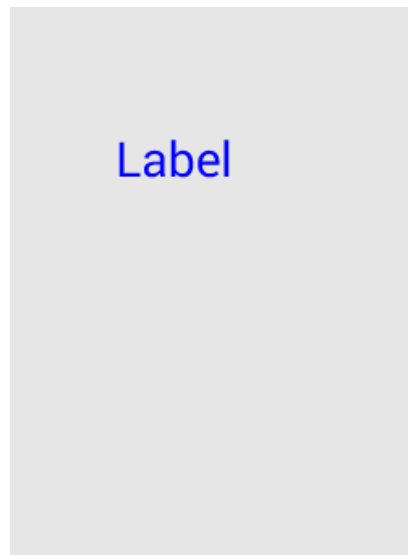
```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

5. Now let's bind the script to the object - open the Object Properties, the "General" tab, the "Scripts" field and fill in the "Collections":



6. So, we have a Text object with the Text Input property set by script . To copy this object and bind the Text Output property to the tags - State2, Speed2, State3, Speed3, State4, Speed4, we don't need to set up a script for each Text object, we only need to duplicate the object and change the user-defined Number property of the new object on the Property Sheet:



Screen: Scre... **Object: Text/...**

Search

▼ 01.General

Name: Text/EditField2

Text: Pump speed within normal limits: 100

Font type: Roboto Regular

Underline: ☐

Font size: 20

Text placement: LEFT

Text color: #4d001a

Border: ☐

Border width: 2

Fill: ☐

Width: 653.0

Height: 80.0

Position X: 127.0

Position Y: 0.0

Angle: 0

Scripts: Collection

Number: 2

7. Let's [Run simulation](#) to check the settings:

Project: FillObjectColor

► Screens

► Scripts

► Servers

▼ Tags

Name	Value
▼ Pump1	
Speed1	0
State1	0
▼ Pump2	
Speed2	0
State2	0
▼ Pump3	
Speed3	0
State3	0
▼ Pump4	
Speed4	0
State4	0

Pump off

Pump off

Pump off

Pump off

You can download this project [here](#).

9.5 Call popup

This chapter contains examples of project to call popup windows:

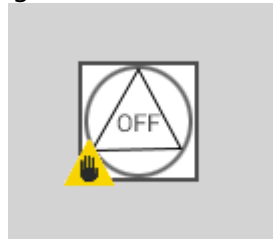
- [Complex call popup with scripts](#)

9.5.1 Complex call popup with scripts

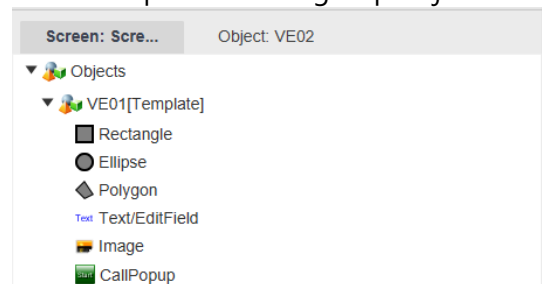
Suppose we have several objects of the same type on the screen, each object has several parameters. When we click on an object, we want to see a popup window with information about the state (value) of the tags and be able to set values for some tags directly in the pop-up window.

1. Let's create a complex group object, let's call it VE01, which consists of primitive objects - [Rectangle](#), [Ellipse](#), [Polygon](#), [Image](#), [Text/EditField](#) and on top of these objects we placed a transparent [Button](#) - CallPopup:

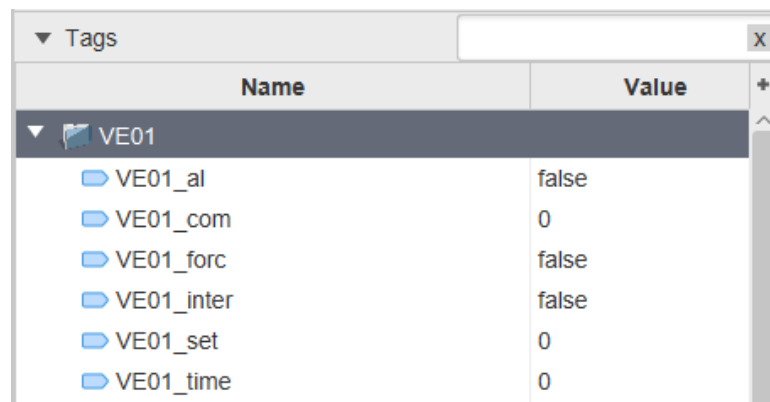
Image of the finished object



Composition of a group object



2. Let's create tags and bind them to objects from the group:



To make it easier to scale this project and be able to quickly copy this group of tags to the next similar object, we associated the name of the tags with the group using the {group} keyword. Example:

Tag properties

General

Group: VE01

Subgroup:

Name: {group}_al

Data type: Boolean

Number of elements: 10

1 element:

Access mode: ReadWrite

Initial PV: false

Access level: 0

Input/Output

PV Input server: Local

PV Input tag:

☐ Output differs from Input:

PV Output server: Local

PV Output tag:

Description:

OK Cancel

3. We want to call a popup screen by clicking on this group object (by clicking on the transparent button - CallPopup, to be precise) and display all the properties in the corresponding fields.

Let's create a popup window named FbMotorAOVentil (to do this you need to create a new screen and specify the screen type - "Pop-up"), which is also a group object.

Popup window

IDENTIFICATION

Object name Text Mnemonique

N° electric Text N° Schema

Installation Text Installation

INFORMATIONS

Command MAN-OFF Setting Value

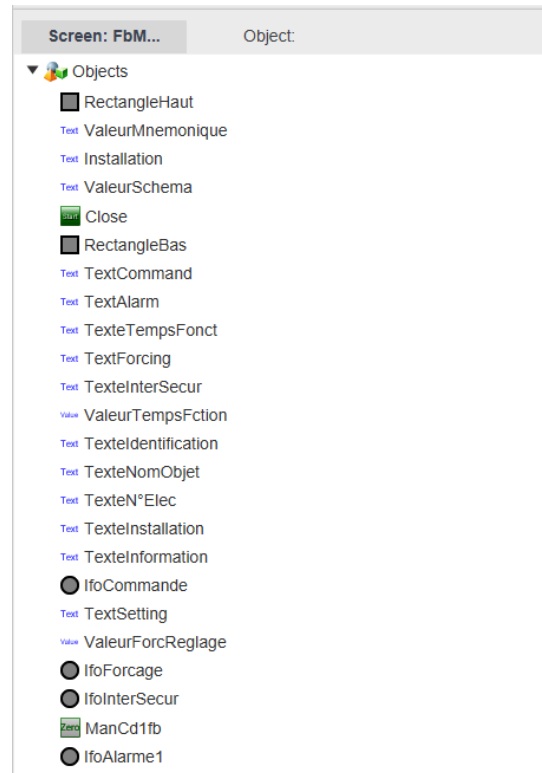
Forcing

Alarm Inter. security

Operating time Value

Close

Popup Composition

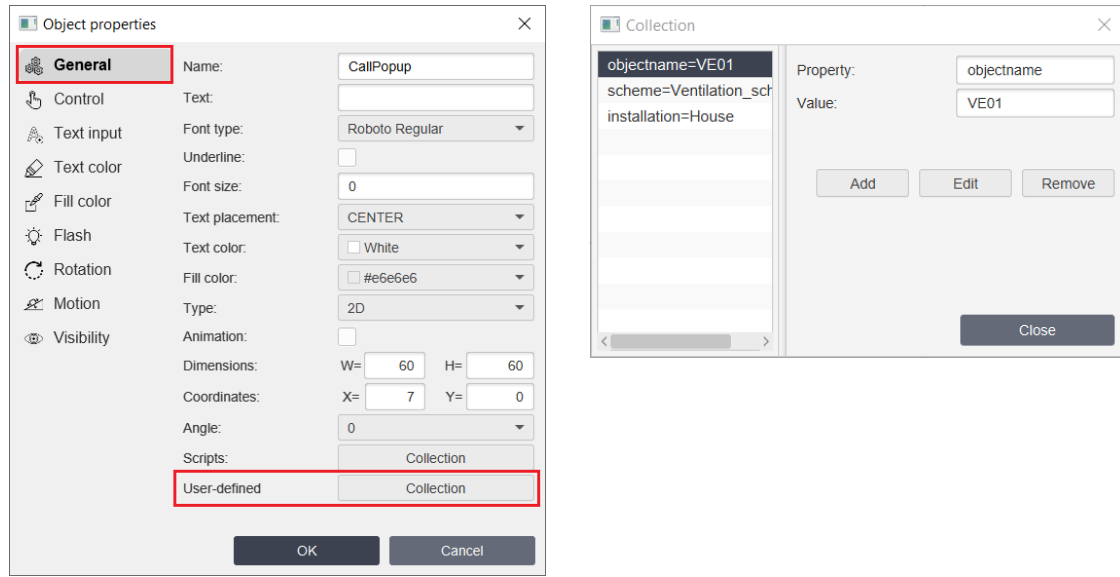


4. Let's create tags for this pop-up window:

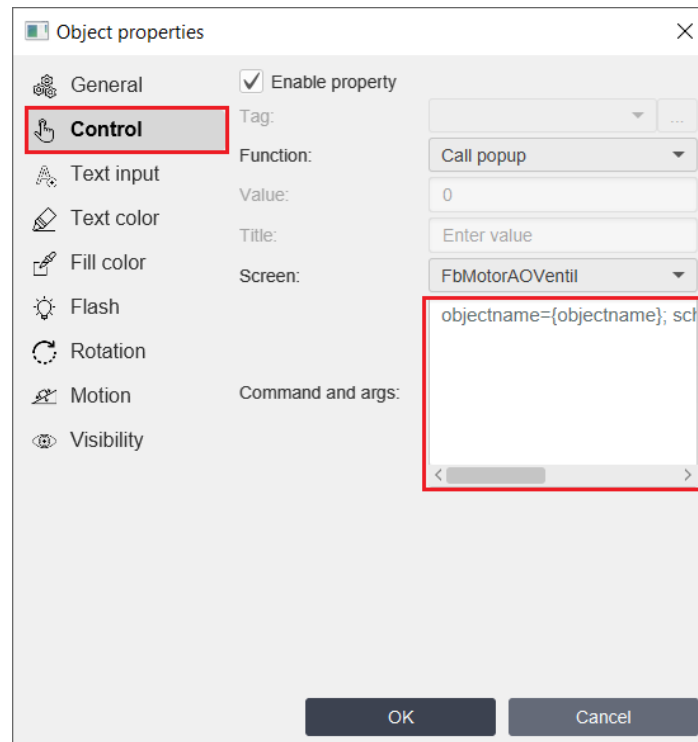
The screenshot shows a window titled 'Tags' with a search bar and a table of tags. The table has two columns: 'Name' and 'Value'. The tags are organized into a hierarchy under 'VE01' and 'Popup'.

Name	Value
VE01	
Popup	
Popup_al	false
Popup_com	0
Popup_forc	false
Popup_install	
Popup_inter	false
Popup_name	
Popup_schema	
Popup_set	0
Popup_time	0

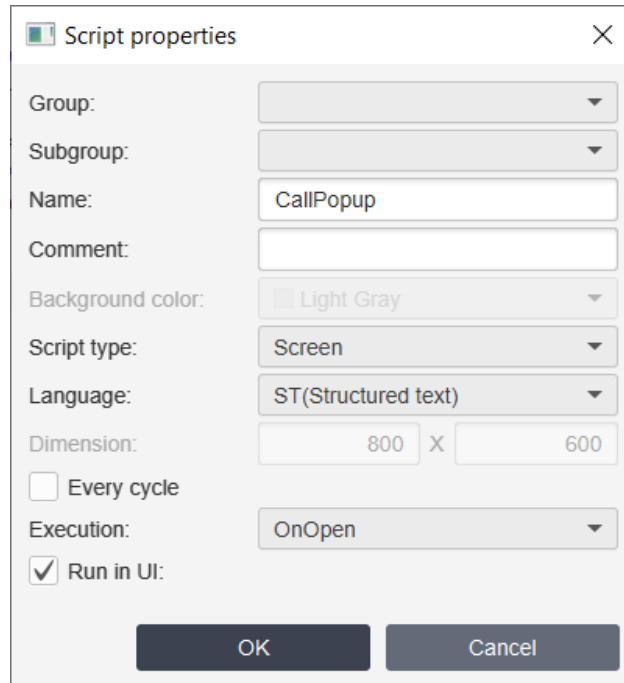
5. In order to have possibility to send some information from the group object to the pop-up window, let's create user-defined properties (we set user-defined properties for the "Button" object, which causes the pop-up window):



6. Now, let's configure the pop-up window call. In the Properties of the "Button", which causes a pop-up window in the "Control" tab, in the "Commands and Arguments" field we will pass the arguments (which we set as user-defined properties in the paragraph above):



7. Now let's create a script that will use the arguments that we wrote down in the paragraph above and which will be used when opening our popup window. Create a script called CallPopup with type "Screen" in ST language and execution "onOpen":



The 'Script properties' dialog box contains the following fields and controls:


- Group:** A dropdown menu.
- Subgroup:** A dropdown menu.
- Name:** A text input field containing 'CallPopup'.
- Comment:** An empty text input field.
- Background color:** A color selection dropdown showing 'Light Gray'.
- Script type:** A dropdown menu showing 'Screen'.
- Language:** A dropdown menu showing 'ST(Structured text)'.
- Dimension:** Two input fields for width and height, both set to '800' and '600' respectively, separated by an 'X'.
- Every cycle:** An unchecked checkbox.
- Execution:** A dropdown menu showing 'OnOpen'.
- Run in UI:** A checked checkbox.
- Buttons:** 'OK' and 'Cancel' buttons at the bottom.

Let's write a script:

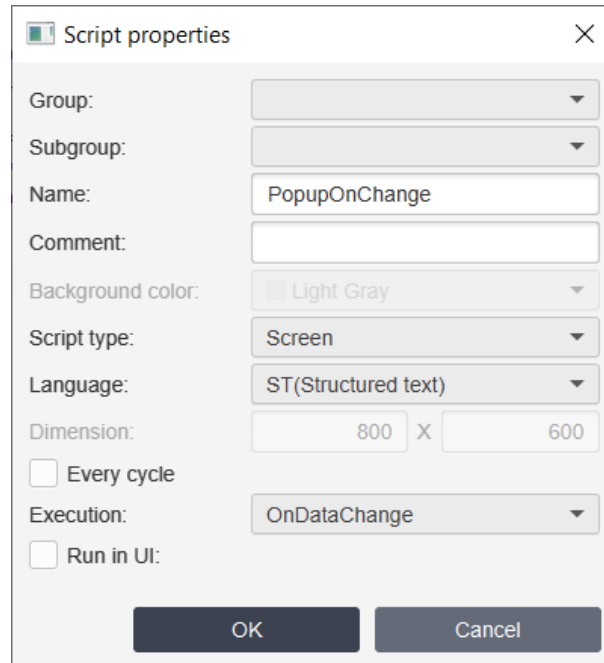
```

1 string objectname = getglobalargument("objectname",""); // get user-defined properties from bundle of global arguments
2 Tags.Popup_name = objectname;
3 Tags.Popup_schema = getglobalargument("scheme","");
4 Tags.Popup_install = getglobalargument("installation","");
5
6 int numberoftags = 6;
7 string tags[6] = {"_al", "_com", "_forc", "_inter", "_set", "_time"}; // all our tags used in group objects and in popups
8 for (int i=0; i<numberoftags;i++){
9
10     double value = gettagvalue(objectname+tags[i],0); //get group's object tag value
11     settagvalue("Popup"+tags[i],value); // bind it to popup tag's value
12     putglobalargument(tags[i],value); //save it in global arguments to make possibility
13                                     //to catch changes in the popup tags
14 }
15

```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

8. Let's create another script to pass changes in the pop-up tags to the tags of the group object (if we change the tag value in the pop-up window, then it will be transferred to the object tag), and vice versa, to catch changes in the tag value of the group object and set the value of the pop-ups tags (if the value of an object's tag changes while the popup is open, the value in the popup will also change):



The 'Script properties' dialog box is shown with the following settings:

- Group: (empty dropdown)
- Subgroup: (empty dropdown)
- Name: PopupOnChange
- Comment: (empty text box)
- Background color: Light Gray
- Script type: Screen
- Language: ST(Structured text)
- Dimension: 800 X 600
- ☐ Every cycle
- Execution: OnDataChange
- ☐ Run in UI:


Buttons: OK, Cancel

Let's write a script:

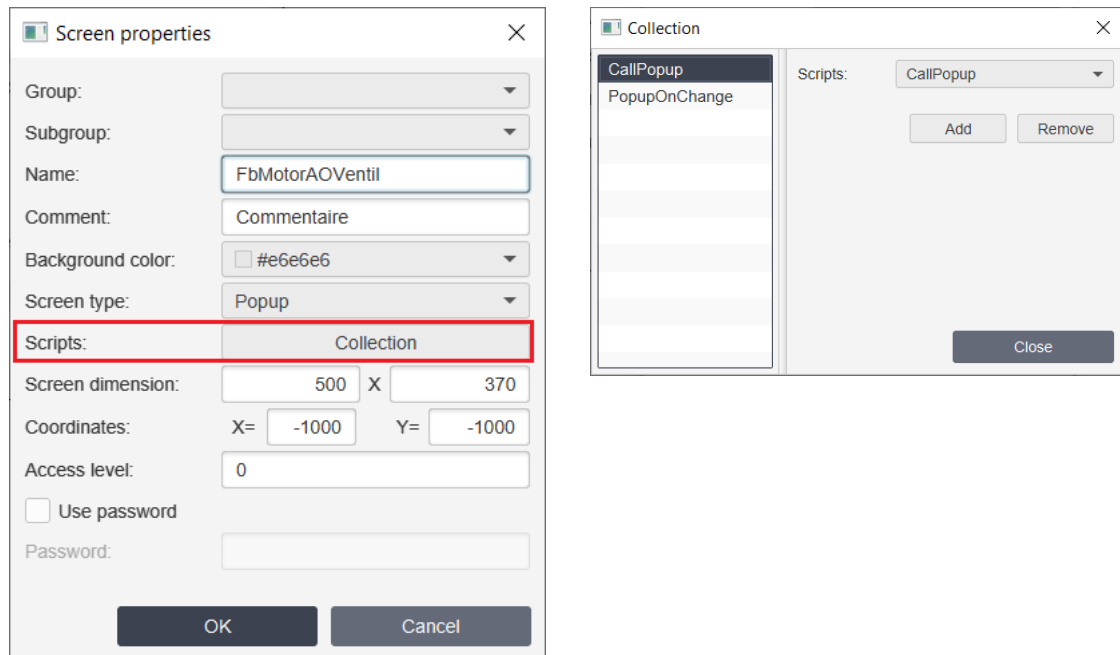
```

1 string objectname = getglobalargument("objectname","");
2 int numberoftags = 6;
3 string tags[6] = ["_al", "_com", "_forc", "_inter", "_set", "_time"]; //tags of the object and popup
4 for (int i=0; i<numberoftags;i++){
5     double popupvalue = gettagvalue("Popup"+tags[i],0); //read popup tag
6     double storevalue = getglobalargument(tags[i], 0); //read tag's value from the bundle of global arguments
7     double value = gettagvalue(objectname+tags[i],0); //read object's tag value
8     if (popupvalue!=storevalue){ //if popup tag is not equal stored value other values are changed
9         putglobalargument(tags[i],popupvalue);
10        settagvalue(objectname+tags[i],popupvalue);
11    }
12    else if (value!=storevalue){ //if object's value is not equal stored value other values are chan
13        settagvalue("Popup"+tags[i],value);
14        putglobalargument(tags[i],value);
15    }
16 }
17 }
18

```

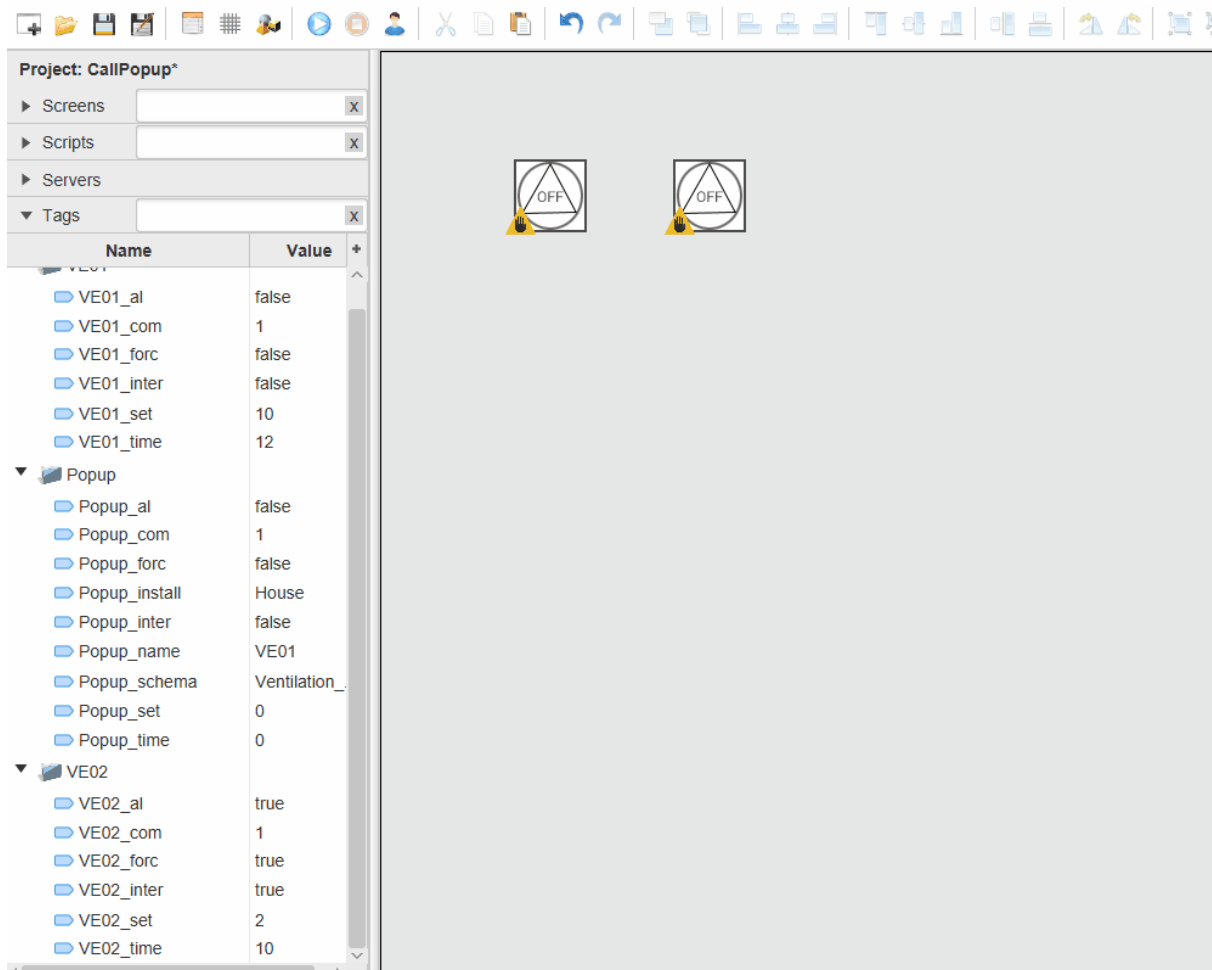
After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

9. Let's bind the scripts to our pop-up window:



10. So, we have created a pop-up window (One for all objects), into which the tag values are transferred and from which you can change the tag values for the object. We also configured the group object VE01 using custom properties. Now let's duplicate the object as many times as we need and change the values in the custom properties.

11. Let's [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

9.6 HTTP requests

In TeslaSCADA2 you can send HTTP POST/GET requests to third party servers to read data from them. Below are examples of retrieving data from some popular HTTP servers. To use these features, you can look into the HTTP library. Below are examples of working with this library:

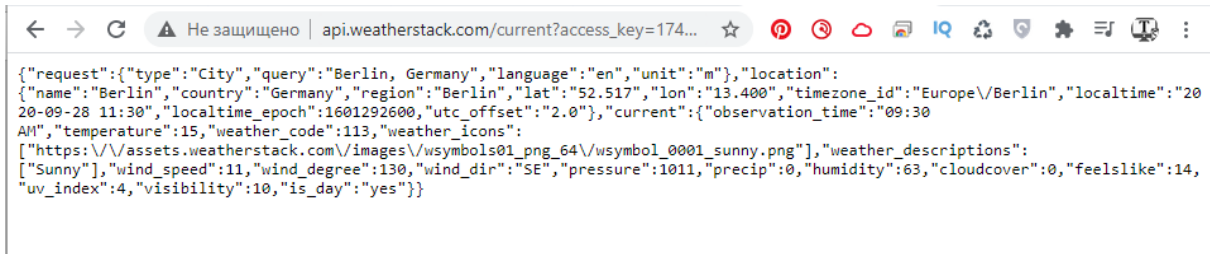
- [Weather from weatherstack.com](#)

9.6.1 Weather from weatherstack.com

[weatherstack.com](#) has a convenient API for reading weather data. After registering on the site, you will receive a unique access key (API Access Key), which must be used in GET requests to obtain weather data. In the weatherstack documentation you can look at examples of requests and create a request, for example, for Berlin it should be like this:

http://api.weatherstack.com/current?access_key=API_ACCESS_KEY&query=Berlin

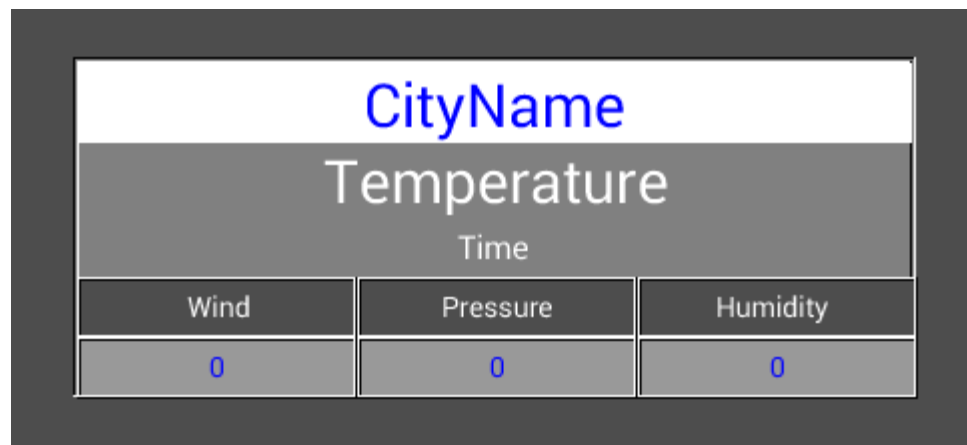
Instead of `API_ACCESS_KEY`, you need to insert the access key received during registration. Please note that if you need to pass a parameter containing a space, for example "New York", then the space must be replaced with "%20", that is, "New%20York". You can check the validity of the request by pasting it into the address bar of your browser:



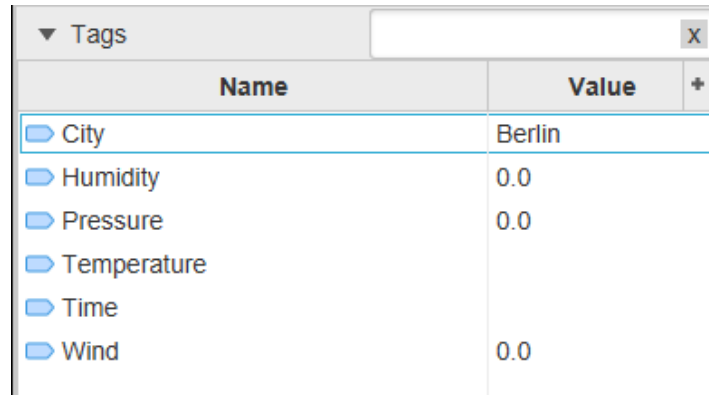
The browser displayed a response with the correct data, indicating that the request was made correctly. As you can see, the response is sent in JSON format, later we will extract the properties we need from it.

Now we can start solving the problem in TeslaSCADA2.

1. First, let's create an interface in the project. Temperature, pressure, wind, humidity and local time will be displayed using `Text/TextField` objects. In the `CityName` field we activate the `Output value` property to be able to change the name of the city. The image below shows the created interface and the names we gave to the components:

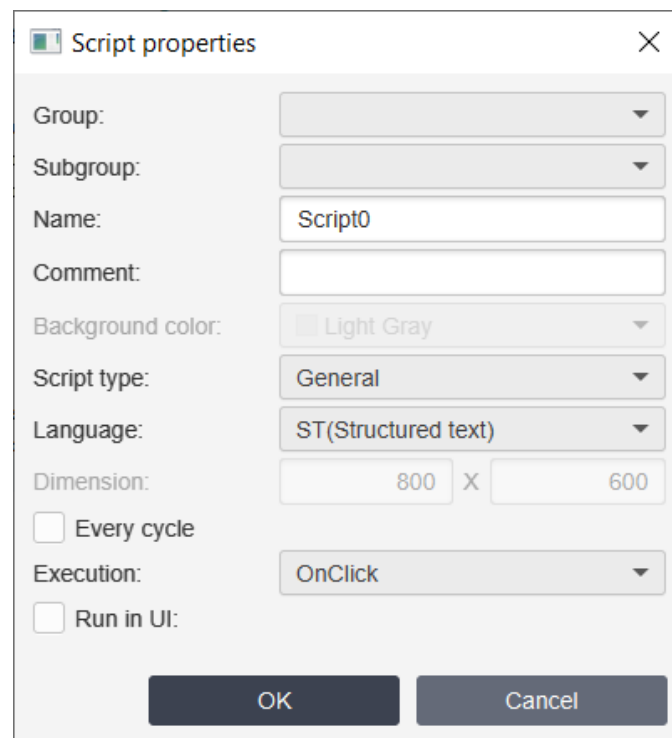


2. Create tags for each text object and bind them:



Name	Value
City	Berlin
Humidity	0.0
Pressure	0.0
Temperature	
Time	
Wind	0.0

3. Now let's create a script in ST language that will be executed when you click on the screen:



Script properties

Group:

Subgroup:

Name:

Comment:

Background color:

Script type:

Language:

Dimension: X

☐ Every cycle

Execution:

☐ Run in UI:

OK Cancel


The text of ST script below:

```

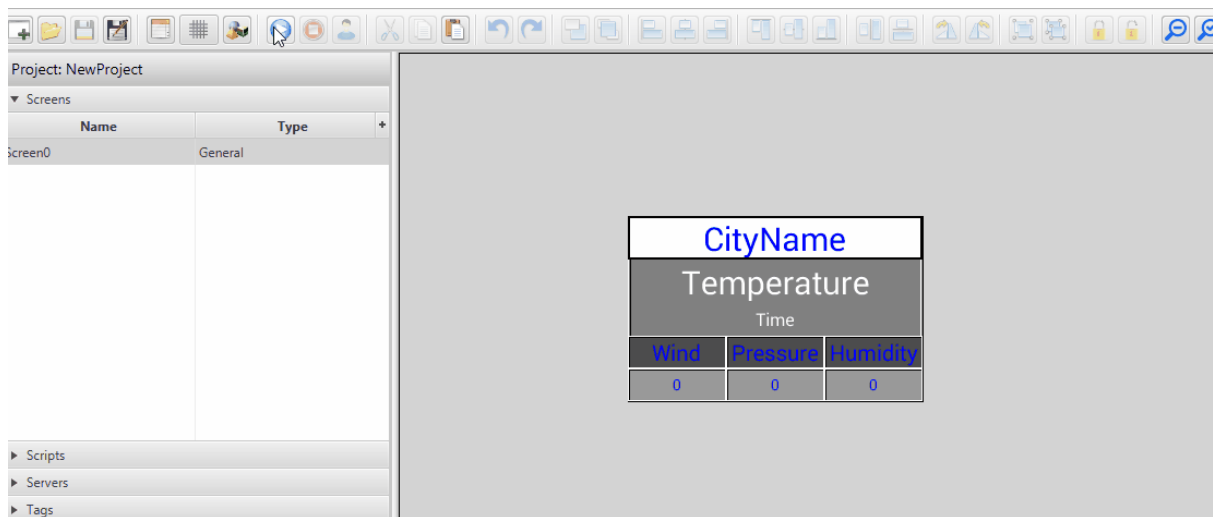
1 string aQuery = "http://api.weatherstack.com/current?access_key=" +
2   "API_ACCESS_KEY" + "&query="+Tags.City; //set query request to weatherstack.com with your API_KEY
3 httpstpostcreate("namehttpstpost", aQuery); //create http post request
4 string response = httpstpostexecute("namehttpstpost"); //execute created request
5 print(response); //for debug purposes check response
6 string current = httpstpostgetvalue(response, "current"); //get "current" value from the response
7
8 string temperature = httpstpostgetvalue(current, "temperature"); //get values from the "current" part of the response
9 string windspeed = httpstpostgetvalue(current, "wind_speed");
10 string pressure = httpstpostgetvalue(current, "pressure");
11 string humidity = httpstpostgetvalue(current, "humidity");
12
13 Tags.Temperature = temperature;
14 Tags.Wind = windspeed;
15 Tags.Pressure = pressure;
16 Tags.Humidity = humidity;
17
18 string location = httpstpostgetvalue(response, "location"); //get "location" value from the response
19 string time = httpstpostgetvalue(location, "localtime");
20 Tags.Time = time;

```

Change API_ACCESS_KEY to your key that you get from site.

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

4. Let's [Run simulation](#)⁷⁰ to check the settings:



You can download this project [here](#).

9.7 Trends

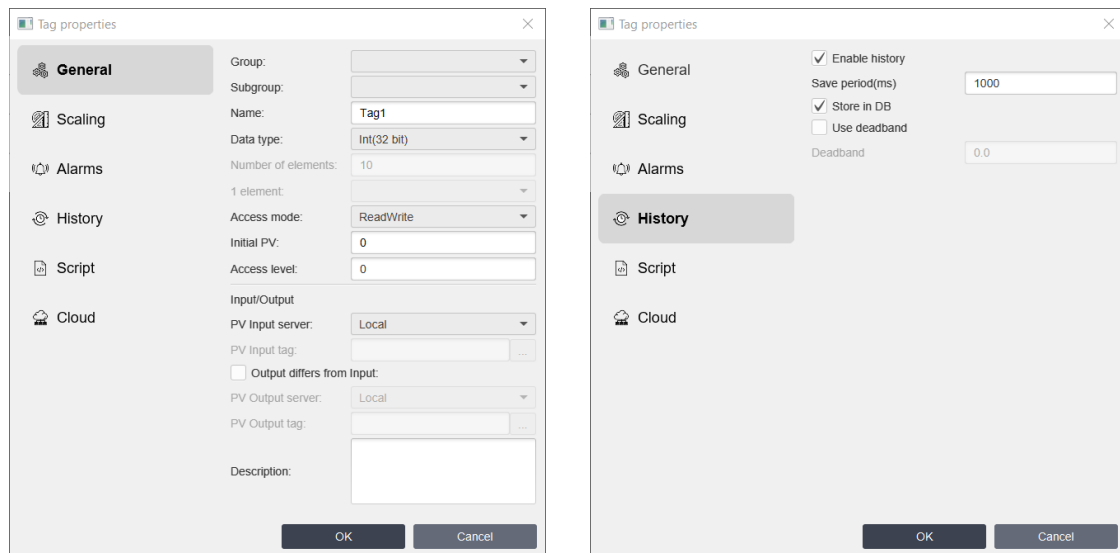
Below are examples for working with history and trends:

- [Simple trend example](#)⁵⁸⁵
- [Trend example with Y axis change](#)⁵⁸⁷
- [Add and remove curve to/from trend dynamically](#)⁵⁹⁰

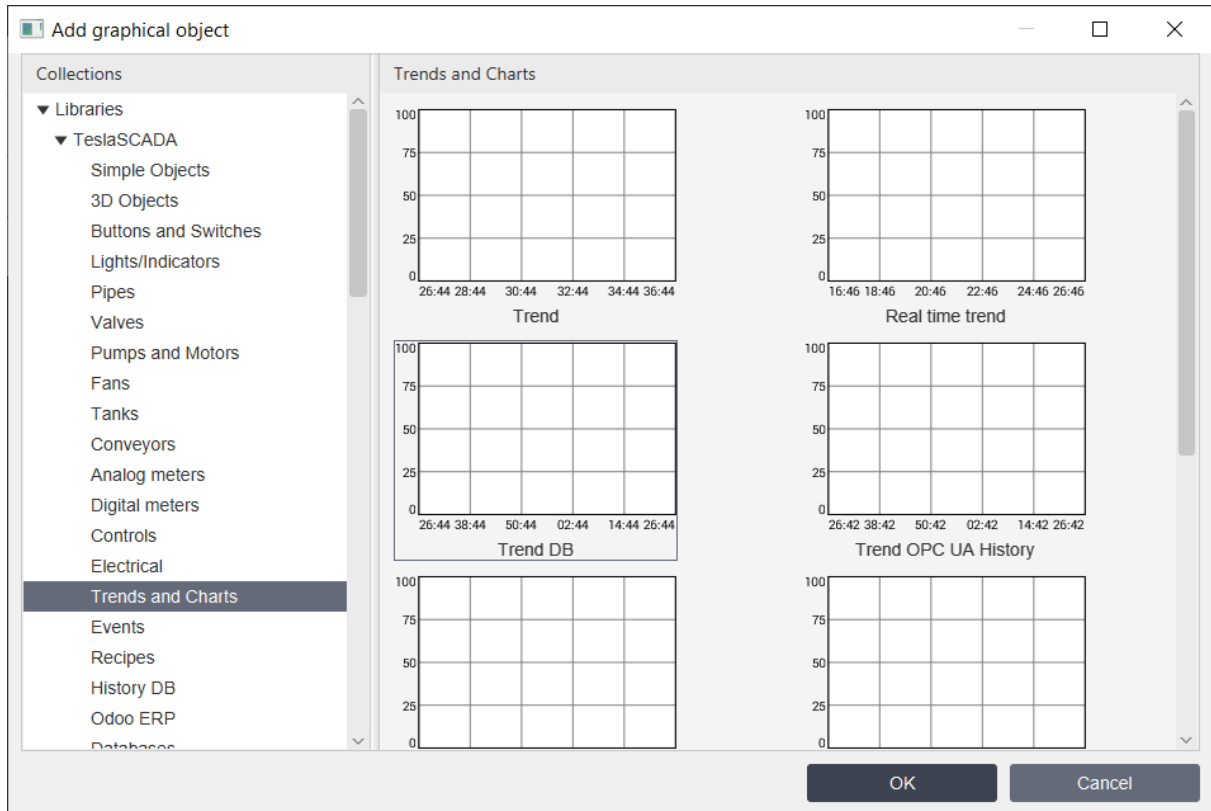
9.7.1 Simple trend example

Quite often you need to look at the dynamics of the values of certain parameters (tags). We can display this data on a chart using graphical objects; in the example below we will use the Trend DB.

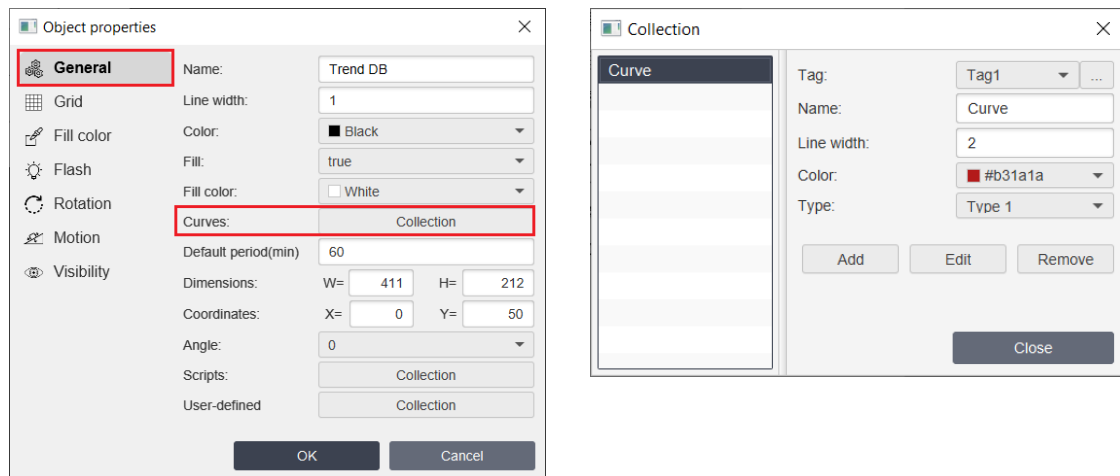
1. Suppose we want to look at the dynamics of values for a certain tag Tag1, the values of which will be collected in the general [SQLite database](#)²⁹¹.



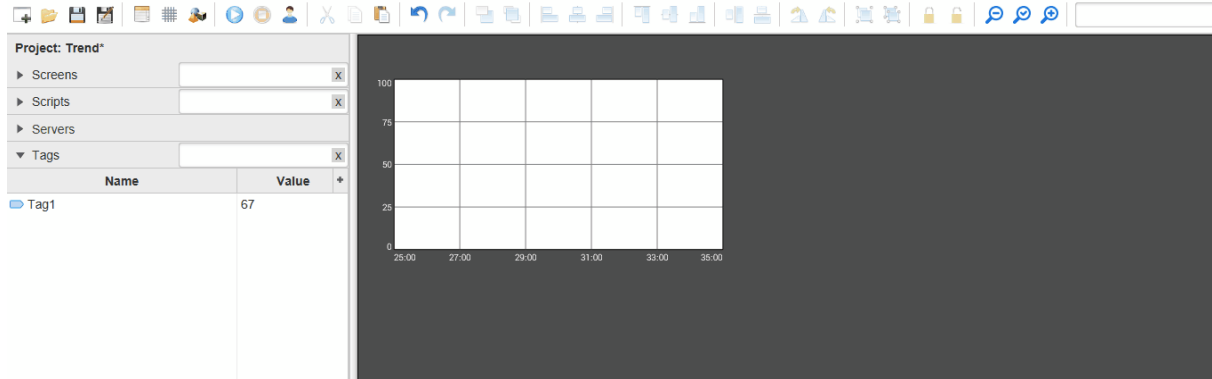
2. We want to display history information about Tag1 values on the Trend DB . Let's place the [Trend DB](#)²²⁸ object on the screen:



3. Bind the tag to our trend. To do this, open the properties of the Trend DB and fill in the "Collection" in the "Curves" field:



4. [Run simulation](#) ⁷⁰ to check the settings. Within a few minutes, set different tag values to keep the information in the database. Then, by clicking on the trend, we will select the period for which we want to obtain data. In our example, we will select data from the previous few minutes when we recorded data.



You can download this project [here](#).

9.7.2 Trend example with Y axis change

In the previous example, we could set the trend time range (X-axis) arbitrarily by clicking on it with the mouse and specifying the "Start" and "End" of the period in the pop-up window:

The screenshot shows a dialog box titled 'Select the start and end time'. It has a close button (X) in the top right corner. Inside the dialog, there are two input fields: 'Start:' with the value '2023-11-22 13:44' and 'End:' with the value '2023-11-22 14:44'. Below these fields is a section labeled 'Select curves:' with a checked checkbox and the text 'Curve'. At the bottom of the dialog, there are four buttons: 'Save report...', 'Print', 'OK', and 'Cancel'.

If we want to change the range of the Y axis we need to use a script.

Let's take the project from our previous example as a basis, where a tag and a graphic object have already been created and configured.

1. Let's create an intermediate tag named max, which will change the maximum trend range (Y-axis) through a script:

Tag properties

General

Group:

Subgroup:

Name:

Data type:

Number of elements:

1 element:

Access mode:

Initial PV:

Access level:

Input/Output

PV Input server:

PV Input tag:

☐ Output differs from Input:

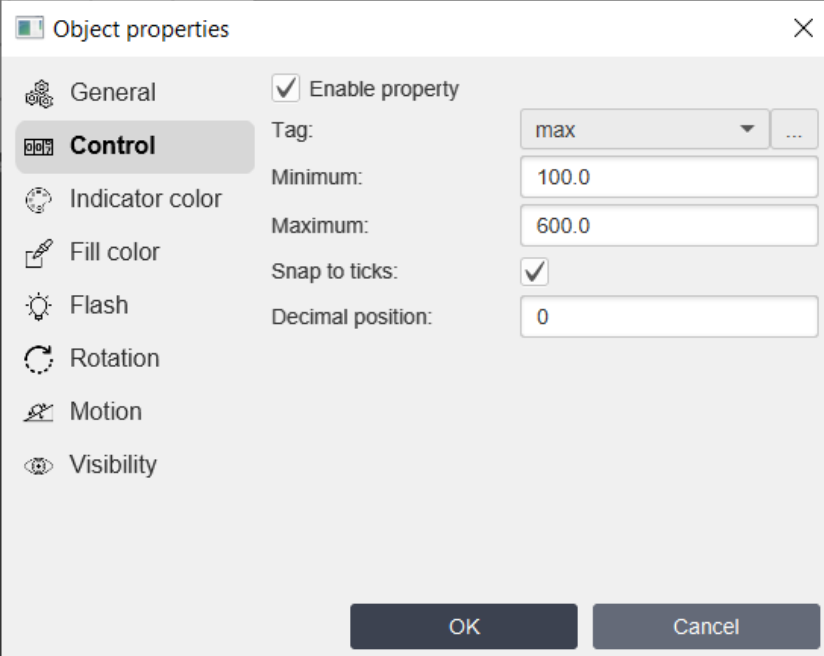
PV Output server:

PV Output tag:

Description:

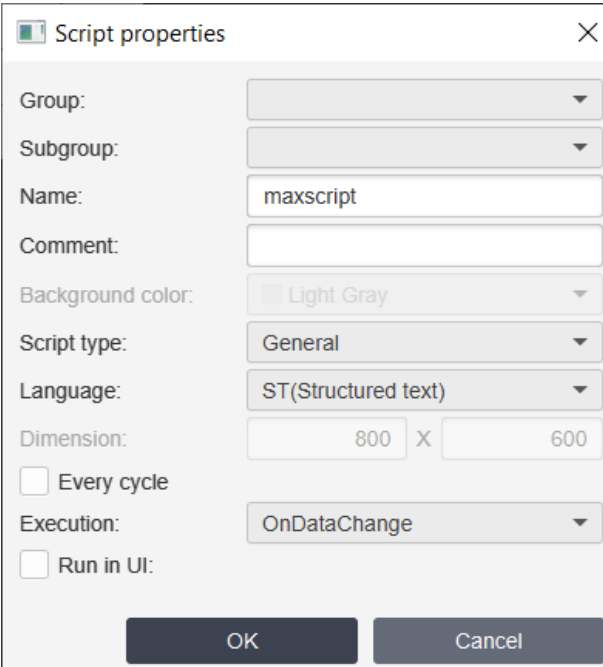
OK Cancel

2. To set the value for the Y axis of the Trend DB, place a [Slider](#)²¹⁴ on the screen and bind the max tag to it through the “Control” property:



The 'Object properties' dialog box is shown with the 'Control' tab selected. It features a sidebar with icons for General, Control, Indicator color, Fill color, Flash, Rotation, Motion, and Visibility. The 'Control' tab is active, displaying settings for 'Enable property' (checked), 'Tag' (max), 'Minimum' (100.0), 'Maximum' (600.0), 'Snap to ticks' (checked), and 'Decimal position' (0). The dialog has 'OK' and 'Cancel' buttons at the bottom.


3. We will also bind the value of the max tag to the maximum property of the Trend using the ST script. Let's create a script that will be called when the value of the max tag changes:




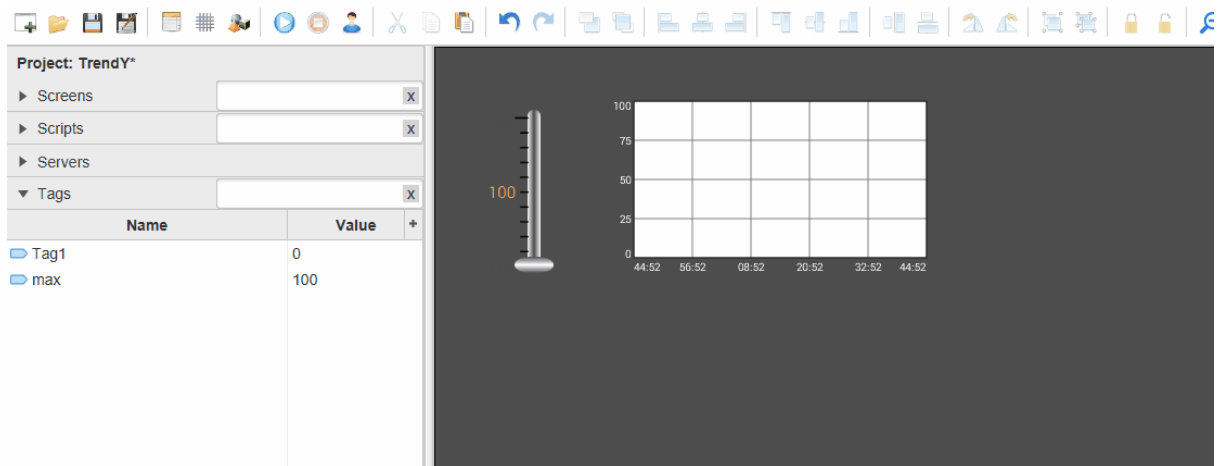
The 'Script properties' dialog box is shown with the following settings: 'Group' and 'Subgroup' are empty dropdowns; 'Name' is 'maxscript'; 'Comment' is empty; 'Background color' is 'Light Gray'; 'Script type' is 'General'; 'Language' is 'ST(Structured text)'; 'Dimension' is '800 X 600'; 'Every cycle' is unchecked; 'Execution' is 'OnDataChange'; and 'Run in UI' is unchecked. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Let's write a script:

```
1 Objects.TrendDB.maximum = Tags.max;
```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

4. Let's [Run simulation](#)  to check the settings (using the slider we will set the value for the Y-axis of the Trend):



You can download this project [here](#).

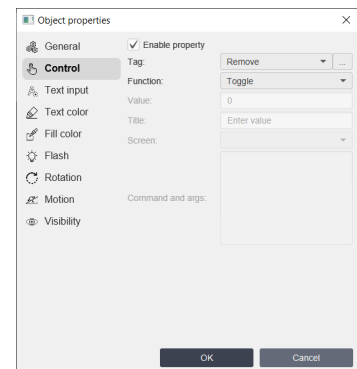
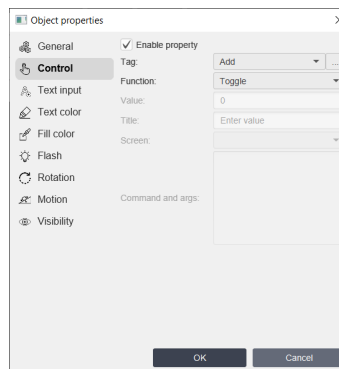
9.7.3 Add and remove curve to/from trend dynamically

If you want to add or remove curves to/from a trend dynamically, you should use scripts. Let's look at an example how to do this. Let's take the project from the previous example as a basis, where we have already created and configured tags and graphic objects.

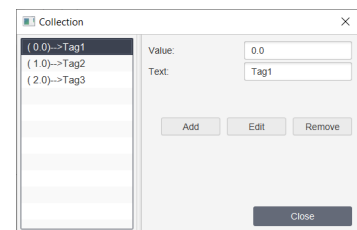
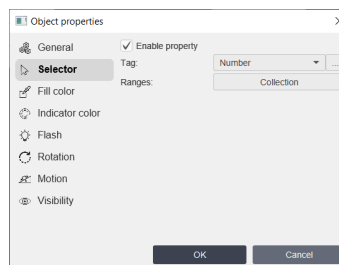
1. Suppose we want to see the dynamics of two more parameters on the same trend. In this case, having copied Tag1, we will additionally create Tag2 and Tag3,
2. Let's create 2 intermediate tags - Add and Remove. When the values of these tags change from FALSE to TRUE, we will add or remove a curve.
3. Let's create an intermediate tag Number, it will contain information about which tag we want to add or remove. All our tags look like below:

▼ Tags		
Name		Value
▶ Add		false
▶ Number		1
▶ Remove		false
▶ Tag1		0
▶ Tag2		0
▶ Tag3		0
▶ max		100

2. Create 2 [Buttons](#)¹⁸¹ "Add curve" and "Remove curve" and bind them to the Add and Remove tags, respectively, through the Control Property:



3. Create a ComboBox object and bind the Number tag to it through the "Selector" property and fill the "Collection" with tag names:



4. Now let's create 2 scripts to add and remove a curve:

Script properties dialog box. Fields include: Group (dropdown), Subgroup (dropdown), Name (addcurve), Comment (empty), Background color (Light Gray), Script type (Tag), Language (ST(Structured text)), Dimension (800 X 600), Every cycle (checkbox), Execution (OnChange), and Run in UI (checkbox). OK and Cancel buttons are at the bottom.


Script properties dialog box. Fields include: Group (dropdown), Subgroup (dropdown), Name (removecurve), Comment (empty), Background color (Light Gray), Script type (Tag), Language (ST(Structured text)), Dimension (800 X 600), Every cycle (checkbox), Execution (OnChange), and Run in UI (checkbox). OK and Cancel buttons are at the bottom.

Let's write a script to add a curve using the `addcurve` function from the [Trend's curve library](#) ⁴⁵¹:

```

1 if (Tags.Number==0){
2     addcurve("TrendDB","curve1","Tag1",2,255,0,0,1); // add curve1 for Tag1 with Red color
3 }
4 else if (Tags.Number==1){
5     addcurve("TrendDB","curve2","Tag2",2,0,255,0,1); // add curve2 for Tag2 with Green color
6 }
7 else if (Tags.Number==2){
8     addcurve("TrendDB","curve3","Tag3",2,0,0,255,1); // add curve3 for Tag3 with Blue color
9 }
10 Tags.Add=false; //reset Add tag
11 Objects.TrendDB.update=true; //update trend to redraw it after adding

```


After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

Let's write a script to delete a curve using the `removecurve` function from the [Trend's curve library](#) ⁴⁵¹:

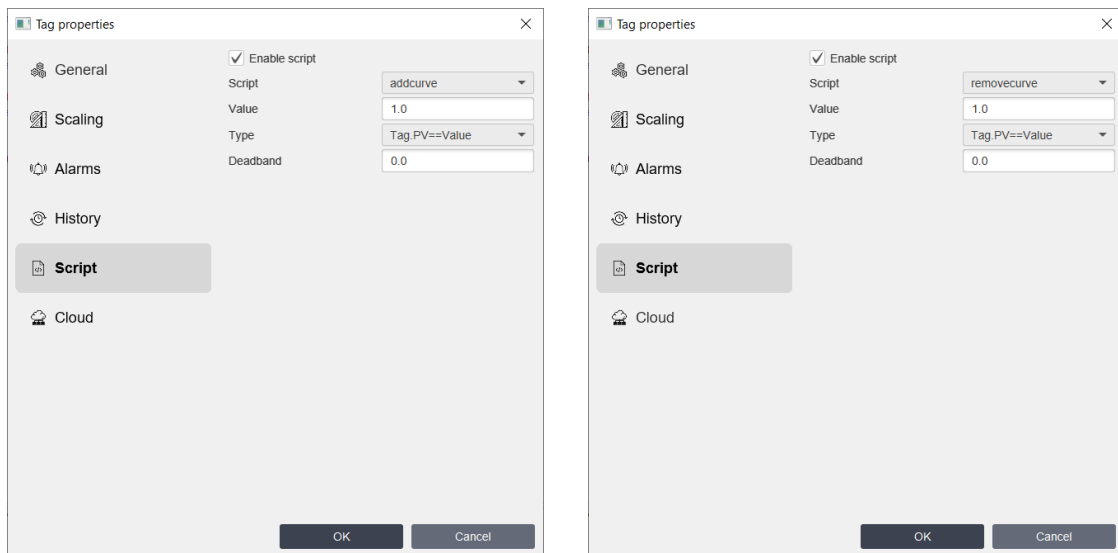
```

1 if (Tags.Number==0){
2     removecurve("TrendDB","curve1"); //remove curve1 from the trend
3 }
4 else if (Tags.Number==1){
5     removecurve("TrendDB","curve2"); //remove curve2 from the trend
6 }
7 else if (Tags.Number==2){
8     removecurve("TrendDB","curve3"); //remove curve3 from the trend
9 }
10 Tags.Remove=false;
11 Objects.TrendDB.update=true; //update trend to redraw it after removing

```

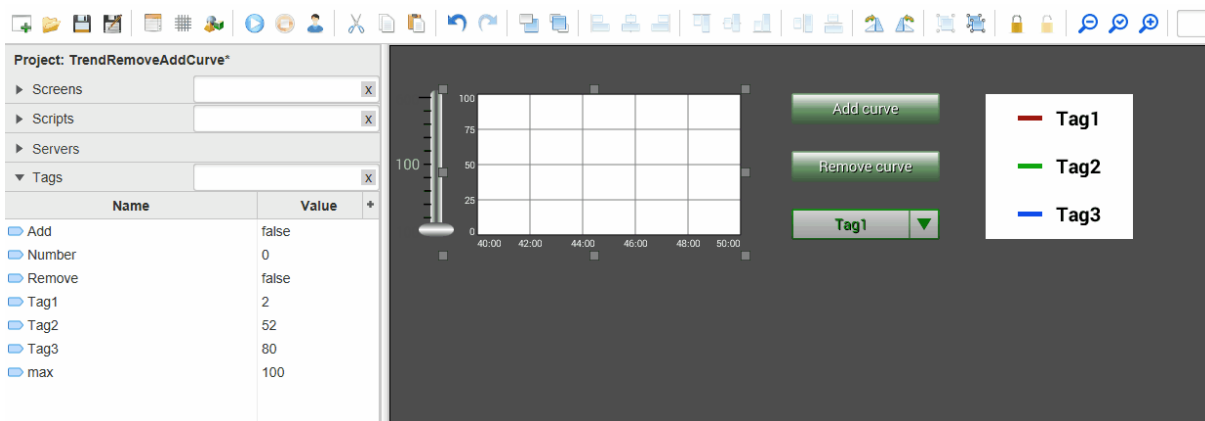
After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

5. Link the scripts to the tags - Add and Remove:



Now when we click the Add and Remove buttons we call the corresponding scripts.

6. [Run simulation](#)^[70] to check the settings:



You can download this project [here](#).

9.8 Change tag's value

The easiest way to change the value of a tag is to use Buttons via the Control property. You can also use Text/Input Field. The example you can find [here](#)^[566]. You can also use objects from [Controls library](#)^[213]. For more complex task you could use scripts:

- [Change values of 2 tags by one click](#)^[594]
- [Write value when screen is opened and closed](#)^[595]

9.8.1 Change values of 2 tags by one click

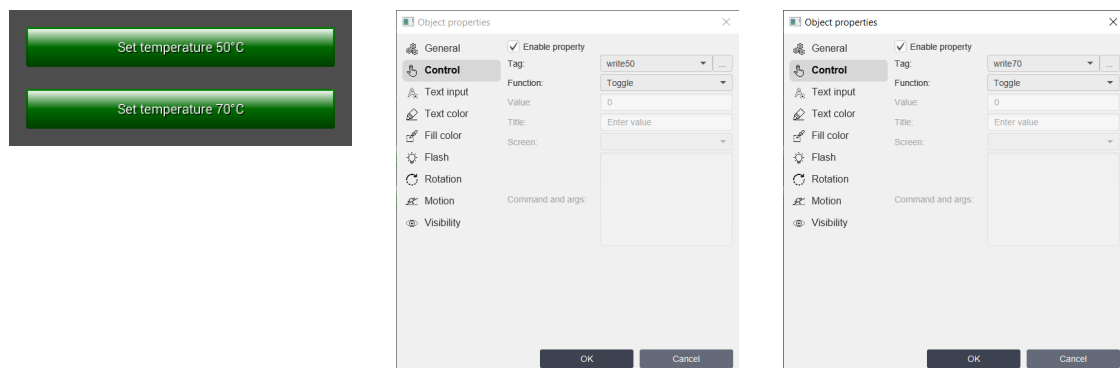
In this example, we'll show you how to change the values of two tags with one click. Suppose we have two containers with liquid that needs to be heated to either 50°C or 70°C. By pressing one button we will set the temperature in both containers - 50°C, and by pressing the other - 70°C.

Let's create two tags - Tag2 and Tag3. We will change the values of these tags simultaneously by pressing the buttons. These buttons will toggle the intermediate tags - write50 and write70.

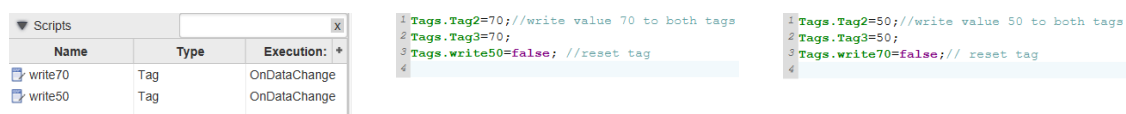
1. Let's create Tags:




2. Create buttons and bind write50 and write70 tags to two buttons:

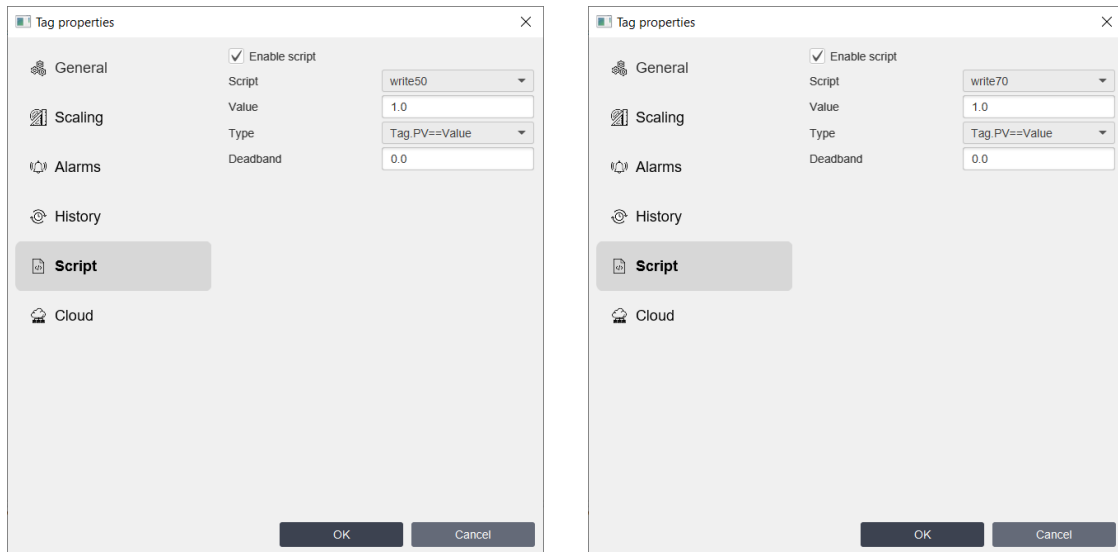


3. Now let's create 2 scripts that will be called when the values of these two tags are switched from FALSE to TRUE:




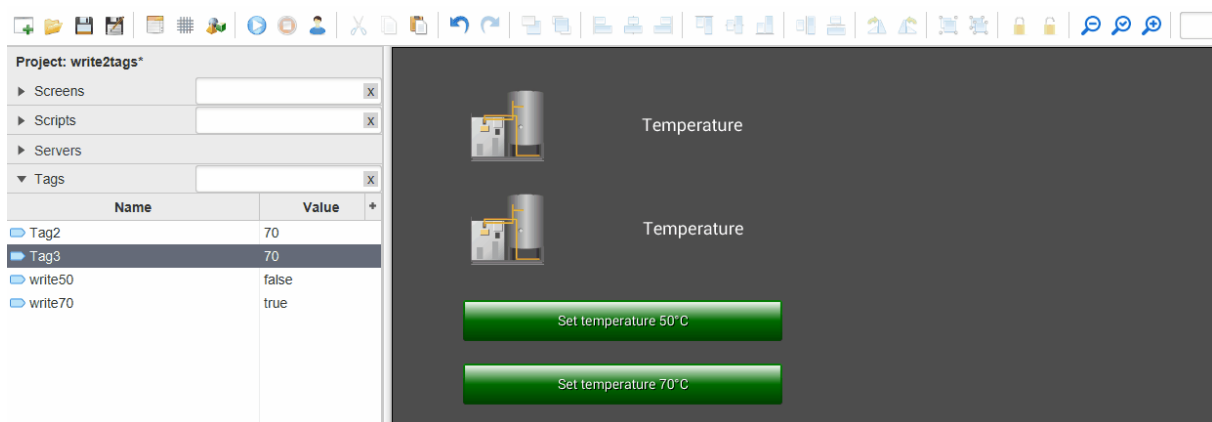
After you have recorded the script, be sure to launch it by clicking the button on the toolbar: 

4. And let's bind these 2 scripts to tags - write50 and write70:



Now when tags write50 and write70 switch from FALSE to TRUE, the corresponding script is called.

5. [Run simulation](#)  to check the settings:



You can download this project [here](#).

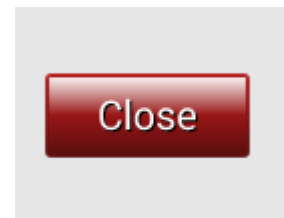
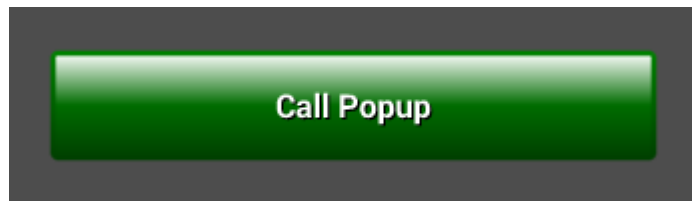
9.8.2 Write value when screen is opened and closed

In this example, we'll show how to record a value when opening and closing a screen.

1. Create a tag named Tag:

Tags		X
Name		Value
Tag		0

2. Let's create 2 screens - General and Popup. On the general screen we will place a button that will open the Pop-up screen and on the pop-up screen we will place a button to close the pop-up screen:



3. Let's create 2 scripts of the Screen type. One is executed when the popup screen opens. The second one is executed when the popup window is closed:

Script properties

Group:

Subgroup:

Name:

Comment:

Background color:

Script type:

Language:

Dimension: X

☐ Every cycle

Execution:

☒ Run in UI:

OK Cancel

Script properties

Group:

Subgroup:

Name:

Comment:

Background color:

Script type:

Language:

Dimension: X

☐ Every cycle

Execution:

☒ Run in UI:

OK Cancel

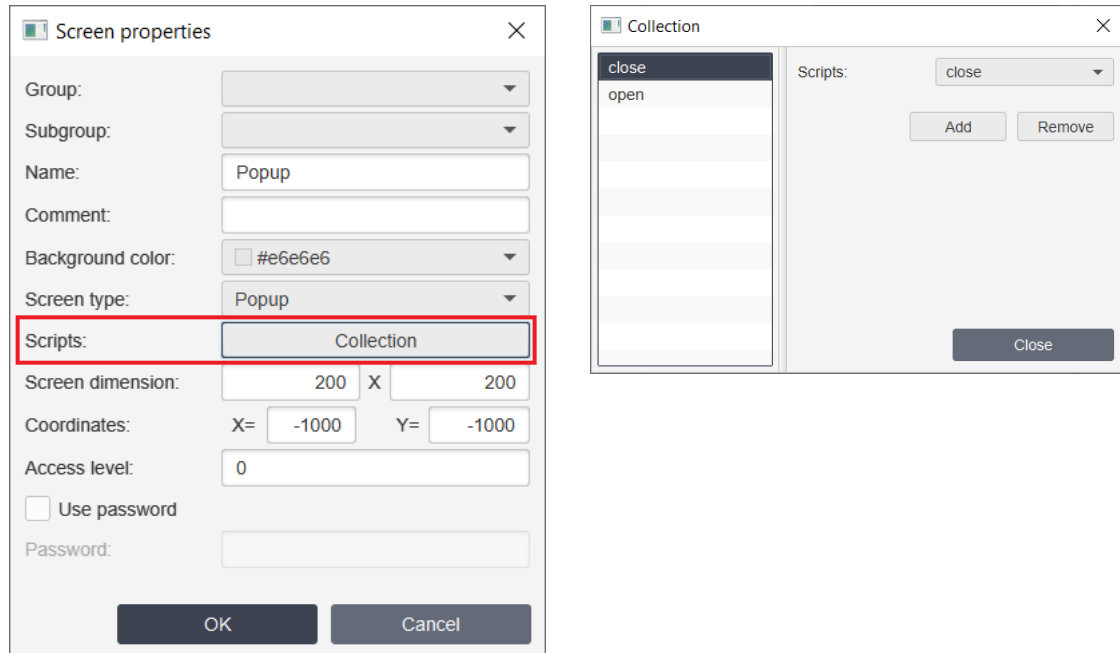
Let's write scripts:

```
1 Tags.Tag=10;
```

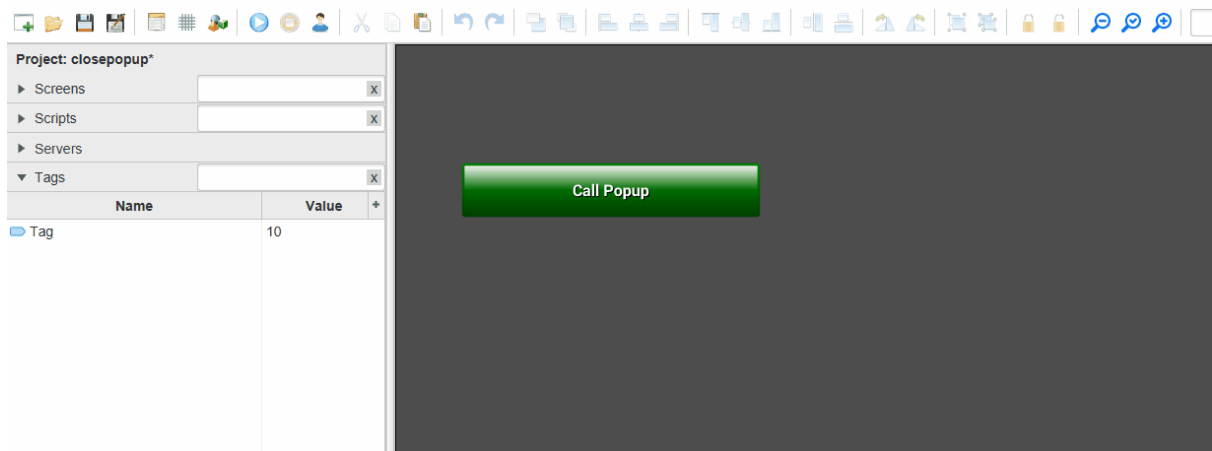
```
1 Tags.Tag=0;
```

After you have recorded the script, be sure to launch it by clicking the button on the toolbar:

4. Let's link these scripts to the pop-up screen:



5. [Run simulation](#) ⁷⁰ to check the settings:



You can download this project [here](#).

9.9 IOT clouds

Examples of working with clouds:

- [IBM Watson IOT](#) ⁵⁹⁸
- [Yandex cloud](#) ⁶¹⁵

9.9.1 IBM Watson IoT

IBM Cloud is a variety of different services. In this example we'll require only one service - Watson IoT. In the given example TeslaSCADA2 Runtime reads data from Modbus device and sends them to IBM Cloud via MQTT protocol in JSON format by using MQTT publisher.

Free (Lite) plan of "Watson IoT" can be used for testing. It includes:

- up to 500 devices,
- up to 500 connections,
- monthly limits
 - up to 200 Mb of traffic,
 - up to 200 Mb of analyzed data
 - up to 200 MB of locally analyzed data (Edge).

More:

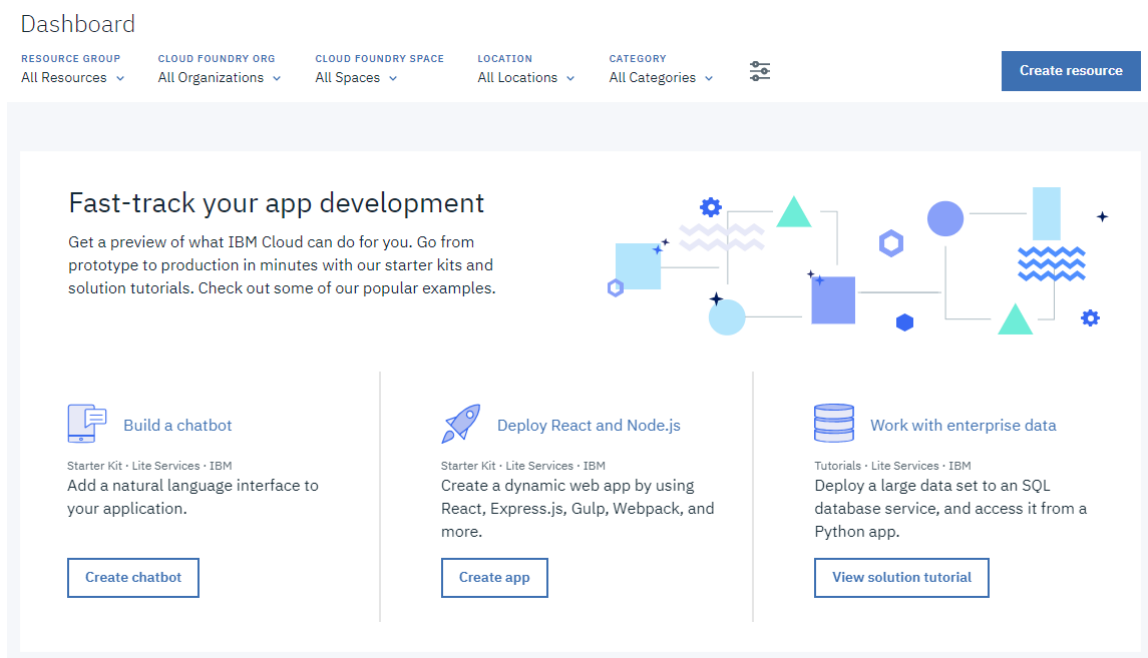
- [Watson IoT](#) (in English)

Setting IBM Watson IoT

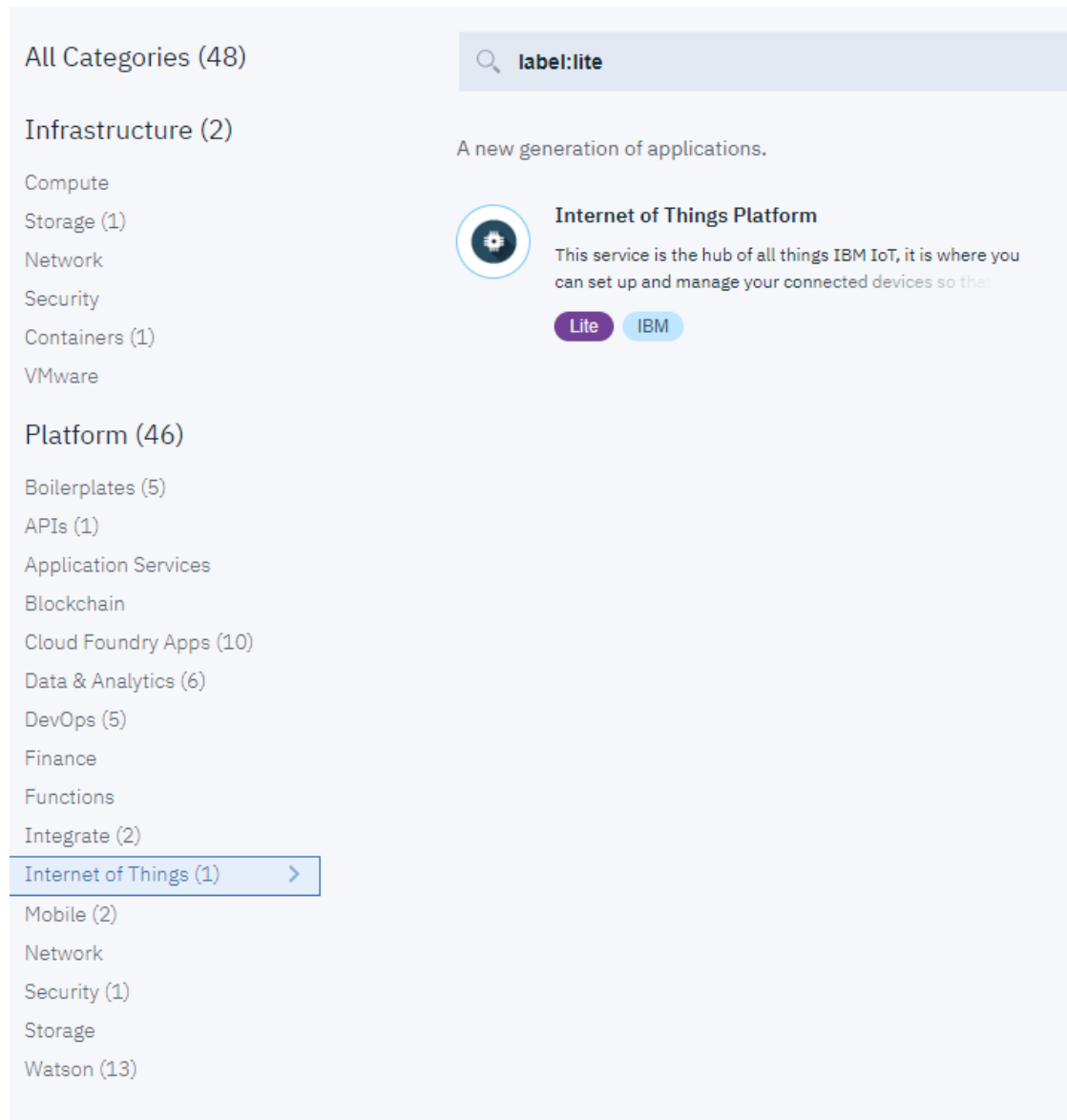
To connect to IBM Watson IoT platform, it is required: [to get IBM Cloud \(IBMid\) account](#).

Creating IBM Watson IoT instance

1. Enter your account and go to [Dashboard](#). Click «Create Resource» button.



2. Select **Internet of Things** category and click **Internet of Things Platform**.



3. Select a region in the parameters, for example, US South (more functions are available for this region).

The screenshot shows the configuration page for the 'Internet of Things Platform' service. On the left, a text block describes the service: 'This service is the hub for IBM Watson IoT and lets you communicate with and consume data from connected devices and gateways. Use the built-in web console dashboards to monitor your IoT data and analyze it in real time. Then, enhance and customize your IBM Watson IoT Platform experience by building and connecting your own apps by using messaging and REST APIs.' The right side contains configuration fields: 'Service name:' with the value 'Internet of Things Platform-zc', 'Choose a region/location to deploy in:' with a dropdown menu showing 'United Kingdom', 'Choose an organization:' with the value 'mtttb@yandex.ru', and 'Choose a space:' with the value 'dev'.

4. Select buying plan (for example, Lite) and click «Create» button:

Pricing Plans Monthly prices shown are for country or region: [Russian Federation](#)

PLAN	FEATURES	PRICING
✓ Lite	Includes up to 500 registered devices, and a maximum of 200 MB of each data metric Maximum of 500 registered devices Maximum of 500 application bindings Maximum of 200 MB of each of data exchanged, data analyzed and edge data analyzed	Free
<p>The Lite service plan for Internet of Things Platform includes up to 500 registered devices, and a maximum of 200 MB each of data exchanged, data analyzed, and edge data analyzed per month.</p> <p>Lite plan services are deleted after 30 days of inactivity.</p>		
Standard	The Standard service plan for Internet of Things Platform includes your free tier of 200 MB each of data exchanged, data analyzed and edge data analyzed per month at no cost. Above the free quota, all three metrics are tiered by usage in MB Charge per MB of data exchanged Charge per MB of data analyzed Charge per MB of edge data analyzed Multi-Tiered	Expand each section to view details
Advanced Security	The Advanced Security service plan for Internet of Things Platform includes your free tier of 200 MB each of data exchanged, data analyzed and edge data analyzed, just as for Standard Plan. Additionally, included in your free tier, Advanced Risk & Security Management features are provided. Above the free quota, all three metrics are tiered by usage in MB When your free tier MB use is exceeded, charges will apply. These are: Charge per MB of data exchanged Charge per MB of data analyzed Charge per MB of edge data analyzed Multi-Tiered	Expand each section to view details

Create

5. The added service is displayed in the list on the dashboard. Click «Launch» button in the window that appears.

[Internet of Things](#) /

Internet of Things Platform-zc



Location: United Kingdom

Org: mtttx@yandex.ru

Space: dev

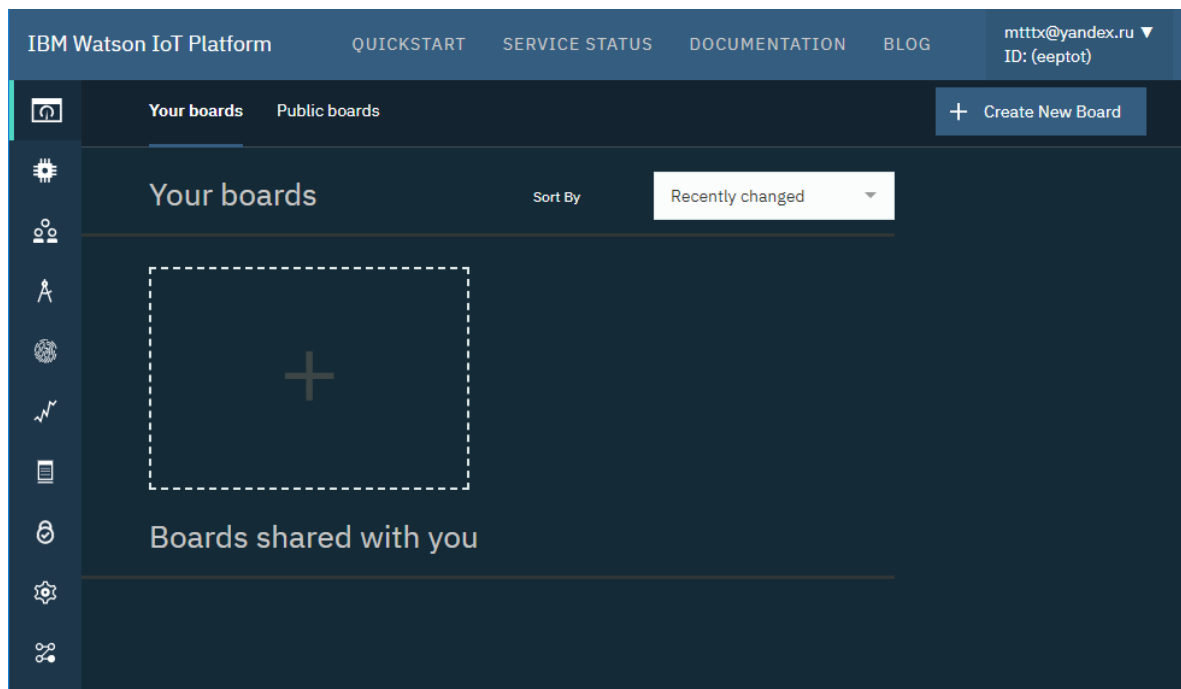


Let's get started with Watson IoT Platform

Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

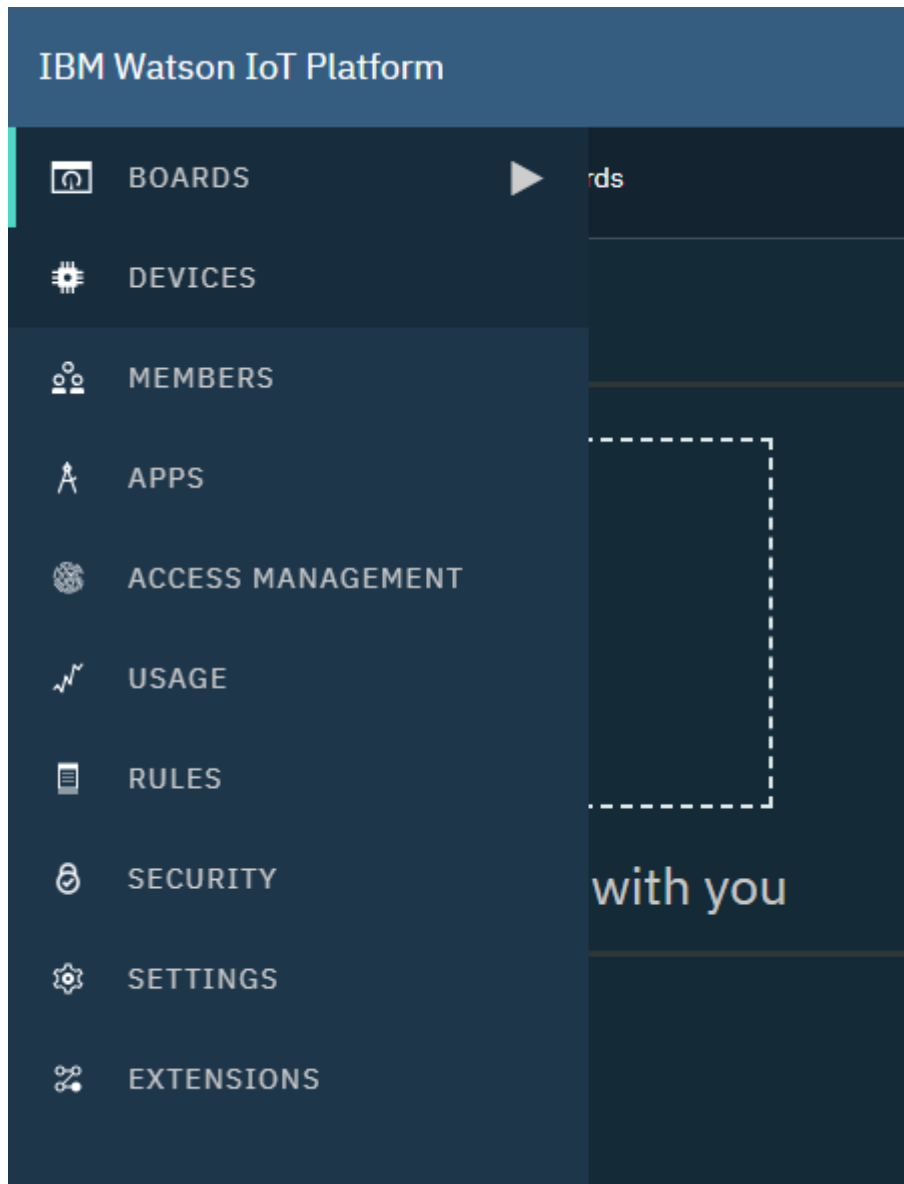
[Launch](#)[Docs](#)

6. A panel to control IoT Platform opens in a new window.

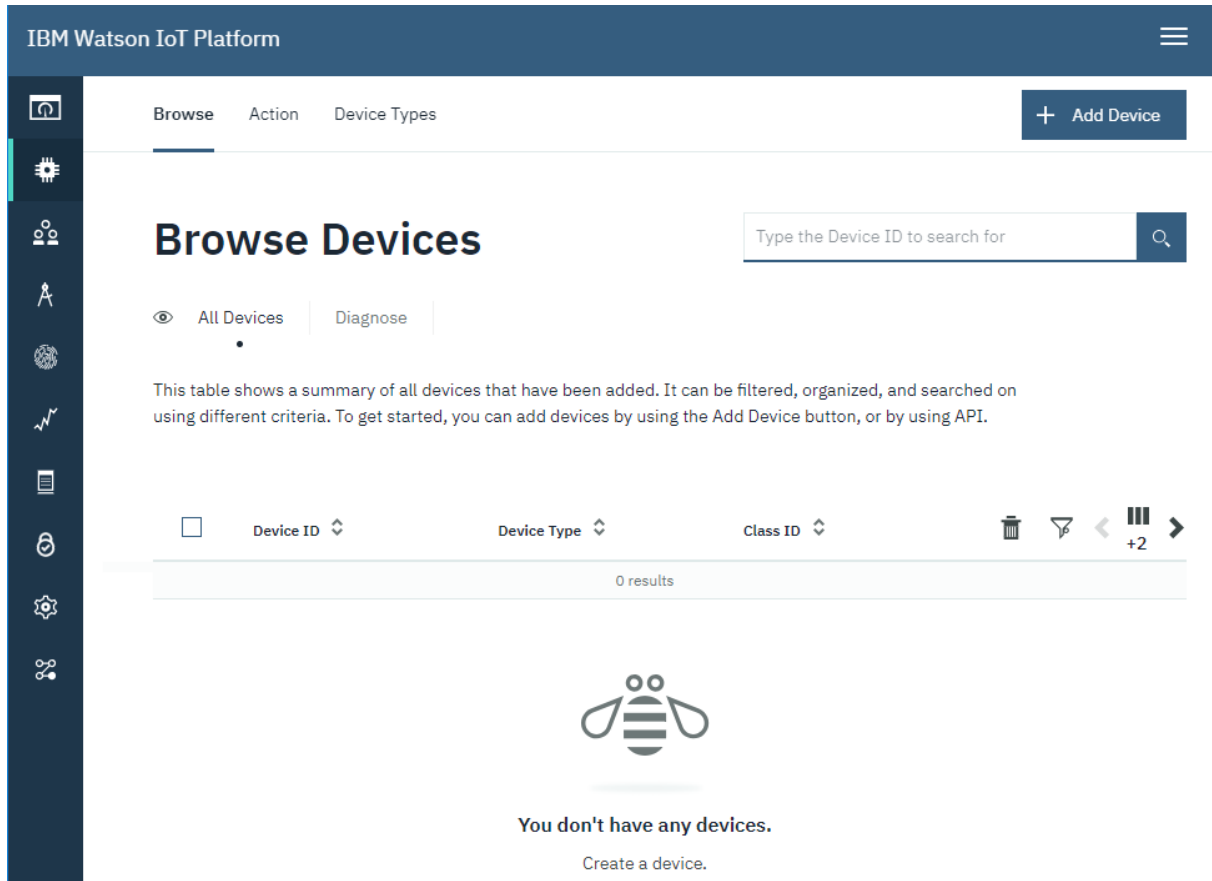


Adding devices

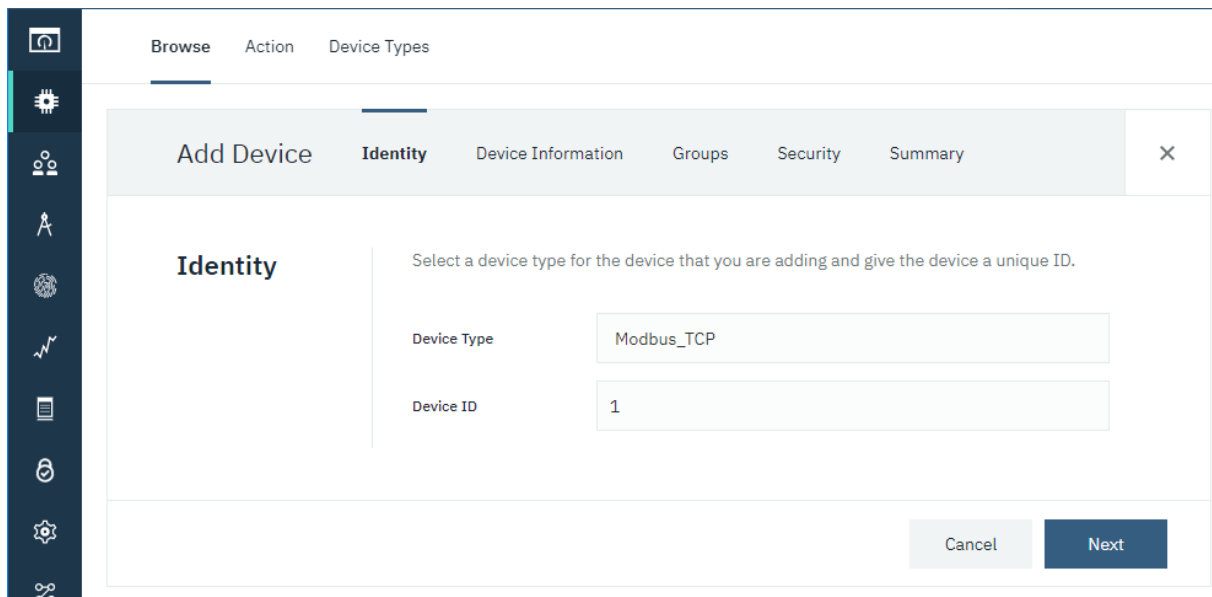
1. Go to Devices tab on the dashboard of IoT Platform.



2. Click «Add Device» button to add a device.



3. Set Device Type and Device ID in the window that appears and click «Next».



4. Enter information about the device and click «Next».

Browse Action Device Types

Add Device Identity **Device Information** Groups Security Summary X

Device Information

You can modify the default device information and enter more information about the device for identification purposes.

Serial Number	0022112	Manufacturer	Enter Manufacturer
Model	Enter Model	Device Class	Enter Device Class
Description	Enter Description	Firmware Version	1.0
Hardware Version	Enter Hardware Version	Descriptive Location	Enter Descriptive Location

+ Add Metadata

< Next


5. Add a group (you can skip this step), click «Next».

Add Device Identity Device Information **Groups** Security Summary X

Groups Beta

This table shows the groups that this device belongs to. For more information about groups, see [Managing groups](#).

Add to Groups

Group name	Number of Devices
	
You currently don't have any groups assigned to the device.	
Add to Groups	

< Next

6. Create a token in Security window for authentication (if the fields is left empty, the token is generated automatically). Click «Next».

Browse

Action

Device Types

×

Device Security

There are two options for selecting a device authentication token.

Auto-generated authentication token (default)

Allow the service to generate an authentication token for you. Tokens are 18 characters and contain a mix of alphanumeric characters and symbols. The token is returned to you at the end of the device registration process.

Self-provided authentication token

Provide your own authentication token for this device. The token must be between 8 and 36 characters and contain a mix lowercase and uppercase letters, numbers, and symbols, which can include hyphens, underscores, and periods. Do not use repeated characters, dictionary words, user names, or other predefined sequences.

Authentication Token

ⓘ

Make a note of the generated token. Lost authentication tokens cannot be recovered. Tokens are encrypted before being stored.

Authentication token are encrypted before we store them.

◀

Next

7. See the result of creating a device and click «Done».

Browse

Action

Device Types

Add Device

Identity

Device Information

Groups

Security

Summary

×

Summary

Verify that the following information is correct then select Done

Device Type

Modbus_TCP

Device ID

1

Serial Number

0022112

Firmware Version

1.0

View Metadata

Security Token

To be generated

◀

Done

8. A message appears that a device was registered and you'll get information to connect the device to the platform. If the token field was left empty, you'll get an automatically generated token.

DEVICE DRILLDOWN

- Device Credentials
- Connection Information
- Recent Events
- State
- Device Information
- Groups
- Metadata
- Extension Configuration
- Diagnostics
- Connection Logs
- Device Actions

Device 1

Device Credentials

You registered your device to the organization. Add these credentials to the device to connect it to the platform. After the device is connected, you can navigate to view connection and event details.

Organization ID	eeptot
Device Type	Modbus_TCP
Device ID	1
Authentication Method	use-token-auth
Authentication Token	eCan5GKWm)H7IQ+fvo

 Authentication tokens are non-recoverable. If you misplace this token, you will need to re-register the device to generate a new authentication token.

[Find out how to add these credentials to your device](#)

Authentication token (password) is given only once. Save it.

Setting MQTT publisher

1. Enable MQTT publisher. And setup it:

Edit Project

General Events/History OPC UA **MQTT Publisher** Web-server

☒ Enable MQTT Publisher

Broker URL: ssl://eepot.messaging.internetofthings.ibmcloud.com:8883

Username: use-token-auth

Password: eCan5GKWm)H7IQ+fvo

Client ID: d:eepot:Modbus_TCP:1

Write topic format: iot-2/evt/{tagname}/fmt/txt

Read topic format: iot-2/cmd/{tagname}/fmt/txt

QoS: QoS 2

☐ Enable TLS/SSL

Protocol: TLSv1.2

Certificate filename:

☐ Enable Client Certificate

Client Certificate:

Client Private Key:

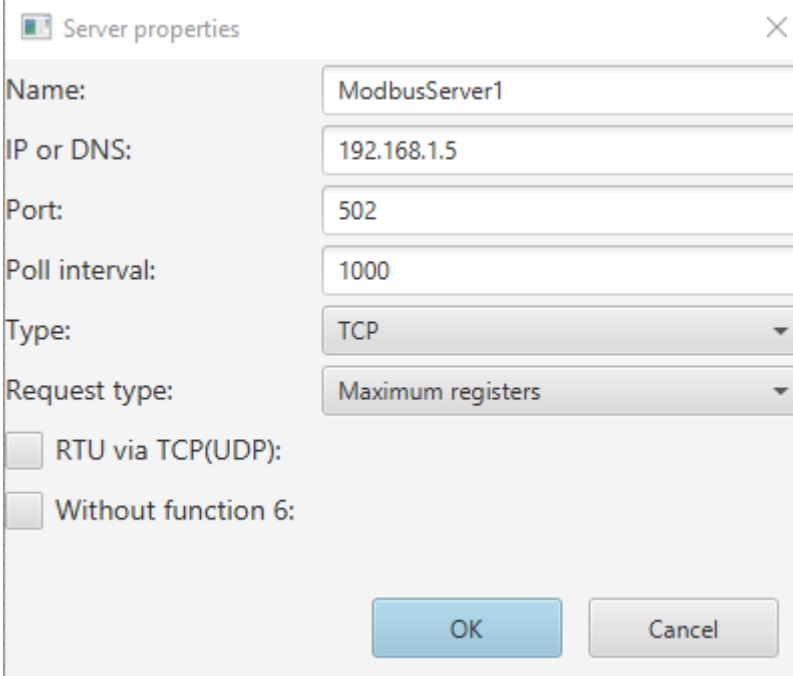
Private Key Password:

☐ PEM Formatted

OK Cancel

- **Broker URL** - address and port of the IoT cloud interface, it is coded the following way: ssl://<your_orgID>.messaging.internetofthings.ibmcloud.com:8883
- **Username** - login, use-token-auth fixed value must be entered here
- **Password** - created or generated Authentication Token
- **Client ID** - client identifier is coded this way: d:<your_orgID>:<your_Type>:<your_Device>
- **Write topic format** - format of writing tags in the topic: iot-2/evt/{tagname}/fmt/txt , {tagname} - name of tags in your project.
- **Read topic format** - format of reading tags in the topic: iot-2/cmd/{tagname}/fmt/txt, {tagname} - name of tags in your project.
- **QoS** - type of the MQTT message.

2. Create Modbus server and setup it:

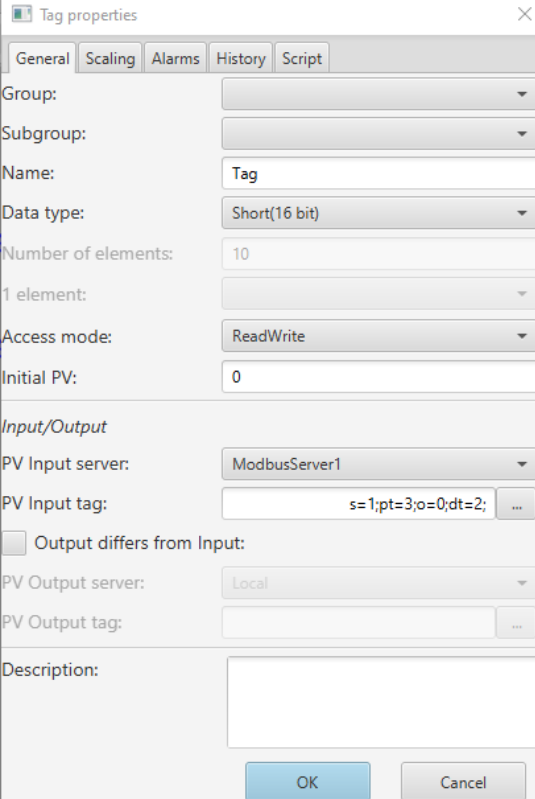


The 'Server properties' dialog box is shown with the following settings:

- Name: ModbusServer1
- IP or DNS: 192.168.1.5
- Port: 502
- Poll interval: 1000
- Type: TCP
- Request type: Maximum registers
- ☐ RTU via TCP(UDP):
- ☐ Without function 6:

Buttons: OK, Cancel

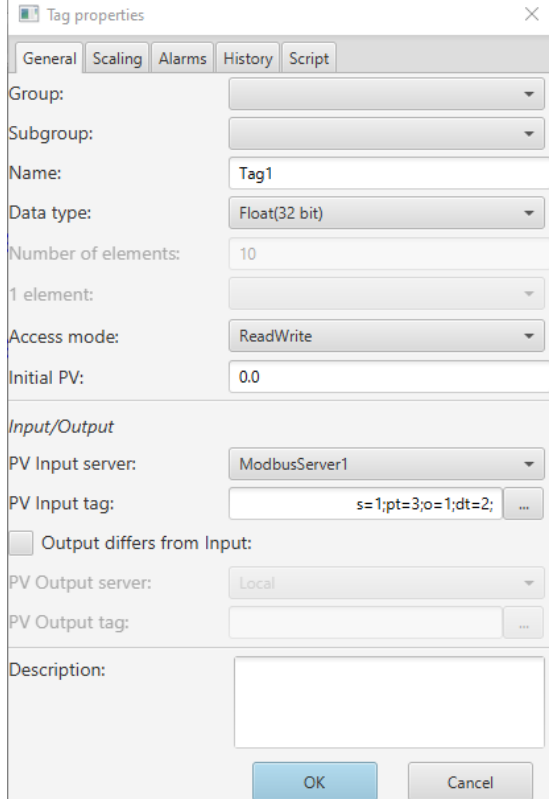
3. Create 2 tags and bind to 2 first registers. And setup it:



The 'Tag properties' dialog box for 'Tag' is shown with the following settings:

- Group: (empty)
- Subgroup: (empty)
- Name: Tag
- Data type: Short(16 bit)
- Number of elements: 10
- 1 element: (empty)
- Access mode: ReadWrite
- Initial PV: 0
- Input/Output
 - PV Input server: ModbusServer1
 - PV Input tag: s=1;pt=3;o=0;dt=2;
 - ☐ Output differs from Input:
 - PV Output server: Local
 - PV Output tag: (empty)
- Description: (empty)

Buttons: OK, Cancel



The 'Tag properties' dialog box for 'Tag1' is shown with the following settings:

- Group: (empty)
- Subgroup: (empty)
- Name: Tag1
- Data type: Float(32 bit)
- Number of elements: 10
- 1 element: (empty)
- Access mode: ReadWrite
- Initial PV: 0.0
- Input/Output
 - PV Input server: ModbusServer1
 - PV Input tag: s=1;pt=3;o=1;dt=2;
 - ☐ Output differs from Input:
 - PV Output server: Local
 - PV Output tag: (empty)
- Description: (empty)

Buttons: OK, Cancel

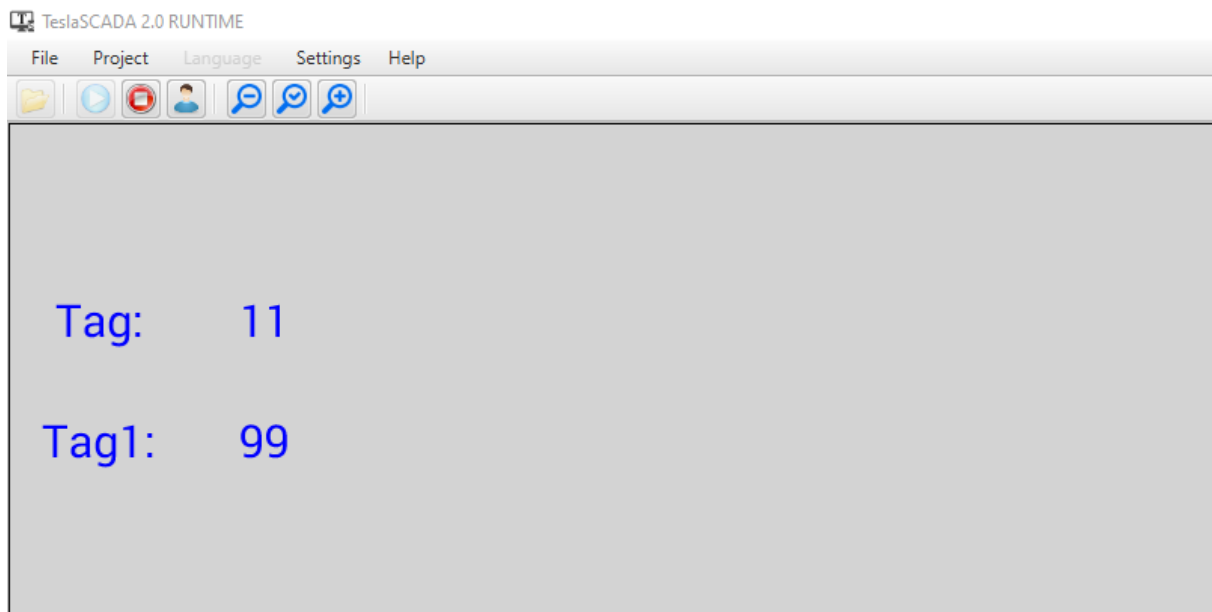
Now when you run this project in TeslaSCADA2 Runtime. It's connecting to the cloud and publish tags values in the cloud:

In the simulator:

ID = 1: F = 03

	Alias	00000
0		11
1		99
2		0
3		0
4		0
5		0
6		0
7		0
8		0
9		0


In the project:



In the cloud:

Состояние

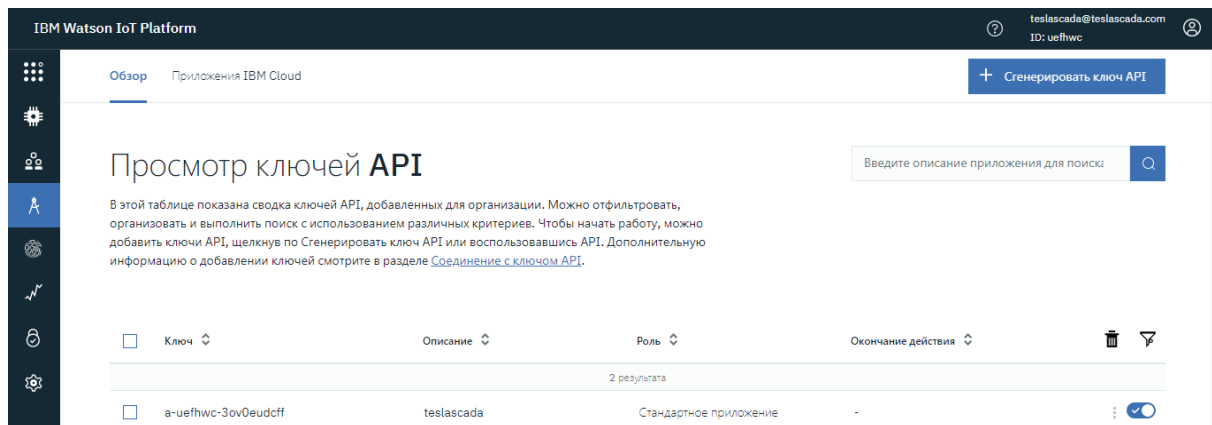
В этой таблице показан список точек данных, полученных от этого устройства.

 Вывод необработанных данных | Нет доступных интерфейсов

Свойство	Значение	Тип	Событие	Последнее получение
Tag (txt)	11	Число	Tag	несколько секунд назад
Tag1 (txt)	99	Число	Tag1	несколько секунд назад

Setting MQTT server

1. For getting information from the cloud we have to create Application in IOT Watson (russian language):



2. Now we can create MQTT server in the new project:

The screenshot shows a 'Server properties' dialog box with the following fields and values:

- Name: MQTTServer1
- URI: ssl://eeptot.messaging.internetofthings.ibmcloud.com:8883
- Username: a-uefhwc-3ov0eudcffi
- Password: CxgCvs5HAVIk?*vHDA
- Client ID: a:uefhwc:teslascada
- ☐ Enable TLS/SSL
- Protocol: TLSv1.2
- Certificate filename: (empty)
- ☐ Enable Client Certificate
- Client Certificate: (empty)
- Client Private Key: (empty)
- Private Key Password: (empty)
- ☐ PEM Formatted

Buttons: OK, Cancel

- **URI** - address and port of the IoT cloud interface, it is coded the following way:
ssl://<your_orgID>.messaging.internetofthings.ibmcloud.com:8883
- **Username** - login, It is coded in the following way:
{api key} - you can see it in the picture. It contains a -{your_orgID}-{code}. In the picture other {your_orgID}. Should be eeptot.
- **Password** - created or generated Authentication Token
- **Client ID** - client identifier is coded this way: d:<your_orgID>:<name of the application>

3.Create 2 tags for reading from the cloud:

The image shows two side-by-side screenshots of software configuration windows.

The left window is titled "Tag properties" and has tabs for "General", "Scaling", "Alarms", "History", and "Script". The "General" tab is active. It contains the following fields:

- Group: (empty dropdown)
- Subgroup: (empty dropdown)
- Name: Tag
- Data type: Short(16 bit)
- Number of elements: 10
- 1 element: (empty dropdown)
- Access mode: ReadWrite
- Initial PV: 0

The "Input/Output" section contains:

- PV Input server: MQTTServer1
- PV Input tag: t=iot-2/type/TeslaSCADA/id/1/evt/T. ...
- ☒ Output differs from Input:
- PV Output server: MQTTServer1
- PV Output tag: t=iot-2/type/TeslaSCADA/id/1/cmd/ ...
- Description: (empty text area)

At the bottom are "OK" and "Cancel" buttons.

The right window is titled "Pointer settings" and contains the following fields:

- Topic: iot-2/type/TeslaSCADA/id/1/evt/Tag/fmt/b
- QoS: QoS0
- ☒ Retained
- JSON path: (empty text area)

At the bottom are "OK" and "Cancel" buttons.

The format of the topic is interesting for us:

iot-2/type/{device_type}/id/{device_id}/evt/{event_id}/fmt/{format_string}

device_type - type of the device

device_id - ID of the device

event_id - name of the Tag you setup in project with publisher.

format_string - format of the topic.(txt in our case).

For writing topic should like this:

iot-2/type/{device_type}/id/{device_id}/cmd/{cmd_id}/fmt/{format_string}

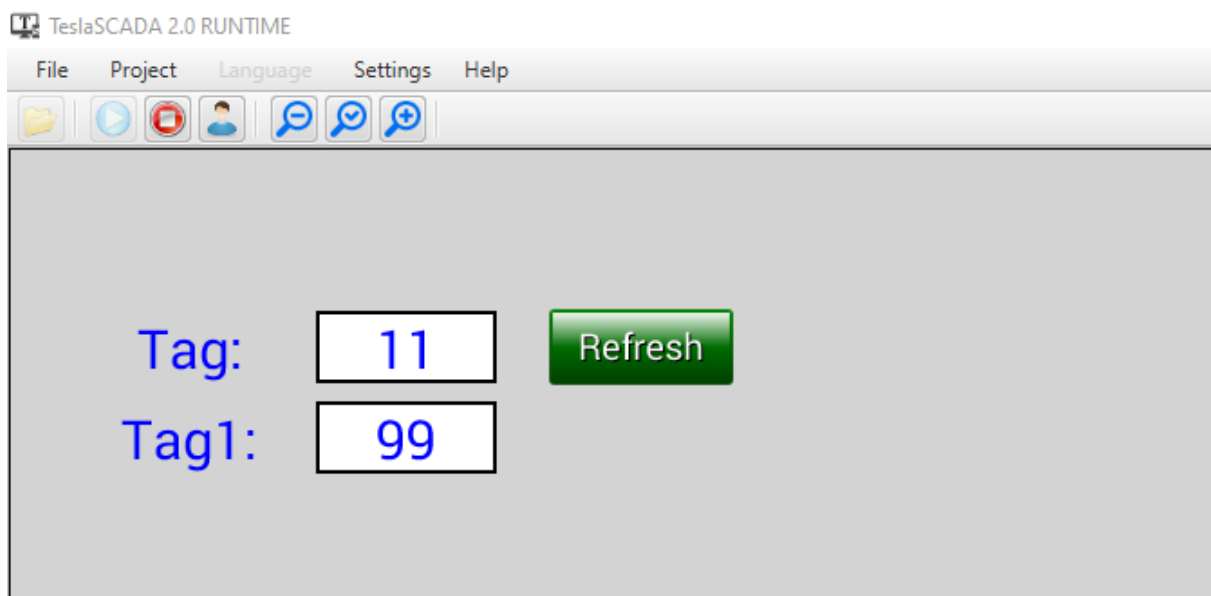
4. In some reasons when you connect to IBM cloud and subscribe to the topics values are not renew.

We create some possibility to renew values. You have to create tag in the MQTT to refresh publisher:

The image shows two dialog boxes side-by-side. The 'Tag properties' dialog has tabs for General, Scaling, Alarms, History, and Script. The General tab is active, showing fields for Group, Subgroup, Name (refreshpublisher), Data type (Boolean), Number of elements (10), 1 element, Access mode (ReadWrite), Initial PV (false), Input/Output section with PV Input server (MQTTSer1), PV Input tag (t=iot-2/type/TeslaSCADA/id/1/cmd/), a checkbox for 'Output differs from Input', PV Output server (Local), PV Output tag, and a Description field. The 'Pointer settings' dialog has fields for Topic (iot-2/type/TeslaSCADA/id/1/cmd/refreshpt), QoS (QoS0), a checked 'Retained' checkbox, and a JSON path field. Both dialogs have OK and Cancel buttons.

Topic should look like this: **iot-2/type/TeslaSCADA/id/1/cmd/refreshpublisher/fmt/txt**
cmd_id - refreshpublisher

5. After starting project with MQTT client and refresh values we'll get:



9.9.2 Yandex cloud

Yandex IoT Core is a cloud-based fail-safe MQTT broker that ensures secure two-way communication between devices and local or cloud resources.

Devices and registries interact using X.509 certificates:

- If you have a certificate, just add it to the device in the registry.
- If don't have one, you can create a certificate, for example, with [OpenSSL](#):

```
openssl req -x509 \
-newkey rsa:4096 \
-keyout key.pem \
-out cert.pem \
-nodes \
-days 365 \
-subj '/CN=localhost'
```

Create registry

The screenshot shows the Yandex Cloud IoT Core console. The left sidebar contains a menu with 'tesla Registry' selected, and sub-items: 'Overview' (active), 'Devices', 'Logs', and 'Operations'. The main content area is titled 'Overview' and includes sections for 'General information', 'Passwords', and 'Certificates'.

General information

Name	tesla
Id	are11i61o5qr0af5j0t
Description	IoT cloud
Date created	05 July 2021, at 09:51

Passwords

Id	Date created
arequc674s2saq3m2rcu	08 July 2021, at 20:45

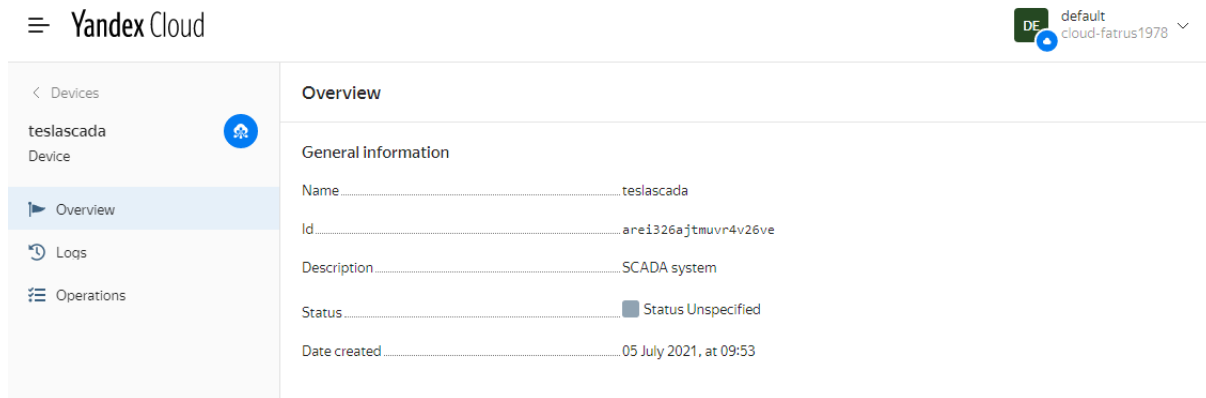
[Add password](#)

Certificates

Digital fingerprint	Contents	Date
ec21043832baa766888e2db7321dd53d5cc8cc32	-----BEGIN CERTIFICATE----- MIIEpDCCAowCCQCzQadJ5...	06 July 2021, at 17:21

You also have to add certificate you created.

Create device



Setup MQTT publisher

The 'Edit Project' dialog box has several tabs: 'General', 'Events/History', 'OPC UA', 'MQTT Publisher' (selected), and 'Web-server'. Under the 'MQTT Publisher' tab, the following settings are visible:

- ☒ Enable MQTT Publisher
- Broker URL: ssl://mqtt.cloud.yandex.net:8883
- Username: arei326ajtmuvr4v26ve
- Password: password
- Client ID: (empty field)
- Write topic format: \$devices/arei326ajtmuvr4v26ve/events/{tagname}
- Read topic format: \$devices/arei326ajtmuvr4v26ve/commands/{tag}
- QoS: QoS 0 (dropdown menu)
- ☒ Enable TLS/SSL
- Protocol: TLSv1.2 (dropdown menu)
- Certificate filename: rootCA.crt
- ☐ Enable Client Certificate
- Client Certificate: cert.pem
- Client Private Key: key.pem
- Private Key Password: (empty field)
- ☒ PEM Formatted

At the bottom right are 'OK' and 'Cancel' buttons.

where:

Username - Device ID

Password - password of the device.

Write topic format - in our case \$devices/arei326ajtmuvr4v26ve/events/{tagname}.

It contains ID of the device and keyword {tagname} for publishing tag's values.

Read topic format - in our case \$devices/arei326ajtmuvr4v26ve/commands/{tagname}

It contains ID of the device and keyword {tagname} for subscribing to tag's values.

Certificate filename - you have to download certificate file from here:

<https://cloud.yandex.com/en/docs/iot-core/concepts/mqtt-properties>

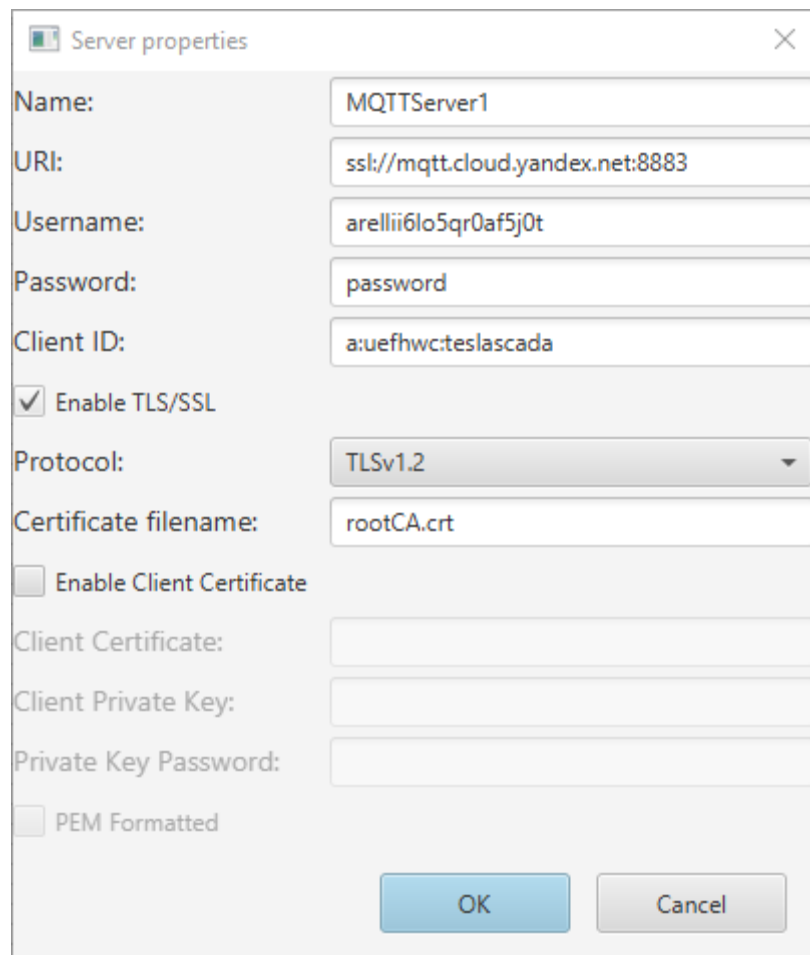
And place this file in the folder private where installed TeslaSCADA2 Runtime.

Now when you run the project created with this MQTT publisher settings all tags values used in this project will be published in the broker.

The published values don't have retain properties.

Setup MQTT client

To read data from the Yandex IOT we create new project and setup MQTT server:



Server properties

Name: MQTTServer1

URI: ssl://mqtt.cloud.yandex.net:8883

Username: arellii6lo5qr0af5j0t

Password: password

Client ID: a:uefhwc:teslascada

☒ Enable TLS/SSL

Protocol: TLSv1.2

Certificate filename: rootCA.crt

☐ Enable Client Certificate

Client Certificate:

Client Private Key:

Private Key Password:

☐ PEM Formatted

OK Cancel

where:

Username - Registry ID

Password - password of the registry.

Certificate filename - you have to download certificate file from here:

<https://cloud.yandex.com/en/docs/iot-core/concepts/mqtt-properties>

And place this file in the folder private where installed TeslaSCADA2 Runtime.

Setup Tag

Tag properties

General Scaling Alarms History Script

Group:

Subgroup:

Name:

Data type:

Number of elements:

1 element:

Access mode:

Initial PV:

Input/Output

PV Input server:

PV Input tag: ...

☒ Output differs from Input:

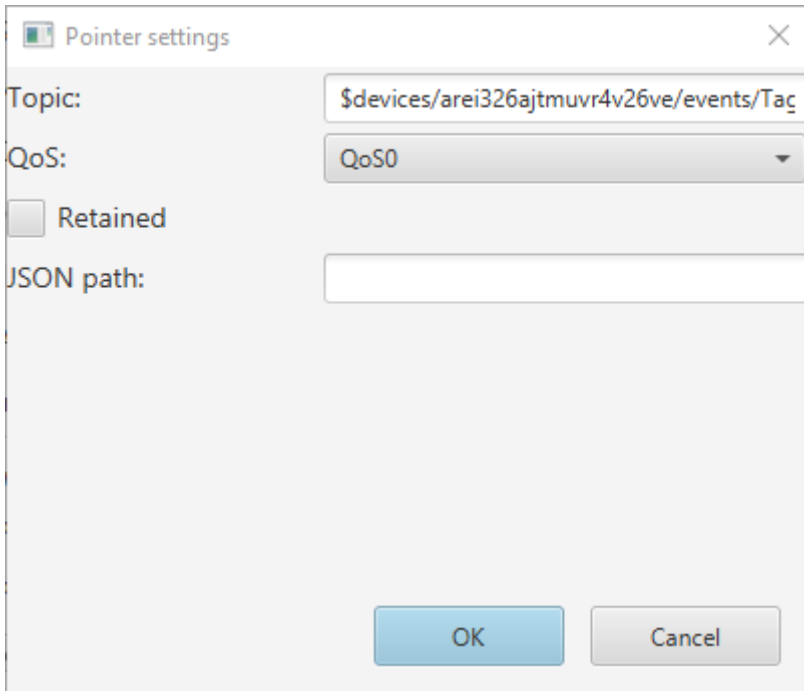
PV Output server:

PV Output tag: ...

Description:

OK Cancel

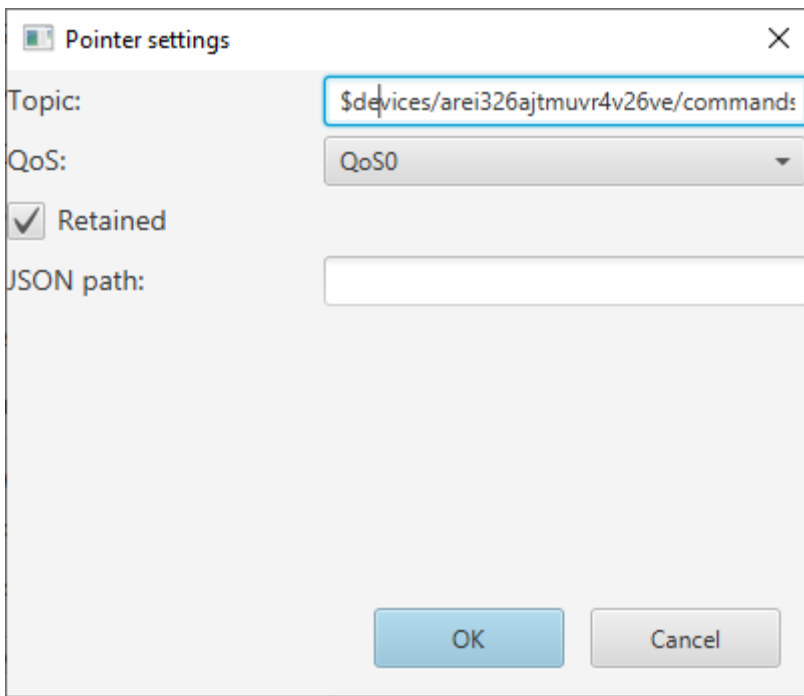
PV Input tag



A screenshot of a 'Pointer settings' dialog box. The 'Topic' field contains the text '\$devices/arei326ajtmuvr4v26ve/events/Tag'. The 'QoS' dropdown menu is set to 'QoS0'. The 'Retained' checkbox is unchecked. The 'JSON path' field is empty. At the bottom right are 'OK' and 'Cancel' buttons.

It contains ID of the device you setup in Yandex IOT core, keyword events and name of the tag you want to read from the publisher project.

PV Output tag



A screenshot of a 'Pointer settings' dialog box, similar to the one above but with different settings. The 'Topic' field contains the text '\$devices/arei326ajtmuvr4v26ve/commands'. The 'QoS' dropdown menu is set to 'QoS0'. The 'Retained' checkbox is checked. The 'JSON path' field is empty. At the bottom right are 'OK' and 'Cancel' buttons.

It contains ID of the device you setup in Yandex IOT core, keyword commands and name of the tag you want to write to the publisher project.

Now you can read values from the Yandex cloud by using this project. And write commands also.