# Table of Content

About TeslaSCADA IDE ................................................................. 10

Requirements ................................................................. 10
  - Windows ........................................................................... 10
  - Mac OS ............................................................................. 10
  - Linux .............................................................................. 11

Installation ............................................................................. 11
  - Windows ........................................................................... 11
  - Mac OS ............................................................................. 11
  - Linux .............................................................................. 11

Start TeslaSCADA IDE ............................................................... 12

Project .................................................................................... 15
  - Create project ................................................................. 15
  - Save project ..................................................................... 17
  - Open project .................................................................... 17
  - Edit project properties ..................................................... 18

Screens .................................................................................... 19
  - Create screen .................................................................. 19
  - Open screen .................................................................... 19
  - Copy screen ..................................................................... 19
  - Delete screen .................................................................... 19
  - Edit screen properties ...................................................... 19
  - New Screen Group ............................................................ 19
  - Add to group .................................................................... 19
  - Export screen ................................................................. 19
  - Import screen ................................................................... 20

Servers .................................................................................... 21
  - Create server ................................................................... 21
    - Modbus server ........................................................... 21
    - Siemens server ......................................................... 21
    - Allen Bradley server .................................................. 22
    - OPC UA server .......................................................... 22
    - MQTT server .................................................................. 22
    - Omron server .............................................................. 23
  - Open server properties .................................................... 24
  - Copy server ...................................................................... 24
  - Delete server .................................................................... 24
Scripts.........................................................................................................................25
Create script...............................................................................................................25
Open script .................................................................................................................25
Copy script ................................................................................................................25
Delete script..............................................................................................................25
Edit script properties...............................................................................................25
New script group ......................................................................................................25
Add to group .............................................................................................................25
Export script ............................................................................................................25
Import script ............................................................................................................25
Tags..................................................................................................................................26
Create tag.....................................................................................................................26
  Modbus tag settings ................................................................................................26
  Siemens tag settings ...............................................................................................27
  AllenBradley tag settings .......................................................................................27
  Micrologix tag settings .........................................................................................27
  OPC UA tag settings .............................................................................................27
  MQTT tag settings ...............................................................................................28
  Omron tag settings ..............................................................................................28
Copy tag......................................................................................................................29
Delete tag...................................................................................................................29
Edit tag properties....................................................................................................30
New group tags .........................................................................................................30
Add to group .............................................................................................................30
Export all tags ..........................................................................................................30
Import tags .................................................................................................................30
Users............................................................................................................................31
Create user..................................................................................................................31
Open user properties ...............................................................................................31
Copy user....................................................................................................................31
Delete user..................................................................................................................31
Databases.....................................................................................................................33
Create recipe..............................................................................................................33
Create history............................................................................................................33
Create Odoo ERP ......................................................................................................34
Open database properties .......................................................................................35
Copy database..........................................................................................................35
Delete database .......................................................................................................35
Design screen ................................................................. 36
Create graphical object ..................................................... 36
Resize graphical object ....................................................... 36
Select several objects .......................................................... 36
Move graphical object .......................................................... 36
Open graphical object properties .......................................... 36
Copy graphical object .......................................................... 37
Cut graphical object ............................................................ 37
Paste graphical object ........................................................ 37
Erase graphical object ........................................................ 37
Duplicate graphical object ..................................................... 37
Send to back graphical object ............................................... 37
Bring to front graphical object ............................................... 37
Rotate clockwise graphical object .......................................... 37
Rotate counterclockwise graphical object .............................. 38
Align graphical objects ........................................................ 38
Group graphical objects ......................................................... 38
Ungroup graphical objects ..................................................... 38

Graphical objects .................................................................. 39

Simple Objects library .......................................................... 39
Line ................................................................................. 39
Rectangle .......................................................................... 39
Ellipse ............................................................................... 40
Polyline ............................................................................. 40
Polygon ............................................................................. 40
Sector ............................................................................... 40
Text .................................................................................. 41
Border ................................................................................. 41
Image ................................................................................ 41
Scale .................................................................................. 41

3D Objects library ................................................................ 43
Sphere ............................................................................... 43
Cylinder .............................................................................. 43
Cone ................................................................................... 43
Sector 3D .............................................................................. 44
Polygon 3D ........................................................................... 44
Tank ...................................................................................... 44
Border 3D ................................................................. 44
Text 3D ...................................................................... 45
Value with History and Event ...................................... 45
Buttons and Switches library ...................................... 46
  Button ...................................................................... 46
  Image Button ............................................................ 46
  Switch and Apple switch .......................................... 47
  Switch 3 Pos. ............................................................. 47
Lights/Indicators library .............................................. 48
  Light ....................................................................... 48
  Indicator .................................................................. 48
Pipes library ............................................................. 49
  Pipe ........................................................................ 49
  Pipeline ................................................................... 49
Valves library ............................................................ 50
  Valve ...................................................................... 50
  Ball valve ................................................................ 51
  Position valve .......................................................... 51
Pumps library ............................................................ 52
  Pump ...................................................................... 52
Fans library ............................................................... 52
  Fan ........................................................................ 52
Tanks library .............................................................. 53
  Vertical tank ............................................................ 53
Conveyers library ....................................................... 53
  Belt conveyer .......................................................... 53
Analog meters library .................................................. 54
  Analog meter ........................................................... 54
  Range Indicator ........................................................ 54
  Other analog meters ................................................. 55
Digital meters library .................................................. 56
  Digital meters .......................................................... 56
Controls library .......................................................... 57
  Slider ...................................................................... 57
  Slider vertical and horizontal .................................... 57
  Counter and counter rectangle ............................... 58
  Selector, ComboBox and MenuBox ........................... 58
Flash..............................................................................................................................74
Rotation..........................................................................................................................75
Motion..............................................................................................................................75
Visibility..........................................................................................................................76
Line color .........................................................................................................................76
Fill color ..........................................................................................................................77
Filling ................................................................................................................................77
Text color .........................................................................................................................78
Control (for buttons) ........................................................................................................78
Text input ..........................................................................................................................80
Output ..................................................................................................................................80
Indicator ............................................................................................................................80
Rotation Indicator ............................................................................................................81
Image ....................................................................................................................................81
Color ....................................................................................................................................81
Control (slider) ...................................................................................................................82
Control (counter) ...............................................................................................................82
Value (for meters) ...............................................................................................................82
Value (for Range indicator and Gauges) ..............................................................................83
Switch control ....................................................................................................................84
Switch control (3 pos.) .....................................................................................................84
Selector ................................................................................................................................85
Row number .......................................................................................................................85

Design script .....................................................................................................................86
Create script object ...........................................................................................................86
Connect script objects .......................................................................................................86
Bind script object to the tag ...............................................................................................86
Enter value to the value script object ...............................................................................86
Duplicate script object ......................................................................................................86
Erase script object .............................................................................................................86
Erase connection line .........................................................................................................86

Script objects of FBD language ......................................................................................87
Input/Output library ..........................................................................................................87
Logical library .....................................................................................................................87
Bitmap operations library .................................................................................................87
Arithmetic library ...............................................................................................................87
Compare library ................................................................................................................88
Select library ......................................................................................................................88
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.

Requirements

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

Windows

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

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

**Memory:** 512MB of RAM (1 GB recommended).

**Disc Space:** 256MB of free disc space.

Mac OS

**Processors:** Dual-Core Intel, PowerPC G5

**Operating systems:** 10.7.3 or greater

**Memory:** 512MB of RAM (1 GB recommended).

**Disc Space:** 256MB of free disc space.
**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:** 512MB of RAM (1 GB recommended).

**Disc Space:** 256MB of free disc 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.

**Installation**

**Windows**
To install TeslaSCADA IDE download EXE package for your operating system. Run installation file and go through installation procedure.

**Mac OS**
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.

**Linux**
To install TeslaSCADA IDE download RPM package for your operating system. 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)
Start TeslaSCADA IDE

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

Main menu

**File** - manipulation with project files.

**Edit** - manipulation with objects (cut, copy, paste and etc.).

**Arrange** - arrange manipulation with objects (align, rotate and etc.).

**Project** - 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
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.

**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 Counterclockwise** – rotate counterclockwise selected object(s).
**Project window**

Project window contains all the information about the project and consists:

- **Screens** - contains all screens of the project.
- **Scripts** - contains all scripts of the project.
- **Servers** - contains all servers of the project.
- **Tags** - contains all tags of the project.
- **Users** - contains all users of the project.

**Screen window**

Screen window contains all objects of the current screen.

**Status bar**

Status bar contains information about path of the current project, information about selected object (x,y coordinates and dimension) and information about run or not simulation mode.

**Canvas**

Place for the design screen or script.
Project

Create project
To create a new project TeslaSCADA IDE must be started.

1. Click on the **New** icon in the toolbar or use the command **New** from the main menu **File**. You’ll see the following window:

![Edit Project Window](image)

2. On the **General** tab:

   2.1. In the **Project name** enter the name of the project.
   
   2.2. In the **Author** write the author of the project if you want.
   
   2.3. When you create a new project the **Start screen** combobox is empty. You can choose the start screen after creating screens of the project.
   
   2.4. Choose **Default font**. **System** font let you use Chinese, Arabian and etc language symbols.
   
   2.5. In the **Update interval(ms)** enter update interval of the project. It’s an interval of updating objects of the current screen.
   
   2.6. Enter default dimensions of your design screen in the **Screen dimensions** fields.
   
   2.7. If the screen dimensions of you target device differs check **Runtime differs** and enter its **Screen dimensions**.
   
   2.8. Optionally, specify a meaningful **Description** yet.
   
   2.9. If you want to protect your project from opening by non-authorised person check **Use project protection**.
   
   2.10. Enter **Password** for protecting your project.
2.11. Check **Save tags values** if you want to save tag’s value when you close application and load them when you open project.

2.12. Enter **Save DB name** where tag’s values will be saved.

3. On the *Events/History* tab:

3.1. Select the time period during which data will be stored in databases in the **Storage DB period** combobox.

3.2. Enter databases names in the **Events DB name** and **History DB name**. If you choose the simple names like *events* or *history* application will create SQLite database in the application directory. If you choose names beginning with *jdbc:mysql:* like *jdbc:mysql://192.168.0.104:3306/test* the application will connect to MySQL database and create events or history table. *Don’t create big MySQL databases for connecting from Android devices (MySQL databases need a wide network bandwidth for sending and receiving data).*

3.3. Enter **Username** and **Password** if you use MySQL database.

3.4. Enter **Notifications(Priority<)**. Events with a priority lower than this will be notified about it by using the pop-up window and sound.

3.5. Click Sounds **Collection** to set up sounds of events notifications depending on priority. After clicking you’ll see the window. Where:

3.5.1. **Name** of sound.

3.5.2. **From priority** and **To priority** - priority interval within which sound will play.

3.5.3. Button to **Play** sound and button to **Open** sound (wav) file.

3.5.4. Buttons to **Add**, **Edit** and **Remove** sounds.

3.6. If you check **Show servers events** you’ll get information about disconnection, lost or restore servers.

3.7. Choose **Report folder** where reports and screenshots will be saved by default.

3.8. If you want to use Telegram bot in your project check **Use Telegram Bot**.

For more information about using telegram bot in your project see the chapter below.

3.9. Enter **Bot’s name**. You get Telegram Bot’s name from BotFather when you creating your bot.

3.10. Enter **Bot’s token**. You get Telegram Bot’s token from BotFather when you creating your bot.

3.11. Check **Use E-mail client** if you want to use E-mail notifications about Alarms. All event messages that have priority < **Notifications(Priority<)** will be sent by E-mail.

3.12. Enter E-mail **Host**.

3.13. Enter E-mail **Port**.

3.14. Choose **Type** of the connection - TLS or SSL.

3.15. Enter **From** which **E-mail address** the mail will be sent.

3.16. Check **Authentication** if you use Username and Password.

3.17. Enter **Username** of the E-mail.
3.18. Enter **Password** of the E-mail.

3.19. Enter **To** which **E-mail addresses** the mail will be sent. Use commas to separate addresses.

4. If you 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. The certificate stored in the {app}/private directory.

If you want to enable OPC UA server of TeslaSCADA2 check **Use OPC UA server**.

4.3. Enter **TCP port** of your OPC UA server.

4.4. Check **Use Anonymous policy** if you want to use this policy in OPC UA server.

4.5. Check **Use Username/Password policy** if you want to use this policy in OPC UA server.

4.6. Enter **Certificate name** of your OPC UA server.

4.7. Enter validation **Period (days)** of the certificate.

5. If you want to use **MQTT Publisher** check **Enable MQTT Publisher**.

5.1. Enter **Broker URL** of the MQTT server.

5.2. Enter **Username** and **Password** of the MQTT server.

5.3. Choose **QoS** of MQTT messages.

5.4. Check **Enable TLS/SSL** if you want to use server certificate for encryption messages.

5.5. Enter **Certificate filename**. File should be placed in /private/ folder in the directory where TeslaSCADA2 Runtime execution file.

5.6. Check **Enable Client Certificate** if you want to use client certificate for encryption messages.

5.7. Enter **Client certificate** filename. File also should be placed in /private/ folder.

5.8. Enter **Client Private key** filename. File also should be placed in /private/ folder.

5.9. Enter **Private key password**.

5.10. Check **PEM formatted** if your certificate and key files are PEM formatted.

Publisher’s topics consists of the «name of the project +/Tags/+tagname» for tags and «name of the project+/Events/+tagname» for events.

**Save project**

To save project:

1. Click on the **Save** icon in the toolbar or select the menu item **File** and **Save**. The first time you save a new project, you will be asked for a location.

2. Now select the location and click the button **Save** (TeslaSCADA project extension .tsp2).

**Open project**

To open project:

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

**Edit project properties**

To edit project properties:

1. Click on the **Properties** icon in the toolbar or select the menu item **Project and Properties**.
Screens

Create screen
To create a new screen select the menu item Project and New Screen or choose Screens on the Project Window, click right button on it and choose New Screen item. You’ll see the following window:
1. In the Name enter the name of the screen.
2. Optionally, specify a meaningful Comment.
3. Choose Background color.
4. Select Screen type: General or Popup.
5. Add Collection of Scripts for this screen if you want.
6. Enter Screen dimension.
7. Check Use password if you want to use screen security.
8. Enter Password of the screen.
9. If you choose Popup screen you can enter Coordinates where this screen will appear. If you enter value < 0 the screen will appear at the center.

Open screen
To open screen:
   1. Right click on the screen you want to open and choose Open item.
   or
   2. Double click on the screen you want to open.

Copy screen
To copy screen:
   1. Right click on the screen you want to copy and choose Copy item.

Delete screen
To delete screen:
   1. Right click on the screen you want to delete and choose Delete item.

Edit screen properties
To edit screen properties:
   1. Right click on the screen you want to edit and choose Screen properties item.

New Screen Group
Create new group of the screens.

Add to group
Add this screen to one of the available group.

Export screen
To export screen:
   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 (TeslaSCADA screen extension .tsp2screen).
**Import screen**

To import screen:

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).
Servers

Create server
To create a new server select the menu item Project and New Server or choose Servers on the Project Window, click right button on it and choose New Server item. Choose server you want to add to your project.

Modbus server
To create a new Modbus server select the menu item Modbus. You’ll see the following window:
1. In the Name enter the name of the Modbus server.
2. Write IP address or DNS in the IP or DNS field (ipaddress).
3. Enter Modbus server port in the Port (port).
4. Define the polling interval of the server in the Poll interval field (interval).
5. Choose communication protocol in the Type (type).
6. Choose Request type (requesttype):
   - 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.
7. Check RTU via TCP(UDP) if you user Modbus converter from serial into TCP(UDP) protocol (rtuviatcp).
8. Check Without function 6 if your controller doesn’t support Modbus writing function 6 (withoutfun).

Siemens server
To create a new Siemens server select the menu item Siemens. You’ll see the following window:
1. In the Name enter the name of the Siemens server.
2. Write IP address or DNS in the IP or DNS field (ipaddress).
3. Enter Siemens server port in the Port (port).
4. Define the polling interval of the server in the Poll interval field (interval).
5. Choose type of the Siemens PLC in the Controller type (plctype).
6. Choose Request type (requesttype):
   - Maximum registers - if you choose this type the
application during polling will send maximum siemens 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.
7. Enter rack number in the Rack field (rack).
8. Enter slot number in the Slot field (slot).

Allen Bradley server
To create a new Allen Bradley server select the menu item Allen Bradley. You’ll see the following window:
1. In the Name enter the name of the Allen Bradley server.
2. Write IP address or DNS in the IP or DNS field (ipaddress).
3. Enter Allen Bradley server port in the Port field (port).
4. Define the polling interval of the server in the Poll interval field (interval).
5. Choose type of the Allen Bradley PLC in the Controller type (plctype).
6. Enter PLC’s cpu slot number in the CPU slot field (cpuslot).
7. Enter PLC’s backplane number in the Backplane field (ethernetslot).

OPC UA server
To create a new OPC UA server select the menu item OPC UA. You’ll see the following window:
1. In the Name enter the name of the OPC UA server.
2. Write OPC UA server address in the URI field (uri).
3. Define the polling interval of the server in the Poll interval field (interval).
4. Choose security mode in the Mode (mode).
5. Choose security policy in the Policy (policy).
6. Check Anonymous if you don’t use User token (anonymous).
7. Enter Username and Password into relevant fields if you use User token (username and password).

MQTT server
To create a new MQTT server select the menu item MQTT. You’ll see the following window:
1. In the Name enter the name of the MQTT server.
2. Write MQTT server address in the URI field (uri).
3. Enter Username and Password into relevant fields (username and password).
4. Check **Enable TLS/SSL** if you want to use server certificate for encryption messages (*enablessl*).

5. Enter **Certificate filename**. File should be placed in `/private/` folder in the directory where TeslaSCADA2 Runtime execution file (*sslfilename*).

6. Check **Enable Client Certificate** if you want to use client certificate for encryption messages (*enableclientcert*).

7. Enter **Client certificate** filename. File also should be placed in `/private/` folder (*clientcertname*).

8. Enter **Client Private key** filename. File also should be placed in `/private/` folder (*clientprivatekey*).

9. Enter **Private key password** (*privatekeypassword*).

10. Check **PEM formatted** if your certificate and key files are PEM formatted (*pem*).

* 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 uncheck. All .p12 files are PEM formatted.

**Omron server**

To create a new Omron server select the menu item **Omron**. You’ll see the following window:

1. In the **Name** enter the name of the Omron server.
2. Write IP address or DNS in the **IP or DNS** field (*ipaddress*).
3. Enter Omron server port in the **Port** (*port*).
4. Define the polling interval of the server in the **Poll interval** (*interval*).
5. Choose communication protocol in the **Type** (*type*).
6. Enter **Network address (DNA)** (*dna*).
7. Enter **Node address (DA1)**. For TCP protocol it will be chosen automatically during communication (*da1*).
8. Enter **Unit number (DA2)** (*da2*).
Open server properties
To open server properties:
   1. Double click on the server properties which you want to open.
   or
   2. Right click on the server properties which you want to open and choose Server properties item.

Copy server
To copy server:
   1. Right click on the server you want to copy and choose Copy server item.

Delete server
To delete server:
   1. Right click on the server you want to delete and choose Delete server item.
Scripts

Create script
To create a new script select the menu item Project and New Script or choose Scripts on the Project Window, click right button on it and choose New Script item.

You’ll see the following window:
1. In the Name enter the name of the screen.
2. Optionally, specify a meaningful Comment.
3. Choose Background color.
4. Select Script type: General or Screen. General script bind to the whole project. Screen script bind to the Screen.
5. Choose Language you use in this script.
6. Enter Dimension of the script’s design screen.

Open script
To open script:
   1. Right click on the script you want to open and choose Open script item.
   or
   2. Double click on the script you want to open.

Copy script
To copy script:
   1. Right click on the script you want to copy and choose Copy script item.

Delete script
To delete script:
   1. Right click on the script you want to delete and choose Delete script item.

Edit script properties
To edit script properties:
   1. Right click on the script you want to edit and choose Script properties item.

New script group
Create new group of the scripts.

Add to group
Add this script to one of the available group.

Export script
To export script:
   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 (TeslaSCADA script extension .tsp2script).

Import script
To import script:
   1. Right click on the script window and choose Import script item.
   2. Now select the script file and click Open (TeslaSCADA screen extension .tsp2script).
**Tags**

**Create tag**

To create a new tag select the menu item *Project* and *New Tag* or choose *Tags* on the *Project Window*, click right button on it and choose *New Tag* item.

You’ll see the following window:

**On the General tab:**
1. Choose **Group** of the tag.
2. In the **Name** enter the name of the tag. The name should be unique for the project.
3. Choose **Data type**.
4. If you select *String* or *Array* data types enter **Number of elements** (letters).
5. If you select *String* or *Array* data types choose data type of **1 element** (letter).
6. Choose **Access mode** to the tag: *Read*, *Write* or *ReadWrite*.
7. Enter default tag’s value into **Initial PV**.
8. 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**.
9. 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**.

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**

You’ll see the following window:
1. Enter **SlaveID** of the modbus device.
2. Choose **Point type** of the register.
3. Write offset of the register into **Offset**.
4. Choose **Data type** of the modbus tag.
5. Choose number of **Bit** if the point type is boolean.
Siemens tag settings
You’ll see the following window:
1. Choose **Storage area** of the siemens tag: \( I,Q,M \) or \( DB \).
2. Write DB number in the **DB№** field if you choose DB storage area.
3. Choose **Data type** of the siemens tag.
4. Enter byte number of the area into **Byte№** field.
5. Choose number of **Bit** if the data type is **Bit**.

AllenBradley tag settings
You’ll see the following window:
1. Enter **Tag name**.
2. Choose **Data type** of the allen bradley tag.

Micrologix tag settings
If you choose Micrologix or SLC500 controller type in the Allen Bradley server settings you’ll see the following window:
1. Choose **File type** of the server’s tag.
2. Write **File number** in the field.
3. Enter **Element** of the servers tag.
4. Choose **Word** for some file types.
5. Choose number of **Bit**.

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.
**MQTT tag settings**

You’ll see the following window:

1. Enter **Topic**.
2. Choose **QoS** of the MQTT tag.
3. Check **Retained** if you want to use this property.
4. 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 field. If response is not JSON format left field empty. If response contains multi dimension JSON format, separate keys by commas without blank spaces.

**Omron tag settings**

You’ll see the following window:

1. Choose address **Area**.
2. Enter **Address** of the tag.
3. Choose **Data type**.
4. Choose **Bit** for Binary data type.

**On the Scaling tab of the Tag properties window:**

1. Check **Enable I/O scaling** if you want to scale a value get from the server.
2. Enter minimum server tag’s value into **Raw value minimum** field.
3. Enter maximum server tag’s value into **Raw value maximum** field.
4. Enter minimum tag’s value in engineer units into **EU value minimum** field.
5. Enter maximum tag’s value in engineer units into **EU value maximum** field.
6. Write tag’s value offset int **EU value offset**.

When you get some value from the server application use this formula:

\[
value = \frac{(value_{-rawmin}) \times (eu_{-max} - eu_{-min})}{(raw_{-max} - raw_{-min})} + eu_{-min} + offset
\]
On the **Alarms** tab of the **Tag properties** window:

1. Check **Enable alarms** if you want to use alarms for this tag.

2. Check **HiHi**, **Hi**, **Lo**, **LoLo** or **Normal** if you want to use the correspondent alarm(event).

3. Write **Limit** 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 Event database. 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 Event database.

4. Enter **Priority** for the correspondent alarm(event). If the priority of the alarm(event) is less than value of **Notifications(Priority<)** you set in the project properties the notification dialog will be called.

5. Enter **Message** for the correspondent alarm(event).

6. Check **Enable OPC UA event** if you bind this tag to the OPC UA server tag(node) and you want to use EventNotifier of this tag(node).

**On the History tab of the Tag properties window:**

1. Check **Enable history** if you want to storage values of this tag.

2. Enter **Storage period(ms)**.

3. Check **Store in DB** if you want to store data in history database.

**On the Script tab of the Tag properties window:**

1. Check **Enable script** if you want to use script bind to this tag’s value.

2. Choose **Script** you want to bind to this tag’s value.

3. Enter **Value** you want to compare with current value.

4. Choose **Type** of the compare operation.

5. Enter **Deadband** for the value.

**Copy tag**

To copy tag:

1. Right click on the tag you want to copy and choose **Copy tag** item.

**Delete tag**

To delete tag:

1. Right click on the tag you want to delete and choose **Delete tag** item.
**Edit tag properties**
To edit tag properties:
1. Right click on the script you want to edit and choose *Tag properties* item.
   or
2. Double click on the tag you want to edit.

**New group tags**
Create new group of the tags.

**Add to group**
Add this tag to one of the available group.

**Export all tags**
To export all tags:
1. Right click on the tags window and choose *Export all tags* item.
2. Now select the location and click the button *Save* (TeslaSCADA tags extension .tsp2tags).

**Import tags**
To import tags:
1. Right click on the tags window and choose *Import tags* item.
2. Now select the tags file and click *Open* (TeslaSCADA screen extension .tsp2tags).
**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* and *New User* or choose *Users* on the *Project Window*, click right button on it and choose *New User* item.

You’ll see the following window:

1. In the **Name** enter the name of the user.
2. Write **Password** for the current user (*password*).
3. Check **Control functions** if you want that current user can write values into the server’s tags (*controlfunctions*).
4. Check **Acknowledge events** if you want that current user can acknowledge events in events database (*acknowledgeevents*).
5. Check **Delete events** if you want that current user can delete events from events database (*deletedevents*).
6. Check **Insert events** if you want that runtime application insert events into events database when current user is logged in (*insertevents*).
7. Check **Insert history** if you want that runtime application insert history data into history database when current user is logged in (*inserthistory*).
8. Check **Settings** if you want current user can enter *Settings* menu of runtime application (*settings*).
9. Check **Edit recipes** if you want current user can Add, Edit and delete recipes fields (*editrecipes*).
10. Check **Save control operation** if you want to save user control operations in Events database. (it will be saved if you check Enable alarms in Tag properties) (*savecontroloperations*).
11. **Priority** of the user control operations events (*priority*).

**Open user properties**

To open user properties:

1. Right click on the user you want to open and choose *User properties* item.

   or

2. Double click on the user properties which you want to open.

**Copy user**

To copy user:

1. Right click on the user you want to copy and choose *Copy user* item.

**Delete user**

To delete user:
1. Right click on the user you want to delete and choose *Delete user* item.
**Databases**

Database is not a mandatory element of the project. You can use or not use it. Database consists of 3 types: Recipe, History and Odoo ERP.

**Create recipe**

To create a new recipe select the menu item `Project` and `New Database` -> `Recipe` or choose **Databases** on the **Project Window**, click right button on it and choose `New Database` -> `Recipe` item.

You’ll see the following window of recipe properties:

1. In the **Name** enter the name of the recipe.
2. Write **DB name** for the current recipe. If you enter the simple name like `recipes` for example you will connect to the SQLite database. The SQLite database file `.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 (for iOS you can’t use MySQL databases at this time).
3. Write **Table name** of the current database.
4. Write **Username** if needed for MySQL databases.
5. Write **Password** if needed for MySQL database.
6. Click **Collection** to fill up ingredients of the recipe.

After clicking **Collection** button you’ll see the following window. Where:

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.

**Create history**

To create a new history select the menu item `Project` and `New Database` -> `History` or choose **Databases** on the **Project Window**, click right button on it and choose `New Database` -> `History` item.

You’ll see the following window of recipe properties:
1. In the Name enter the name of the history db.
2. Write DB name for the current history. If you enter the simple name like history for example you will connect to the SQLLite database. The SQLLite database file .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 (for iOS you can’t use MySQL databases at this time).
3. Write Table name of the current database.
4. Write Username if needed for MySQL databases.
5. Write Password if needed for MySQL database.
6. Choose Storage type - Time or Tag. If you choose Time, enter Save Period. If you choose Tag, choose tag.
7. Enter Save Period(ms). The retention period of the selected tags in the database.
8. Choose Tag dependent of which value (transition from false to true) collection values will be saved in history database.
9. Choose Storage DB period. It’s a period of storing history information.
10. Click Collection to fill up ingredients of the recipe.

After clicking Collection button you’ll see the following window. Where:
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.

Create Odoo ERP
To create a new Odoo ERP connection select the menu item Project and New Database -> Odoo ERP or choose Databases on the Project Window, click right button on it and choose New Database->Odoo ERP item.
You’ll see the following window of Odoo ERP properties:
1. In the Name enter the name of the Odoo ERP connection.
2. Enter URL of the Odoo ERP.
3. Enter Port of the Odoo ERP.
4. Enter DB name for the current Odoo ERP database.
5. Enter Username for connecting to Odoo ERP.
6. Enter Password for connecting to Odoo ERP.
7. Choose Refresh type to renew data information.
8. Click Collection to fill up model names of the Odoo ERP.

After clicking Collection button you’ll see the following window. Where:
1. Enter Name of the model.
Open database properties
To open database properties:
1. Right click on the recipe you want to open and choose *Database properties* item.
   or
2. Double click on the database properties which you want to open.

Copy database
To copy database:
1. Right click on the database you want to copy and choose *Copy database* item.

Delete database
To delete database:
1. Right click on the database you want to delete and choose *Delete database* item.
Design screen

To start designing the screen you want, you should double click on it or click right button on the Project window->Screens and choose Open screen.

Create graphical object

Add new graphical object on the screen you can in several ways:

1. Select the menu item Project and New Object.
2. Click New Object button on the Toolbar.
3. Click right button on the Screen window and choose New object item.
4. Click right button on the Canvas and choose New object 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:

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

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 Drag and Drop technology.

Open graphical object properties

You can open graphical object properties on the Screen Window or on the Canvas. To open graphical object properties:

1. Right click on the object you want to open and choose Object properties item.

or
2. Double click on the object properties which you want to open.

**Copy graphical object**
You can copy graphical object:
1. Right click on the object you want to copy and choose *Copy* item.
2. Select the object you want to copy and choose *Edit->Copy* menu item.
3. Select the object you want to copy and click *Copy* button on the **Toolbar**.

**Cut graphical object**
You can cut graphical object:
1. Right click on the object you want to cut and choose *Cut* item.
2. Select the object you want to cut and choose *Edit->Cut* menu item.
3. Select the object you want to cut and click *Cut* button on the **Toolbar**.

**Paste graphical object**
You can paste (before cut or copied) graphical object:
1. Right click on the **Canvas** and choose *Paste* item.
2. Choose *Edit->Paste* menu item.
3. Click *Paste* button on the **Toolbar**.

**Erase graphical object**
You can erase graphical object:
1. Right click on the object you want to erase and choose *Erase* item.
2. Select the object you want to erase and choose *Edit->Erase* menu item.
3. Right click on the object in the **Screen Window** and choose *Delete object* item.

**Duplicate graphical object**
You can duplicate graphical object:
1. Right click on the object you want to duplicate and choose *Duplicate* item.
2. Select the object you want to duplicate and choose *Edit->Duplicate* menu item.

**Send to back graphical object**
You can send to back graphical object relative to other objects of the screen:
1. Right click on the object you want to send to back and choose *Send to Back* item.
2. Select the object you want to send to back and choose *Arrange->Send to Back* menu item.
3. Select the object you want to send to back and click *Send to Back* button on the **Toolbar**.

**Bring to front graphical object**
You can bring to front graphical object relative to other objects of the screen:
1. Right click on the object you want to bring to front and choose *Bring to Front* item.
2. Select the object you want to bring to front and choose *Arrange->Bring to Front* menu item.
3. Select the object you want to bring to front and click *Bring to Front* button on the **Toolbar**.

**Rotate clockwise graphical object**
You can rotate clockwise graphical object clockwise:
1. Select the object you want to rotate clockwise and click **Rotate Clockwise** button on the **Toolbar**.
2. Select the object you want to rotate clockwise and choose **Arrange->Rotate Clockwise** menu item.

**Rotate counterclockwise graphical object**
You can rotate counterclockwise graphical object clockwise:
1. Select the object you want to rotate counterclockwise and click **Rotate CounterClockwise** button on the **Toolbar**.
2. Select the object you want to rotate counterclockwise and choose **Arrange->Rotate CounterClockwise** menu item.

**Align graphical objects**
You can align objects relative to each other on the screen. Choose objects you want to align by selecting square. And:
1. Choose **Arrange->Align** menu items.
2. Click **Align** buttons on the **Toolbar**.
3. Right click on selecting square and choose **Align** item.
For more information about each alignment operation you can read above in section **Start TeslaSCADA IDE ->Toolbar**.

**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. And:
1. Select **Arrange->Group objects** menu item.
2. Click **Group objects** button on the **Toolbar**.
3. Right click on selecting square and choose **Group objects** item.

**Ungroup graphical objects**
You can ungroup objects. Choose group of objects you want to ungroup by clicking on it. And:
1. Select **Arrange->Ungroup objects** menu item.
2. Click **Ungroup objects** button on the **Toolbar**.
3. Right click on selecting square and choose **Ungroup objects** item.
Graphical objects

Every graphical object has several group of properties. All properties you can edit in Property sheet 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. In this chapter we describe one group for every object - General.

This group is responsible for the appearance of the object. Each object has the following properties:
1. Name - write name of the object in this field.
2. Dimensions - dimensions of the graphical object. Enter width of the object in the W(width-field name you can use in ST scripts) field and enter height of the object in the H (height) field.
3. Coordinates - 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.
4. Angle(angle) - select the angle of rotation of the object.
5. Almost all objects has Type property to select the type of the object - 2D or 3D.

Simple Objects library

Simple objects library contains the following objects: Line, Rectangle, Ellipse, Polyline, Polygon, Sector, Text, Border, Image and Scale.

Line
1. Write width of the line in the Line width field (linewidth).
2. Choose Color of the line (color).
3. Choose Line style: Solid, Dash, Dot or DashDot (linestyle).
4. Choose Begin marker: Flat, Arrow, Square or Circle (beginmarker).
5. Choose Begin marker: Flat, Arrow, Square or Circle (endmarker).

Rectangle
1. Write width of the line in the Line width field (linewidth).
2. Choose Color of the border line (color).
3. Select fill or not this rectangle in the Fill combobox (fill).
4. Choose Fill color of the rectangle (fillcolor).
**Ellipse**
1. Write width of the line in the **Line width** field (**linewidth**).
2. Choose **Color** of the border line (**color**).
3. Select fill or not this ellipse in the **Fill** combobox (**fill**).
4. Choose **Fill color** of the ellipse (**fillcolor**).

**Polyline**
1. Write width of the line in the **Line width** field (**linewidth**).
2. Choose **Color** of the line (**color**).
3. When you click **Collection** button the Collection window will appear. You can **Add**, **Edit** or **Remove** nodes of the polyline.

**Polygon**
1. Write width of the line in the **Line width** field (**linewidth**).
2. Choose **Color** of the border line (**color**).
3. Select fill or not this polygon in the **Fill** combobox (**fill**).
4. Choose **Fill color** of the polygon (**fillcolor**).
5. When you click **Collection** button the Collection window will appear. You can **Add**, **Edit** or **Remove** nodes of the polygon.

**Sector**
1. Write width of the line in the **Line width** field (**linewidth**).
2. Choose **Color** of the border line (**color**).
3. Select fill or not this sector in the **Fill** combobox (**fill**).
4. Choose **Fill color** of the sector (**fillcolor**).
5. Enter **Start angle** of the sector in the field. 0 degrees is right middle point of the dimensions rectangle (**startangle**).
6. Write **Rotation angle** in the field. Counterclockwise rotation (**rotationangle**).
Text
1. Write text in the Text field (text).
2. Choose Font type of the text (fonttype).
3. Check Underline if you want to use underline font (underline).
4. Enter Font size in the field (fontsize).
5. Select Text placement: Left, Center or Right (textplacement).
7. Select use or not Border around text (useborder).
8. Write width of the border in the Border width field (linewidth).
9. Choose Border color (bordercolor).
10. Select fill or not text background in the Fill combobox (fill).
11. Choose Fill color of the text background (fillcolor).

Border
1. Write width of the border in the Line width field (linewidth).
2. Select Inner or not border style (inner).
3. Select fill or not this border in the Fill combobox (fill).
4. Choose Fill color of the border (fillcolor).

Image
1. Select Image you want to add to the project by clicking «...» button. Open file dialog will appear. Choose file with image you want to add to the project and click Open button.

Scale
1. Write width of the line in the Line width field (linewidth).
2. Choose Color of the border and scale lines (color).
3. Select use or not Border for this scale (useborder).
4. Select use or not Scale №2 for this scale (scale2).
5. Select use or not Scale №3 for this scale (scale3).
6. Enter Scale interval №1 in the field (scaleinterval1).
7. Enter Scale interval №2 in the field (scaleinterval2).
8. Enter Scale interval №3 in the field (scaleinterval3).
9. Write width of the scale №1 in the Marker №1 size field (sizemarkers1).
10. Write width of the scale №2 in the Marker №2 size field (sizemarkers2).
11. Write width of the scale №3 in the **Marker №3 size** field (sizemarkers3).

12. Choose **Type** of the scale: Left, Right, Top or Bottom (type).

13. Check **Use digital** if you want to bind numeration to Scale №1 (usedigit).

14. Enter **Minimum** value of Scale №1 (min).

15. Enter **Maximum** value of Scale №1 (max).

16. Enter **Decimal position** of scale’s numbers (decimalpos).
3D Objects library
3D objects library contains the following objects: Sphere, Cylinder, Cone, Sector 3D, Polygon 3D, Tank, Border 3D, Text 3D, Value with History and Event.

**Sphere**
1. Choose Fill color of the sphere (fillcolor).

**Cylinder**
1. Write width of the line in the Line width field (linewidth).
2. Choose Color of the border line (color).
3. Select fill or not this cylinder in the Fill combobox (fill).
4. Choose Fill color of the cylinder (fillcolor).

**Cone**
1. Write width of the line in the Line width field (linewidth).
2. Choose Color of the border line (color).
3. Select fill or not this cone in the Fill combobox (fill).
4. Choose Fill color of the cone (fillcolor).
5. Select Aspect ratio of the cone (aspectratio).
Sector 3D
1. Write width of the line in the **Line width** field (linewidth).
2. Choose **Color** of the border line (color).
3. Select fill or not this sector in the **Fill** combobox (color).
4. Choose **Fill color** of the sector (fillcolor).
5. Enter **Start angle** of the sector in the field. 0 degrees is right middle point of the dimensions rectangle (startangle).
6. Write **Rotation angle** in the field. Counterclockwise rotation (Rotationangle).

Polygon 3D
1. Write width of the line in the **Line width** field (linewidth).
2. Choose **Color** of the border line (color).
3. Select fill or not this polygon in the **Fill** combobox (fill).
4. Choose **Fill color** of the polygon (fillcolor).
5. When you click **Collection** button the Collection window will appear. You can Add, Edit or Remove nodes of the polygon.

Tank
1. Choose **Fill color** of the tank (fillcolor).
2. Select **Ratio** of the tank (ratio).
3. Select vertical or horizontal will be tank in **Vertical** combobox (vertical).

Border 3D
1. Write width of the border in the **Line width** field (linewidth).
2. Choose **Color** of the border (color).
3. Select fill or not this border in the **Fill** combobox (fill).
4. Choose **Fill color** of the border (fillcolor).
5. Enter radius of the border’s corner in the **Corner radius** field (cornerradius).
6. Select use or not **Glass** effect (glass).
**Text 3D**

1. Write text in the **Text** field (text).
2. Choose **Font type** of the text (fonttype).
3. Check **Underline** if you want to use underline font (underline).
4. Enter **Font size** in the field (fontsize).
5. Select **Text placement**: Left, Center or Right (textplacement).
7. Select use or not **Border** around text (useborder).
8. Write width of the border in the **Border width** field (linewidth).
9. Choose **Border color** (bordercolor).
10. Select fill or not text background in the **Fill** combobox (fill).
11. Choose **Fill color** of the text background (fillcolor).

**Value with History and Event**

1. Write text in the **Text** field (text).
2. Choose **Font type** of the text (fonttype).
3. Enter **Font size** in the field (fontsize).
4. Select **Text placement**: Left, Center or Right (textplacement).
5. Choose **Text color** (textcolor).
6. Select use or not **Border** around text (useborder).
7. Write width of the border in the **Border width** field (linewidth).
8. Choose **Border color** (bordercolor).
9. Select fill or not text background in the **Fill** combobox (fill).
10. Choose **Fill color** of the text background (fillcolor).
11. Choose 2D or 3D **Type** (type3d).
12. Write number of trend’s horizontal grid lines in **Horizontally** field (horizontally).
13. Write number of trend’s vertical grid lines in **Vertically** field (vertically).
14. Enter **Time format** of the trend (timeformat).
15. On the **Grid** tab choose Right or Left **Side** of placement of the trend and event table (side).
16. Enter **Line width** of the curve (linewidth).
17. Choose **Curve color** (gridlinecolor).
18. Choose **Line style** (linestyle).
19. Enter **Grid width** of the trend and event table (gridwidth).
20. Enter **Grid height** of the trend and event table (gridheight).
21. Enter **Font size** of the marks of the trend (fontsize).
22. Choose **Mark color** (markcolor).
**Buttons and Switches library**

Buttons and Switches library contains the following objects: Button, Press button, Toggle button, Light button, Oval light button, Image button, Oval jump, Rectangle jump, Right Arrow, Left Arrow, Switch, Switch 3 Pos and Apple switch. All buttons except Image button have the same General group properties. Below we describe there only for 5 graphical objects - **Button**, **Image button**, **Switch**, **Switch 3 Pos** and **Apple switch**.

---

**Button**

1. Write text of the button in the Text field (**text**).
2. Choose Text color (**textcolor**).
3. Choose Fill color of the button (**fillcolor**).
4. Choose Font type of the button’s label (**fonttype**).
5. Check Underline if you want to use underline font (**underline**).
6. Select Text placement: Left, Center or Right (**textplacement**).
7. Check Animation if you want to use animation for button (**animation**).

**Image Button**

1. Choose Fill color of the button’s background (**fillcolor**).
Switch and Apple switch
1. Write label of the switch in the Text field (text).
2. Choose Text color (textcolor).
3. Choose Fill color of the switch (fillcolor).
4. Write label for ON position of the switch in Text ON (texton).
5. Write label for OFF position of the switch in Text OFF (textoff).
6. For Apple switch you can choose Function type Toggle or Push (clicktype).

Switch 3 Pos.
1. Write label of the switch in the Text field (text).
2. Choose Text color (textcolor).
3. Choose Fill color of the button (fillcolor).
4. Write label for ON position of the switch in Text ON (texton).
5. Write label for OFF position of the switch in Text OFF (textoff).
Lights/Indicators library
Lights/Indicators library contains the following objects: Light, Rectangle light, Triangle light, Triangle light 2, Stack light, Indicator, Rectangle Indicator and Oval Indicator. All lights have the same General group properties and all indicators have the same General group properties. Below we describe it only for two graphical objects - Light and Indicator.

**Light**
1. Write text of the light in the Text field (text).
2. Choose Text color (textcolor).
3. Choose Fill color of the light (fillcolor).

**Indicator**
1. Choose Color TRUE of the indicator (truecolor).
2. Choose Color FALSE of the indicator (falsecolor).
Pipes library
Pipes library contains the following pipes objects: Straight, End, Bow, Tee, Intersect, Elbow and Pipeline. All pipes have the same General group properties. Below we describe it only for two graphical objects - **Straight and Pipeline**.

Pipe
1. Choose **Fill color** of the pipe (*fillcolor*).

Pipeline
1. Choose **Color** of the pipe (*fillcolor*).
2. Set **Pipe width** of the pipeline (*linewidth*).
3. Choose **Type** of the pipeline 2D or 3D (*type*).
4. When you click **Collection** button the Collection window will appear. You can **Add**, **Edit** or **Remove** nodes of the pipeline.
Valves library
Valves library contains the following objects: Valve, Round valve, Ball valve, Position valve and Valve ISA. Valve, Round valve and Valve ISA have the same General properties:

Valve
1. Choose Fill color for the valve (fillcolor).
**Ball valve**
1. Choose Fill color for the valve (fillcolor).
2. Choose Indicator color (indicatorcolor).

**Indicator** property like indicator property for other graphical object. 
**Indicator color** property like other color properties.

---

**Position valve**
1. Choose Fill color for the valve (fillcolor).
2. Choose Arrow color (indicatorcolor).

**Value** property like value property for analog meters. 
**Arrow color** property like other color properties.
**Pumps library**
Pumps library contains the following objects: Pump, Motor, Stand-up pump, Centrifugal pump and Air blower pump. All pumps have the same General group properties. Below we describe there only for one graphical object - **Pump**.

**Pump**
1. Choose **Fill color** of the pump (*fillcolor*).

**Fans library**
Fans library contains the following objects: Fan, Round fan and Square fan. All fans have the same properties:

**Fan**
1. Choose **Fill color** of the fan (*fillcolor*).

Set up **Rotation ind.** properties to rotate fan.
**Tanks library**

Tanks library contains the following objects: Vertical tank, Horizontal tank, Vertical tank with trend, Horizontal tank with trend and Cone tank. All tanks have the same General group properties. Below we describe it only for one graphical object - **Vertical tank**.

**Vertical tank**

1. Choose background color of the tank in **Color** (*color*).
2. Choose filling color of the tank in **Fill color** (*fillcolor*).
3. Enter text in the **Text** field (*text*).

**Conveyers library**

Conveyers library contains the following objects: Screw conveyer and Belt conveyer. All conveyers have the same General group properties. Below we describe it only for one graphical object - **Belt conveyer**.

**Belt conveyer**

1. Choose **Fill color** of the conveyer (*fillcolor*).
2. Select incline or not in **Incline** combobox (*inclined*).
Analog meters library
Analog meters library contains 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 fill, Analog meter horizontal, Analog meter horizontal fill, Range Indicator, Gauge, Gauge 180 degrees, Gauge 90 degrees and Gauge 90 degrees 2.

Analog meter
1. Write text of the label in the Text field (text).
2. Choose color of the arrow in the Color (color).
3. Choose filling color of the meter in Fill color (fillcolor).

Range Indicator
1. Choose Needle(Fill) color (color).
2. Choose Fill color (fillcolor).
3. Choose Border color (bordercolor).
4. Choose Type of the indicator: Left or Right (type).
Other analog meters
Analog meter rectangle, Analog meter 90 degrees, Analog meter 90 degrees 2, Analog meter 90 round, Analog meter vertical, Analog meter vertical fill, Analog meter horizontal and Analog meter horizontal fill have the same General properties:
1. Write text of the label in the Text field (text).
2. Choose color of the arrow in the Color (color).
3. Choose border color of the meter in Border color (bordercolor).
4. Write unit text in the Unit field (unit).
5. Enter the number of intervals of the meter in the № of intervals field (interval).
6. Check Use digital if you want to use digital meter (usedigital).
**Digital meters library**
Digital meters library contains the following objects: Digital meter, 4 digit meter, 6 digit meter, 8 digit meter and Meter with history and event.

**Digital meters**
All digital meters have the same general properties:
1. Write text of the label in the **Text** field (*text*).
2. Choose **Text color** (*textcolor*).
3. Choose color of the border in the **Border color** (*bordercolor*).
4. Choose filling color of the meter in **Fill color** (*fillcolor*).

For Meter with history and event also you can choose **Side** (*side*) where history trend or event table will appear after clicking on meter.
**Controls library**

Controls library contains the following objects: Slider, Slider vertical, Slider horizontal, Counter, Counter rectangle, Apple slider, Selector, ComboBox, CheckBox, MenuBox, MenuCheckList and CheckList.

**Slider**
1. Choose color of the background in the **Color** \((color)\).
2. Choose filling color of the slider in **Fill color** \((fillcolor)\).

**Slider vertical and horizontal**
1. Choose color of the background in the **Color** \((color)\).
2. Choose filling color of the slider in **Fill color** \((fillcolor)\).
3. Enter label text in the **Text** field \((text)\).
4. Write unit text in the **Unit** field \((unit)\).
5. Enter the number of intervals of the slider in the **№ of intervals** field \((interval)\).
6. Check **Use digital** if you want to use digital meter \((usedigital)\).
Counter and counter rectangle
General properties for counter and counter rectangle are the same:
1. Choose color of the background in the Color (color).
2. Choose Text color (textcolor).

Selector, ComboBox and MenuBox
General properties for the selector:
1. Choose colour of the background in the Color (color).
2. Choose Fill color for the background of the selected button (fillcolor).
3. For MenuBox choose Expanded type of the menu: horizontally or vertically (expandedtype).
4. For MenuBox check Animation possibility of expanding menu items (animation).
CheckBox, MenuCheckList and CheckList

General properties for the CheckBox, Menu Box and CheckList:
1. Choose colour of the text in the Color (color).
2. Choose Fill color for the background of the selected object (fillcolor).
3. Click Collection to fill up CheckBox items.
4. For MenuCheckList choose Expanded type of the menu: horizontally or vertically (expandedtype).
5. For MenuCheckList check Animation possibility of expanding menu items (animation).

After clicking Collection you’ll see window:
1. Choose Tag bind to checkbox item.
2. Enter the Value which will be written after selecting the item of the checkbox.
3. Enter the Uncheck value which will be written after unselecting the item of the checkbox.
4. Enter Text of the checkbox item.

Parameter list

General properties for the Parameter list:
1. Choose colour of the background in the Fill color (fillcolor).
2. Choose Color for the text of the selected object (color).
3. Click Collection to fill up Parameter list items.
4. Check Write simultaneously to enter values simultaneously in tags (simultaneously).

After clicking Collection you’ll see window:
1. Choose Tag bind to parameter list item.
2. Enter Name of the parameter list item.
3. Enter Decimal position of the value.
**Electrical library**
Electrical library contains the following objects: Electrical switch, 2-Way Electrical switch, 2-Way Electrical switch with neutral, Transformer. Below description of the Electrical switch. For the other switches properties are the same.

**Electrical switch**
1. Write width of the line in the **Line width** field (*linewidth*).
2. Choose **Color** of the line (*color*).
3. Choose **Beginmarker** of the switch (*beginmarker*).
4. Choose **Endmarker** of the switch (*endmarker*).

**Transformer**
1. Write width of the line in the **Line width** field (*linewidth*).
2. Choose **Color** of the line (*color*).
**Trends and Charts library**

Trends library contains the following objects: Trend, Real time trend, Trend DB, Trend OPC UA History, Pie Chart and Bar chart. *Trend* and *Real time trend* draw curves based on tags that used history data collection (check **Enable history** in Tags properties). *Trend DB* draws curves based on tags that used data stored in database (check **Store in DB** in Tags properties). *Trend OPC UA History* draw curves based on tags that bind to OPC UA nodes supported Historyzing property. All trends have the same General and Grid group properties. Below we describe their only for one graphical object - *Trend*.

**Trend**

1. Enter width of line in the **Line width** (*linewidth*).
2. Choose background color of the trend in **Color** (*color*).
3. Select fill or not in the **Fill** combobox (*fill*).
4. Choose filling color of the trend in **Fill color** (*fillcolor*).
5. To add curve click **Collection** button.
6. Enter **Default period** (or time interval period) in minutes (*defaultperiod*).

Collection window will appear:

1. Select tag that you want to bind to this curve in the **Tag**.
2. Enter curve’s name in the **Name** field.
3. Write width of curve’s line in **Line width** field.
4. Choose **Color** of the curve.
5. Select **Type** of the curve. *Type 1* just draw the line. *Type 2* draw line with filling till axis X. *Type 3* draw a ladder line. *Type 4* draw a filled ladder line.

On the **Grid** tab:

1. Enter width of grid’s lines in the **Line width** field (*linewidth*).
2. Choose **Color** of grid’s lines (*color*).
3. Select **Line style**: *Solid, Dash, Dot or DashDot* (*linestyle*).
4. Write number of horizontal grid’s lines in **Horizontally** field (*horizontally*).
5. Write number of vertical grid’s lines in **Vertically** field (*vertically*).
6. Enter **Maximum** value of the trend (*maximum*).
7. Enter **Minimum** value of the trend (*minimum*).
8. Write **Font size** of the marks (*fontsize*).
9. Choose **Mark color** (*markcolor*).
10. Enter **Time format** of trend’s time axis (*timeformat*).

Also Trend object has several properties that you can’t setup by using settings dialog box, but you can setup by using ST script:
1. `begin` - start time for trend information. Time represented in minutes from current period. 
   \[\text{start time} = \text{current time} - \text{begin}\].
2. `end` - finish time for trend information. Time represented in minutes from current period. 
   \[\text{finish time} = \text{current time} - \text{end}\].
3. `title` - title for the trend’s report representation.
4. `filename` - name of the report’s file.
5. `number` - report’s frequency of writing values.
6. `savereport` - when this value become **true** trend’s report will be created.

**Pie Chart**

1. Check **Use legends** if you want to add legends to the chart (*uselegends*).
2. Check **Donut** if you want to use ring type chart (*donut*).
3. To add, edit and remove sector(s) click **Collection**.

Collection window will appear:
1. Select tag that you want to bind to this sector in the **Tag**.
2. Enter sector’s name in the **Name** field.
3. Choose **Color** of the sector.

**Bar chart**

1. Check **Use legends** if you want to add legends to the chart.
2. Choose **Fill** if you want to fill background.
3. Choose **Fill color**.
4. To add, edit and remove bar(s) click **Collection**.

On the **Grid** tab:
1. Enter width of grid’s lines in the **Line width** field (*linewidth*).
2. Choose **Color** of grid’s lines (*color*).
3. Select **Line style**: Solid, Dash, Dot or DashDot (*linestyle*).
4. Write number of vertical grid’s lines in **Vertically** field (*vertically*).
5. Enter **Maximum** value of the trend (*maximum*).
6. Enter **Minimum** value of the trend (*minimum*).
7. Write **Font size** of the marks (*fontsize*).
8. Choose **Mark color** (*markcolor*).

Collection window will appear:
1. Select tag that you want to bind to this sector in the **Tag**.
2. Enter sector’s name in the **Name** field.
3. Choose **Color** of the sector.
Events library

Events library contains the following object: Events log. Events log collects tag’s events (check Enable alarms and check events you want to collect in Tags properties).

Events log

1. Enter title of the table in the **Title** field (*title*).
2. Write size of text in the **Font size** (*fontsize*).
3. Choose row’s background color of acknowledged events in the **Acknowledge color** (*ackcolor*).
4. Choose row’s background color of not acknowledged events in the **Unacknowledge color** (*unackcolor*).
5. Choose color of the event’s text by clicking **Collection** button.
6. Enter **Time format** of the time’s text (*timeformat*).

After clicking **Collection** button you’ll see the window:

1. Enter the priority of the event from which be used this color in the **From** field.
2. Enter the priority of the event to which be used this color in the **To** field.
3. Choose **Color** of the event’s text.

On the **Columns** tab:

1. Select columns that you want to use in the table.
2. Write titles of the columns in correspondent **Title** field (*nametitle, timetitle, typetitle and etc*).
3. Enter **Width** of the correspondent column (*namewidth, timewidth, typewidth and etc*).

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:

1. **begin** - start time for log information. Time represented in minutes from current period. (*start\text{\ time} = \text{current time} - \text{begin})*.
2. **end** - finish time for log information. Time represented in minutes from current period. (*finish\text{\ time} = \text{current time} - \text{end})*.
3. **filename** - name of the report’s file.
4. **savereport** - when this value become **true** trend’s report will be created.
Recipes library
Recipes library contains the following objects: Recipe table, Recipe selector, Parameter table and Schedule table.

Recipe table
1. Choose Recipe you want to bind to the table. During running you can add, edit and delete fields of the recipe database by clicking right button on the table and choosing operation (recipename).
2. Enter title of the table in the Title field (title).
3. Write size of text in the Font size (fontsize).
4. Enter Name column width. Set width of the name’s column (namecolumnwidth).
5. Enter Other column width. Set width of other columns (othercolumnwidth).

Recipe selector
1. Choose Recipe you want to bind to the selector. During running you can select fields of the recipe database by clicking on the recipe selector (recipename).
2. Choose Text color of the selector (textcolor).
3. Choose Fill color of the selector (fillcolor).

Parameter table
1. Enter Parameter column name (parametercolumn).
2. Enter Value column name (valuecolumn).
3. Check Use DB value if you want to use DB value column (usedb).
4. Enter DB value column name (dbcolumn).
5. Choose Recipe you want to bind to the table. During running you can add, edit and delete fields of the recipe database by clicking right button on the table and choosing operation (recipename).
6. Enter Row number of the database which be used in DB value column (rownumber).
7. Write size of text in the Font size.
8. Enter Name column width. Set width of the name’s column (namecolumnwidth).
9. Enter Other column width. Set width of other columns (othercolumnwidth).

After clicking Collection button you’ll see the window:
1. Choose Tag you want to use in the table.
2. Enter row Name for this tag.
3. Enter Decimal position of the tag’s value.

Schedule table
1. Choose Default schedule from Recipes you want to bind to the table. During running you can add, edit and delete fields of the recipe database by clicking right button on the table and choosing operation (recipename).
2. Enter Title of the table (title).
3. Write size of text in the Font size (fontsize).
4. Enter Name column width. Set width of the name’s column (namecolumnwidth).
5. Enter Other column width. Set width of other columns (othercolumnwidth).
6. Choose Time interval tag. Depending of this tag’s value will be duration of the next step(row) of schedule table (timertagname).
7. Check Repeat if you want to repeat all schedule steps (rows) (repeat).
History DB library

History DB library contains the following objects: History DB table, History DB trend, History Excel Report and History Max and Min Report.

History DB table
1. Choose History DB you want to bind to the table (historydbname).
2. Enter title of the table in the Title field (title).
3. Write size of text in the Font size (fontsize).
4. Enter Date column width. Set width of the date’s column (datecolumnwidth).
5. Enter Time column width. Set width of the time’s column (timecolumnwidth).
6. Enter Other column width. Set width of other columns (othercolumnwidth).
7. Enter Decimal position of tag’s values entered in the table (decimalpos).
8. Check Auto refresh if you want update table every time when new tag’s value added into database (autorefresh).
9. Choose Time order by of the database rows ASC or DESC (orderby).

Also History DB Table object has several properties that you can’t setup by using settings dialog box, but you can setup by using ST script:

1. begin - start time for table information. Time represented in minutes from current period. (start time = current time - begin).
2. end - finish time for table information. Time represented in minutes from current period. (finish time = current time - end).
3. filename - name of the report’s file.
4. savereport - when this value become true trend’s report will be created.
**History DB trend**

All properties of History DB trend are similar to properties of Trend object. And additional properties:

1. Check **Auto refresh** if you want update table every time when new tag’s value added into database (*autofresh*).
2. Choose **History DB** you want to bind to the trend (*historydbname*).

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:

1. **begin** - start time for trend information. Time represented in minutes from current period. (*start time = current time - begin*).
2. **end** - finish time for trend information. Time represented in minutes from current period. (*finish time = current time - end*).
3. **filename** - name of the report’s file.
4. **savereport** - when this value become **true** trend’s report will be created.

**History Excel report and History Max and Min report**

1. Choose **History DB** you want to bind to the report (*historydbname*).
2. Enter title of the table of the report in the **Title** field (*title*).
3. Enter second title of the table in the **Title 2** field (*title2*).
4. Enter **Decimal position** of tag’s values entered in the table (*decimalpos*).
5. Check **Transparent background** if you want to make background of the button invisible (*transparent*).
6. Choose **Fill colour** of the button (*fillcolor*).
7. Choose **Type** of the button (*type*).

Also reports object has several properties that you can’t setup by using settings dialog box, but you can setup by using ST script:

1. **begin** - start time for report information. Time represented in minutes from current period. (*start time = current time - begin*).
2. **end** - finish time for report information. Time represented in minutes from current period. (*finish time = current time - end*).
3. **filename** - name of the report’s file.
4. **savereport** - when this value become **true** trend’s report will be created.
**Odoo ERP library**
Odoo ERP library contains the following object: Odoo ERP table. *Odoo ERP table* collects rows of Odoo ERP.

**Odoo ERP table**
1. Choose *Odoo ERP* bind to this table (*odooerpname*).
2. Choose *Model* of the Odoo ERP (*modelname*).
3. Enter *Title* of the table (*title*).
4. Enter *Font size* of the table (*fontsize*).
5. Enter *Fields* by clicking *Collection* button.
6. Enter *Filters* by clicking *Collection* button.
7. Enter *Functions* by clicking *Collection* button.
8. Check *Auto refresh* to refresh table automatically (*autorefresh*).

After clicking Fields *Collection* button you’ll see the window:
1. Enter *Name* of the field.
2. Enter *Field* of the Odoo ERP model.
3. Enter *Width* of the field’s column.
4. Check *Use relation* to get data from the relation model.
5. Check *Read only* if you don’t want to let change field.

After clicking Filters *Collection* button you’ll see the window:
1. Enter *Name* of the filter.
2. Enter *Field* of the Odoo ERP model.
3. Choose *Comparison* operation for filter.
4. Enter *Value* for comparison.
5. Check *Use* if you want to use this filter for the table by default.
6. Choose *Color* for rows that fits for this filter conditions.

After clicking Functions *Collection* button you’ll see the window:
1. Enter *Name* of the function.
2. Enter *Function* of the Odoo ERP model.
3. Check *Bind tag* if you want to bind tag to the button.
4. Choose *Tag* of the function.
5. Enter *Value* will be written to the Tag.
6. Check *Use filter* to bind button of the function to the filter (if check the button enable if filter condition is TRUE).
7. Choose *Filter* bind to the function.

When you click on the row of the table you will select the row and you can use it in in the script by using fields: *selectrowfield* and *selectrowvalue*. At the beginning first you should select field of the row and then get or set value of the row.
Databases library
Databases library contains the following object: Database table.

**Database table**
1. Enter **Database** name. If database name contains “jdbc:mysql” it means address of MySQL database (*databasename*).
2. Enter **Username** for MySQL database (*username*).
3. Enter **Password** for MySQL database (*password*).
4. Enter **Table name** (*tablename*).
5. Enter **Title** of the table (*title*).
6. Enter **Font size** (*fontsize*).
7. Enter **Column width** (*columnwidth*).
**Widgets library**

Widgets library contains the following object:
WebView, Video, Analog Clock, Digital Clock, Color picker and Color rectangle.

**Web View**

1. Enter url of the internet resource in the **Url** field (*url*).

**Video**

1. Enter url of the internet resource in the **Url** field (*url*).
2. Choose **Type** of video signal (*type*).
3. Check **Authentication** if your video camera use username and password for login (*security*).
4. Enter **Username** in the field (*username*).
5. Enter **Password** in the field (*password*).
6. Check **Motion detect** for detecting motion by using this camera (*motiondetect*).
7. Enter **Time interval** in ms for comparing 2 frames (*interval*).
8. Choose **Tag** for writing the value of comparing 2 frames in %.
9. Check **Auto save image** if you want to save images from video camera depending on the value of motion detect (*autosaveimage*).
10. Choose **Save condition** (*savecondition*).
11. Enter **Difference** between 2 frames in % during motion detect (*diff*).

Items 3-11 use only PC versions. These features doesn’t work on Android and iOS.
**Analog clock**

1. Enter label of the clock in the **Text** field (*text*).

**Digital clock**

1. Write text of the label in the **Text** field (*text*).
2. Choose **Text color** (*textcolor*).
3. Choose color of the border in the **Border color** (*bordercolor*).
4. Choose filling color of the meter in **Fill color** (*fillcolor*).

**Color Picker and Color rectangle**

1. Choose **Red color** tag (*redcolortagstring*).
2. Choose **Green color** tag (*greencolortagstring*).
3. Choose **Blue color** tag (*bluecolortagstring*).
4. Choose **Opacity** tag (*opacitycolortagstring*).
**User-defined library**

**Create user-defined library**
You can create your own library by clicking right button on Collections section of the Add graphical object window and choosing New library menu item. You can add graphical object in your library by clicking right button on the object on Canvas or Screen window 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-defined library**
To rename library:

1. Right click on the library you want to rename and choose Rename library item.

**Delete user-defined library**
To delete library:

1. Right click on the library you want to delete and choose Delete library item.

**Export user-defined library**
To export library:

1. Right click on the library you want to export and choose Export library item.
2. Now select the location and click the button Save (TeslaSCADA library extension .tsp2lib).

**Import user-defined library**
To import library:

1. Right click on the Collections window and choose Import library item.
2. Now select the library file and click Open (TeslaSCADA library extension .tsp2lib).
Properties
Each graphical object has several group of properties. To use property of the graphical object check **Enable Property**. You can select tag that be bound to the property by using ComboBox (you can choose it by beginning entering name of the tag when ComboBox focused) or use Button (…). Each object has **Flash, Rotation, Motion** and **Visibility** properties. Other properties depend on the object.

Flash
Flash property lets object to flash when conditions is TRUE or FALSE. To edit flash property click **Flash** tab on the object property window.

1. Select the **Tag** value of which will be compared.
2. Enter the comparison **Value**.
3. Select **Type** of comparison.
4. Write period’s time in milliseconds of objects flashing if the comparison is true in the **Duration TRUE(ms)** field. If you enter 0 the object will not flashing.
5. Write period’s time in milliseconds of objects flashing if the comparison is false in the **Duration FALSE(ms)** field. If you enter 0 the object will not flashing.
6. If you select **Tag.PV in the range** in the **Type** combobox and click **Collection** button. You’ll see the window:

   1. Enter the value from which the object will flash with this periodicity in the **From** field.
   2. Enter the value to which the object will flash with this periodicity in the **To** field.
   3. Enter period of flashing in the **Duration(ms)** field.
   
   You can **Add, Edit or Remove** collection element of flashing conditions.
**Rotation**

Rotation property lets to rotate the object proportional to the value of the tag. To edit rotation property click **Rotation** tab on the object property window.

1. Select the **Tag** value of which will be compared.
2. Enter the minimum of rotation angle in the **Rotation angle(min)** field.
3. Enter the maximum of rotation angle in the **Rotation angle(max)** field.
4. Write the minimum of the tag’s value in the **Rotation value(min)**.
5. Write the maximum of the tag’s value in the **Rotation value(max)**.
6. Enter X coordinate of the pivot in **PivotX** field.
7. Enter Y coordinate of the pivot in **PivotY** field.

**Motion**

Motion property lets to move the object depending on value of the tag. To edit motion property click **Motion** tab on the object property window.

1. Select the **Tag** depending on value of which the object will change location coordinates.
2. Click Collection button to edit move conditions coordinates.

After clicking you’ll see window:

1. Enter the value from which the object will change coordinates in the **From** field.
2. Enter the value to which the object will change coordinates in the **To** field.
3. Write **TranslationX** coordinates (X offset of the object position on the screen).
4. Write **TranslationY** coordinates (Y offset of the object position on the screen).
Visibility
Visibility property lets to make the object visible or not depending on the tag’s value. To edit visibility property click **Visibility** tab on the object property window.

1. Select the **Tag** value of which will be compared.
2. Enter the comparison **Value** (*visibilityvalue*).
3. Select **Type** of comparison.

Line color
Line color property lets object to change color of its line when conditions is TRUE or FALSE. To edit line color property click **Line color** tab on the object property window.

1. Select the **Tag** value of which will be compared.
2. Enter the comparison **Value**.
3. Select **Type** of comparison.
4. Choose a color that will result if the comparison is true in **Color TRUE**.
5. Choose a color that will result if the comparison is false in **Color FALSE**.
6. If you select *Tag.PV in the range* in the **Type** combobox and click **Collection** button. You’ll see the window:

   1. Enter the value from which the object will change color in the **From** field.
   2. Enter the value to which the object will change color in the **To** field.
   3. Choose **Color**.
   You can Add, Edit or Remove collection element of line color conditions.
**Fill color**

Fill color property lets object to change color of its filling when conditions is TRUE or FALSE. To edit fill color property click **Fill color** tab on the object property window.

1. Select the **Tag** value of which will be compared.
2. Enter the comparison **Value**.
3. Select **Type** of comparison.
4. Choose a color that will result if the comparison is true in **Color TRUE**.
5. Choose a color that will result if the comparison is false in **Color FALSE**.
6. If you select *Tag.PV in the range* in the **Type** combobox and click **Collection** button. You’ll see the window:

   1. Enter the value from which the object will change color in the **From** field.
   2. Enter the value to which the object will change color in the **To** field.
   3. Choose **Color**.

   You can Add, Edit or Remove collection element of fill color conditions.

**Filling**

Filling property lets to control filling of the object depending on value of the tag. To edit filling property click **Filling** tab on the object property window.

1. Select the **Tag** value of which will be used to control filling.
2. Enter minimum value of the object’s filling in the **Minimum** field (*minimum*).
3. Enter maximum value of the object’s filling in the **Maximum** field (*maximum*).
**Text color**

Text color property lets object to change color of text when conditions is TRUE or FALSE. To edit text color property click **Text color** tab on the object property window.

1. Select the **Tag** value of which will be compared.
2. Enter the comparison **Value**.
3. Select **Type** of comparison.
4. Choose a color that will result if the comparison is true in **Color TRUE**.
5. Choose a color that will result if the comparison is false in **Color FALSE**.
6. If you select **Tag.PV in the range** in the **Type** combobox and click **Collection** button. You’ll see the window:

   1. Enter the value from which the object will change color in the **From** field.
   2. Enter the value to which the object will change color in the **To** field.
   3. Choose **Color**.

   You can **Add**, **Edit** or **Remove** collection element of text color conditions.

**Control (for buttons)**

Control property lets to write value to the tag. To edit control property click **Control** tab on the object property window.

1. Select **Tag** which will be recorded value.
2. Select **Function** of writing value: **Set** - will write true to the tag; **Reset** - will write false to the tag; **Toggle** - if current tag’s value true will write false, if currents tag’s value false will write true; **Push** - during pressing button will write true; **Set value** - will write Value to the tag; **Enter value** - will call dialog that lets you enter value; **Call screen** - will call selected screen; **Call popup** - will call selected popup screen, **Close popup** - will close popup screen, **Call external software** - lets call external software by using command and arguments of OS.

3. When you select **Set value Function** write **Value** that will be written to the tag.
4. When you select **Enter value Function** write **Title** of the called dialog that lets you enter value.
5. When you select *Call screen* or *Call popup Function* choose *Screen* that will be called after clicking button.

6. Enter **Command and args** of the OS for calling external software. Examples:
   
   for MacOS: `open /Applications/TextEdit.app`
   for Windows: `C:/Program~1/somesoftware.exe`
   for Android: `ops.tesla.scada (name of the Android application package)`
   for iOS: `http://www.youtube.com/watch?v=VIDEO_IDENTIFIER` (youtube scheme for calling in iOS).
Text input
Text input property lets object to control display tag’s value or text when conditions is TRUE or FALSE. To edit text input property click Text input tab on the object property window.
1. Select the Tag value of which will be compared.
2. Enter the comparison Value.
3. Select Type of comparison or display tag’s value.
4. Enter text that will result if the comparison is true in Text TRUE.
5. Enter text that will result if the comparison is false in Text FALSE.
6. Write the text will be shown before displayed text in the Text before.
7. Write the text will be shown after displayed text in the Text after.
8. Enter Decimal position of displayed text in the field.
9. If you select Tag.PV in the range in the Type combobox and click Collection button. You’ll see the window:
   1. Enter the value from which the object will change text in the From field.
   2. Enter the value to which the object will change text in the To field.
   3. Write displayed Text.
You can Add, Edit or Remove collection element of displayed text conditions.

Output
Output property lets write value to the tag. To edit output property click Output tab on the object property window.
1. Select the Tag where value will be written.
2. Enter Title of the dialog that will be used to write value to the tag.

Indicator
Indicator property lets to indicate object depending on value of the tag. To edit indicator property click Indicator tab on the object property window.
1. Select the Tag value of which will be compared.
2. Enter the comparison Value.
3. Select Type of comparison.
Rotation Indicator
Rotation Indicator property lets to rotate object around its center depending on value of the tag. To edit indicator property click Rotation ind. tab on the object property window.
1. Select the Tag value of which will be compared.
2. Enter the comparison Value.
3. Select Type of comparison.
4. Enter rotation speed in Speed(ms).

Image
Image property lets object to change image when conditions is TRUE or FALSE. To edit image property click Image tab on the object property window.
1. Select the Tag value of which will be compared.
2. Enter the comparison Value.
3. Select Type of comparison.
4. Choose image that will result if the comparison is true by clicking Image TRUE button.
5. Choose image that will result if the comparison is false by clicking Image FALSE button.

Color
Color property lets object to change color of its when conditions is TRUE or FALSE. To edit color property click Color tab on the object property window.

1. Select the Tag value of which will be compared.
2. Enter the comparison Value.
3. Select Type of comparison.
4. Choose a color that will result if the comparison is true in Color TRUE.
5. Choose a color that will result if the comparison is false in Color FALSE.
6. If you select Tag.PV in the range in the Type combobox and click Collection button. You’ll see the window:
   1. Enter the value from which the object will change color in the From field.
   2. Enter the value to which the object will change color in the To field.
   3. Choose Color.
You can Add, Edit or Remove collection element of color conditions.
**Control (slider)**
Control property lets object to write value to the tag. To edit control property click **Control** tab on the object property window.

1. Select the **Tag** value of which will be changed.
2. Enter **Minimum** value of the control *(minimum)*.
3. Enter **Maximum** value of the control *(maximum)*.
4. Check **Snap to ticks** if you want to bind control’s value.
5. Enter **Decimal position** of displayed text in the field.

**Control (counter)**
Control property lets object to write value to the tag. To edit control property click **Control** tab on the object property window.

1. Select the **Tag** value of which will be changed.
2. Enter **Minimum** value of the control *(minimum)*.
3. Enter **Maximum** value of the control *(maximum)*.
4. Write **Delta** in the field.
5. Enter **Decimal position** of displayed text in the field.

**Value (for meters)**
Value property lets to control values of analog and digital meters depending on value of the tag. To edit value property click **Value** tab on the object property window.

1. Select the **Tag** value of which will be used to control value of meter.
2. Enter minimum value of the meter in the **Minimum** field *(minimum)*.
3. Enter maximum value of the meter in the **Maximum** field *(maximum)*.
4. Enter **Decimal position** of displayed text in the field *(decimalpos)*.
Value (for Range indicator and Gauges)

Value property lets to indicate depending on value of the tag. To edit value property click Value tab on the object property window.

1. Select the Tag value of which will be used to control value of the indicator.
2. Enter minimum value of the indicator in the Minimum field (minimum).
3. Enter maximum value of the indicator in the Maximum field (maximum).
4. Enter Decimal position of displayed text in the field (decimalpos).
5. Check Show value if you want to make visible number representation.
6. Check Show history if you want to make visible history information of the tag.
7. Choose Period of the history information.
8. Choose Color of the history information.
9. Click Collection of the Ranges if you want to add Color ranges of the indicator. You’ll see the window:

1. Enter the value from which the object will change color in the From field.
2. Enter the value to which the object will change color in the To field.
3. Choose Color.

You can Add, Edit or Remove collection element of line color conditions.
**Switch control**
Switch control property lets to switch boolean value of the tag. To edit switch control property click **Switch control** tab on the object property window.

1. Select the **Tag** value of which will be controlled by the switch.

**Switch control (3 pos.)**
Switch control property lets to switch boolean values of the tags. To edit switch control property click **Switch control** tab on the object property window.

1. Select the **Tag ON** value of which will be controlled by the switch.
2. Select the **Tag OFF** value of which 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.
**Selector**
Selector property lets to enter values by clicking selector buttons. To edit selector property click **Selector** tab on the object property window.

1. Select the **Tag** which value will be changed.
2. Click Collection button to edit selector buttons.

After clicking Collection you’ll see window:
1. Enter the **Value** which will be written after clicking the button of the selector.
2. Enter **Text** of the selector’s button.

**Row number**
Row number property lets to choose row number of the Recipe database in Parameter table object. To edit property click **Row number** tab on the object property window.

1. Select the **Tag** which value will choose row number of the Recipe database.
Design script

To start designing the script you want, you should double click on it or click right button on the Project window->Scripts and choose Open script. For creating scripts you should use FBD objects.

Create script object
Add new object on the screen you can in this way: click right button on the Canvas and choose New object 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). Select object you can 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.
**Script objects of FBD language**

Below description of script libraries and object.

### 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.

### 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).

### Bitmap operations library
- **Inverse** - this script object used to inverse input integer value (Output = ~ 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 & ~(1<<Input2)).

### 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 = Log^Input). 

**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).

**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)).

**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 (Output[Input2] = Input1).

**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*.

**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 = \( \sin(\text{Input}) \)).

**Cosine** - this script object used to calculate cos of Input value. (Output = \( \cos(\text{Input}) \)).

**Tangent** - this script object used to calculate tag of Input value. (Output = \( \tan(\text{Input}) \)).

**Arc Sine** - this script object used to calculate arc sin of Input value. (Output = \( \arcsin(\text{Input}) \)).

**Arc Cosine** - this script object used to calculate arc cos of Input value. (Output = \( \arccos(\text{Input}) \)).

**Arc Tangent** - this script object used to calculate arc tang of Input value. (Output = \( \arctan(\text{Input}) \)).

**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.

**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*.

**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 = \( \text{Input1+Input2} \)).

**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.

**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).

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.

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.

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.
Description of ST(Structured text) 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** button on the **Tool bar**. 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.

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 graphics-based 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. Try to see if you can understand the function of this program. Does Structured Text look familiar to you?

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 is 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 has a meaning and is 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**
- **The language is case-sensitive**
- **Spaces have no function**

Structured Text consists of statements and semicolons to separate them.

It is good practice to use upper- and lowercase for readability.

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 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:**

```plaintext
// comment
```

**Multiple line comment:**

```plaintext
/* start comment
...
end comment */
```

**Making Statements with Structured Text**

So, Structured Text consists of statements. But what is statements? A statement is you telling the TeslaSCADA2 what to do. Let’s take the first statement as an example:

```plaintext
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`. 
**Types in Structured Text**
Data types of Structured Text are similar to data types of TeslaSCADA2:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Format</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>Boolean</td>
<td>False/True</td>
</tr>
<tr>
<td>byte</td>
<td>Byte</td>
<td>-128 … 127</td>
</tr>
<tr>
<td>short</td>
<td>Short</td>
<td>-32768 … 32767</td>
</tr>
<tr>
<td>int</td>
<td>Integer</td>
<td>-2^31 … 2^31-1</td>
</tr>
<tr>
<td>long</td>
<td>Long Integer</td>
<td>-2^63 … 2^63-1</td>
</tr>
<tr>
<td>float</td>
<td>Float</td>
<td>±3.40282347E+38F</td>
</tr>
<tr>
<td>double</td>
<td>Double</td>
<td>±1.79769313E+308</td>
</tr>
<tr>
<td>string</td>
<td>Character string</td>
<td>“My string”</td>
</tr>
<tr>
<td>array</td>
<td>Array</td>
<td>byte[], short[], int[], float[]</td>
</tr>
</tbody>
</table>

Examples of variable initialisation:

```plaintext
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];
```
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.
Operators
There are several operators available in Structured Text language:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parenthesization</td>
<td>(expression)</td>
<td>Highest</td>
</tr>
<tr>
<td>Negation</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Complement</td>
<td>!</td>
<td></td>
</tr>
<tr>
<td>Multiply</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Divide</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Modulo</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Subtract</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Left Shift</td>
<td>&lt;&lt;</td>
<td></td>
</tr>
<tr>
<td>Right Shift</td>
<td>&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Comparison</td>
<td>&lt;, &gt;, &lt;=, &gt;=, ==, !=</td>
<td></td>
</tr>
<tr>
<td>Boolean AND</td>
<td>&amp;</td>
<td>Lowest</td>
</tr>
<tr>
<td>Boolean OR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boolean XOR</td>
<td>^</td>
<td></td>
</tr>
</tbody>
</table>

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:

\[ A + B \times C \]

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. \( B \times C \) 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

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

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 or FALSE.
- == (equal)
- < (less than)
- <= (less than or equal)
- > (greater than)
- >= (greater than or equal)
- != (not equal)

Example:
TEMPERATURE = 93.9;
TEMPERATURE >= 100.0
Result:
false
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 or FALSE as a result of the expression.

• &&
• ||
• ^
• !

Example:

limitswitch1 = true;
limitswitch2 = false;
limitswitch1 || limitswitch2

Result:
true

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

<table>
<thead>
<tr>
<th>Bit number</th>
<th>1111 (15)</th>
<th>1000 (8)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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.

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.
Conditional Statements
A 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 TeslaSCADA@:

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.

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. A boolean expression could also look like this:
1 > 2
But this time the boolean expression will evaluate to FALSE, 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:

```python
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:

```python
if (INPUT1 & INPUT2) {
    OUTPUT1 = TRUE;
}
```

Now the expression will evaluate to TRUE, only if INPUT1 and INPUT2 is TRUE.
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.

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.

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 {}.

```java
if (boolean expression) {
    break;
}
```
**User-defined functions**
Also you can use user-defined functions in Structured Text language for TeslaSCADA2. You can find example below:
```plaintext
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.

**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 whitespaces and any signs.

**Example:**
```plaintext
int var = 10;
Tags.Tag1 = var;
```
In this example value of the variable `var` will be assigned to Tag with name Tag1.

**Other Example:**
```plaintext
float f = Tags.Float1;
```
In this example value of the Tag with name Float1 will be assigned to variable `f`.

**Using Object property fields in Structured Text**
You can include Object property fields in your project’s scripts by using keyword `Objects`. Then type dot (.), name of your Object, again type dot (.) and name of property field. For possibility to compile this code the name of the object and object property fields should contain only English letters without whitespaces and any signs.

**Example:**
```plaintext
int width = 100;
Objects.Rectangle.width = var;
```
In this example value of the variable `width` will be assigned to Object with name `Rectangle` and field property name `width`. Name of the property fields you can find out in parentheses of object and property descriptions above.
Using Server parameter fields in Structured Text
You can include Server parameter fields in your project’s scripts by using keyword Servers. Then type dot (.), name of your Server, again type dot (.) and name of parameter field. For possibility to compile this code the name of the server and server parameter fields should contain only English letters without whitespaces 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 field property name ipaddress. Name of the property fields you can find out in parentheses of server and parameter descriptions above. Also for parameters are written in descriptions you can use: lostconnection, connect and connected.

Using User parameter fields in Structured Text
You can include User parameter fields in your project’s scripts by using keyword Users. Then type dot (.), name of your User, again type dot (.) and name of parameter field. For possibility to compile this code the name of the user and user parameter fields should contain only English letters without whitespaces and any signs.

Example:
Users.Operator.controlfunctions = true;
In this example value true will be assigned to the user with name Operator and field property name controlfunctions. Name of the property fields you can find out in parentheses of user and parameter descriptions above.

Embedded functions
In the Structured Text language for TeslaSCADA2 there are number of embedded functions:
print(Input) - print input in the log.
sqrt(Input) - arithmetic operation square root of the input value.
pow(Input1, Input2) - arithmetic operation power for input values. output = Input1^Input2.
log(Input1, Input2) - arithmetic operation logarithm of the input value (Output = Log_{Input2}Input_1).
ln(Input1) - arithmetic operation ln (natural logarithm) of the input value (Output = Ln(Input)).
bytestoshort(Input1, Input2) - used to pack 2 bytes in the short (Output = Input<<8+Input2).
shorttobyte(Input1, Input2) - used to unpack short value in 2 bytes (Output = Input<<Input2).
shorttoint(Input1, Input2) - used to unpack int value in 2 shorts (Output = Input<<16+Input2).
readbit(Input1, Input2) - used to read bit of the input value (Output = Input[Input2]).
setbit(Input1, Input2)- used to set bit of the input value (Output = Input | 1<<Input2).
resetbit(Input1, Input2) - used to reset bit of the input value (Output = Input & ~(1<<Input2)).
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)).
abs(Input) - used to arithmetic operation absolute for input value (Output = |Input|).
sign(Input) - used to arithmetic operation sign for input value (Output = -Input).
int(Input) - used to arithmetic operation for getting integer part of the input value (Output = int(Input)).
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)).
hextoint(Input) - converts hex value into integer.
inttohex(Input) - converts integer value into hex.
stringsequals(Input1, Input) - compare two strings in Inputs and if there are equals it returns true.
stringtodoreal(Input) - converts Input’s string value into double value.
doubletostring(Input) -converts Input’s double value into string value.
stringint(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 defined in Input2 and Input3.
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.
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).
reconnect(Input1,Input2) - used to reconnect to server with name from Input1 to IP address from Input2.
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.
sendemail(Input1, Input2) - send email (if it setup in Project properties) with subject from Input1 and message from Input2.
oodogetmodelcount(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 fits the filter in Input3. Filter is consisted with name of field, 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");

odooreadmodelfield(Input1, Input2, Input3, Input4, Input5) - used to read value of row’s field that you get from Odoo ERP with name in Input1 (Example: “OdooERP0”) and model with name in Input2 (Example: “mrp.workorder”) that fits the filter in Input3. Filter is consisted with name of the field, comparison and value to compare separated by commas (Example:”state,=,cancel” get rows where state == cancel). Name of the field 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 = odooreadmodelfield("OdooERP0", "mrp.workorder", "", "production_id", 1);

odoowritemodelfield(Input1, Input2, Input3, Input4, Input5, Input6) - used to write value to the row’s field that you get from Odoo ERP with name in Input1 (Example: “OdooERP0”) and model with name in Input2 (Example: “mrp.workorder”) that fits the filter in Input3. Filter is consisted with name of the field, comparison and value to compare separated by commas (Example:”state,=,cancel” get rows where state == cancel). Name of the field 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: odoowritemodelfield("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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

callerfunction(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);

Example: saverecipeexcelreport(1,"RecExcel","streport","Title");

excelcreateWorkbook() - this function create workbook for Excel file;

excelsaveworkbook(Input1) - used to save workbook in the Excel with name in Input1.
Example: excelsaveworkbook("filename");

excelcreatesheet(Input1) - create sheet in the workbook of Excel file 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");

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.
Example: makescreenshot("filename");

createdbsqlliteconnection(Input1) - used to create create connection to SQLite database with name in Input1.
Example: createdbsqlliteconnection("filename");

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 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("filename");
**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 fit to condition first row is read)

*Example: 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: 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 fit 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 fit the row(s) in Input3.

*Example: deleterowindb(“databasename”, “tablename”, “_id=1”);*
Simulation

You can simulate behavior of your project. To start simulation select the menu item *Project* and *Run simulation* or click button on the *Toolbar*.

If you use users in your project Login dialog will appear. 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* -> *Tags*. Or you can click by right button on the tag and select *Simulate* and *Set value* menu item. Also you can simulate behavior of the tag:

1. *Random value* - periodically change the value of the tag randomly.
2. *Ramp value* - periodically change the tag value from 1 to 100 by adding 1.

By selecting *Simulate* and *Cancel* you annul the task.

Also it’s possible to change value of the tag using control graphical objects of your project like text, buttons, slider, counter 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* and *Events log* objects. Place these objects on the *Canvas*. 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.
During simulation Events log will be look like this:

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.
Reports

For some graphical objects like Events table, History trends, Recipe table 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:
1. **Format** of the paper.
2. **Orientation** of the paper.
3. **Paper width** and **Paper height**.
4. Set **Pagination** if you want to show page numbers.

**Banner**, where:
1. Choose **Image** of the banner.
2. Setup **Width** and **Height** of the banner.
3. Setup **Horizontal Alignment** of the banner.
4. Use banner **For All Pages** or not.

**Report title**, where:
1. **Title** caption of the report.
2. **Font** of the caption.
3. **Color** of the caption.
4. **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:
1. **Background color** of the caption.
2. **Border** of the caption.
3. **Vertical Alignment** of the caption.
4. **Number of columns** using in report.
5. **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:
1. Check **Highlight Even Row** if you want to do it.
2. Choose **Even Row Background**.
3. 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.
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. For this purpose, first the corresponding TeslaSCADA Runtime mobile app on the Android device or PC apps on the Windows, Linux or MAC OS must be installed and started.

If the app has now been installed on the mobile device or PC, there are 2 ways to load the project to the device.
1. Network method.

Network method

This method must, the PC on which the TeslaSCADA IDE is started, and the mobile device or PC on which TeslaSCADA Runtime started and the project will be stored in a Wi-Fi network (note IP addresses) are.

Procedure:
1. Enable WiFi on your mobile device or PC where installed TeslaSCADA Runtime.
2. Start the TeslaSCADA2 Runtime app.
3. Open it in the editor TeslaSCADA2 IDE the desired project to be transferred.
4. Select the menu item File and Load on Device.
5. It now opens the dialog Load on Device and it will now 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 a normally 5-10s. In individual cases it may happen that this search can take up to 3 minutes.

If you can’t find a device you can try to restart Load on Device dialog and TeslaSCADA2 Runtime application.
6. After a successful search in this dialog box all found mobile devices with active TeslaSCADA Runtime app will be shown.
7. Now select the desired target device and press the Load on Device button.
8. After a successful transfer, the target device with TeslaSCADA2 Runtime load new project.
**Manual method**

Another way to load a project on the mobile device is a file explorer such as: the **Android File Transfer** for Mac OS. Once the TeslaSCADA Runtime installed mobile app and once started on the sd card, a folder called **TeslaSCADA2Runtime->Projects** is created.

Now, if the project which has been stored as .tsp2 file from the Windows, Linux or MacOS system TeslaSCADA IDE(The path was chosen when you first save of the project) will be manually copied to the folder of the sd card of the TeslaSCADA Runtime mobile device, the app can be started normally. Now loads the app, the file from this folder by clicking *Load* on the main menu of TeslaSCADA Runtime.

There is no problem to manually copy to the PC where TeslaSCADA Runtime is installed. You can use local network, flash driver or use any other storage device.
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 and select Import for iOS. When you do it Import for iOS window will appear.

To do import project for iOS devices click Import, file dialog will appear, enter name of the file and click OK.

Import file has *.tsp2db extension. This file based on SQL database format and you can open and check it by using softwares for working with SQL databases. You can also open imported file by clicking Open button.

Imported or opened file will be appeared in the text field. To activate project:
1. Choose license type.
2. Enter license number.
3. Click Activate button (it will change background colour to the green and «License available for activation» message will appear).
4. If you want to deactivate license click Deactivate button (it will change background colour to the 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 has now been installed on the iOS mobile device (iPhone or iPad), there are 2 ways to load the imported project on the device.
1. Network method.

Click Load on iOS device to use Network method to load imported file on your iOS device.

Network method

This method must, the PC on which the TeslaSCADA IDE is started, and the iOS mobile device on which TeslaSCADA2 Runtime started and the project will be stored in a Wi-Fi network (note IP addresses) are.

Procedure:
1. Enable WiFi on your mobile device where installed TeslaSCADA2 Runtime.
2. Start the TeslaSCADA2 Runtime app.
3. In TeslaSCADA2 IDE select the menu item File and Import for iOS.
4. Open the desired imported project. Click Load on iOS device.
5. It now opens the dialog Load on iOS Device and it will now 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 a normally 5-10s. In individual cases it may happen that this search can take up to 3 minutes.

If you can’t find a device you can try to restart **Load on iOS Device** dialog and TeslaSCADA2 Runtime application on iOS device.

6. After a successful search in this dialog box all found mobile devices with active TeslaSCADA2 Runtime app will be shown.

7. Now select the desired target device and press the **Load on Device** button.

8. After a successful transfer, the target device with TeslaSCADA2 Runtime load new project.

**Manual method**

Another way to load an project on the iOS mobile device is iTunes ->File Sharing.

Procedure:
1. Open iTunes on your Mac or PC.
2. Connect your iPhone or iPad to your computer using the USB cable that came with your device.
3. Click your device in iTunes.
4. In the left sidebar, click Apps. Then, scroll to the File Sharing section at the bottom of the page.
5. Select TeslaSCADA2 Runtime see which files are available for sharing in that app on your device.
6. Now you can copy your imported project (*.tsp2db) to the documents folder of TeslaSCADA2 Runtime apps.
Use Telegram Bot
If you want to get events notification from your project in TeslaSCADA2 Runtime you can use Telegram messenger for this purpose. To do this you should create Telegram Bot:
1. You should have Telegram messenger installed on your device and have an account.
2. Open in browser https://telegram.me/botfather
3. Click button «Send message» or «Open in Telegram Web» (you should have login in web telegram client).
4. Open your Telegram client and choose BotFather.
5. Click button Start or type /start.
6. Enter /newbot.
7. Enter your bot’s name. The name should be unique. This name you should enter in Bot’s name field of project properties.
8. Then you should choose username for your bot.
9. After entering username you’ll get the telegram bot’s token. Enter it in Bot’s token field of project properties.

Now you can use telegram bot in getting notification messages from TeslaSCADA2 Runtime. To do this you should find your created bot in your telegram messenger client and click button Start or enter /start. To stop getting notification messages enter /stop. Also you can get some information from your project:
1. Enter /tags to get current values of tags. You’ll get information only from currently monitored tags (tags that enable history, events and tags of objects that displayed on currently opened screen).
2. Enter name of the tag used in your project. You’ll get information about value of this tag and if tags supports history you’ll get trend for last hour period. You can choose other period by clicking proper button.

Warning don’t use underline in the name of the tags. Telegram have problems with working with this kind of names.
At this moment you can use Telegram bot only on desktop versions of TeslaSCADA2 Runtime.