



# myDESIGNER

USER MANUAL  
Version 7

## Table of contents

<b>1</b>	<b>About myDESIGNER .....</b>	<b>7</b>
<b>2</b>	<b>Tutorial Videos.....</b>	<b>8</b>
<b>2.1</b>	<b>myDESIGNER .....</b>	<b>8</b>
<b>2.2</b>	<b>myREPORTS.....</b>	<b>8</b>
<b>2.3</b>	<b>myMOBILE .....</b>	<b>9</b>
<b>3</b>	<b>Getting Started with myDESIGNER.....</b>	<b>10</b>
3.1	Creating a Project .....	13
3.2	Workspace Management.....	18
<b>4</b>	<b>Checking Project .....</b>	<b>21</b>
<b>5</b>	<b>Projects' Visual Appearance .....</b>	<b>24</b>
<b>6</b>	<b>GUI/HMI Editor.....</b>	<b>28</b>
6.1	Creating Views.....	28
6.2	Selecting Objects.....	34
6.3	Drawing Primitives.....	34
6.4	Creating Text Elements.....	35
6.5	Insert Image.....	36
6.6	Poly-lines .....	39
6.7	Moving Objects .....	41
6.8	Resizing Objects .....	41
6.9	Rotating Objects.....	42
6.10	Skewing Objects .....	44
6.11	Filleting.....	44
6.12	Combining Objects .....	45
6.13	Fill and Stroke .....	48
6.14	Rulers and Guides.....	64
6.15	Layers .....	68
6.16	Copying and Pasting Elements.....	77
6.17	Object's Order .....	79
6.18	Grouping.....	80
6.19	Repeated Actions Mode .....	82
6.20	View Scripts .....	82
6.21	Used Tags.....	84
6.22	Zoom on Zone.....	84
6.23	Undo and Redo .....	84
6.24	View Properties.....	84
<b>7</b>	<b>Components .....</b>	<b>89</b>
7.1	Default Components.....	90
7.2	Custom Components .....	92
7.3	Used Components.....	96
7.4	Editing Components.....	99
7.5	Replacements.....	105
7.6	On Touch Actions.....	108
7.7	Entering Advanced Functions (Equations).....	113
<b>8</b>	<b>Combo Box, List Box and Text Box .....</b>	<b>116</b>
8.1	Inserting Into View .....	116
8.2	Components parameters .....	118
8.3	Using Components In View Scripts .....	121
8.4	Supported functions.....	124
<b>9</b>	<b>Active Area .....</b>	<b>125</b>
9.1	Introduction.....	125

9.2	Creating Active Areas.....	125
9.3	External Web Page in Active Area.....	152
9.4	HTML Code.....	152
9.5	DIV Type.....	152
<b>10</b>	<b>Layout Views .....</b>	<b>156</b>
10.1	Page Layout.....	156
10.2	Adding Layout View .....	157
10.3	Creating New Layout.....	157
<b>11</b>	<b>Entering Tags and Math Expressions .....</b>	<b>163</b>
11.1	Entering Tags .....	163
11.2	Entering Mathematical Expressions.....	164
<b>12</b>	<b>Tag Database .....</b>	<b>169</b>
12.1	Changing tag value .....	169
12.2	Engineering units .....	169
12.3	Filtering data .....	171
12.4	Usage Count.....	172
12.5	MS Excel Import and Export .....	172
12.6	Tag Import.....	174
12.7	Deleting unused tags.....	182
12.8	Restoring tag database.....	183
12.9	Specifying a new tag during development.....	183
<b>13</b>	<b>Formatting Numerical values.....</b>	<b>184</b>
<b>14</b>	<b>Linking Views with PLC.....</b>	<b>185</b>
14.1	Introduction.....	185
14.2	Animations .....	185
14.3	Show Value Animation .....	186
14.4	Value to Text Mapping Animation.....	191
14.5	Visibility Animation.....	191
14.6	Opacity Animation .....	192
14.7	Color Animation.....	193
14.8	Moving Animation.....	194
14.9	Size Animation.....	196
14.10	Scale Animation .....	198
14.11	Rotate Animation .....	198
14.12	Circular Sector Animation .....	200
14.13	Zoom Visibility Animation.....	201
14.14	Sounds.....	202
14.15	Sounds Triggered by Tag value .....	204
14.16	Effects.....	205
<b>15</b>	<b>Time Sequence.....</b>	<b>214</b>
15.2	Example.....	215
15.3	Triggering Time Sequences.....	217
15.4	Modifying Time Settings.....	218
15.5	Adding New Sequences.....	219
15.6	Renaming Time Sequence.....	220
15.7	Switching Among Time Sequence.....	220
15.8	Modifying Timing .....	220
15.9	Combining Multiple Time Sequences.....	221
<b>16</b>	<b>Open Command.....</b>	<b>226</b>
16.1	Open Type.....	228
16.2	Target .....	228
16.3	Popup (Face Plate) Window .....	229

16.4	View type .....	230
<b>17</b>	<b>Write/Set Command .....</b>	<b>240</b>
17.1	Using Batches .....	242
17.2	Specifying Set Command Parameters .....	243
17.3	Value Options .....	247
<b>18</b>	<b>Scaling Set Values .....</b>	<b>253</b>
18.1	Set Example: .....	253
<b>19</b>	<b>Key Shortcuts.....</b>	<b>255</b>
<b>20</b>	<b>On Touch .....</b>	<b>257</b>
20.1	On Touch in Views.....	257
20.2	On Touch in Component.....	258
20.3	On Touch Example .....	259
20.4	Lock Element.....	261
20.5	Lock Key .....	264
20.6	Slider.....	266
<b>21</b>	<b>Parametric Views .....</b>	<b>269</b>
21.1	Opening Parametric Views .....	269
21.2	Symbolic Text Replacements.....	272
21.3	Symbolic Tag Creation.....	273
21.4	Replacing Connections .....	274
21.5	Nesting Parametric Views.....	275
21.6	Usage Example: .....	275
<b>22</b>	<b>View Scripts .....</b>	<b>278</b>
22.1	Using Script in Views .....	278
22.2	Declaring Variables.....	281
22.3	Script Writing.....	281
22.4	Using variables in animations and effects.....	282
22.5	Debugging View Scripts .....	286
22.6	Using debug screen on your view.....	286
22.7	Using web browsers integrated debugger .....	288
22.8	Using JavaScript Libraries - Includes .....	290
22.9	Linking External JavaScript Libraries – Remote Includes .....	292
<b>23</b>	<b>View and Server Side Scripts – Common tasks.....</b>	<b>293</b>
23.1	Graphical guides.....	293
23.2	Read/Write data from/to PLC.....	299
23.3	Generating Report.....	301
23.4	Other Guides .....	301
<b>24</b>	<b>Documents .....</b>	<b>305</b>
<b>25</b>	<b>Reports.....</b>	<b>307</b>
25.1	Creating Report Templates .....	307
25.2	Designing Report .....	308
25.3	Inserting Text.....	309
25.4	Adding Picture – logo.....	310
25.5	Table Data.....	311
25.6	Table Summary .....	314
25.7	Previewing Report during design.....	316
25.8	Showing Report in Runtime.....	317
25.9	Creating Custom Report in Runtime.....	317
25.10	Creating Report on Demand .....	319
<b>26</b>	<b>CAS Alarms .....</b>	<b>322</b>
26.1	Digital Alarms.....	322

26.2	Analog Alarms.....	322
26.3	Alarm Window.....	327
26.4	Alarm History.....	327
26.5	CAS Alarms in View - Example.....	329
<b>27</b>	<b>Data Logging .....</b>	<b>332</b>
27.1	Data-logs .....	332
27.2	Defining Connection.....	334
27.3	Defining Data Points.....	334
27.4	Continuous Data Logging.....	337
27.5	Triggered Data Logs.....	338
27.6	Triggered Logging Example .....	339
27.7	Simple Periodic Export to CSV and Microsoft Power BI.....	340
27.8	Data-logs Properties.....	342
<b>28</b>	<b>Data-Log Views .....</b>	<b>346</b>
28.1	Data Points Selection .....	347
28.2	Data Grouping.....	348
28.3	Filtering Data.....	350
<b>29</b>	<b>Aggregated Data logs .....</b>	<b>353</b>
29.1	Creating aggregated data log.....	353
29.2	Aggregated Values: .....	354
29.3	Time Aggregates .....	357
29.4	Value Change Aggregates.....	358
29.5	Aggregates based on Alarm Activation.....	360
29.6	Running the Aggregation Periodically.....	362
<b>30</b>	<b>Advanced Trends .....</b>	<b>363</b>
30.1	Using Multiple Axes .....	367
30.2	Advanced Trends – Visual Appearance.....	368
<b>31</b>	<b>Connections .....</b>	<b>372</b>
31.1	Creating New Connection.....	373
31.2	Creating PLC Type Connection .....	374
31.3	Creating Database Type Connection.....	374
31.4	Creating IoT Type Connection.....	378
31.5	Deleting Connections.....	378
31.6	Advanced Options - Optimizations.....	379
<b>32</b>	<b>User Accesses .....</b>	<b>381</b>
32.1	Access Levels.....	381
32.2	Access Groups.....	381
32.3	User Accounts .....	381
32.4	Limiting Access for Whole Project.....	383
32.5	Limited Access for Views and Trends .....	384
32.6	Limited Access of Arbitrary Object in Views.....	385
<b>33</b>	<b>Multi-language support.....</b>	<b>388</b>
33.1	Providing translations inside a Project .....	390
33.2	Translating Names of Views.....	391
33.3	Translating Data-logs.....	392
33.4	Translating CAS Alarms .....	393
33.5	Translating Advanced Trends.....	394
<b>34</b>	<b>Server-side Scripts.....</b>	<b>396</b>
34.1	Introduction.....	396
34.2	Server-side Scripts folder.....	396
34.3	Server-side Scripts Folder Structure.....	397
34.4	Variables Tables.....	397

34.5	Script Data-logs .....	399
34.6	Global Variables .....	400
34.7	Sources Folder .....	402
34.8	Organizing Project into Modules .....	402
34.9	Importing Modules .....	403
34.10	Using the Event-driven Asynchronous Callbacks .....	405
34.11	Creating Server Side Reports .....	406
34.12	mySCADA Specific Functions.....	411
34.13	Debugging.....	412
34.14	Script Status (on myBOX Devices Only).....	414
34.15	Ser2Net (on myBOX Devices Only).....	415
<b>35</b>	<b>View and Server Side Scripts – Common tasks.....</b>	<b>417</b>
35.1	Graphical guides.....	417
35.2	Read/Write data from/to PLC.....	424
35.3	Generating Report.....	426
35.4	Other Guides .....	426
35.5	Server Side Scripts – Examples.....	428
35.6	Reading data from PLC.....	429
35.7	Writing data into PLC.....	432
35.8	Timers – eg. Run code in given time intervals.....	432
35.9	Scheduled Execution e.g run code every Monday at 2:00 PM.....	433
35.10	Generating a report at given time interval .....	434
35.11	Limiting Access to Generated Files .....	437
35.12	Processing data-log data .....	437
35.13	Exporting data-log data into CSV files .....	439
35.14	Using Virtual PLC.....	441
35.15	Sending Data from View Script into Server Side Scripts.....	447
<b>36</b>	<b>Devices .....</b>	<b>451</b>
<b>37</b>	<b>EtherNet/IP Driver .....</b>	<b>397</b>
<b>38</b>	<b>MicroLogix and SLC Driver .....</b>	<b>399</b>
<b>39</b>	<b>Modbus Driver .....</b>	<b>400</b>
39.1	Tag name syntax .....	400
39.2	32-bit registers in Modbus .....	401
39.3	Floating point numbers.....	402
39.4	32-bit Integers .....	403
39.5	Address mapping.....	403
39.6	Signed and unsigned numbers.....	404
<b>40</b>	<b>Siemens S7 family PLCs Driver .....</b>	<b>406</b>
40.1	Memory types .....	406
40.2	S7 Data types .....	406
40.3	S7 1200/1500 notes.....	407
40.4	Protection .....	408
40.5	LOGO! 0BA7/0BA8 configuration.....	409
40.6	S7-200 (via CP243-1) configuration.....	409
<b>41</b>	<b>OPC UA Driver.....</b>	<b>410</b>
41.1	Connection configuration.....	410
41.2	Tag Name Syntax .....	412
<b>42</b>	<b>MELSEC-Q Driver.....</b>	<b>413</b>
42.1	Tag name syntax .....	413
42.2	Connection Settings.....	415
<b>43</b>	<b>SigFox Driver.....</b>	<b>417</b>

43.1	datatype.....	418
43.2	Reverse order.....	419
<b>44</b>	<b>Databases Driver.....</b>	<b>421</b>
44.1	Reading values from SQL database.....	422
44.2	Examples.....	423
44.3	Writing values to SQL database.....	424
44.4	Using SQL connectivity in Server Side Scripts.....	425
<b>45</b>	<b>Download/Upload from/to Device .....</b>	<b>426</b>
45.1	Download to Device.....	426
45.2	Upload from Device .....	428

# 1 About myDESIGNER

---

*myDESIGNER*, part of the *mySCADA* bundle, 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-feature SCADA (Supervisory Control and Data Acquisition) project. With this tool, you can create and manage *mySCADA* projects; configure connections with devices; and enter tags, alarms, and trends. It also allows you to design advanced mimic graphics with specific animations, corresponding with PLC tag values.

An easy-to-use interface allows for simple manipulation of the project's configuration and data processing. The project data are stored in a single directory for easy backup and restoration.

*myDESIGNER* has an integrated GUI (Graphical User Interface) visualization editor for easy creation of professional looking mimic graphics. The graphics are based on the Scalable Vector Graphic (SVG) format, which means that your controlled technology will always look sharp.

## Key Features

- Free for personal and business use
- Simple to use
- GUI design in Scalable Vector Graphics (SVG)
- Animations and effects can be added to any shape or object
- Support for background images (JPEG, GIF, and PNG)
- Ability to attach PDF documents to the project
- Ability to attach MP3 sound files to the project
- Built-in script editor
- Available for Mac OS X, Windows, and Linux

## Drawing Possibilities

- Shape tools: rectangles, circles, ellipses, paths, texts, and images
- Path tools: Bezier curves, conversion to a path, union, subtraction, intersection, merge
- Group editing
- Advanced text support
- Images import (.jpg, .png)
- Transformations: resize, rotate, skew, align, distribute
- Properties manager
- Resource manager: gradients, patterns, and markers
- Layers
- Smart Guidelines, snap to points, grid

## Project Management

Key components of *mySCADA* projects are the visualization screen views. This is where the schematic visualizations of controlled devices are displayed. Single objects or groups of objects can be created and defined especially for communication with connected PLCs. These specified objects can then be animated, changing their visual appearance based on the PLC tag values.

The graphic screens are internally represented as SVG files, defined project connections as configuration JSON files, and alarm states and trends stored in separate JSON files. The configuration files can be edited directly in *myDESIGNER*, so you do not have to edit them individually. The visual and server side scripts are also stored into external files with

the .js extension. Complete *mySCADA* projects consist of configuration files, SVG files, scripts, documents and sound files, and they are saved directly to your hard drive. For more information about exact project structure, see the following table:

File	Description
settings.json	Complete settings of project
*.dlg.json	Data-log definition file
datalogs.json	Settings for data-logs such as its names, translations, size distribution, etc.
trends.json	Settings for trends such as its names, translations, etc.
views.json	Settings for views such as its names, translations, etc.
*.trnd	Advanced trend definition file
cas.json	CAS Alarms definition file
*.svg	Mimic diagram (view) definition.
*.svg.js	View Script. Has the same name as the corresponding view. Contains a list of functions in JavaScript language.
scripts	Directory containing all server side scripts. Can be empty if no server-side scripts are defined.

In the following sections, you will learn how to create a project; add, configure, or delete connections; define alarms and trends; and manage other project settings.

## 2 Tutorial Videos

---

To get you started using myDESIGNER quickly and efficiently, we have prepared a set of tutorial videos. These videos cover most of myDESIGNER's functionalities. Here is the list of all tutorial videos:

### 2.1 myDESIGNER

Creating a New Project	<a href="https://www.youtube.com/watch?v=lpXeE13HyZl">https://www.youtube.com/watch?v=lpXeE13HyZl</a>
Exporting/Importing Projects	<a href="https://www.youtube.com/watch?v=d8jvR75eGz0">https://www.youtube.com/watch?v=d8jvR75eGz0</a>
Creating PLC Connections	<a href="https://www.youtube.com/watch?v=9_jUEjRtFAC">https://www.youtube.com/watch?v=9_jUEjRtFAC</a>
Creating Views	<a href="https://www.youtube.com/watch?v=HtLLUt3O5dc">https://www.youtube.com/watch?v=HtLLUt3O5dc</a>
Layout Views	<a href="https://www.youtube.com/watch?v=sVvFPCMYUZ8">https://www.youtube.com/watch?v=sVvFPCMYUZ8</a>
Customizing the look	<a href="https://www.youtube.com/watch?v=So78TI730KY">https://www.youtube.com/watch?v=So78TI730KY</a>
Drawing primitives	<a href="https://www.youtube.com/watch?v=ryduKCgzFr0">https://www.youtube.com/watch?v=ryduKCgzFr0</a>
Active Areas [part1]	<a href="https://www.youtube.com/watch?v=q_V91pm8C1c">https://www.youtube.com/watch?v=q_V91pm8C1c</a>
Active Areas [part2]	<a href="https://www.youtube.com/watch?v=uq-hHTbrwrs">https://www.youtube.com/watch?v=uq-hHTbrwrs</a>
Creating Custom Component	<a href="https://www.youtube.com/watch?v=tYh4jD-2f2M">https://www.youtube.com/watch?v=tYh4jD-2f2M</a>
Master Components	<a href="https://www.youtube.com/watch?v=jeJNHUKfWEk">https://www.youtube.com/watch?v=jeJNHUKfWEk</a>
Rulers and Guides	<a href="https://www.youtube.com/watch?v=jiTpwXYE_I">https://www.youtube.com/watch?v=jiTpwXYE_I</a>
Layers	<a href="https://www.youtube.com/watch?v=uogOzykluA0">https://www.youtube.com/watch?v=uogOzykluA0</a>
Animations	<a href="https://www.youtube.com/watch?v=_cPPysaTWKc">https://www.youtube.com/watch?v=_cPPysaTWKc</a>
Effects	<a href="https://www.youtube.com/watch?v=GTOfCwrRTPM">https://www.youtube.com/watch?v=GTOfCwrRTPM</a>
Entering Tags	<a href="https://www.youtube.com/watch?v=XNrNiTUPFDk">https://www.youtube.com/watch?v=XNrNiTUPFDk</a>
Commands	<a href="https://www.youtube.com/watch?v=7z_0_DcmvSQ">https://www.youtube.com/watch?v=7z_0_DcmvSQ</a>
Parametric Views	<a href="https://www.youtube.com/watch?v=pAbXrOpUrd8">https://www.youtube.com/watch?v=pAbXrOpUrd8</a>
Viewing scripts	<a href="https://www.youtube.com/watch?v=t7BtSyaPVss">https://www.youtube.com/watch?v=t7BtSyaPVss</a>
Time Sequence	<a href="https://www.youtube.com/watch?v=VYeJfgMaERQ">https://www.youtube.com/watch?v=VYeJfgMaERQ</a>
Documents	<a href="https://www.youtube.com/watch?v=4Q4HKTUFeE4">https://www.youtube.com/watch?v=4Q4HKTUFeE4</a>
CAS Alarms	<a href="https://www.youtube.com/watch?v=HwE-NMUmKxA">https://www.youtube.com/watch?v=HwE-NMUmKxA</a>
Data-logs	<a href="https://www.youtube.com/watch?v=wzxKQ_AUCn8">https://www.youtube.com/watch?v=wzxKQ_AUCn8</a>
Advanced Trends	<a href="https://www.youtube.com/watch?v=NhvLI-QZL34">https://www.youtube.com/watch?v=NhvLI-QZL34</a>
User Access Levels	<a href="https://www.youtube.com/watch?v=A6lloVDPH4">https://www.youtube.com/watch?v=A6lloVDPH4</a>
User Access Levels (group)	<a href="https://www.youtube.com/watch?v=9maAmwbKPDc">https://www.youtube.com/watch?v=9maAmwbKPDc</a>
Zoom visibility	<a href="https://www.youtube.com/watch?v=9BJjs-p9nfl">https://www.youtube.com/watch?v=9BJjs-p9nfl</a>
Getting Started	<a href="https://www.youtube.com/watch?v=hJT0--o1ub8">https://www.youtube.com/watch?v=hJT0--o1ub8</a>
Exporting Data-logs to MS PowerBI	<a href="https://www.youtube.com/watch?v=vVZyWoRu59c">https://www.youtube.com/watch?v=vVZyWoRu59c</a>

### 2.2 myREPORTS

Getting Started	<a href="https://www.youtube.com/watch?v=UtRMkw8VOAw">https://www.youtube.com/watch?v=UtRMkw8VOAw</a>
Exporting to MS Power BI	<a href="https://www.youtube.com/watch?v=rnDPbIOzHMA">https://www.youtube.com/watch?v=rnDPbIOzHMA</a>

## 2.3 myMOBILE

Android App preview

<https://www.youtube.com/watch?v=UXg8vqLRvBo>

iOS App preview

<https://www.youtube.com/watch?v=olnB4t9HOtU>

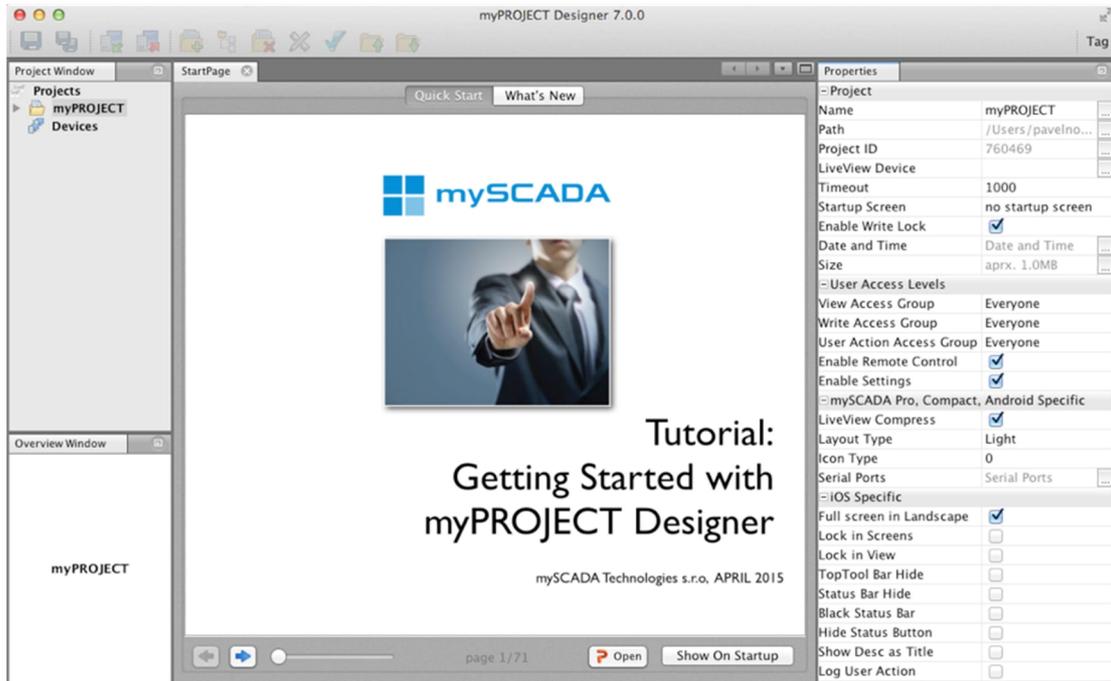
Downloading project to device

<https://www.youtube.com/watch?v=TJ7Qot2iltY>

### 3 Getting Started with myDESIGNER

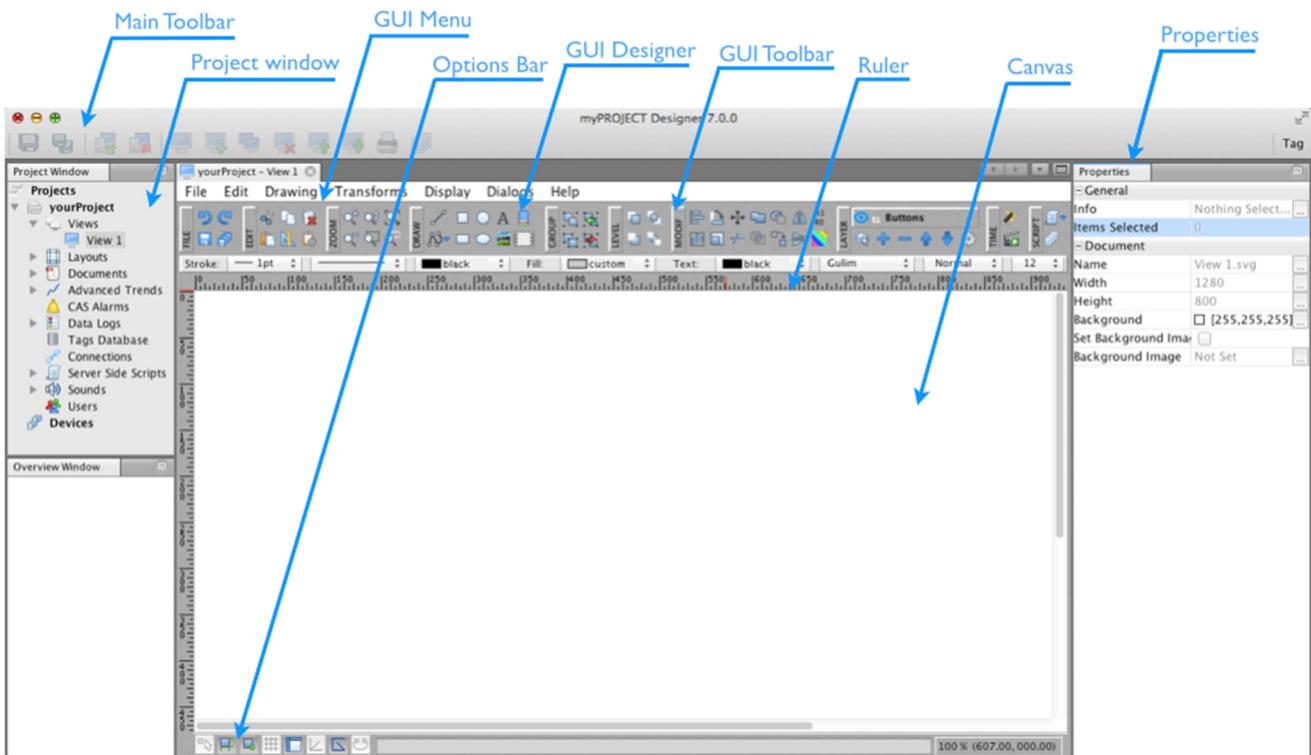
Watch video describing this functionality: <https://www.youtube.com/watch?v=hJTO--o1ub8>

The first screen you will see after opening the application is the *Start Page*.



Click on the blue arrow in the bottom left-hand corner for a brief tutorial to get to know the capabilities of *myDESIGNER*. The *Start Page* always opens up upon the start of the application - if you wish to prevent this, click on the **Show On Startup** button at the bottom. You can close the *Start Page* by clicking on the cross in the header tab. If you wish to display the *Start Page* again, go to the main menu, *Help* section, and select *Start Page*.

Look at the picture below to get to know the *myDESIGNER* interface:



## Main Menu

**File:** manipulation of project files and printing

**Project:** has the same structure as the Project Window

**Projects:** manipulation of projects

**View:** working with views

**Layouts:** viewing Layout definitions

**Documents:** loading, copying, and deletion of attached PDF documents

**Advanced Trends:** opening up and closing trends

**CAS Alarms:** opening up and closing complex alarms

**Data-logs:** defining data logging

**Tag Database:** managing your tags

**Connections:** connections to your PLCs

**Server-side Scripts:** managing *JavaScript* in *mySCADA*

**Sound:** loading and deletion of MP3 files

**Users:** managing users

**Devices:** equipment with installed *mySCADA* application, accessible from editor

**Tools:** language selection

**Window:** direct access to different editor tabs and window control

**Help:** opens the help menu

## Main Toolbar

The basic toolbar consists of the following functions:



**Save** – saves your work



**Save all files** – saves all opened files



**Undo** – undo the last operation



**Redo** – redo last operation



**Upload from a Device** – downloads a project from the selected *mySCADA* device



**Download to a Device** – uploads a project to the selected *mySCADA* device



**New Project** – creates a new empty project or from the wizard, opens existing projects, and imports projects from .mep file



**Check Project** – checks the selected project



**Customize the Look** – customizes colors, fonts, and look of a project



**Project Up** – moves the project up the list



**Project Down** – moves the project down the list

The main toolbar content may change depending on a selected project element (i.e. project, view, sound, script, document, alarm, etc.).

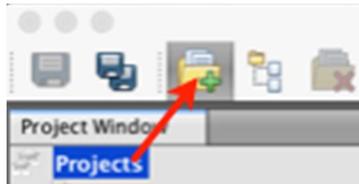
The set of described icons only concerns the project elements; other possible sets are described in the chapters devoted to the relevant elements.

## 3.1 Creating a Project

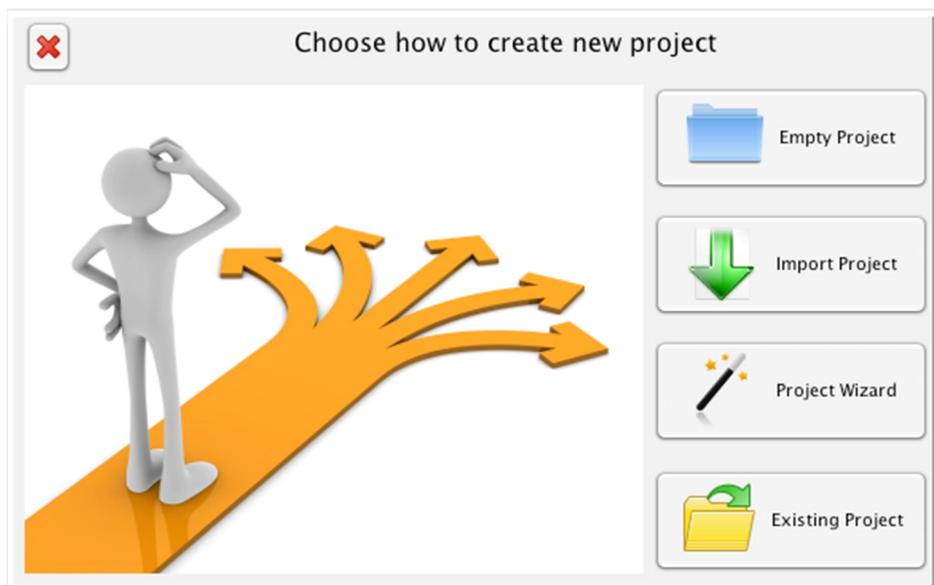
Start using *myDESIGNER* by creating a new project.

Watch video describing this functionality: <https://www.youtube.com/watch?v=lpXeE13HyZl>

- 1) Click on the **New Project** icon in the main toolbar or use the command *New Project* from the main menu *Project-> Projects*.



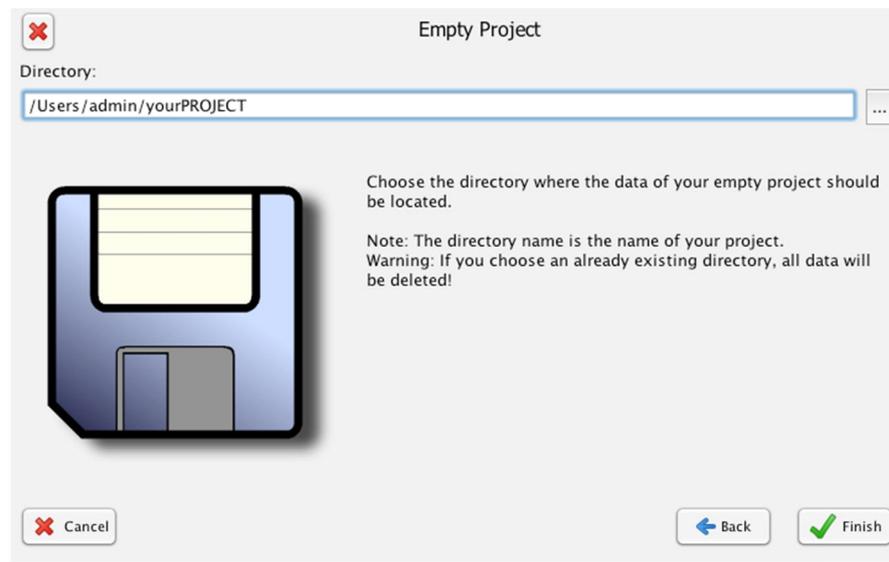
- 2) A new dialog window will show up with 4 options:



### Empty Project

With this option, you can create a new empty project.

- 1) Choose the directory where your empty project will be located.
- 2) If the selected directory is not empty, all its files will be deleted.
- 3) Click on the **Finish** button to create an empty project.



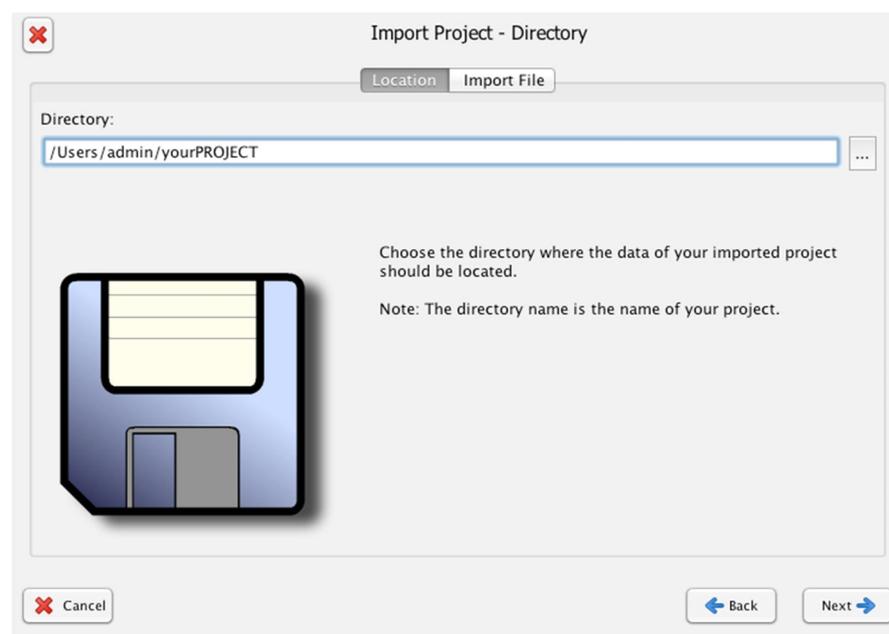
### Import Project

This option imports another project from an MEP file (all exported projects of *mySCADA Designer* use this extension).

Watch video describing this functionality:

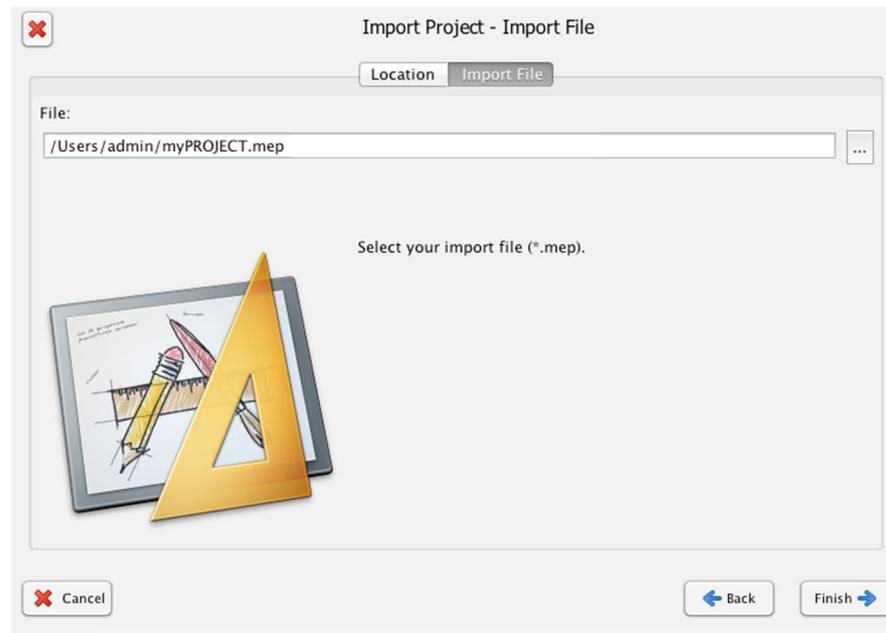
<https://www.youtube.com/watch?v=d8jVR75eGz0>

- 1) Choose the directory where the project to be imported should be located and click **Next**.



- 2) Click on **Import File** (suffix \*.mep) and then on the **Finish** button - an imported project will be created in the selected directory.
- 3) If the directory is not empty, all its files will be deleted!

- 4) Click **Finish** to finalize the import.

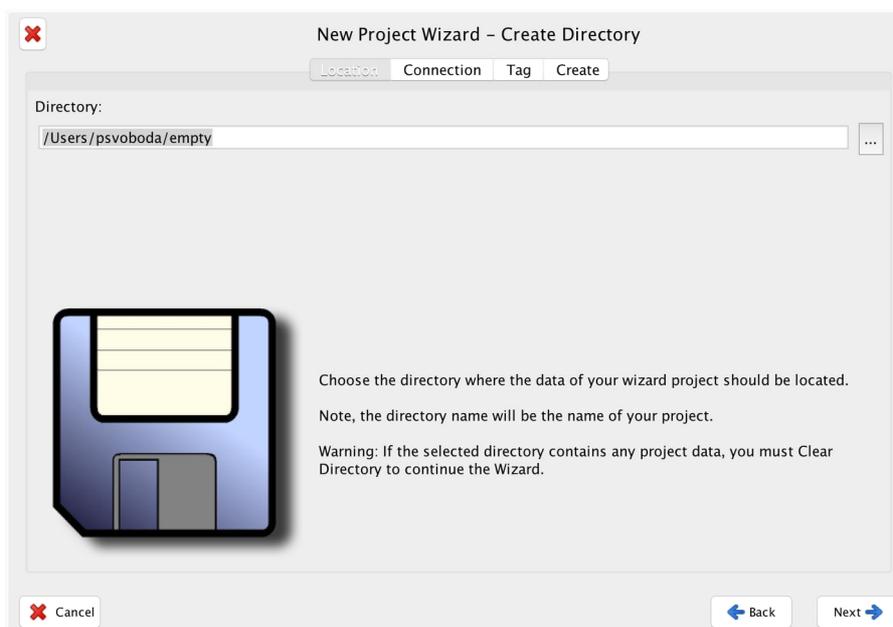


### Project Wizard

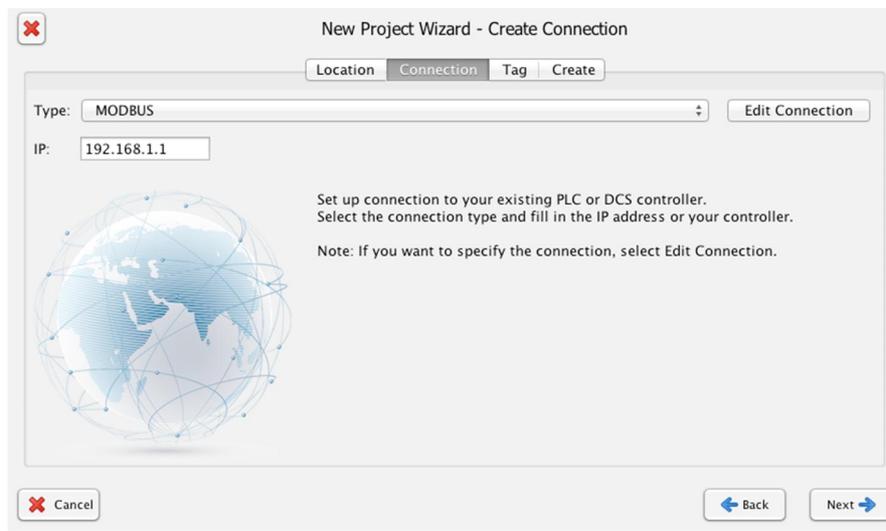
This feature helps you step-by-step to create a functional base for your new project. It will create a connection to the PLC, set up a simple screen with animations, and pre-configure alarms and data logging.

- 1) Select the directory where you want your new project to be located and then click **Next**.

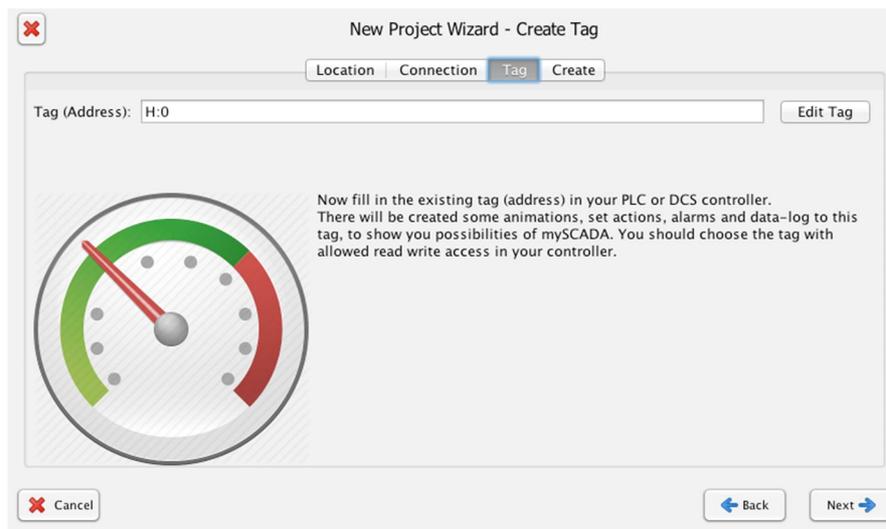
*Note: You have to click on **Clear Directory** first if the selected directory is not empty.*



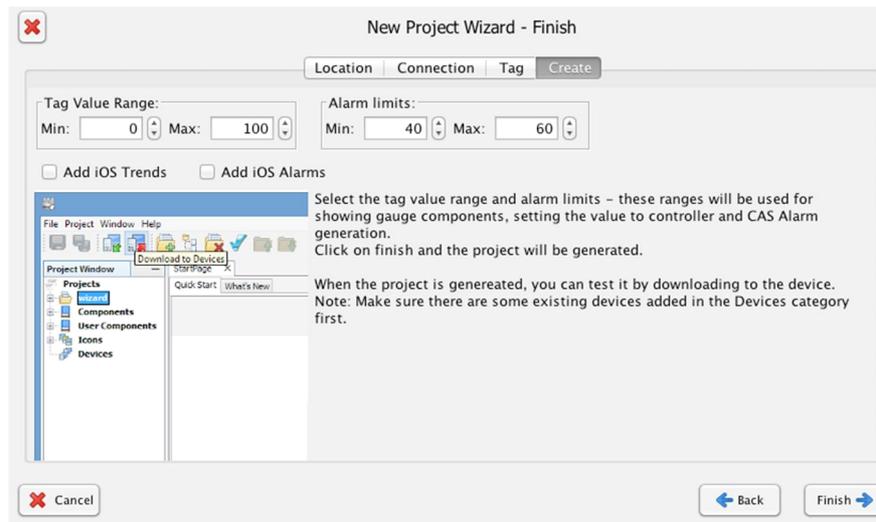
2) Set the connection to your device and click **Next**.



3) Set the Tag (Address) and click **Next**.



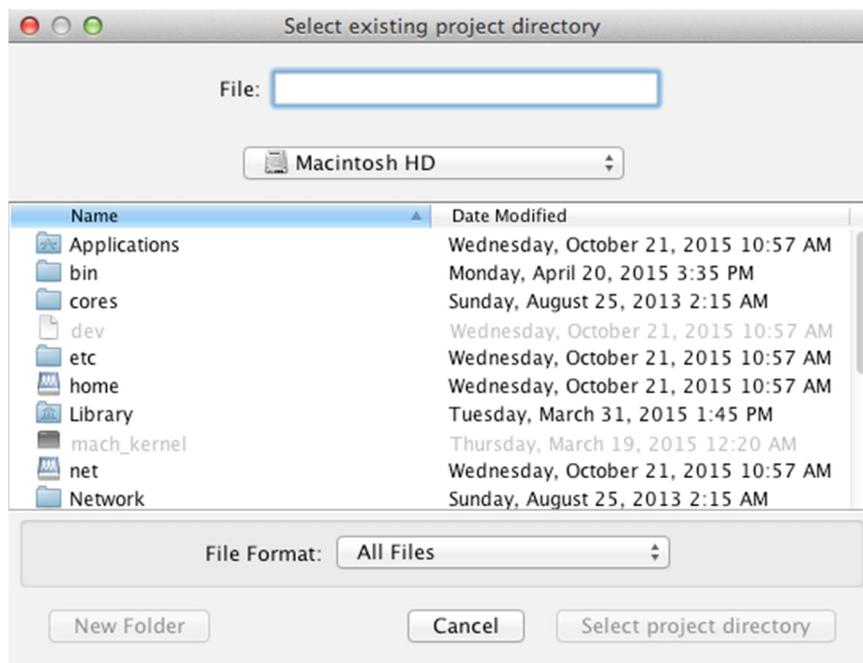
4) You can now set the Tag Value Range and Alarm limits.



5) Click on **Finish** to close the *Project Wizard*.

## Existing Project

With this feature, you can open an already existing project:



Navigate to the project directory and click on “select project directory.” If the directory contains project files, the project will be opened in myDESIGNER.

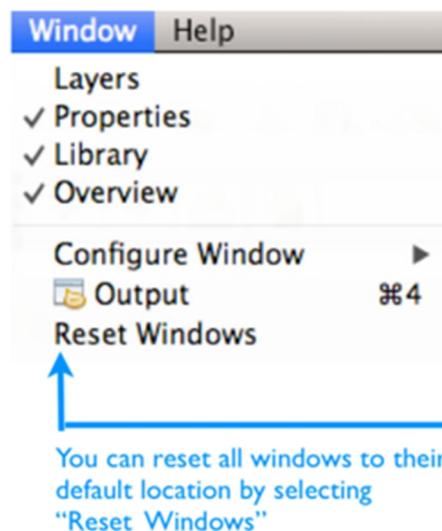
## 3.2 Workspace Management

All *myDESIGNER* windows are organized into panes. You can move the windows arbitrarily as the designer remembers the positions of both automatically and manually closed windows until the next time they are opened.

Each window can be dragged away from the workspace and will stay undocked until you dock them again using the key combination *Alt+Shift+D*. You may resize the windows as well.

### Resetting Windows Positions

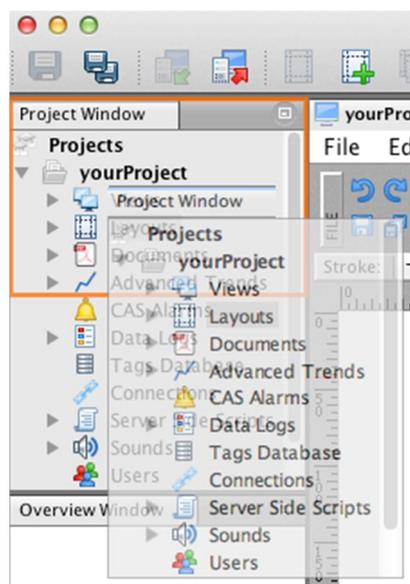
If you wish to return all the windows to their default state, use the *Reset Windows* command from the menu *Window* → *Reset Windows*.



### Moving windows

Click on the window header and drag it to the desired position,

*Note: The red preview box indicates where the window will appear once you drop it.*



## Automatically Appearing Windows

Some windows appear only when you are performing a task to which they are related. For example, the *Layers* window appears only when you are designing a screen view. In the *Window* menu, you can select which windows will appear automatically.

## Useful shortcuts

<b>Shift + Escape</b>	Maximizes a currently opened window
<b>Ctrl + Shift + W</b>	Closes all open documents in the <i>Source Editor</i> .
<b>Alt + Shift + D</b>	Pins a detached window to the Main window.

## Project Window

This window displays a tree structure of all opened projects and the list of devices running mySCADA.

### *Project (Your project name)*

- *Views* – all HMI screens in a project
- *Layouts* – all defined graphical layouts for given project
- *Documents* – PDF-documents attached to the project
- *Advanced Trends* – trends used with data logs for retrieving online and historical data
- *CAS Alarms* – here you can configure the *Complex Alarm System*
- *Data-logs* – configured data-logs
- *Tags Database* – managing your tags
- *Connections* – PLC connections for a given project
- *Server-side Scripts* – user-defined scripts
- *Sounds* – associated sounds in MP3 files
- *Users* – defined project users

### *Devices* – list of available devices from/to which you can load a project

Here you can find all *iPods*, *iPhones*, *iPad Touches*, *Android devices*, *Raspberry PI devices*, *mySCADA Boxes*, *mySCADA Pros*, and server PCs to which you can connect. You can also specify the device manually to access remote network devices. Note that it may take a few minutes to find all devices available in the local network.

## Overview Window

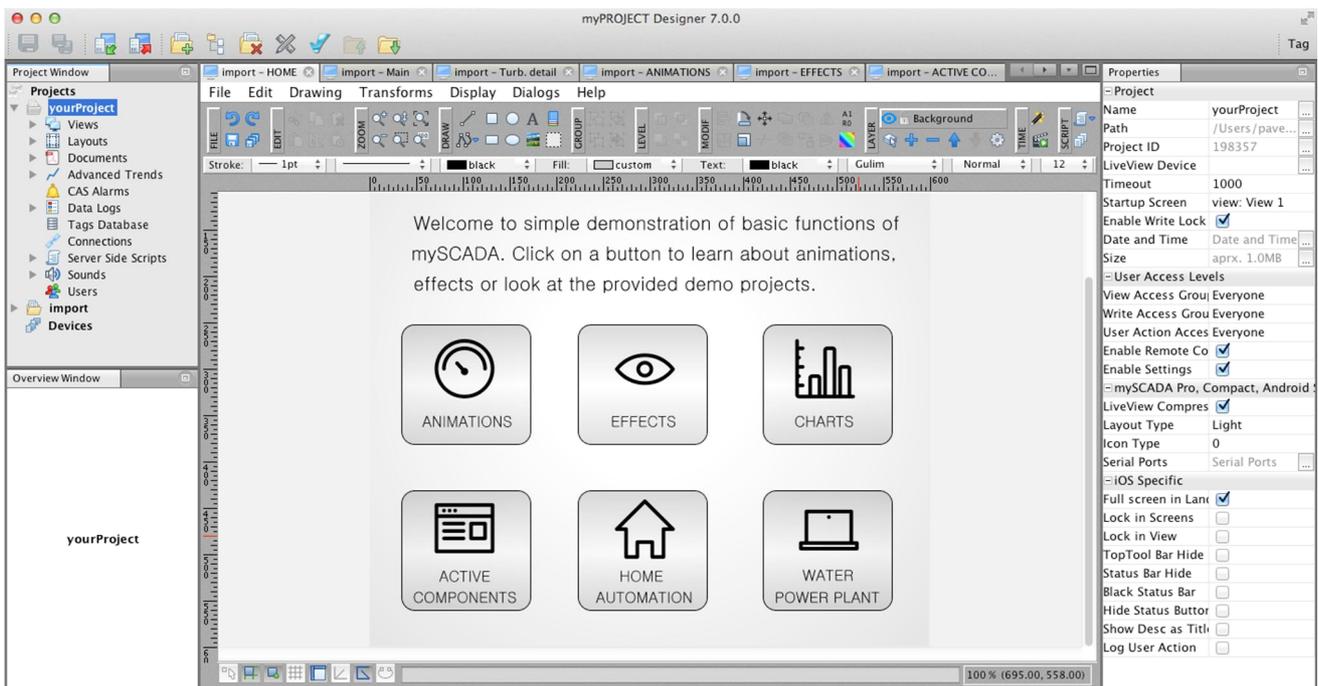
This window shows a preview of the selected view; otherwise, it is blank.

## Properties Window

This window allows access to object and project properties. The properties of specific objects are described in the relevant chapters.

## Main Window

This window is your main working area where the content of all project elements is displayed. It gives you the ability to draw graphic visualizations, change connection details, and alter alarms or trend details. The script editor also uses this window.



-  This button expands the list of all opened projects and selects one of them. The arrow next to the project name will point to the project that is currently being displayed.
-  This button maximizes the main window by docking all other open windows.
-  These buttons are used to switch between different opened tabs when the tabs do not fit into the main window.

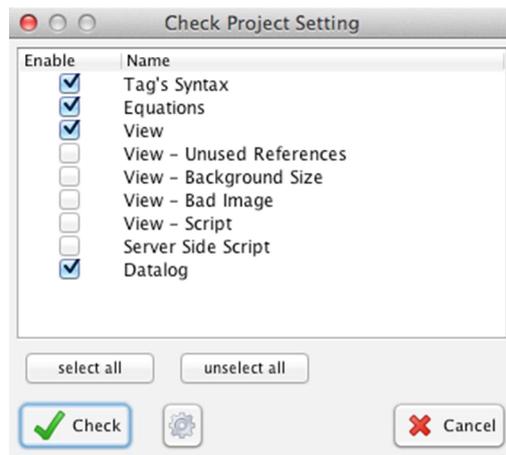
## 4 Checking Project

If you want to make sure your *mySCADA* applications run properly, you have to comply with certain rules when creating a project. For this purpose, you can use the *Check Project* function, which checks if your project complies with these rules.

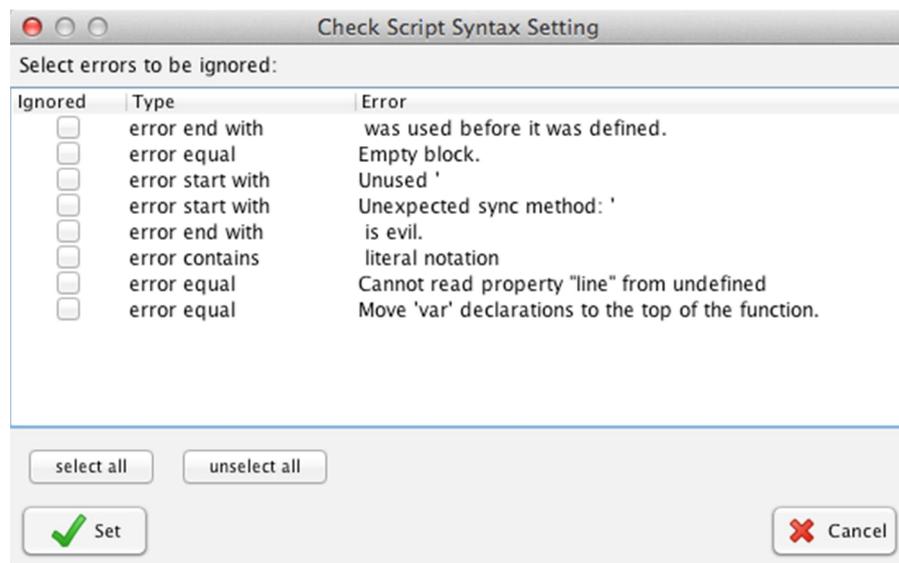
Use the command *Check Project* from *Project->Projects* or click on the **Check Project** icon in the main toolbar.

### Example:

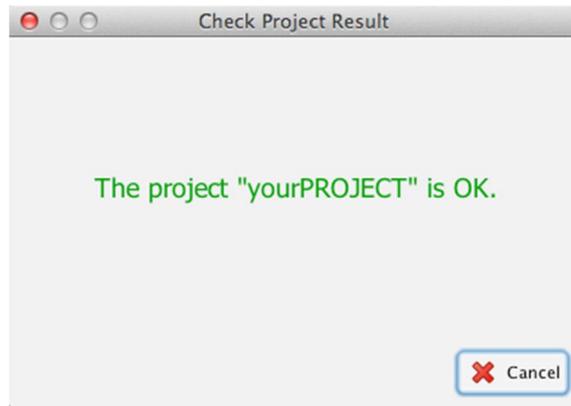
- 1) Select the parts of your project that require checking:



- 2) The **Syntax Settings**  button opens the error list, where you can set the errors to be ignored:



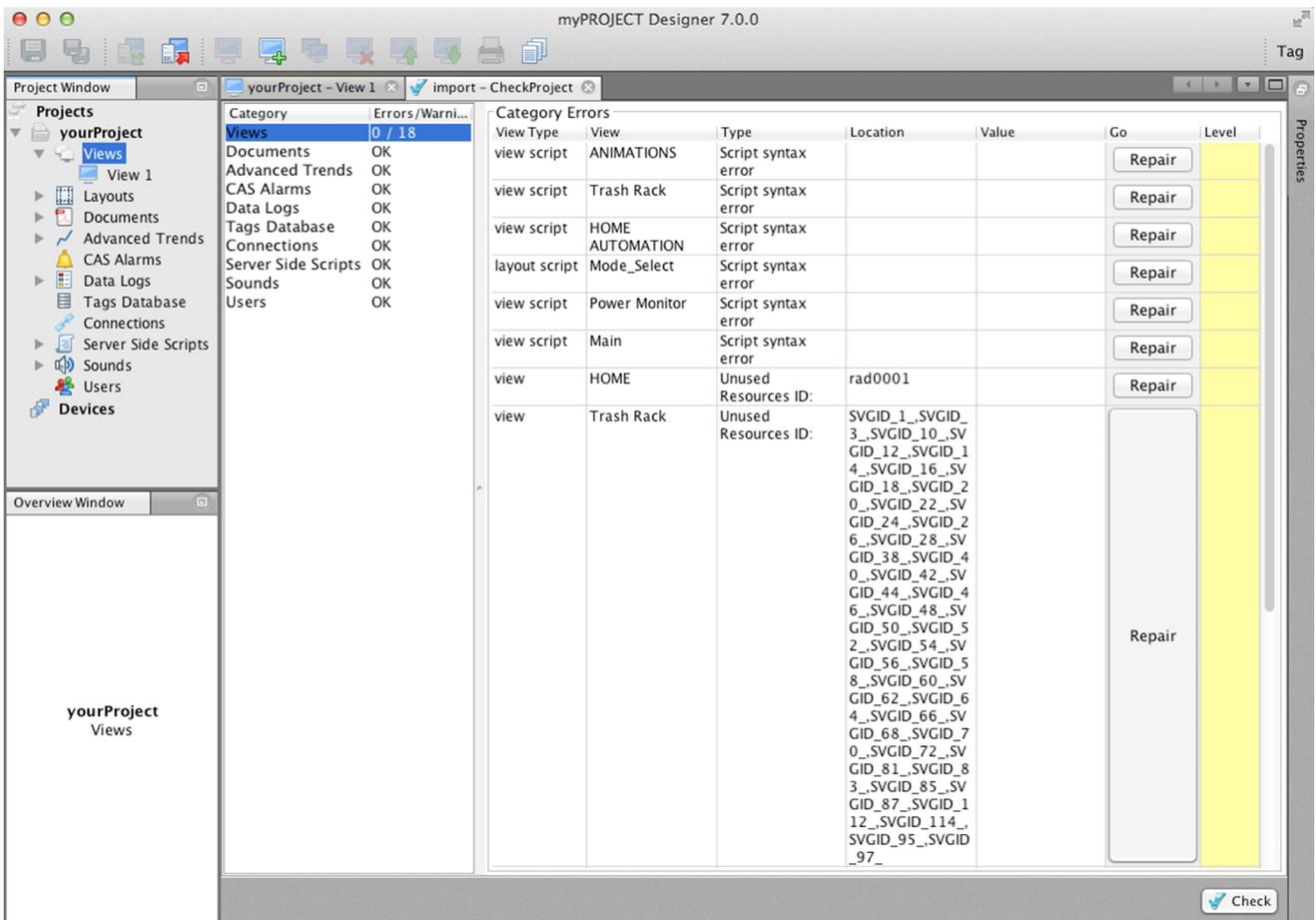
- 3) The **Check** button starts checking the project syntax. If the check is successful, a confirmation dialog box will show up:



If the check finds errors, an error dialog box will show up:



4) Click on the **Show** button to display all errors.



*Note: All errors are divided into categories on the left; selected error categories such as View, Documents, etc. are displayed on the right. If you wish to correct the errors that were found, click on **Repair**.*

## 5 Projects' Visual Appearance

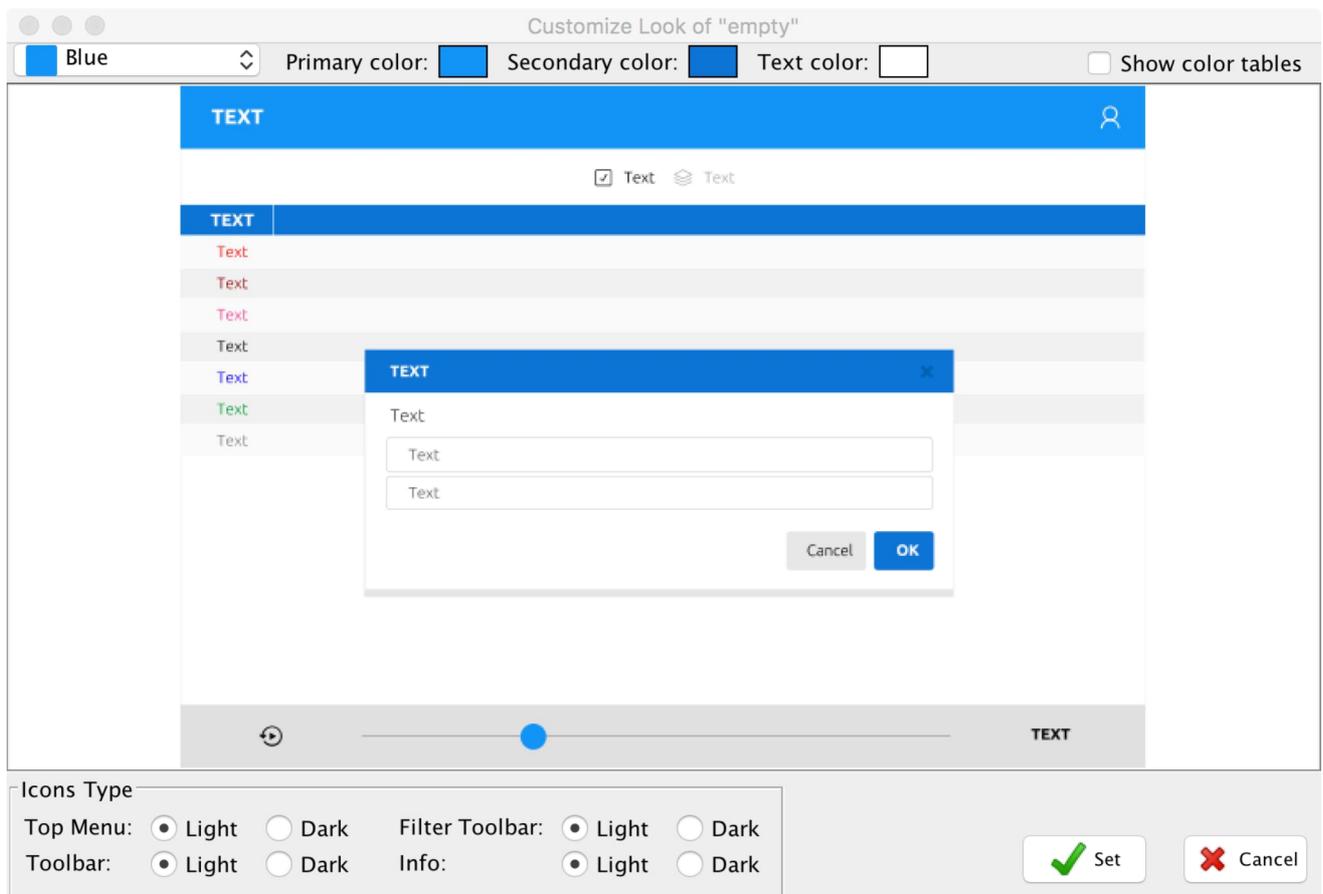
mySCADA is a very flexible tool that allows you to modify the visual appearance of your project in a runtime environment freely. You are free to change the fonts, colors, and icons to fit your company's visual guidelines.

Watch video describing this functionality: <https://www.youtube.com/watch?v=So78TI730KY>

For visual appearance set up, click on your project and then click on the color palette icon in the Main Toolbar.



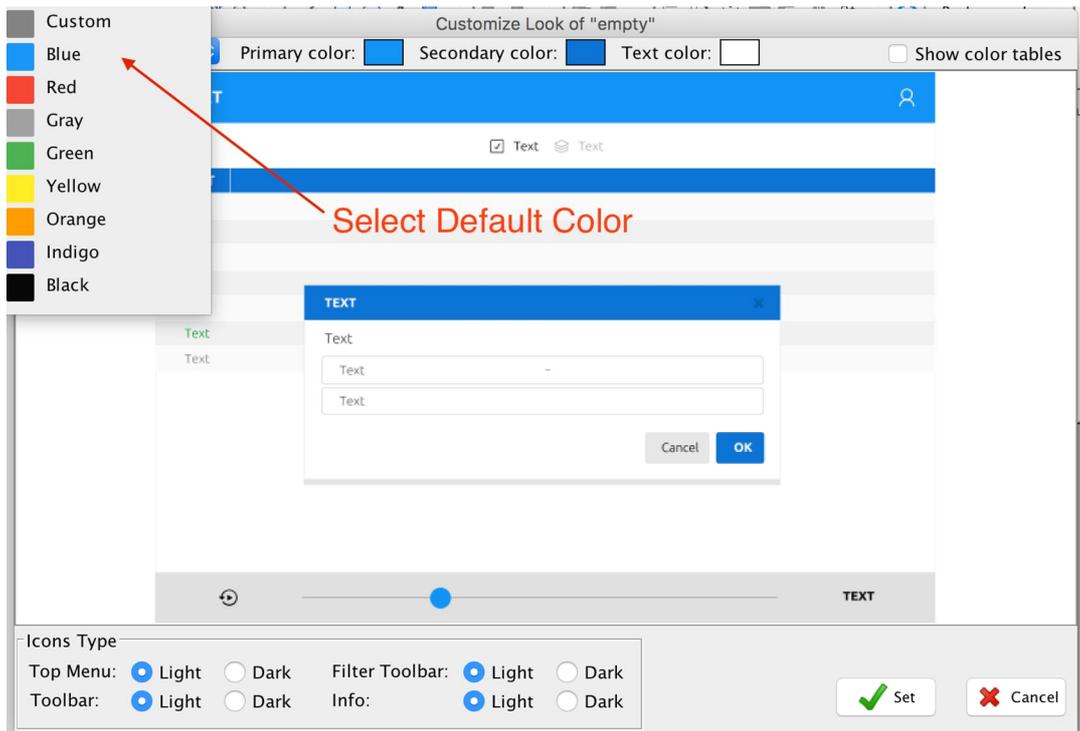
The Visual Appearance Dialog Window will show up:



In the dialog, you can see options for the visual appearance of your project in a runtime environment. If you change any color or visual option, you can immediately see how this option is displayed in the runtime environment.

### Changing the Default Color

The simplest way to change the default visual appearance is to set a new default color. To do so, click on a corresponding color in the "Default Colors" section.



The change is immediately reflected:

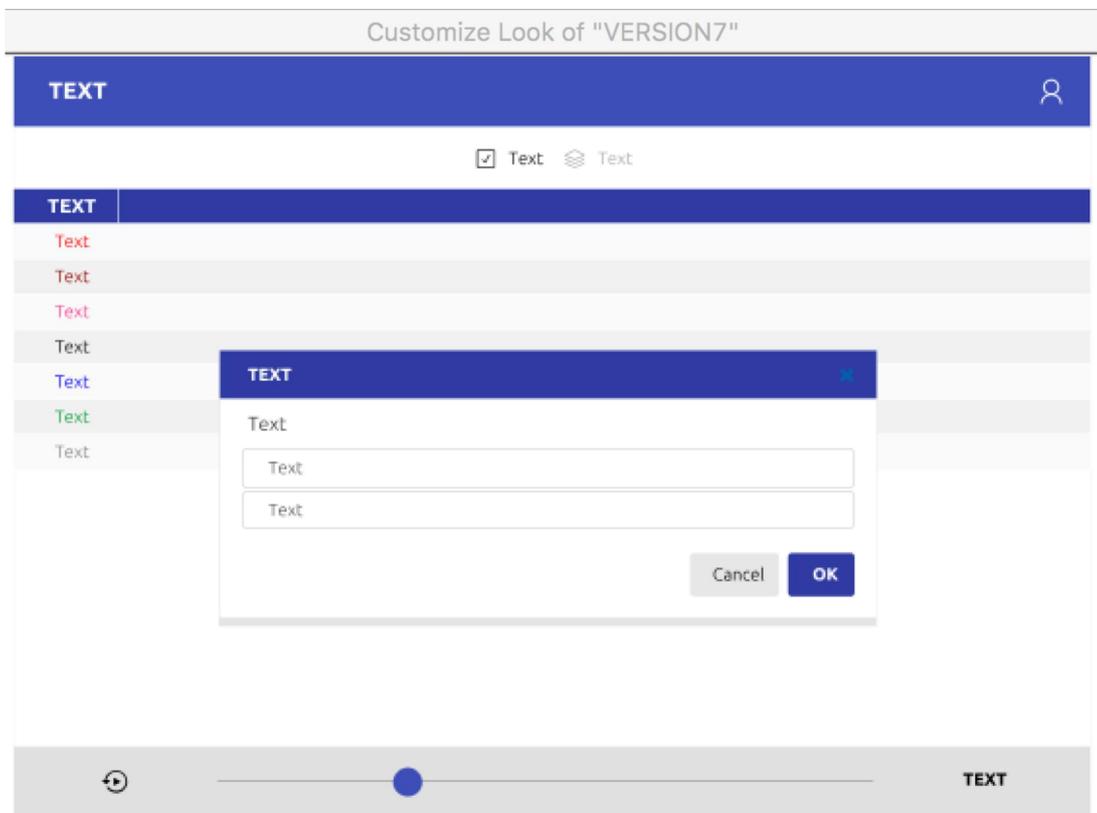
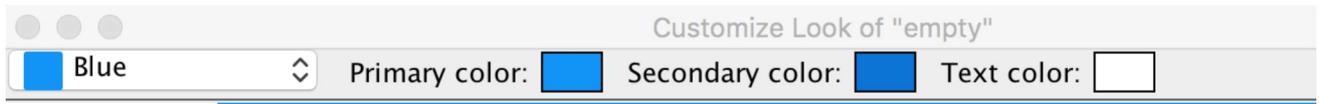


Figure 1 Default Color set to Indigo

### Changing Master Colors

You can very easily change the color scheme to any colors you like. Just click and change the Primary, Secondary, and Text colors.



Changes are reflected in all GUI sections.

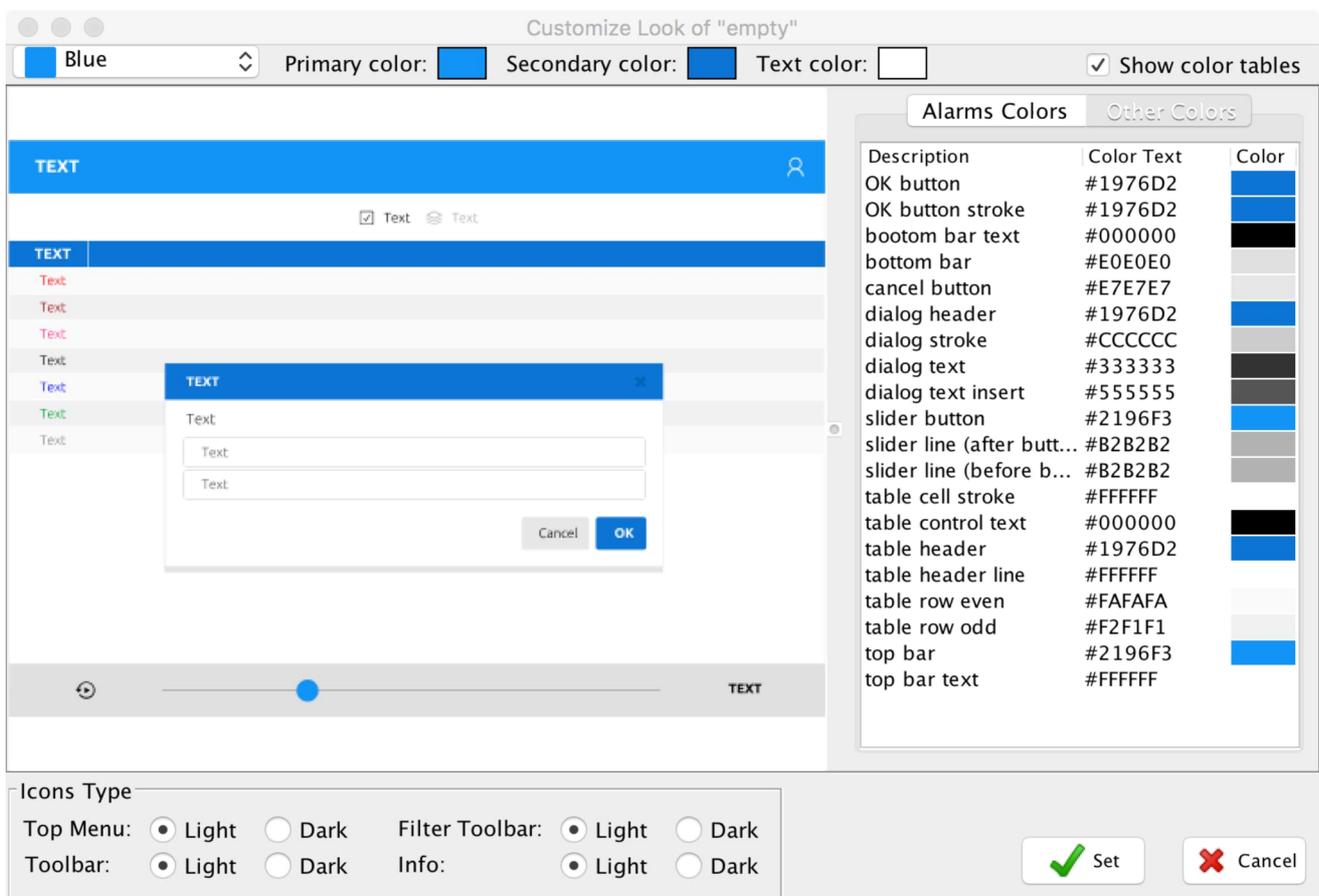
### Changing Icon Colors

You can change the Icon colors in the section "Icon Colors." For each visual section of the runtime environment, you can specify different icon colors.



### Fine Tuning

You can fine-tune the visual appearance of any on-screen element. To do so, click on the option "Show Color Tables." The color tables for your project are shown:

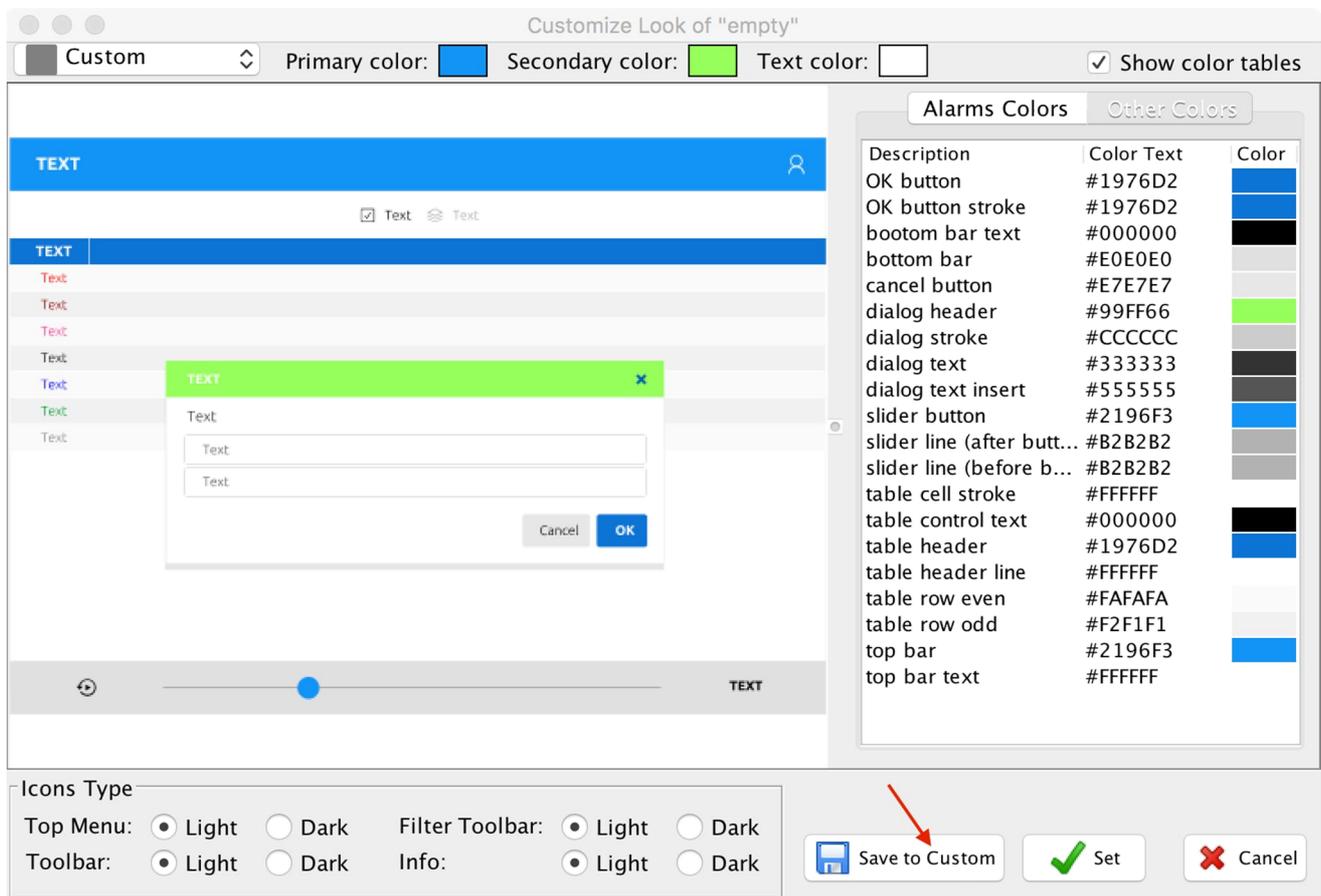


## Projects' Visual Appearance

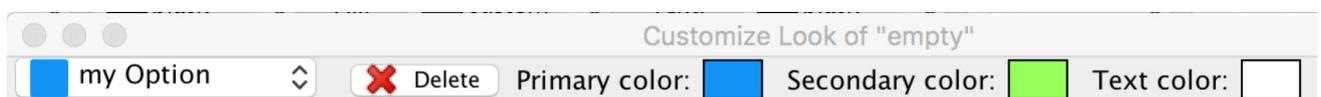
They are split into two parts: Alarms and Other Colors. In the Alarms section, you can set the colors of the text for any type of alarm. In the Other Colors section, you can fine-tune the appearance of almost any element in the runtime environment.

### Saving Custom Appearance

You can save your changes for use in other projects. When you change any color or parameter, the “save to custom” button is automatically shown.



When you press the “Save to Custom” Button, all your changes are saved under a given name. Your custom appearance settings are automatically added to the combo box selection:



You can delete your custom saved appearance by clicking the “Delete” button.

# 6 GUI/HMI Editor

The GUI editor is a drawing interface based on Scalable Vector Graphics (SVG).

## 6.1 Creating Views

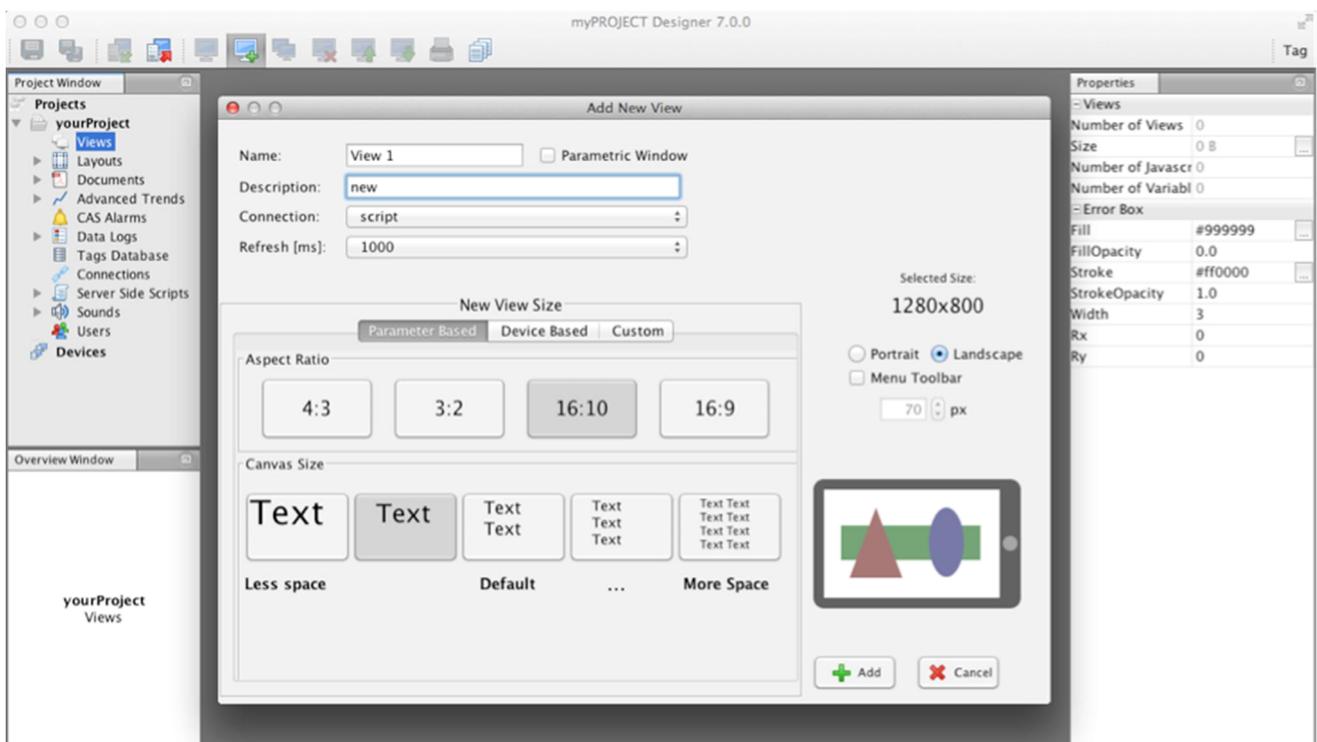
Watch video describing this functionality:  
<https://www.youtube.com/watch?v=HtLLUt3O5dc>

To create views for visualization, open the **Views** folder in the *Project Window*; the main toolbar will show you these options: *Open View*, *Add View*, *Duplicate View*, *Delete View*, *Print View*, and *Show Used Tags*.



- 1) Click on the **Add View** icon located above the main screen or use the command from the main menu *Project->Views->New View*.

A new dialog window opens to allow you to set the visualization parameters:



**Name:** view name

**Description:** additional comments about visualization

**Parametric Window:** option to create a parametric view - use parametric views when you need multiple views different only in a data source; such sources are specified by the received index when a parametric

view is prompted (see *Commands - Open command* and *Parametric window*).

**Connection:** list of PLC connections defined in the current project. This is a default connection - if you enter the tag and do not specify the connection, this connection will be assumed as the *default*.

**Refresh:** refresh rate of the visualization; time in milliseconds

**Page Orientation:**

- *Portrait* predefined vertical display
- *Landscape* predefined horizontal display

*Note: If your device uses a pop-up menu toolbar (iPhone, Smartphones, etc.) you can set up its width in pixels, and it will be automatically subtracted from the final screen size.*

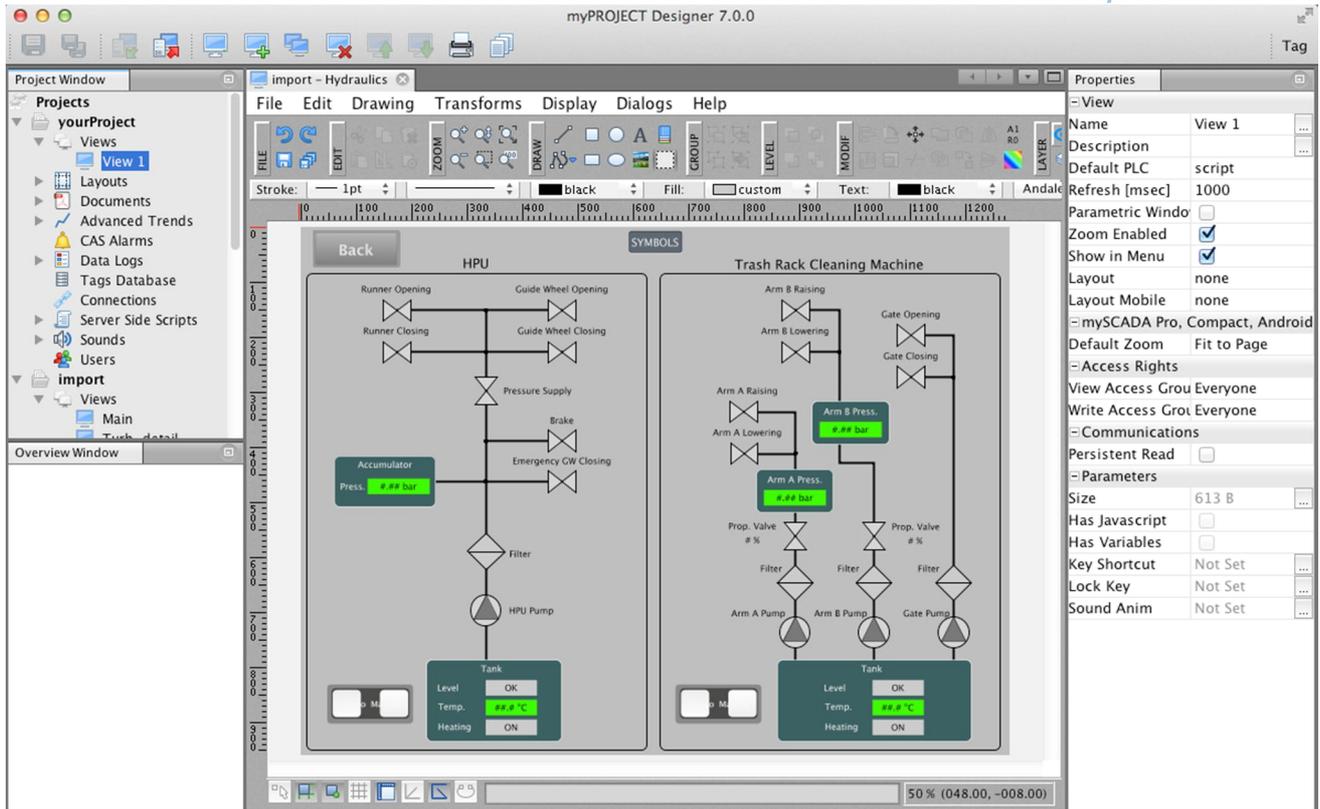
**New View Size:**

- *Parameter Based* - here you can set the aspect ratio and the canvas size
- *Device Based* - variations of pre-defined settings for many types of devices
- *Custom* - allows you to define your own view settings

*Note: You can change the screen dimensions at a later time if you right-click on the canvas and select 'Resize canvas' from the pop-up menu.*

- 2) Click on **Add** to create a view with the selected parameters or press **Cancel** to exit without creating or saving the view.

In the *Project Window*, you can see all the views listed in the **Views** folder. In the *Properties window*, you can see the parameters of the currently selected view. To open an existing view, simply double click on its name. You can also click on the **Open View** icon or on select *Open View* from the *Project->Views* menu. You may also use the other options such as *copy, delete, import a view, etc.*



## Menu

File Edit Drawing Transforms Display Dialogs Help

## File

This menu allows you to manipulate an opened view:

- Export** – exports the current view into other formats: *JPG, PNG, BMP, and PDF*
- Print** – prints the current file (shortcut **Ctrl+P**).
- Check** – performs a check of your view; use this if you experience malfunctions, and it will check for possible duplicity of IDs, objects misplaced outside of the canvas, etc.
- Close** – closes the current file (shortcut **Ctrl+W**).

## Edit

This menu provides basic operations like *undo, redo, copy, cut, delete, group, lock, select, etc.*

- Undo** – erases the last change made in the program (shortcut **Ctrl+Z**)
- Redo** – reverts the change (shortcut **Ctrl+Y**)
- Copy** – creates a copy of a selected object or area on the clipboard (shortcut **Ctrl+C**)
- Paste** – pastes the object or area from the clipboard (shortcut **Ctrl+V**)
- Paste on the same location** – pastes the copied or cut object to the same location as the source.
- Cut** – cuts a selected object and places it on the clipboard (shortcut **Ctrl+X**)

*Delete* - deletes a selected object (shortcut **Ctrl +D**)

*Paste dimensions* - transfers certain parameters of a selected object:

- *Paste size* - modifies selected objects, setting the size to that of the object being copied
- *Paste width* - modifies selected objects, setting the overall width to the size of the object being copied
- *Paste height* - modifies selected objects, setting the overall height to the size of the object being copied
- *Paste size separately* - resizes selected objects one by one
- *Paste width separately* - changes the width of selected objects one by one
- *Paste height separately* - changes the height of selected objects one by one

*Group* - groups multiple objects together (shortcut **Ctrl+G**)

*Ungroup* - splits the grouped objects

*Enter group* - allows access to individual objects inside an object group (shortcut **Ctrl+E**)

*Exit group* - returns to normal selection mode, i.e. the whole group will be selected at once (shortcut **Ctrl+Shift+E**)

*Select all* - selects all objects on the canvas (shortcut **Ctrl+A**)

*Deselect all* - deselects all objects on the canvas

## Drawing

In this menu you can create figures, lines, and curves; add pictures and texts; etc.

It contains the same features as the main *toolbar*; for detailed information, see the *toolbar* section of this chapter.

## Transforms

In this menu you can align, center, distribute, order, flip, and rotate objects and resize the canvas.

## Display

In this menu, you can enable grid and rulers; different zoom options are available too.

## Dialogs

This menu includes the *Memory Monitor utility*.

**Memory Monitor** - shows the amount of memory being used by *myDESIGNER*

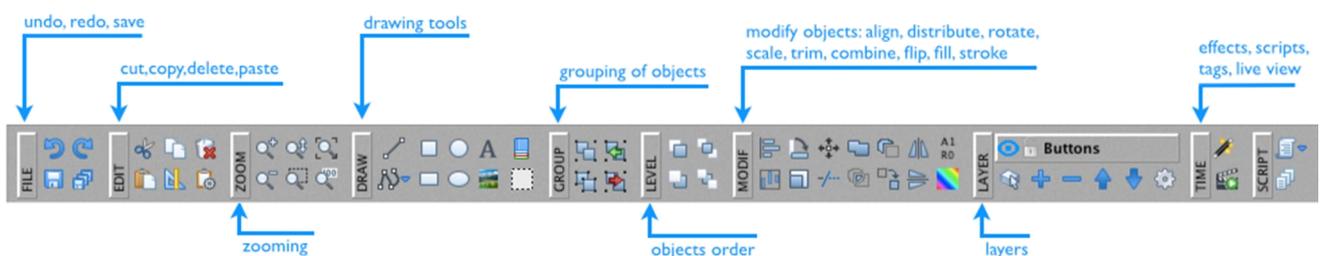
## Help

This menu offers additional information about *myDESIGNER*.

*TIP: The most frequently used functions from these menus are also available in the main toolbar or in the right-click menu.*

## GUI Toolbar

In the upper part of the GUI editor window, there is a secondary GUI toolbar where you can find all of the necessary functions for designing and animating the views.



## Properties Bar

This bar is located right below the GUI toolbar. It allows you to view and change the properties of selected objects or set default drawing properties such as *fill and stroke color; line type and line thickness; font family, type, and size*.

If there is no object selected, you can specify the default properties for new objects to be created. If you click on an existing object, you will see its current properties that you can modify.

In the following example, all of the new objects will have transparent fill and black solid line stroke:

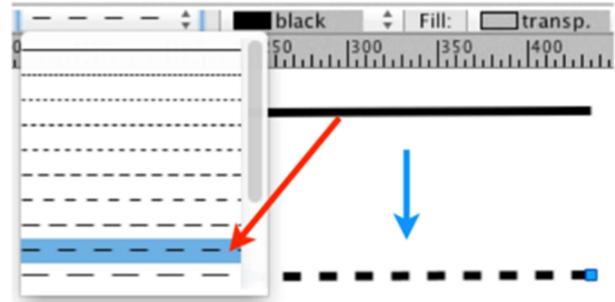


With the *Properties bar*, you can easily change the properties of multiple objects at once: stroke width, stroke style, stroke color, fill color, and also the font family and size.

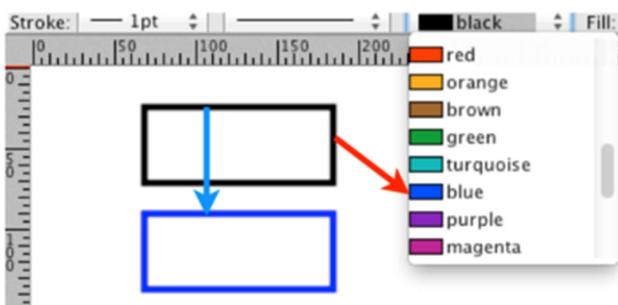
**Stroke width:**



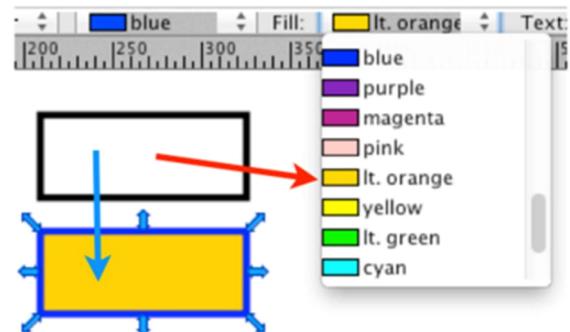
**Line style:**



**Stroke color:**

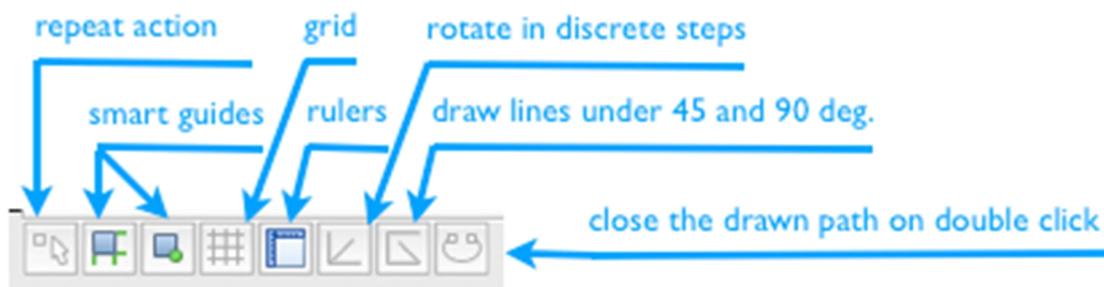


**Fill color:**



**Options Bar**

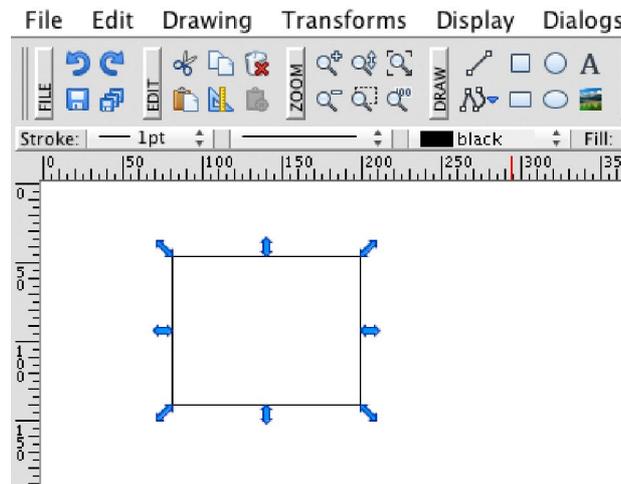
This bar is located in the left lower part of the *Main window*.



Here you can activate the *repeat action mode*, turn on/off the *smart guidelines*, *snap-to-points*, and *square mode* function; enable the *grid*; set the *discrete rotation step*, force horizontal/vertical line drawing; and automatically *close the drawn paths*. Detailed explanations of each function will follow.

## 6.2 Selecting Objects

Click on the canvas and drag the mouse to select object(s). If the objects are selected correctly, the selection border shows up and the object properties will be displayed in the *Properties window*. To add objects to your current selection, press and hold the **Shift** key and click on more objects.



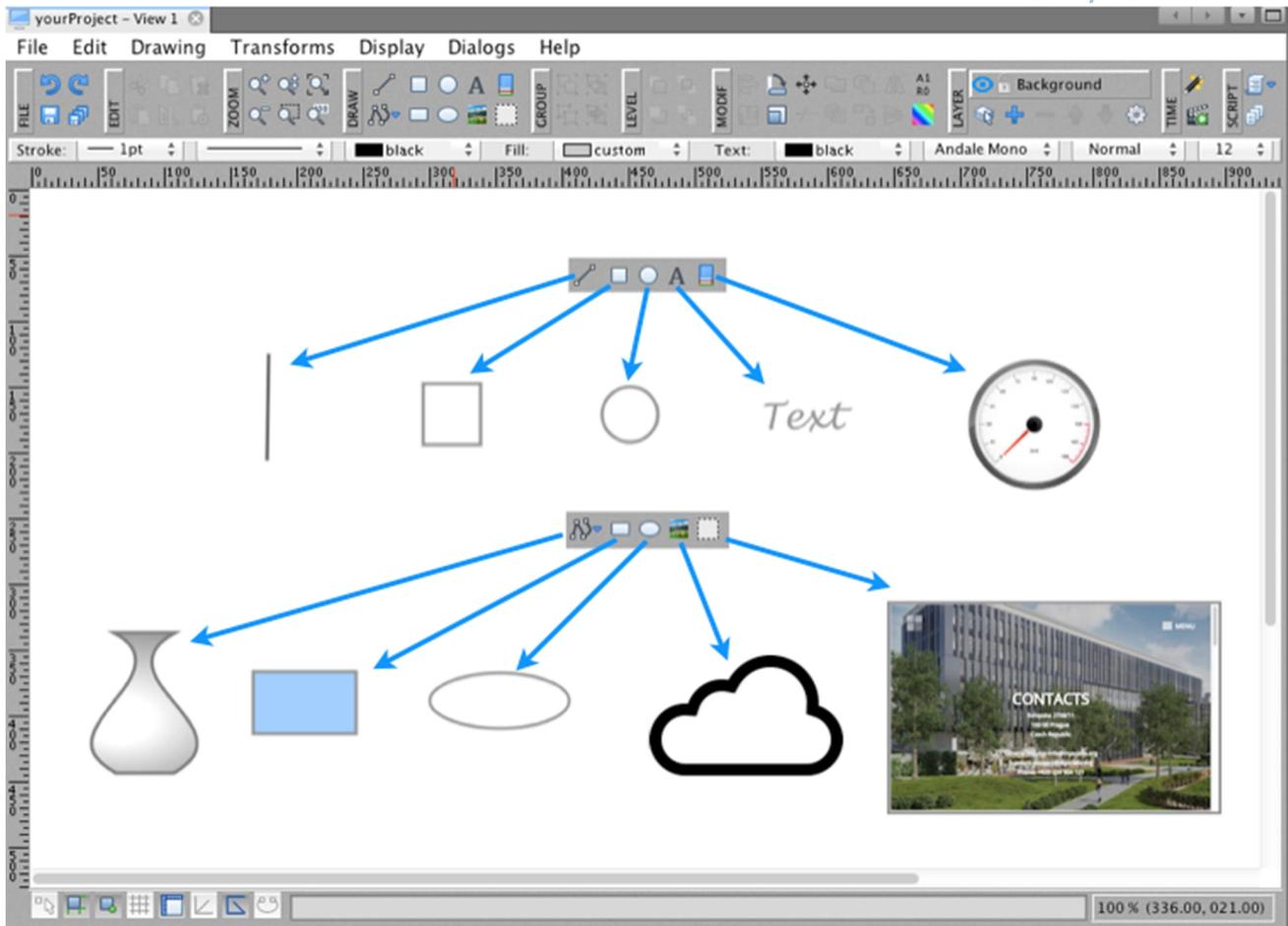
*Note: If you drag the mouse from the left to the right, only the objects fully enclosed in the selection window will be selected. If you drag the mouse from the right to the left, all the objects that intersect with the selection window will be selected.*

## 6.3 Drawing Primitives

Watch video describing this functionality: <https://www.youtube.com/watch?v=ryduKCgzFr0>

In the DRAW section of the GUI toolbar, you can choose the type of object you want to draw (*line, polyline, square, rectangle, circle, ellipse, text*) or insert an active area window, picture, or ready-made components.





## Rectangles and Squares

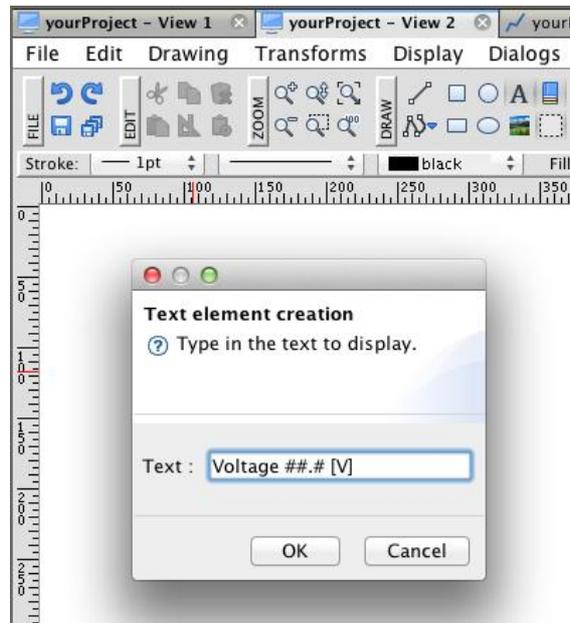
-  To create a rectangles or squares, click on the **Create Rectangle** icon in the GUI toolbar (you can also find it in the menu under *Drawing*). Click on the canvas, hold, and drag the mouse to get the desired shape and size.

## Ellipses and Circles

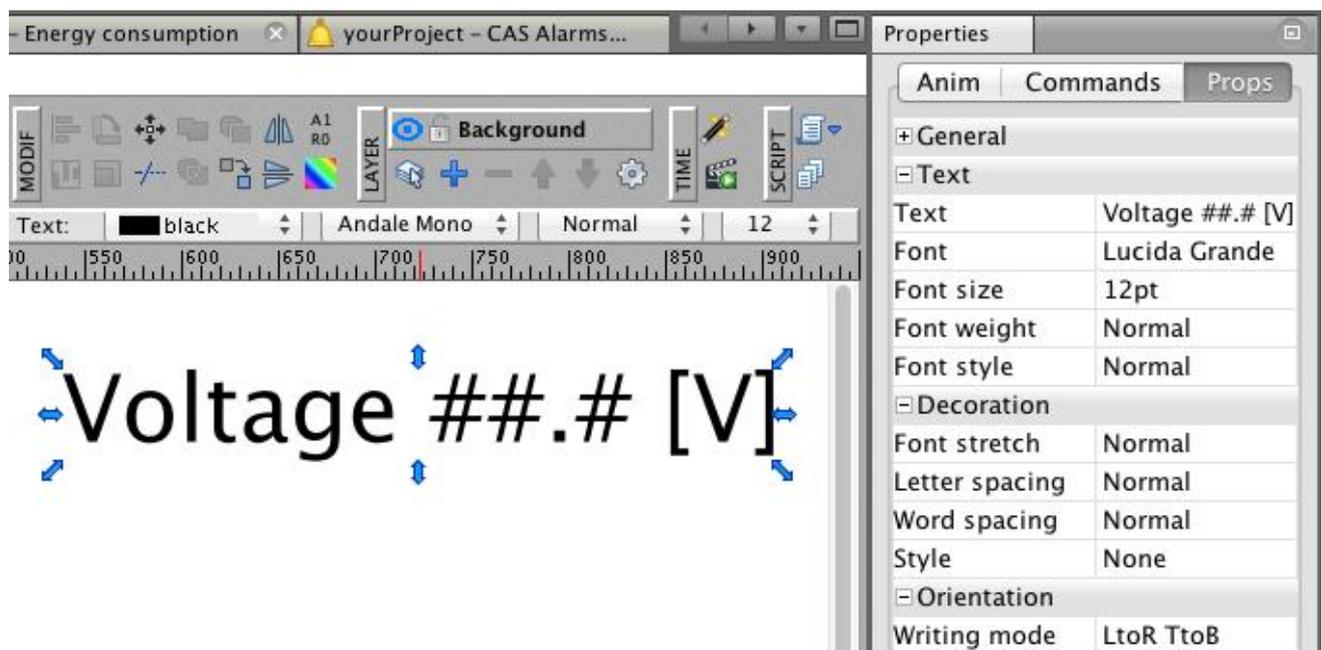
-  To create ellipses or circles, click on the **Create Ellipse** icon in the GUI toolbar (you can also find it in the menu under *Drawing*). Click on the canvas, hold, and drag the mouse to get the desired size and shape of the object.

## 6.4 Creating Text Elements

- A** This feature allows you to add text on the drawing canvas. Click on the **Text** tool icon in the main toolbar and then click on the canvas in the desired location. This will open a new window for the text input.



Type in the text to be displayed and click on **OK**. The text is now placed on the canvas and can be moved, resized, rotated, etc., like any other object.



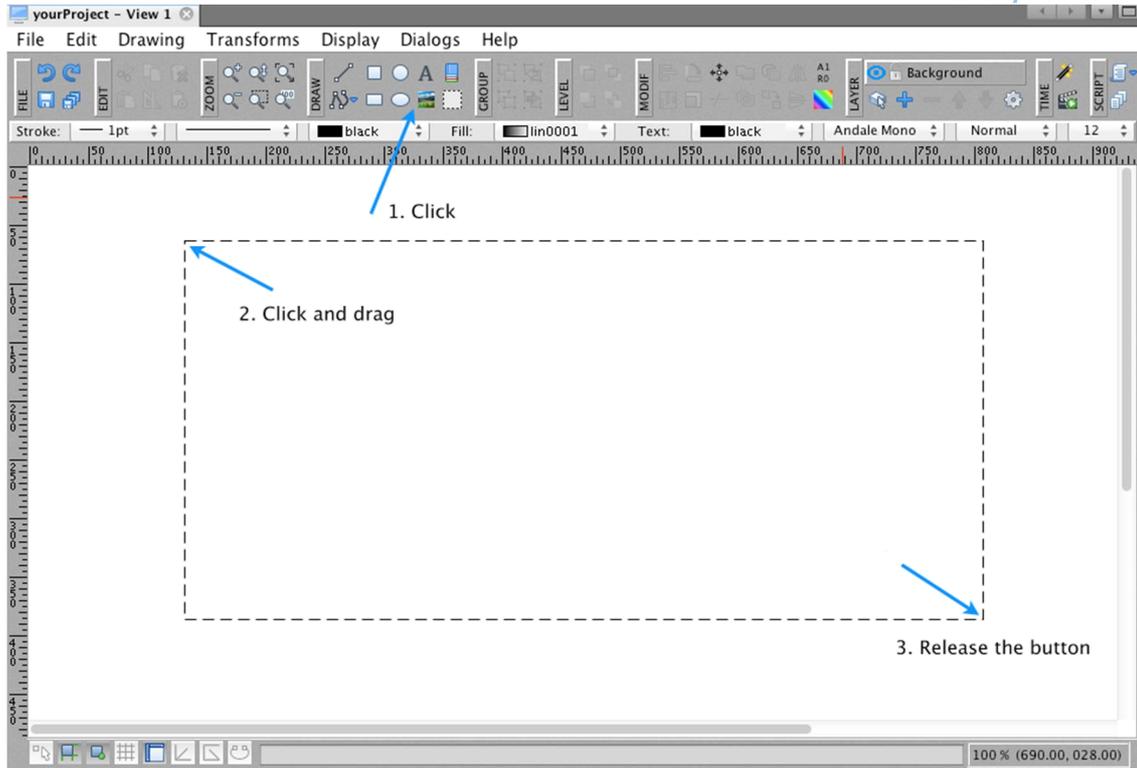
*Note: The text in the element can be changed at a later time - select Properties on the right side of the window and look at the Text cell.*

## 6.5 Insert Image

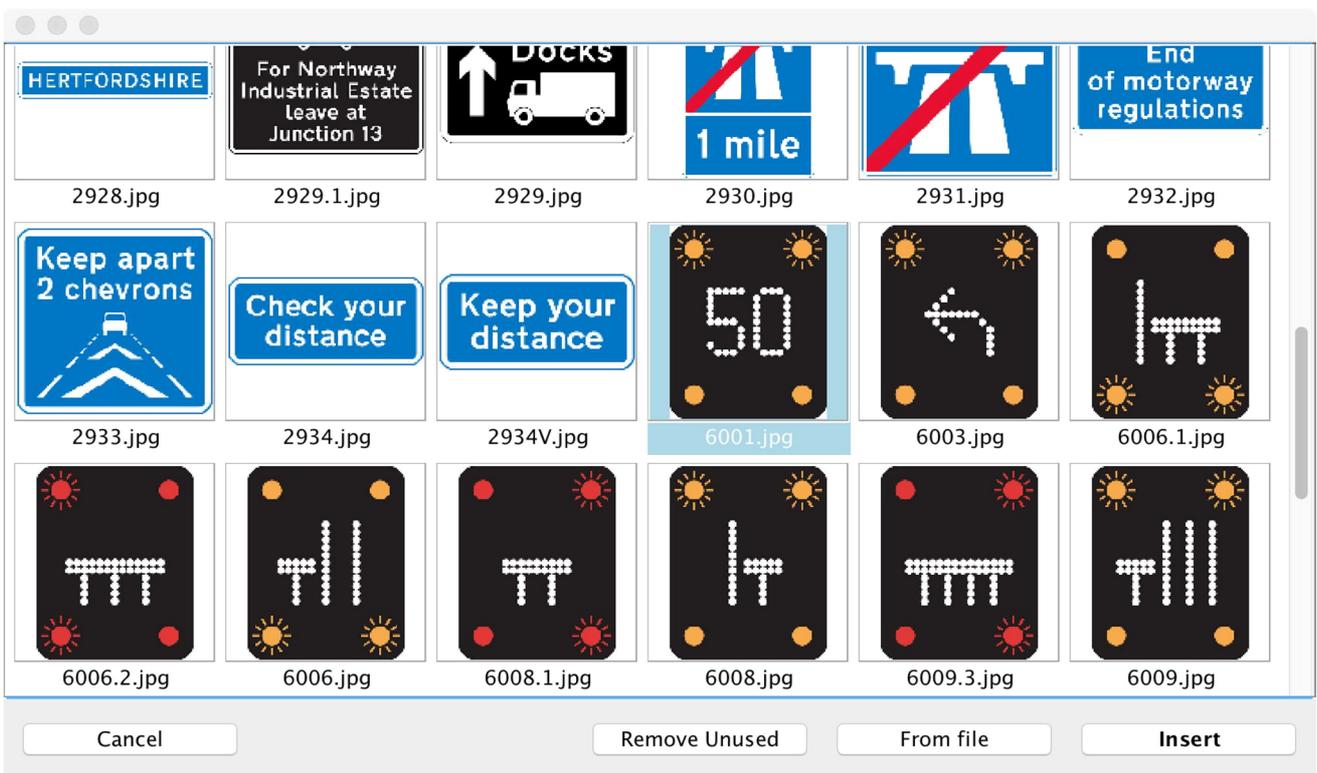


With this function, you can insert images in *PNG, JPG, and JPEG* formats.

Click on the **Insert Image** icon in the main toolbar, then click on the canvas, drag the mouse to set the picture size, and release the mouse button.



This will open a dialog window to select the picture.



In this dialog, you can see all pictures that you have used in your project. You can add new pictures by clicking on the *From File* button.

*TIP: When you want to remove pictures that are not used anywhere in your project, you can use the button "Remove Unused".*

If your project does not have any imported pictures yet, an open File dialog will be shown. If you already have pictures in your project, they will be listed in the Picture Chooser Dialog. Either select pictures that have already been imported or import new ones by clicking on the button “From file”.

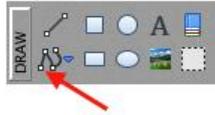


The picture is now on the canvas and can be moved, resized, rotated, etc. like any other object.

*Note: You can import images in their original size by a single-click on the canvas.*

## 6.6 Poly-lines

With this function you can create independent and continuous lines, arcs, splines, or any combination of these elements.



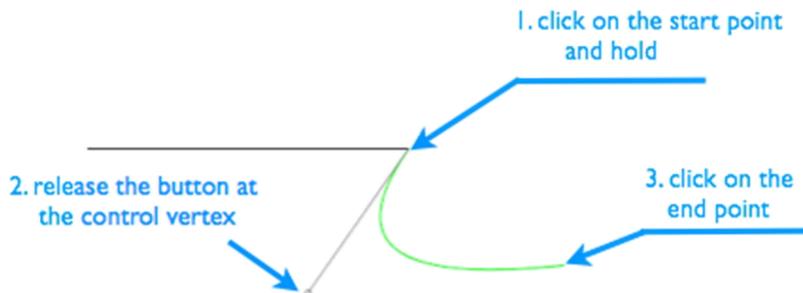
### Line

Click to set the start point, move the mouse to the end point of the line, and then click again.



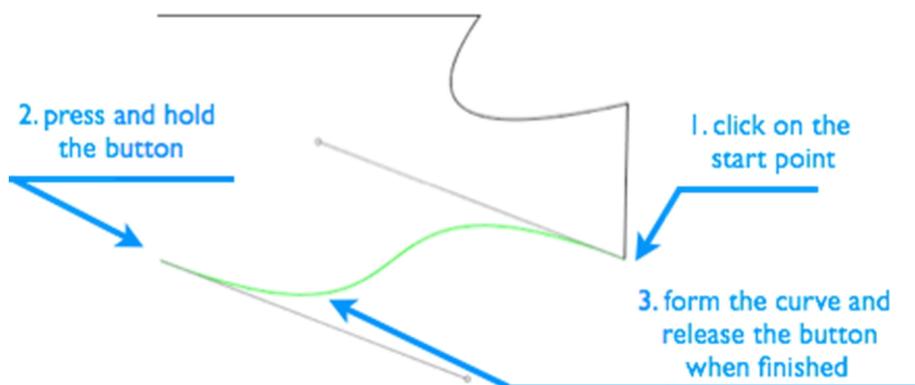
### Arc

Click and hold the mouse button while moving from the start point, release the button at the control vertex point, and then click again to set the end point.

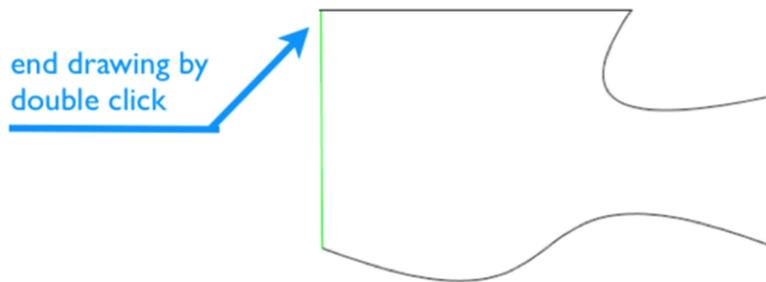


### Spline

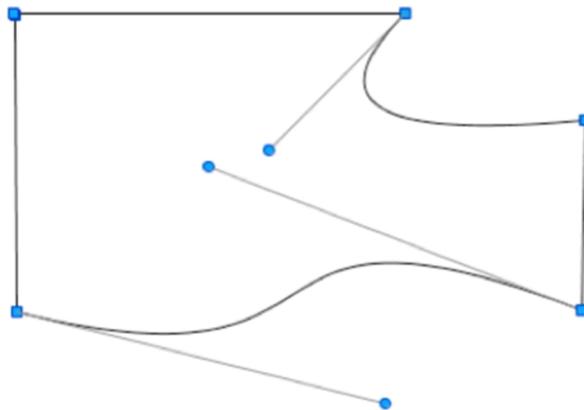
Click on the start point and move the mouse to the endpoint, then click again and hold down the mouse button while forming the curve, and release the mouse button to finish the drawing.



Double-click or press the ESC key to finishing the polyline drawing.

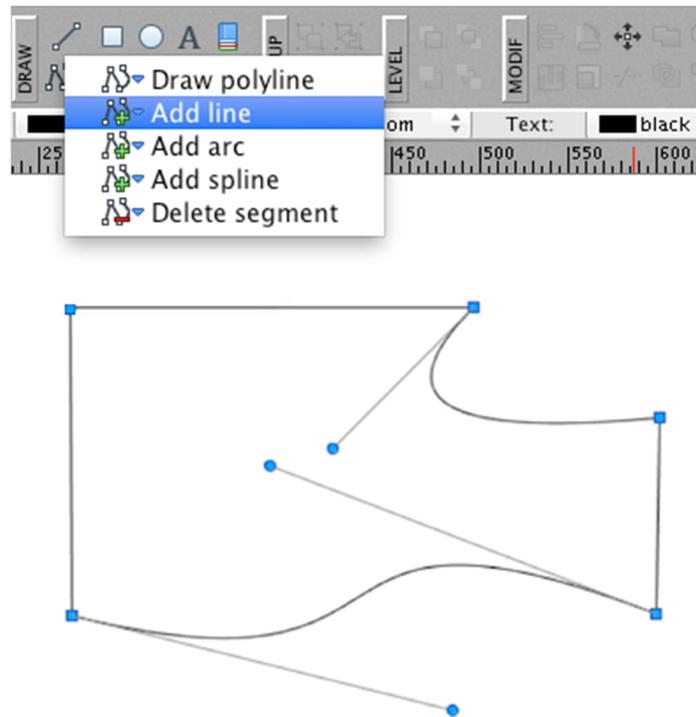


When you finish drawing the object, you will see all the control, end, and vertex points, which you can modify.



### Editing Segments

You can add, remove, or delete segments from a spline you have created. Click on the arrow on the right side of the **Create Spline / Bezier Curves** icon (you can also use the right-click) to display the *Options menu*.



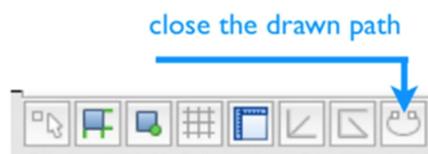
### *Add segment*

Select a segment you would like to add from the options menu. Click on the control point of the polyline where you want to add the new segment.

### *Delete segment*

Select *Delete segment* from the options menu. Click on the end control point of the segment you want to delete.

### *Close path*



If this option is active, all the polylines being drawn will close automatically.

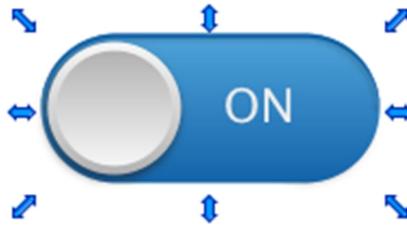
## 6.7 Moving Objects

Select the object(s) you want to move. Move the mouse inside the selection border, click and drag the object to the desired position, and then release the mouse button.

## 6.8 Resizing Objects

You can resize object(s) by dragging the selection border.

- 1) Select an object you want to resize – you will see the blue arrows selection border.



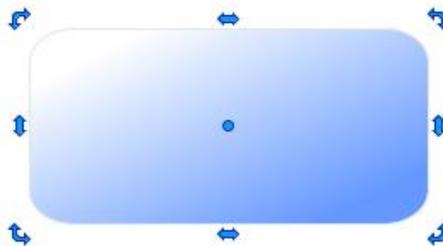
2) Drag the arrows to change the size of the object.

If you drag using the arrows at the corners, the object will retain its aspect ratio. If you use the arrows on the sides, the object will change its horizontal / vertical size only.

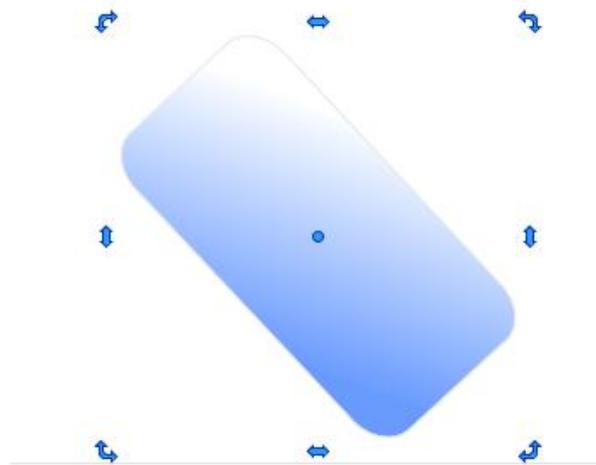


## 6.9 Rotating Objects

1) Double click on an object you want to rotate; blue arrows will appear around the selected object.



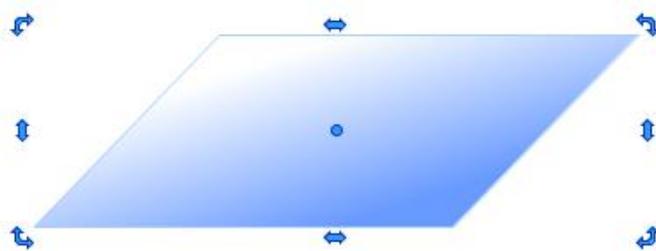
2) Drag the corner arrows until you get the desired angle of rotation.



*Note: You can also change the center of rotation for the selected object. To do so, move the blue circle located at the geometric center of the object.*

## 6.10 Skewing Objects

- 1) Double-click on the object you want to skew
- 2) Drag the blue arrow in the middle to get the desired skew.



## 6.11 Filleting

The rectangles and sq

- 1) Triple click on an object to show the blue point in its corner.

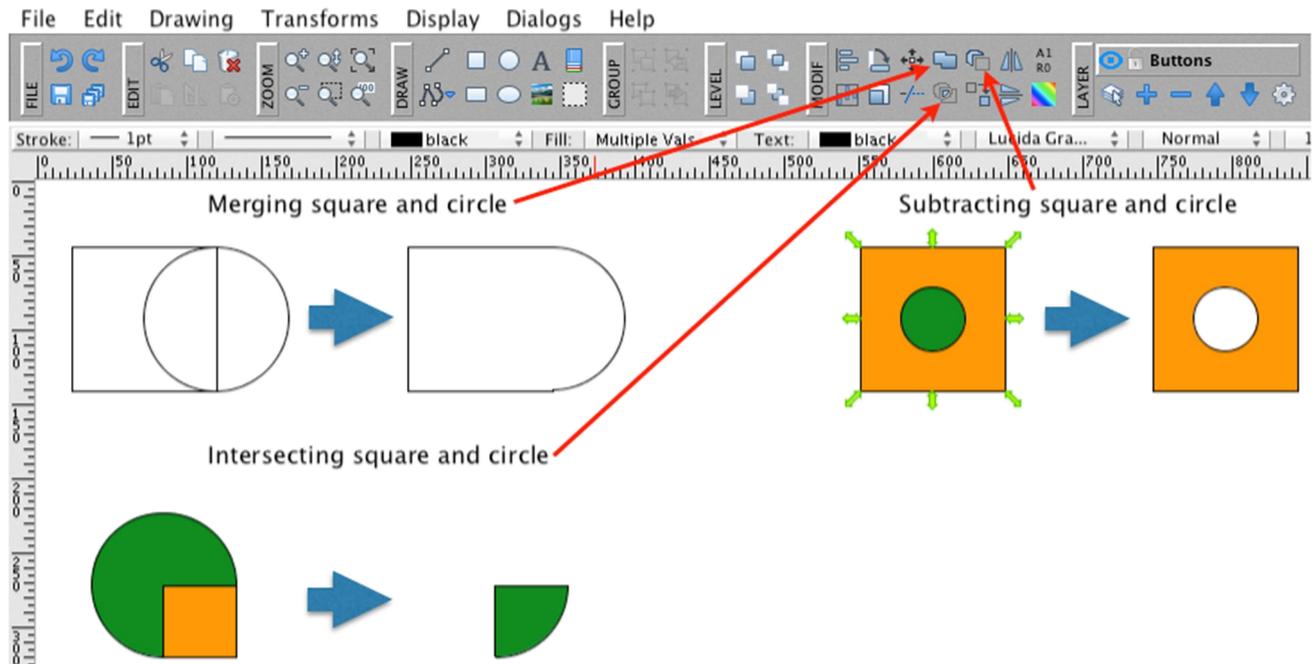


- 2) Click on the blue corner point, hold, and drag the mouse to get the desired shape.



## 6.12 Combining Objects

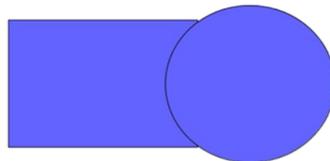
You can combine multiple simple objects that overlap; together with the *Merge*, *Subtract*, *Intersect*, and *Turn-to-Path* functions, you can create one complex object of arbitrary shape and color.



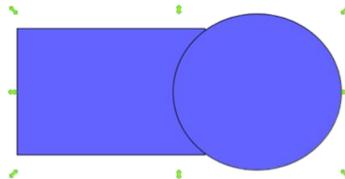
### Merge

This feature can merge two objects into one, including the contours:

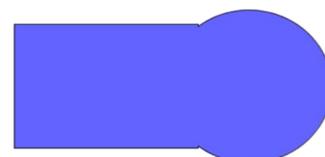
- 1) Create objects you want to use to create the final object.



- 2) Select both objects using your mouse.



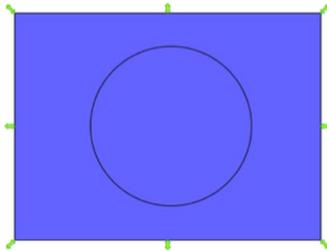
- 3) Go to the menu *Drawing* → *Path operations* and select *Merge selected objects together* - the two objects will merge into one. You can also select the desired function from the GUI Toolbar.



## Subtract

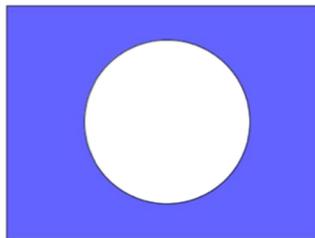
This feature is opposite to the *Merge* function. You can subtract portions of a bigger object to create a new object of the desired shape.

- 1) Select objects and make sure the outline is in the desired shape of the final object.



Then select the inner object that you want to remove.

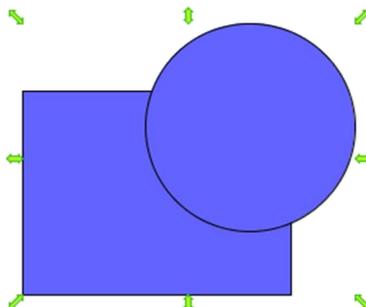
- 2) Go to the menu *Drawing -> Path operations* and select *Subtract objects*. You can also select the desired function from the GUI Toolbar.
- 3) Now you have a new cutting that can be modified - use the same steps as described for shaping objects.



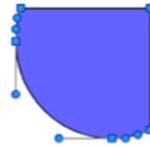
## Intersect

This feature creates a new object from an overlapped area of two different objects.

- 1) Arrange two arbitrary objects so their overlapping pieces make a desired shape, and then select the objects.



- 2) Go into the menu *Drawing ->Path operations* and select *Intersect*. You can also select the desired function from the GUI Toolbar.

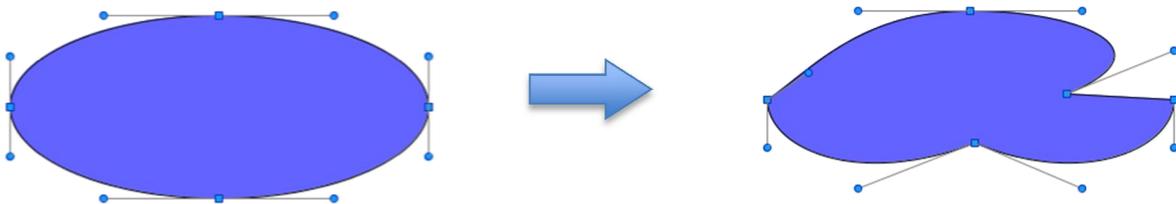


Now you have a new cutting ready to be formed into a desired shape by clicking and dragging the surrounding dots.

### Turn-to-Path

This feature allows you to transform the objects e.g. Changing primitive objects to complex paths. Go to *Drawing->Path operations->Convert Selected Objects to General Path Objects*. You can also select the desired function from the GUI Toolbar. You will see a series of points appear around the object.

Simply drag the surrounding points to achieve the desired shape.

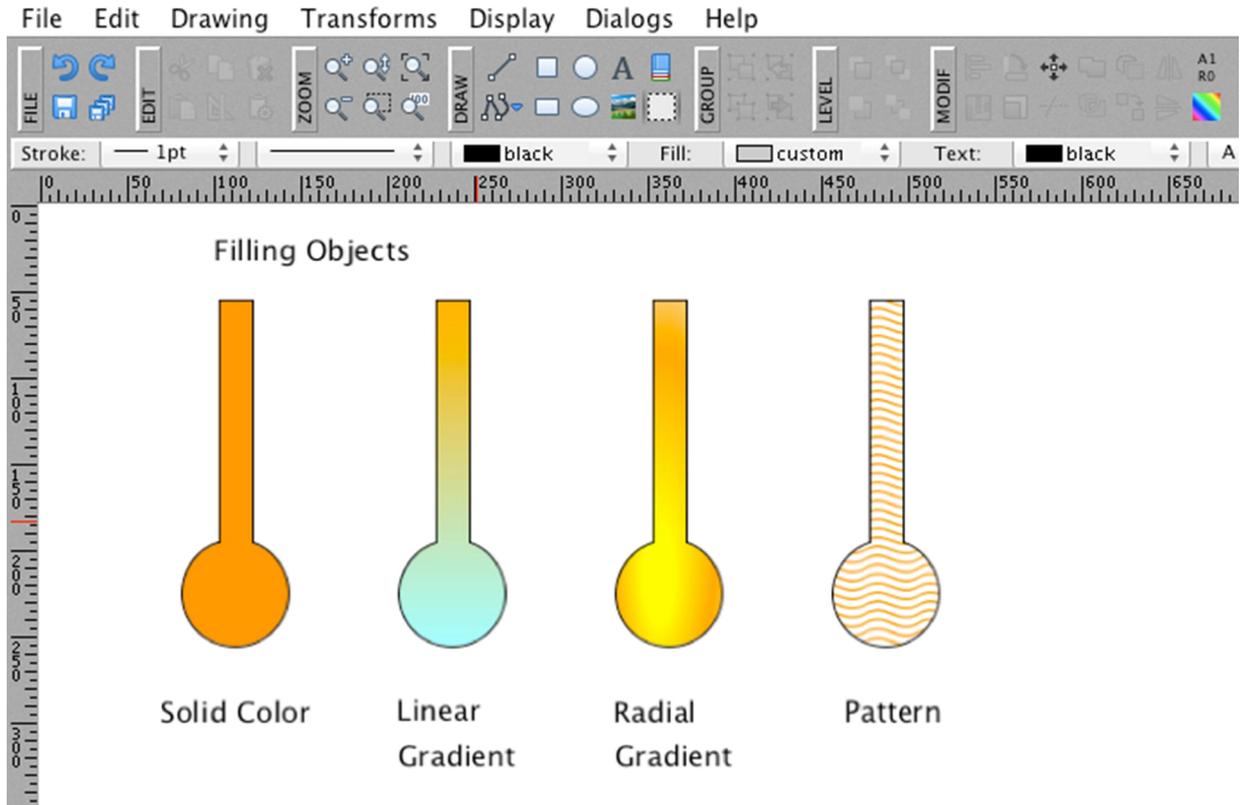


## 6.13 Fill and Stroke



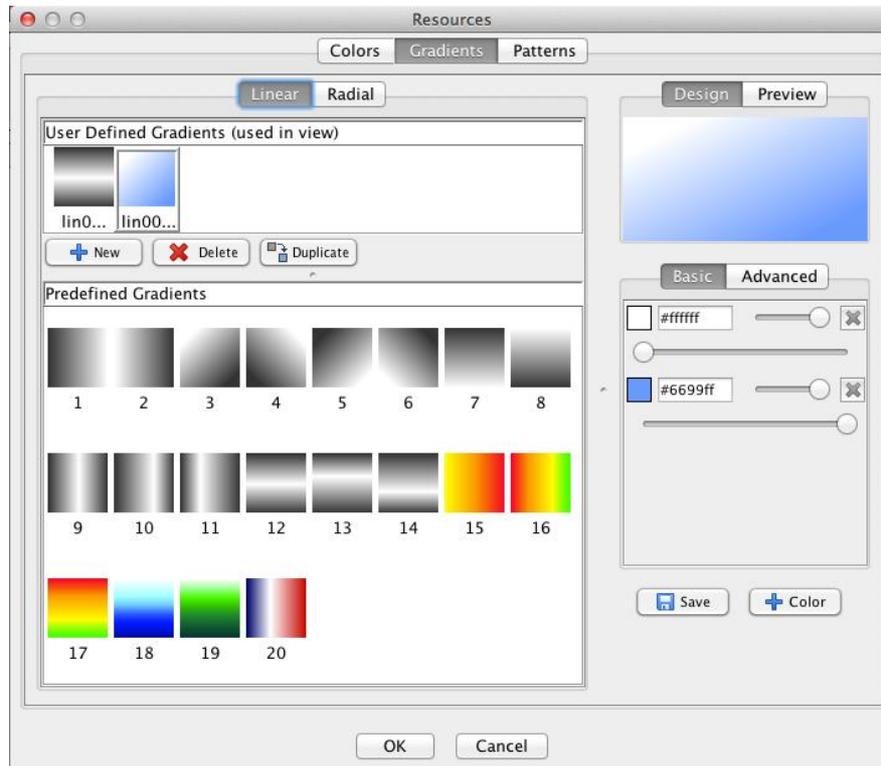
Graphic objects can be filled (or stroked) with a solid color, linear gradient, radial gradient, or complex pattern.

Watch video describing this functionality: <https://www.youtube.com/watch?v=ryduKCgzFr0>



Change a fill or stroke:

- 1) Select the object(s) whose color you want to change.
- 2) Click on the **Color**, **Gradients**, and  **Patterns** button in the GUI Toolbar.

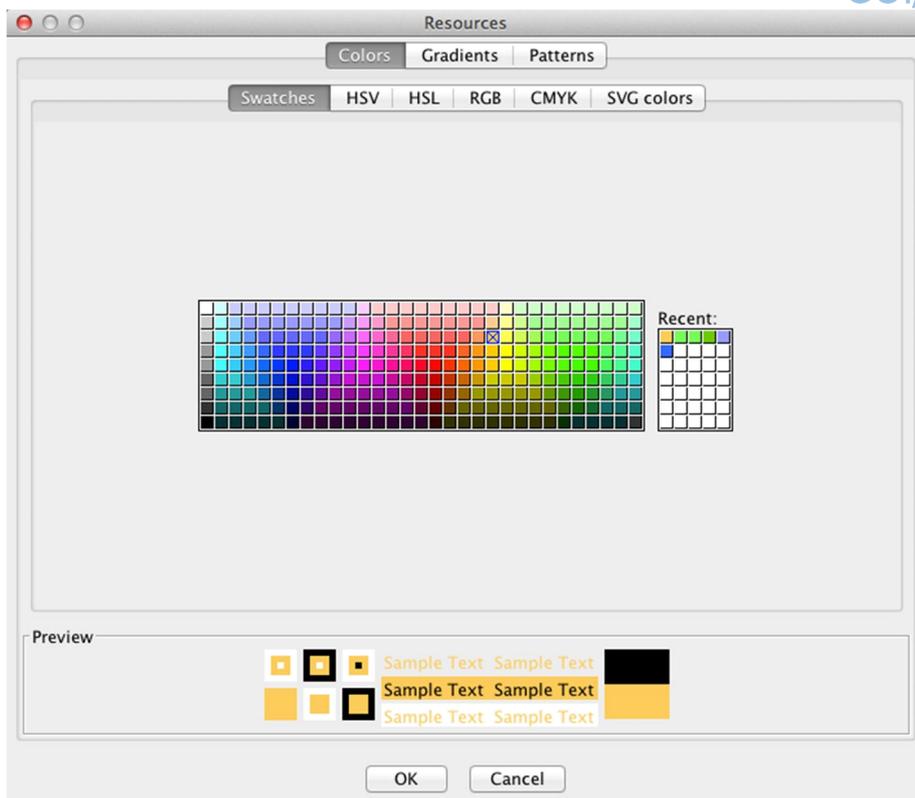


- 3) Choose a solid color, gradient, or pattern.
- 4) Confirm the changes by clicking on the **OK** button.

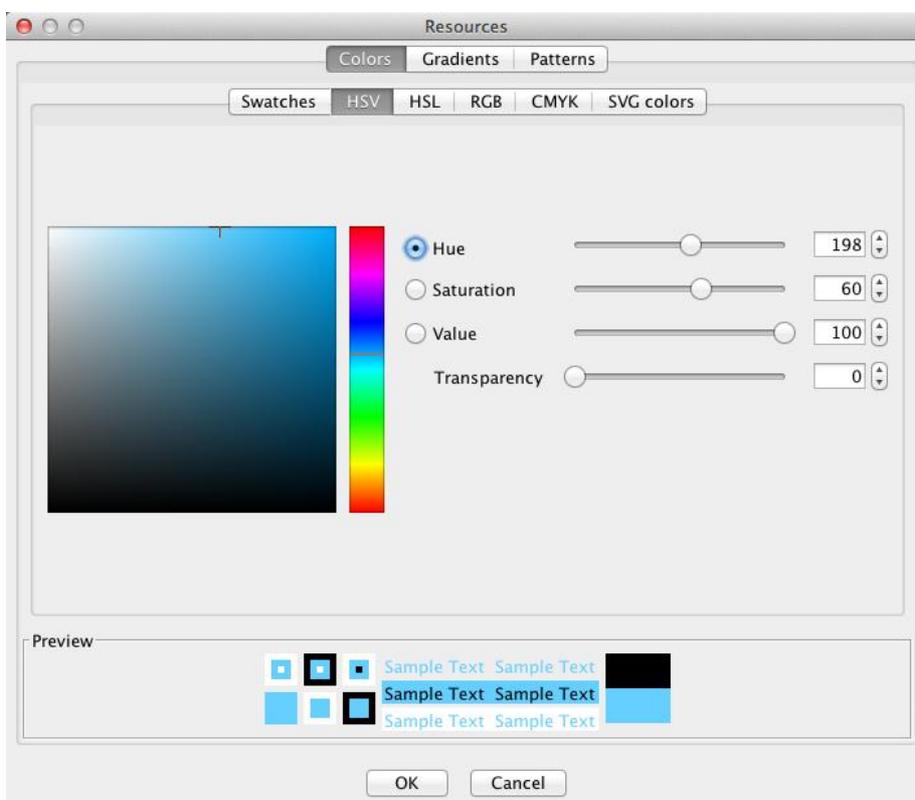
### Solid Color

Navigate to the **Colors** tab in the **Resources** tab where you will see a few options for solid colors:

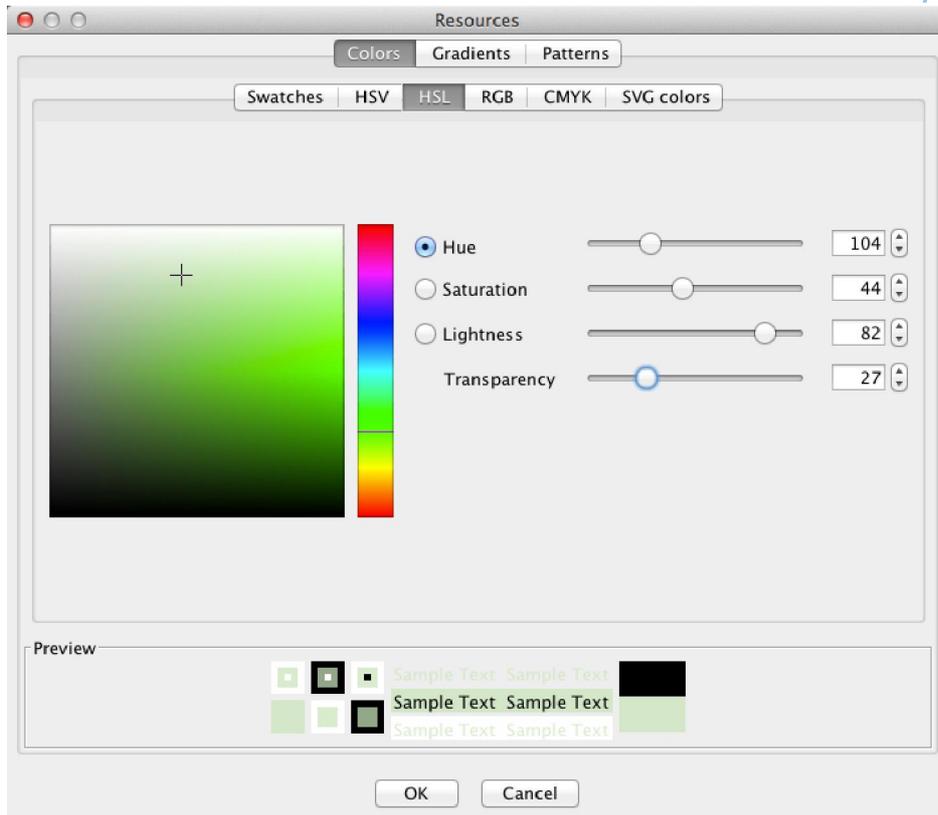
*Swatches* – select the color from the pre-defined palette



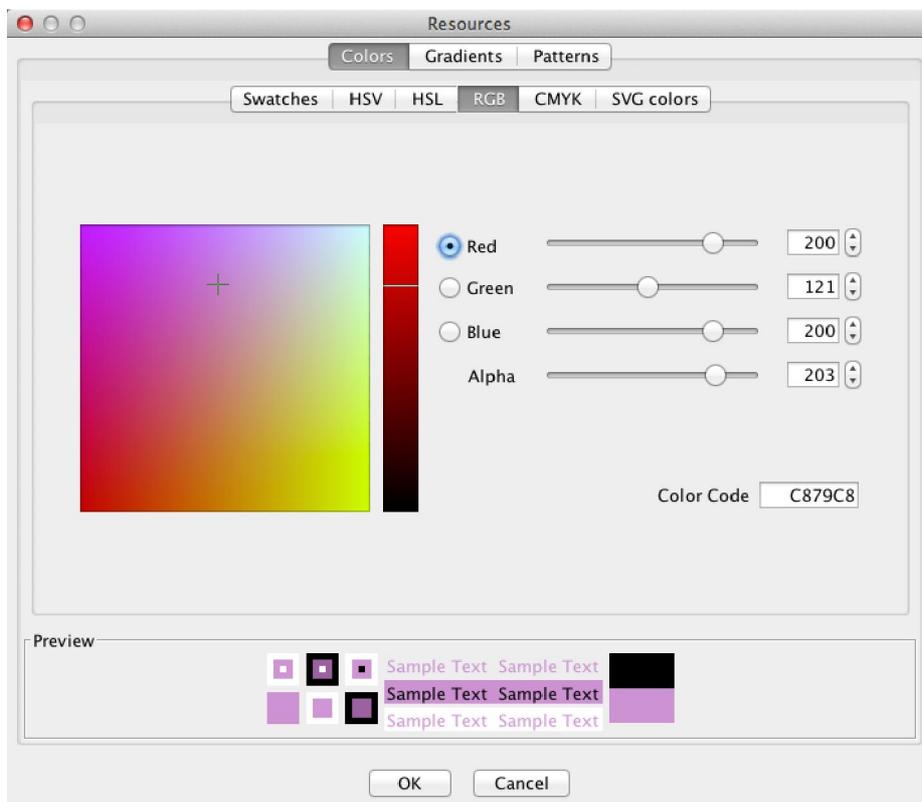
*HSV* - mix your desired color by determining the *Hue*, *Saturation*, and *Value* and setting *Transparency*



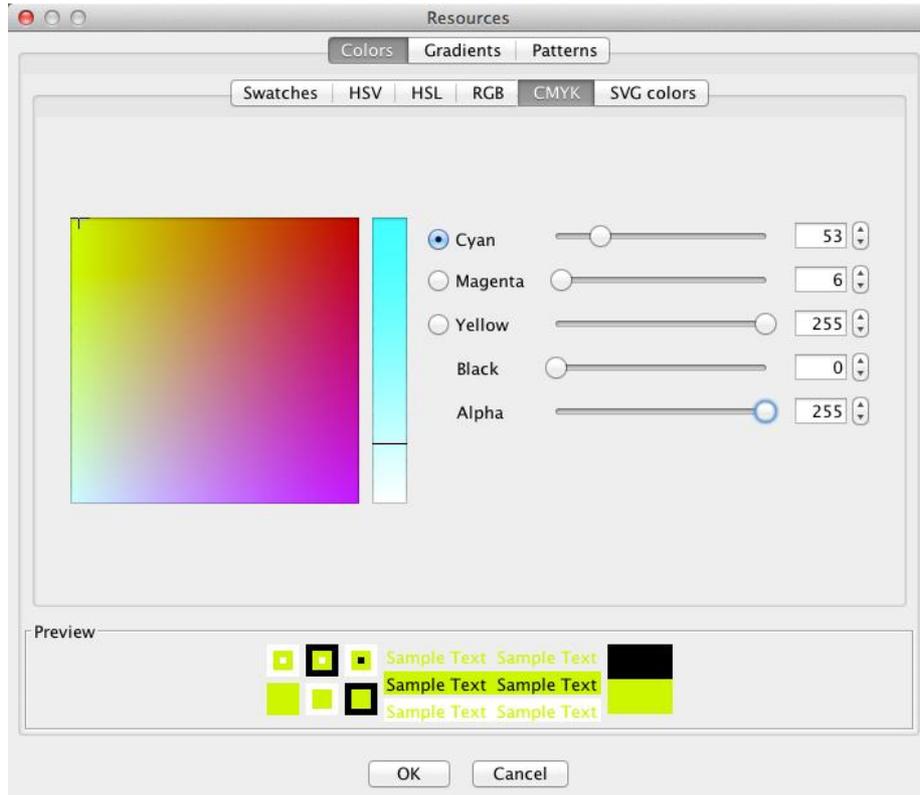
*HSL* - mix your desired color by determining the *Hue*, *Saturation*, and *Lightness* values and setting *Transparency*



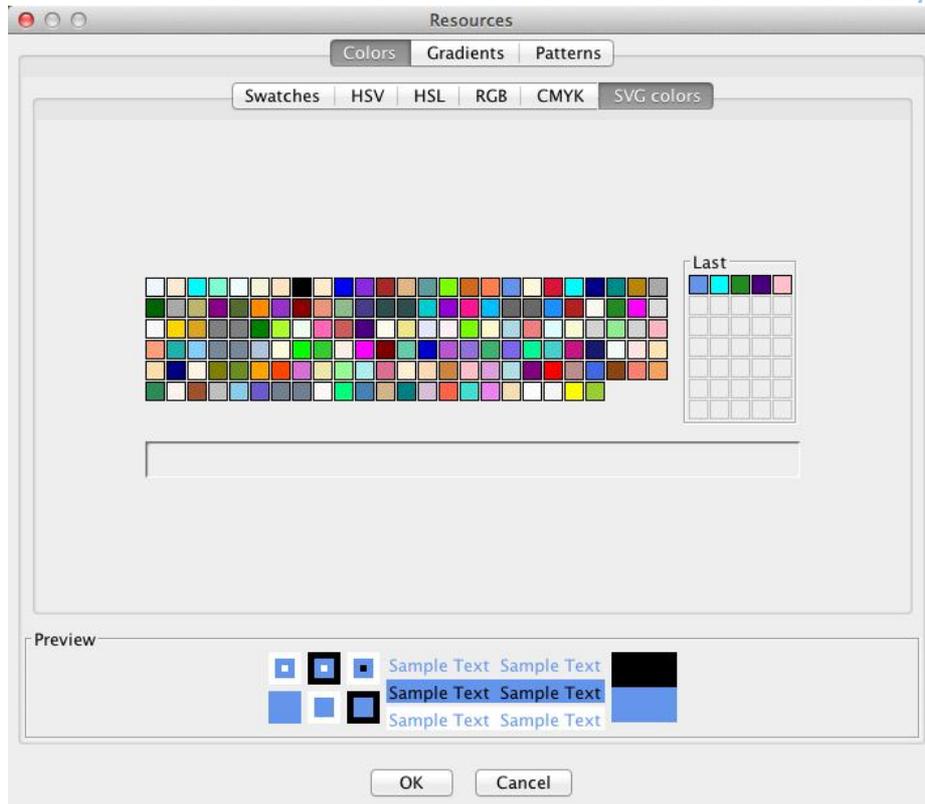
*RGB* - mix your desired color by determining the *Red Green* and *Blue* values and setting *Alpha* level



**CMYK** - select your color from the *Cyan, Magenta, Yellow, Key (Black)* color palette and set *Alpha*

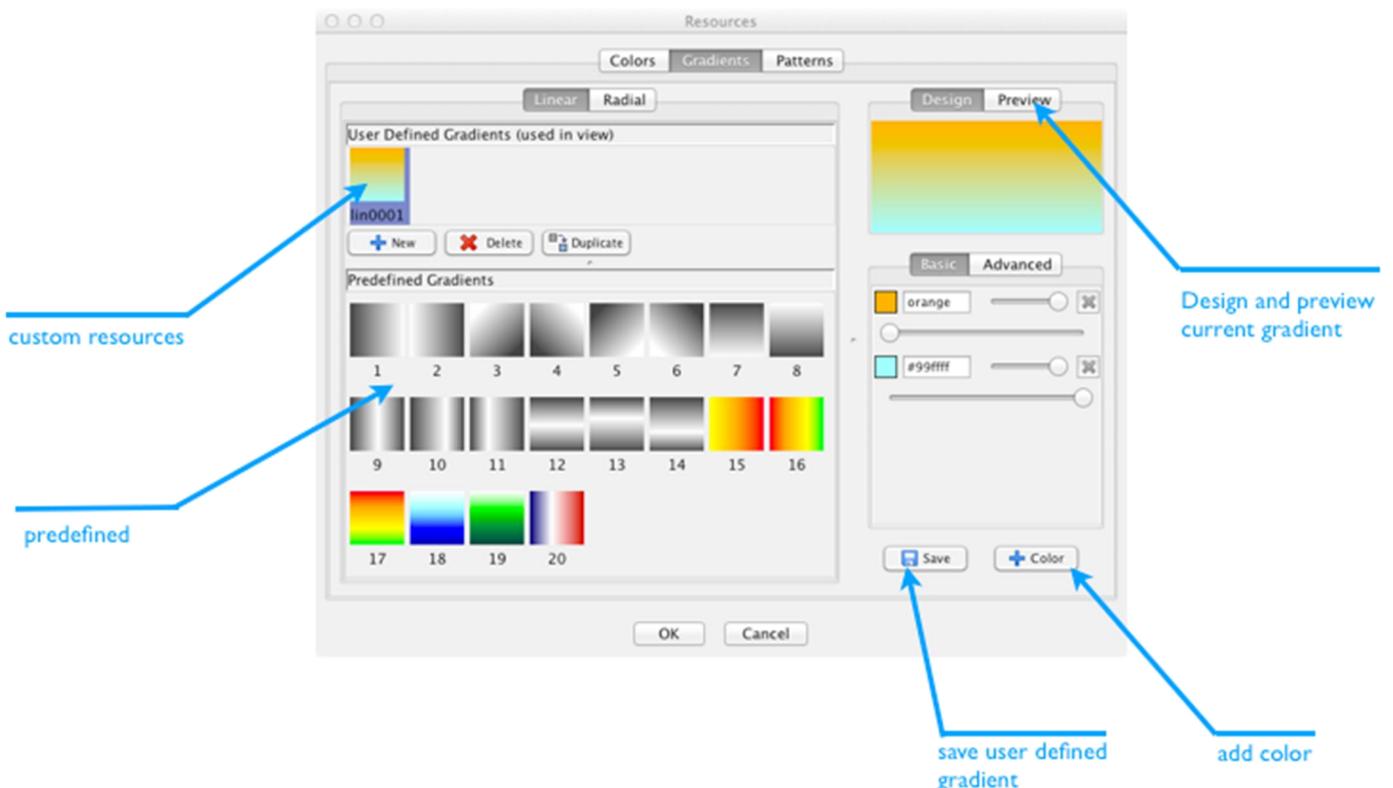


**SVG** - select your color from the pre-defined SVG standard colors palette



## Gradients

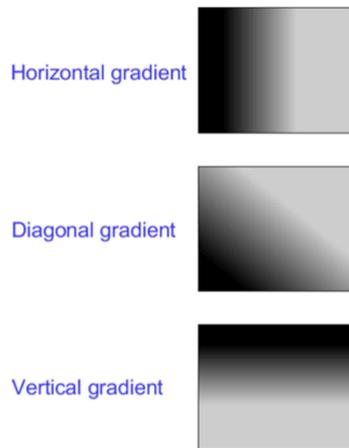
Using gradients enables you to create your own custom color blends and give your objects a plastic look. You can create smooth color gradations over one or more objects and save them for later use on other objects.



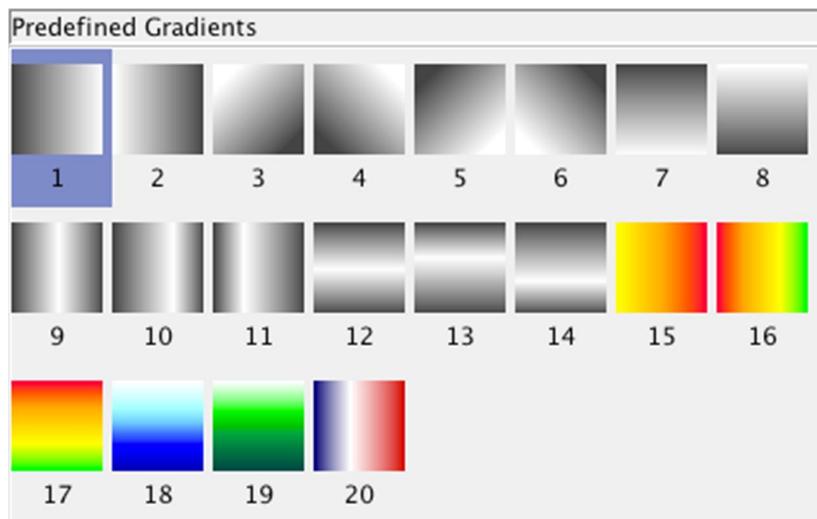
The **Resources** tab contains pre-defined linear and radial gradients and patterns. You may create new gradients and patterns or modify the existing ones.

**Linear gradient**

The linear gradient function enables you to create horizontal, vertical, and diagonal gradient fills.

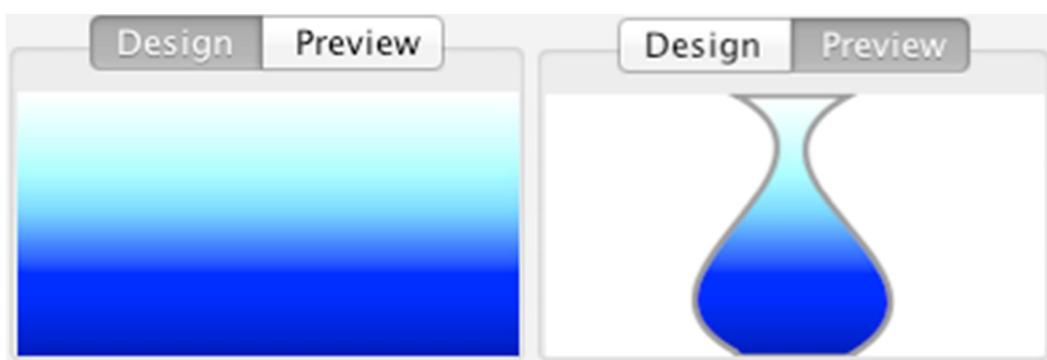


You can choose from pre-defined linear gradients in the **Resources** tab:



When you click on a pre-defined gradient, you will see its properties in the right side panel.

You can see the gradient preview in the small top window, and if you click on the tab **Preview**, you will see how the object will look.

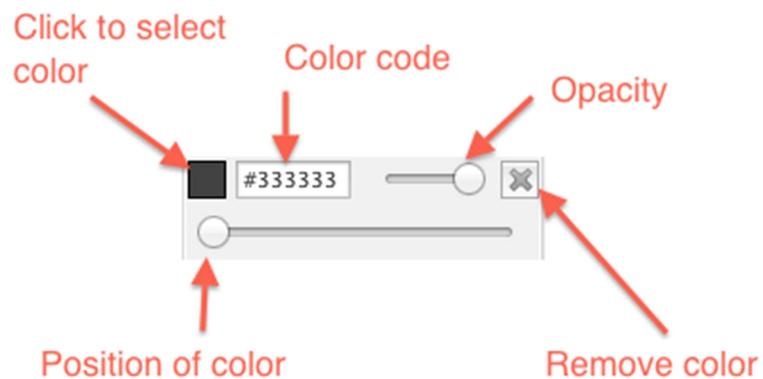


The bottom part of the panel is divided into two tabs, **Basic** and **Advanced**:

**Basic** - you will find all defined colors for a selected gradient

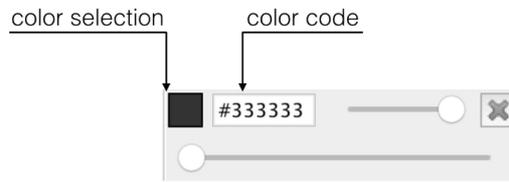


On the basic tab panel, each color has its own controls.



Therefore, each color has the following properties:

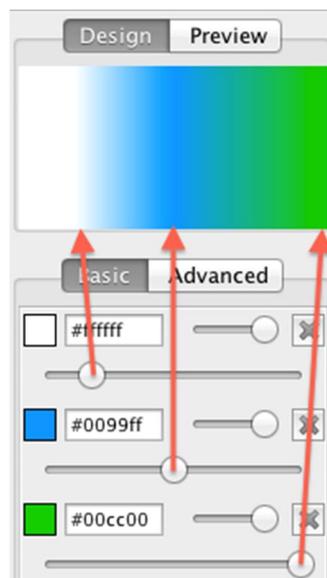
- *Color* - sets the color and corresponding color code



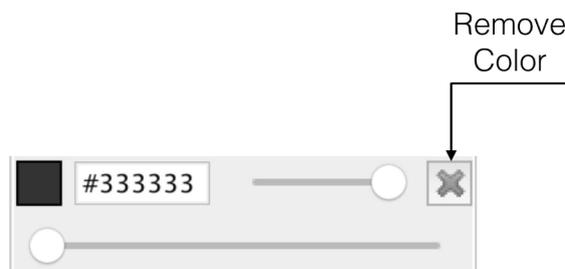
- *Opacity* - specifies the alpha channel (transparency)



- *Position* - specifies the starting point of this color. Move the slider to the side to change the color position.

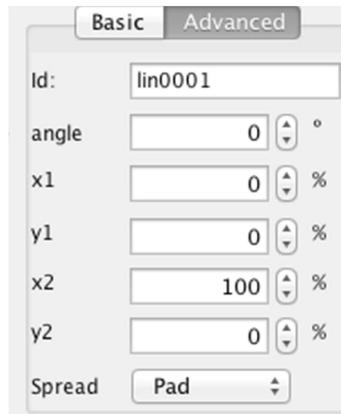


To remove a color, press the "X" button.



To add a new color, press the *+ Add Color* button.

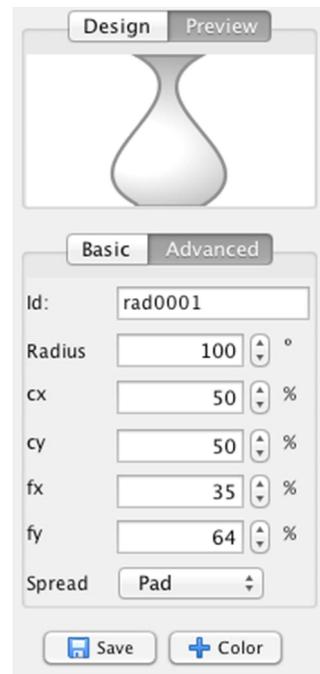
**Advanced** - you can fine-tune your custom gradient and define the id, angle, and fill options



- *Id* - you can name the newly created gradient
- *Angle* - you can change the angle of the gradient from horizontal to a custom angle
- *x1, y1, x2, y2* - linear gradients are defined by the bounding box of an object they fill; x1 and y1 specify the initial point of the gradient of the bounding box; x2 and y2 represent the end point of the gradient
- *Spread method* - Pad - basic fill option, no reflection or repeating
  - Repeat (repeats the shading)
  - Reflect (reflects the shading)

## Radial gradients

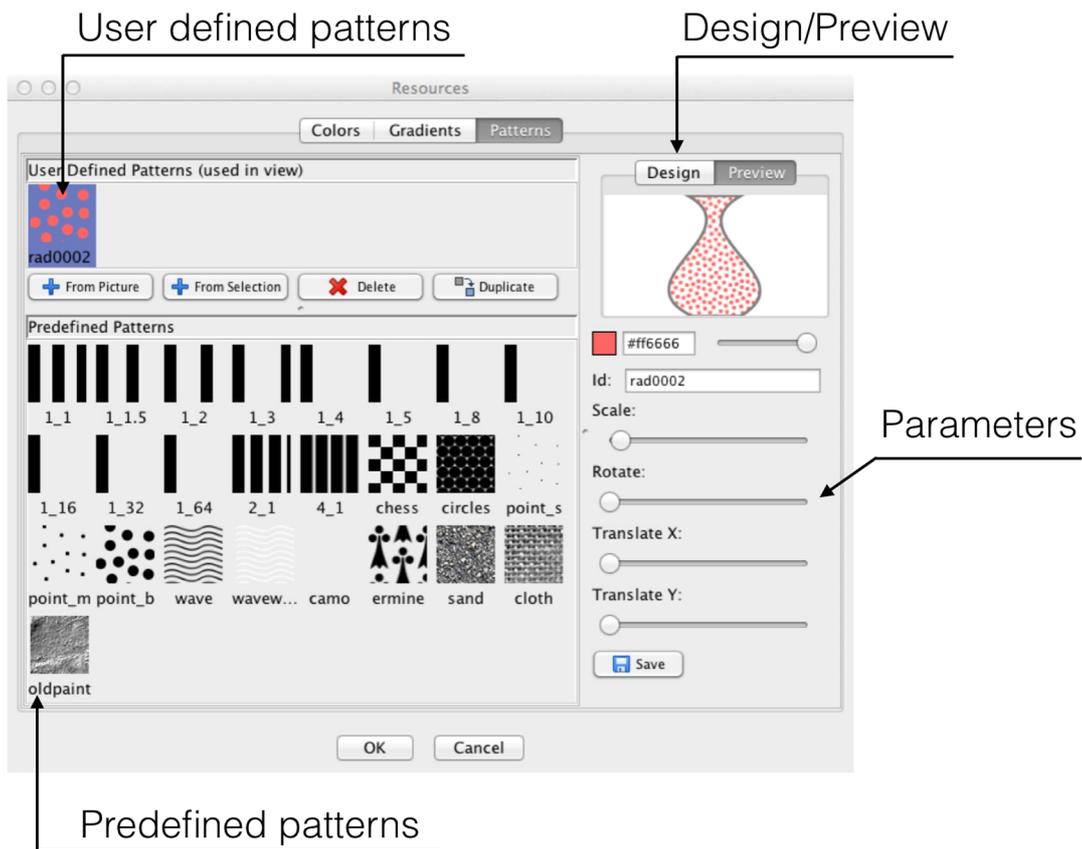
This function operates with circular gradient fills; this is the same principal as for the linear gradients, only the **Advanced** section is slightly different:



- *Id* - you can name the newly created gradient
- *Radius* - sets the gradient radius
- *cx*, *cy*, and *r* define the outermost circle of the radial gradient
- *fx* and *fy* define the focal point of the radial gradient
- *Spread method* - Pad - basic fill option, no reflection or repeating
  - Repeat (repeats the shading)
  - Reflect (reflects the shading)

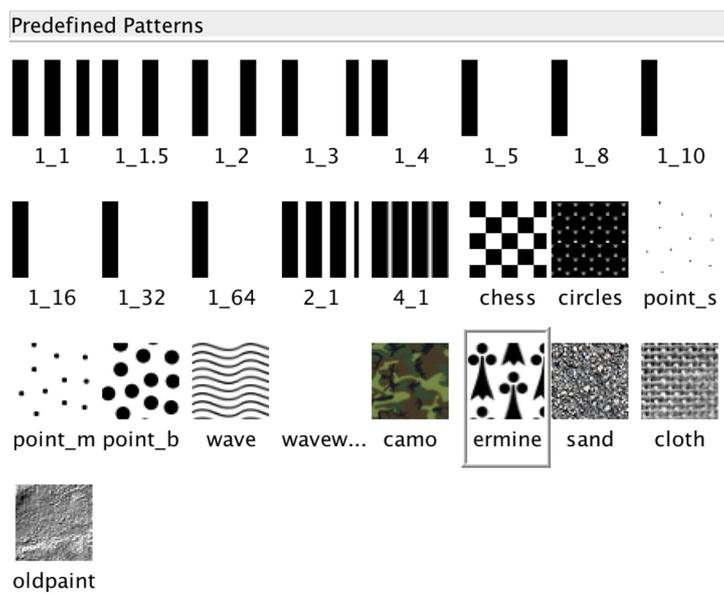
## Patterns

Different objects or images can be used as a pattern for other objects. *myDESIGNER* lets you choose from pre-defined patterns or create new ones, either by modification of the existing ones or by importing a file (jpg, png, jpeg) or selection. Patterns can be either raster images or vectors. If you use raster images, you might experience pixelation on tighter zoom, so if you can choose, use vector image where possible.



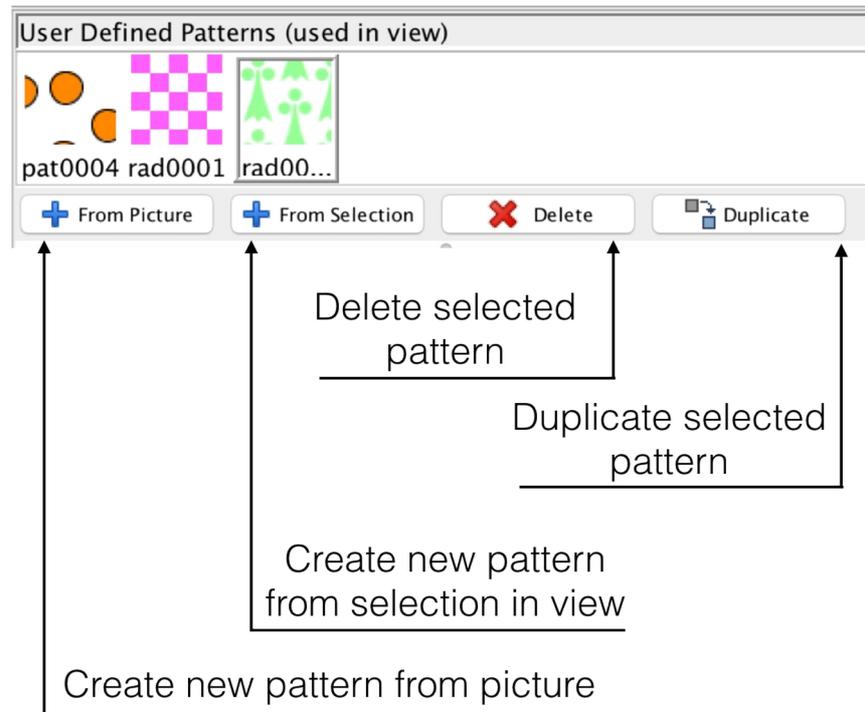
### Predefined Patterns

Predefined patterns are simple but effective patterns for you to use. Please keep in mind that you can very easily change color or rotate gradients to suit your requirements. Usage is quite simple; select a pattern you wish to use, and if necessary, modify the parameters and hit the save button.



## User Defined Patterns

All the patterns used in your view are listed here. You can also create new ones from predefined patterns or import them from files or selections.



To use a user-defined pattern, just select it and click on the *OK* button.

Buttons on the bottom allow you to create new patterns, either from a file or from a selection, as well as delete and duplicate existing patterns.

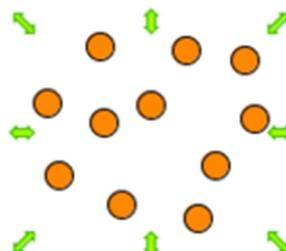
### Pattern Import

#### From File

Click on the *+ From Picture* button and select the source raster image file in the dialog window, then click on the *OK* button.

#### From Selection

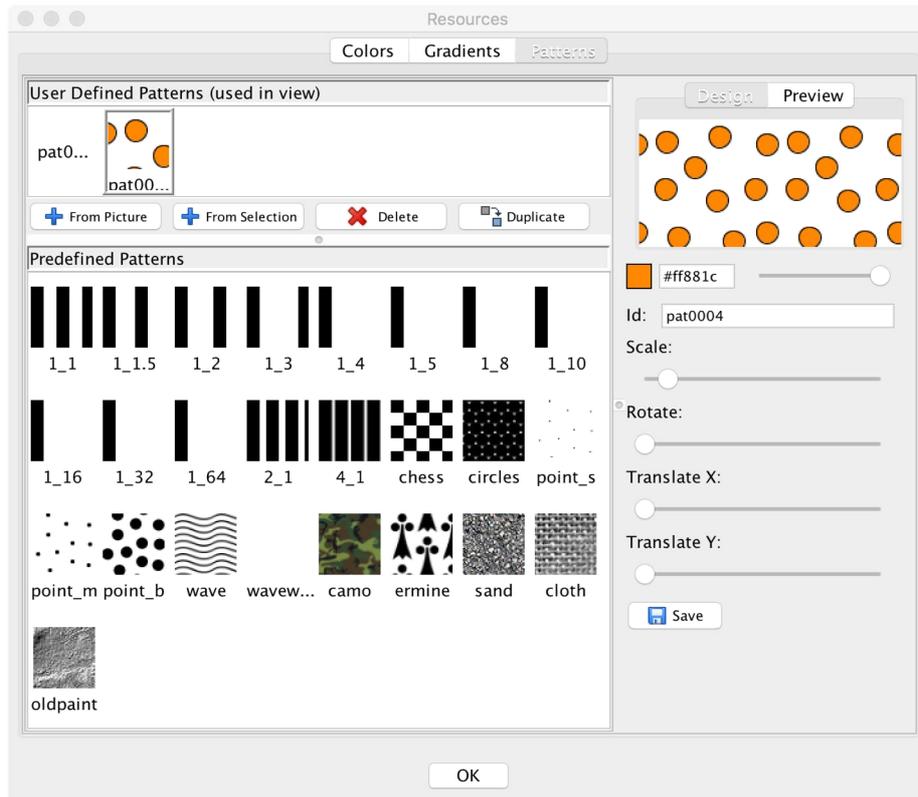
Click on the *+ From Selection* button to create a new pattern from the selection in your current view.



1. Draw the objects you would like to use as a pattern and select them.
2. Now open the resources dialog. Navigate to the Pattern tab. Now click on the *From Selection* button. The new gradient has been created:

**Parameters**

The parameters of the selected gradients can be easily modified. You can, for example, change the main color of the gradient, rotate, or scale it. Please find a list of parameters below, including explanations for each one.

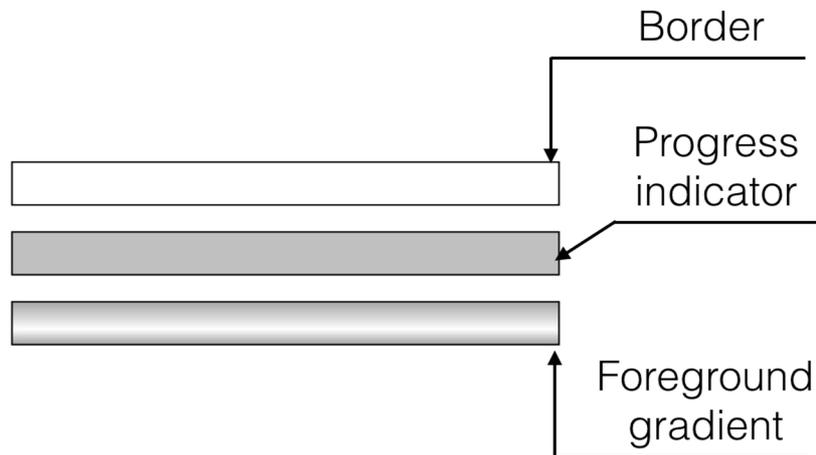


Parameter	Description
Id	Unique ID of the pattern. Each object has its own unique id.
Scale	You can scale the gradient so it appears smaller or larger. You might need to scale the gradient to fit your object's size. You can preview the result in the Preview window.
Rotate	You can rotate the pattern to the desired angle.
Translate	Move the position of the gradient along the X and Y coordinates.

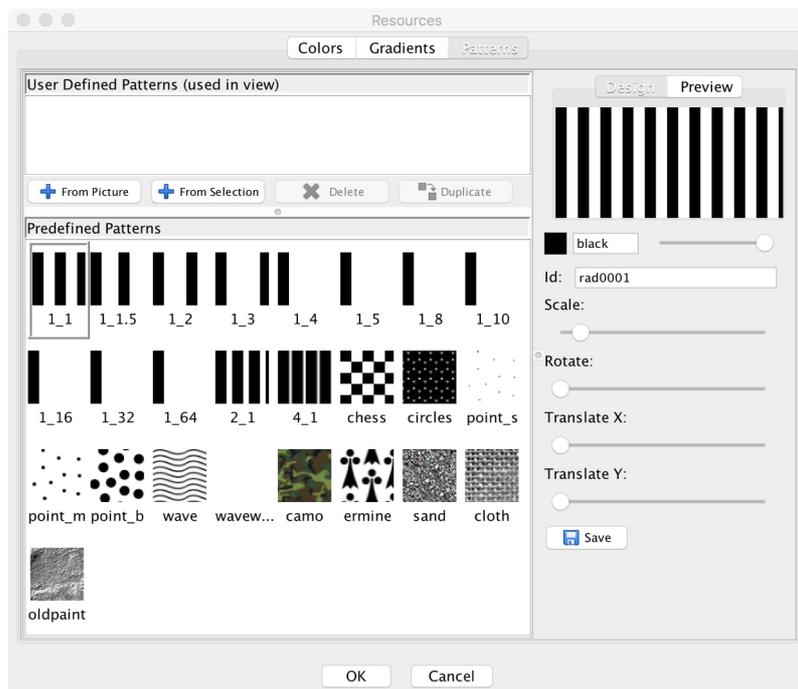
*Example of using a pattern*

We will show you a simple example of how to use a pattern in your design. We will create a simple progress bar by using and modifying a predefined pattern.

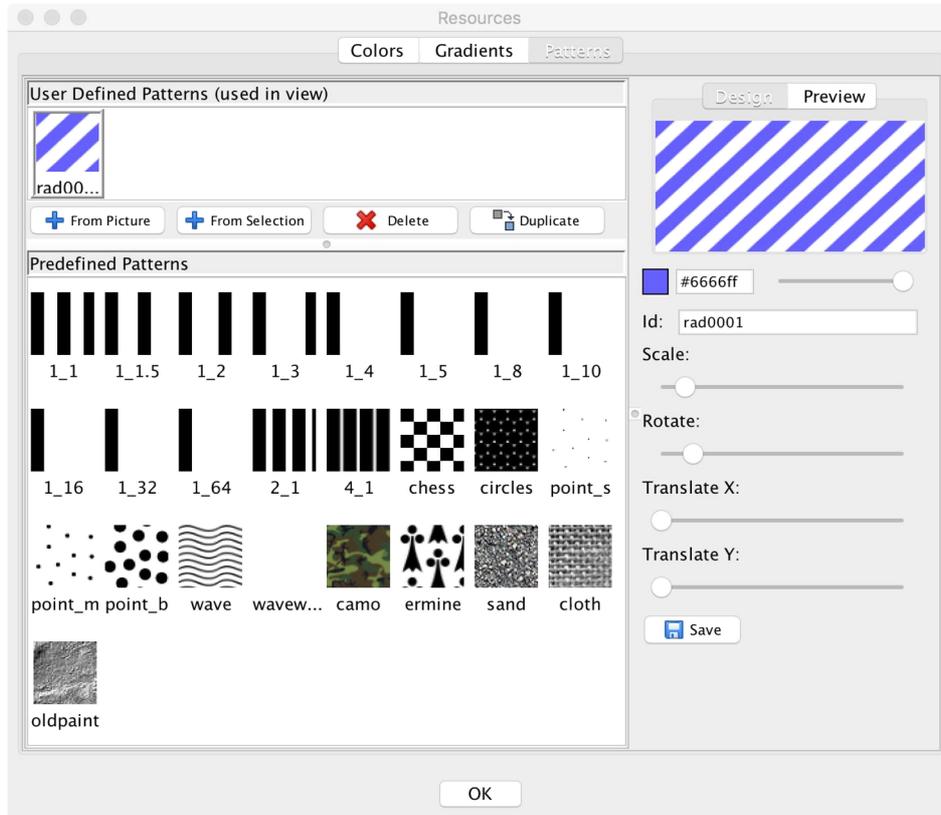
1. First, we will create three rectangles. The top one will be used as a border, the middle one will be used as our progress indicator, and the last one will be used in the foreground to give our progress bar a plastic look.



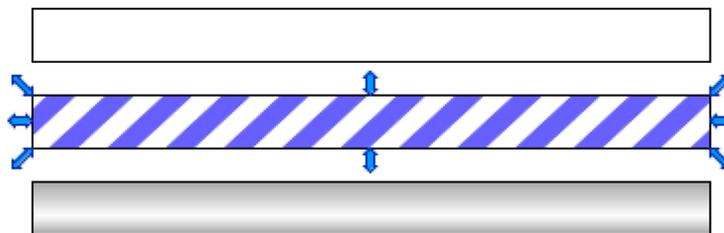
2. Now select the middle rectangle and click on the *Colors, gradients, and patterns* icon. In the pattern section, select a bar pattern.



- In the properties on the right side of the window, click on the color chooser rectangle and select a different color. Then use the *Rotate* slider to rotate the pattern by 45 degrees.



- Click on the *OK* button to apply a pattern to the rectangle.



- Now put all of the rectangles together to finish the progress bar.

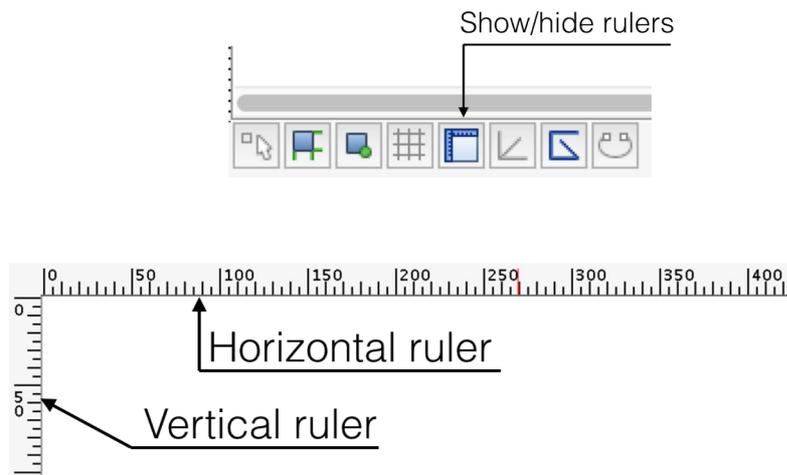


## 6.14 Rulers and Guides

Watch video describing this functionality: [https://www.youtube.com/watch?v=jiTpwXYE\\_I](https://www.youtube.com/watch?v=jiTpwXYE_I)

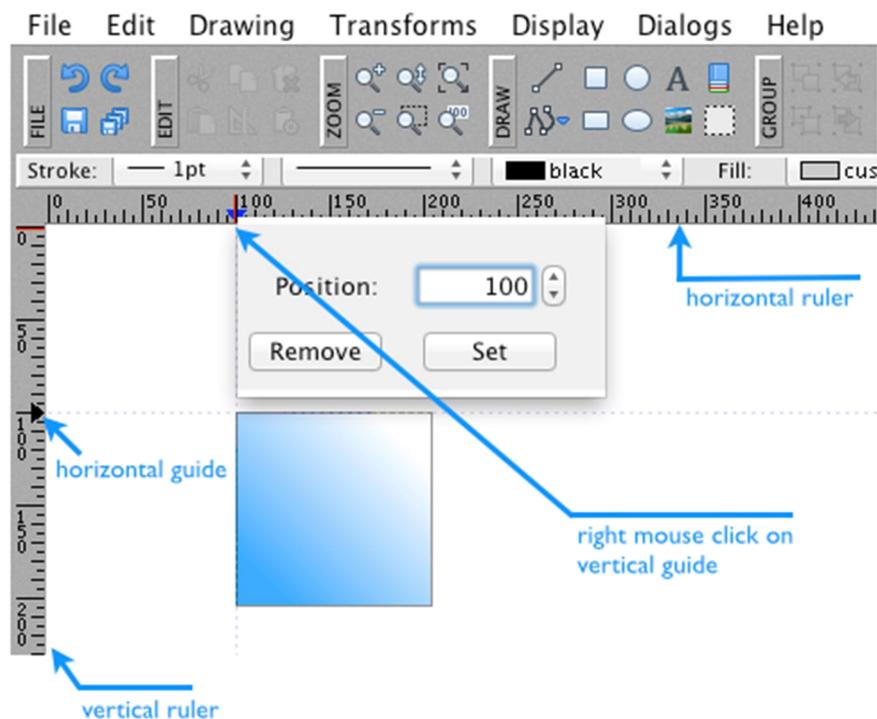
### Rulers

Rulers help you to accurately place and measure objects on the canvas. The common '0' point of the horizontal and vertical rulers is the root. You can show and hide rulers in the options menu located in the left lower corner of your view.



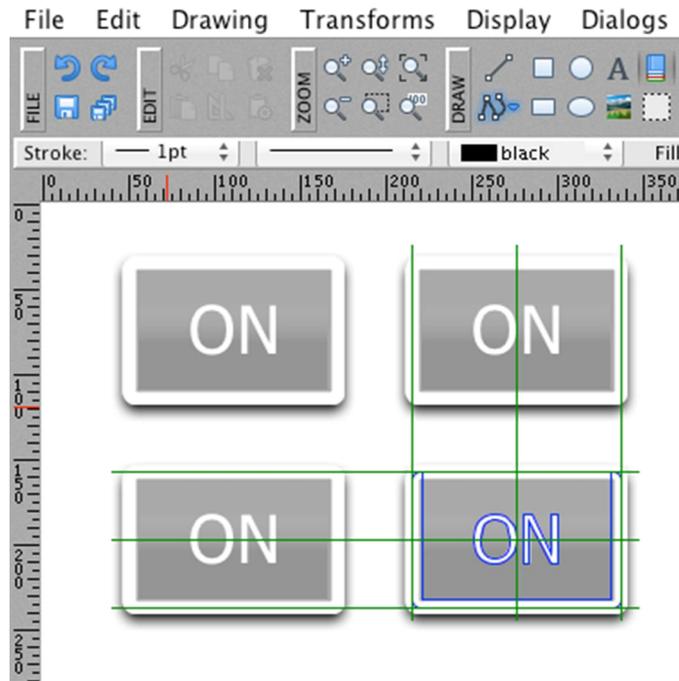
### Guides

Guides help you to align text and graphic objects. You can create ruler guides (straight vertical or horizontal lines) to align the objects. To create a new guide, click on any point on the ruler; the created guide can be moved later. You can set its exact location or delete it by using the right-click menu.

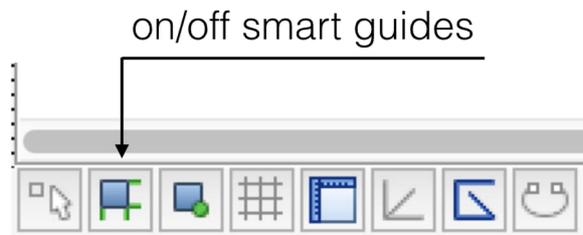


### Smart Guides

Smart guides are temporary orthogonal snap-to-point guides that appear when you are creating or manipulating objects. They help you align, edit, and transform objects in respect to other objects.

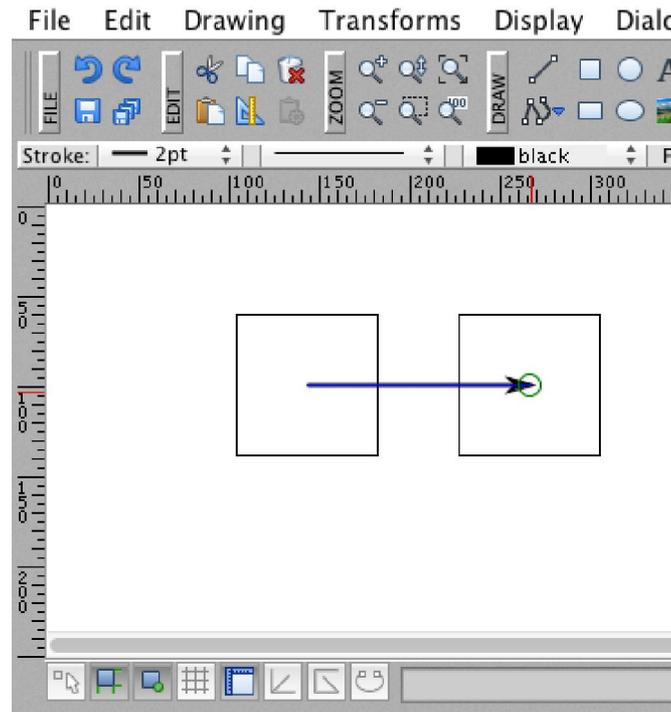


To enable/disable the smart guides, please click on the Smart Guides icon in the options menu located in the left lower corner of your view.

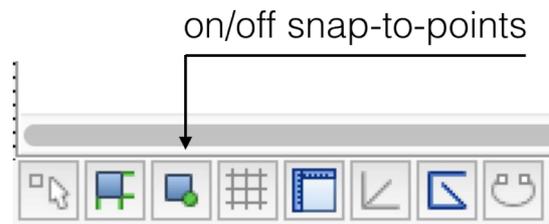


### Snap-to-Point

Snap-to-point guides are temporary bullet points that appear when you are creating or manipulating objects. They help you to align your object to other objects.



To enable/disable the snap-to-point guides, please click on the snap-to-point icon in the

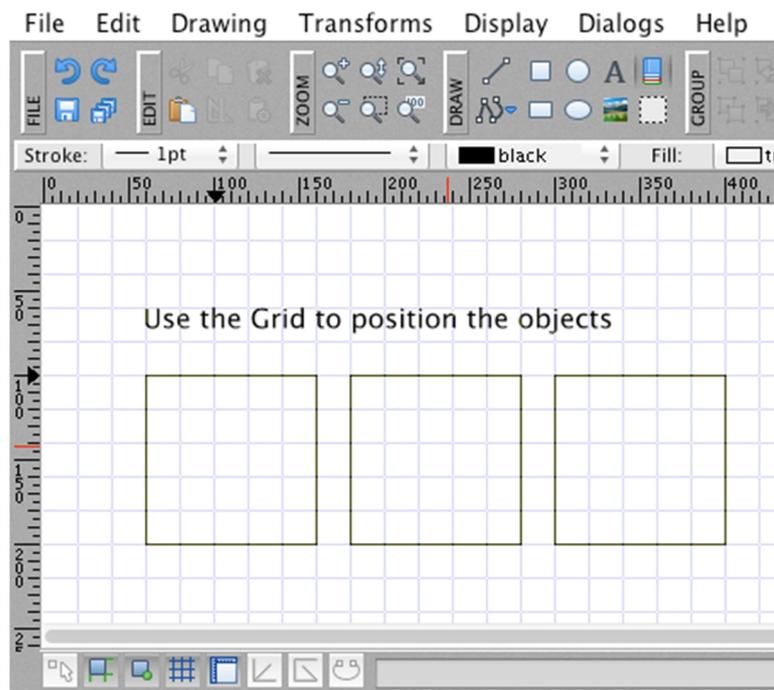


options menu located in the left lower corner of your view.

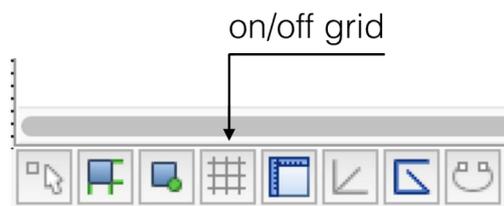
### Grid



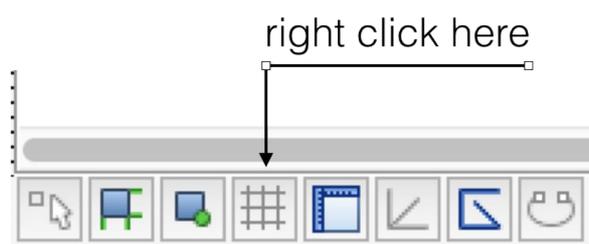
The grid is displayed behind your graphics on the canvas but does not print with the project visualization – objects align automatically when the grid is active. You can activate or deactivate it from the *Display* menu or the *Options bar*.



To enable/disable the grid, please click on the grid icon in the options menu located in the left lower corner of your view.



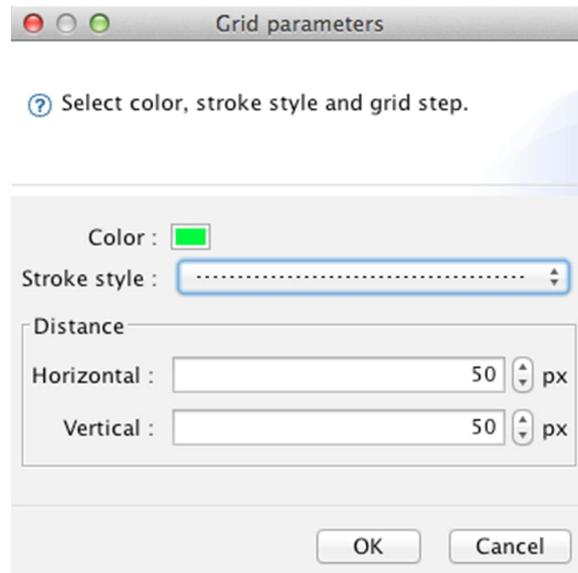
When you right-click on the Grid Icon in the Options bar, the Grid settings dialog is shown.



### Grid settings

Here you can set the grid parameters like color, stroke style, and the grid step.

Go to the menu *Display* -> *Grid settings* or right-click on the grid icon in the Options bar.



## 6.15 Layers

When you are creating complex visualizations, you might find it challenging to keep track of all of the graphic objects on the canvas. The smaller items might be hidden under the larger ones and thus working with them could get difficult. Layers can help you to keep track of your items.

Watch video describing this functionality:

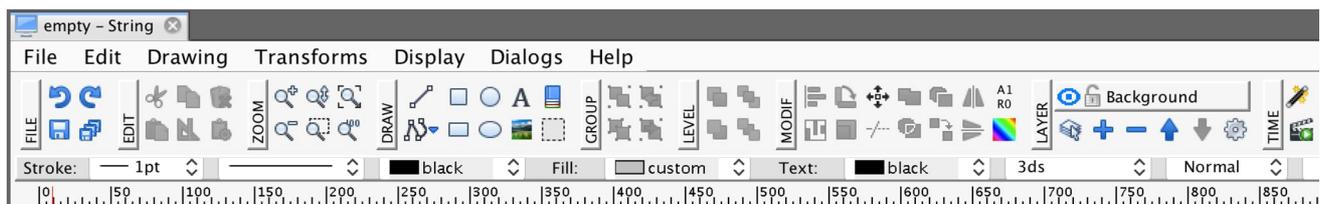
<https://www.youtube.com/watch?v=uogOzyklUA0>

Think of the layers as transparent planes, each containing graphic items that are glued together. If you change the order of the layers, you also change the position (visibility) of the items on the canvas. You can move items within a single layer, or you can group together multiple items spread throughout several layers. If you copy objects, then a new object is copied into the layer of the original object.

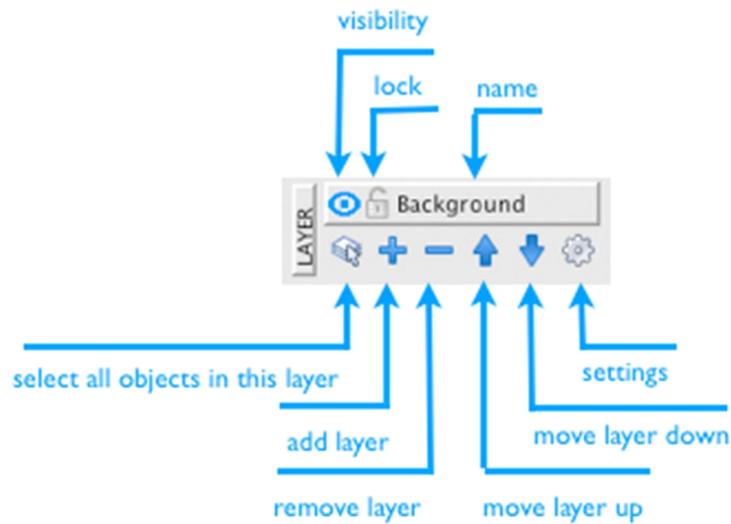
Layers can be controlled from two different places. There is a Layer window located above the properties window (toggle its visibility in the main menu – section Windows), and there is a Layer Combo Item located on the GUI Toolbar.

### Layer Combo

The Layer Combo option is located on the right side of the GUI Toolbar:

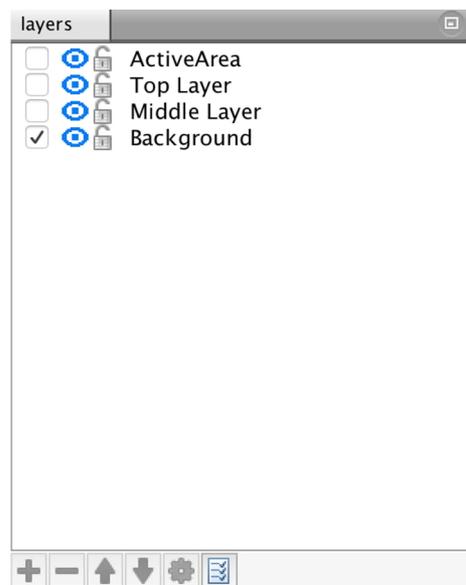


With the Layer Combo option, you can control all aspects of Layers. An explanation of the controls is in the following picture.

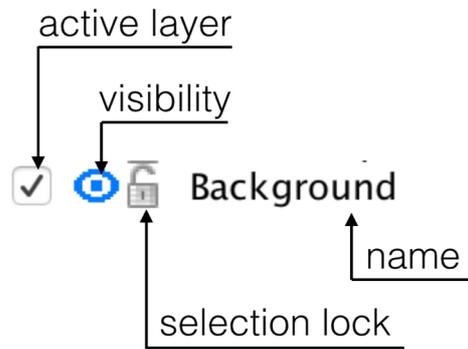


### Layer Window

The Layer window is activated in the main menu. Go to Window and toggle Layer Window visibility on/off.



The Layer Window and the Layer Combo option are complementary elements; both of them allow you to access all functions of Layers. The Layer window occupies more space, and so it allows for more comfortable control of layers than the Layer Combo option. An explanation of the controls is in the following picture:



### Changing Layer Visibility

To change the layer’s visibility, click on the eye icon of the corresponding layer.

	Layer is visible
	Layer is invisible

### Locking selection

You can lock a selection for a given layer by clicking on the Lock icon. If the layer is locked, you will not be able to select any of its elements.

	Layer Locked
	Layer Unlocked

### Setting Active Layer

The Active layer is the one into which new objects are drawn. For example, if you set the active layer to a layer named “Background” and then create a new object, this object will be created in the layer called Background. The Active layer cannot be locked, as you cannot create new elements on locked layers.

### Setting Active Layer in Layer Combo control

If there is no object selected in your view, the name of the active layer is displayed in the Layer combo control.



To select a new active layer:

1. Make sure you have no object selected in your view. If you have selected objects, use the escape key to unselect all objects or click on a canvas)
2. Click on the Layer Combo control, and list of layers is shown.
3. Click on a layer name, which you want to set as the new Active Layer.

### Setting Active Layer in Layer Window

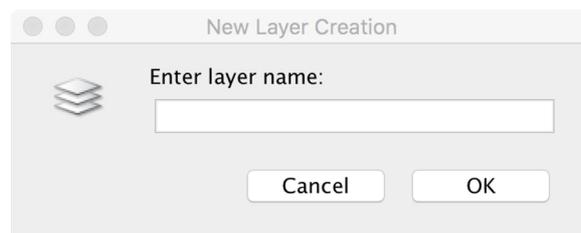
The Active layer is denoted in the Layer Window by the checked checkbox on the left side of the layer control.



To select a new active layer, simply check the checkbox next to the desired layer.

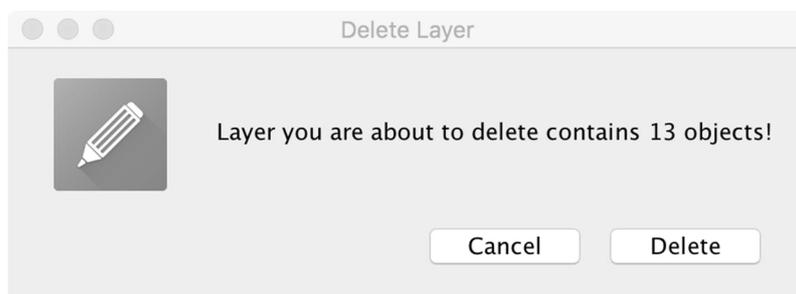
### Creating New Layer

To create a new layer, press the "+" button. A new dialog appears; enter the Layer name and confirm. A new layer will be created.



### Removing Layer

To remove an existing layer, click on its name and press the "-" button. If the layer you are about to delete contains any graphical objects, you will be presented with a warning:



If you continue, the layer will be deleted along with all the graphical objects it contains.

**WARNING:** If you delete a layer, all graphical objects contained in the layer will be removed as well. This action cannot be undone.

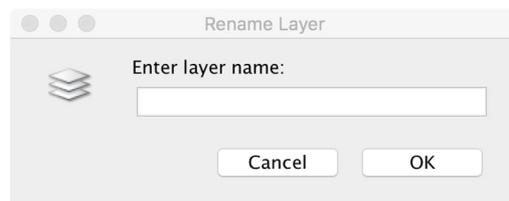
## Selecting all items in given layer

To select all items in given layer, click on the *select all items* icon.



## Changing Layer Name

To change a layer name, double click on the layer name. The Rename dialog will be shown. Change the name and press the *OK* button.

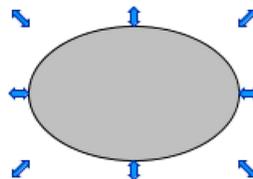
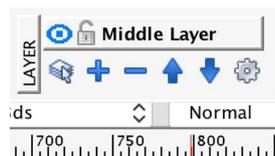


## Special Purpose Layer - ActiveArea

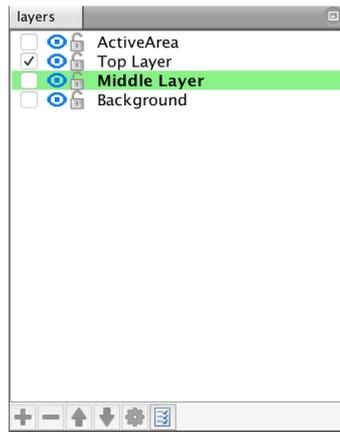
As you might notice, the topmost layer is always a layer named "ActiveArea". This Layer cannot be removed or shifted. It only contains active areas defined in your views. Active areas are always on top of the rest of your drawings.

## Determining Layers of selected objects

If you select an object or multiple objects, the layer containing these objects will be displayed in the Layer Combo box and highlighted in green in the layer list.



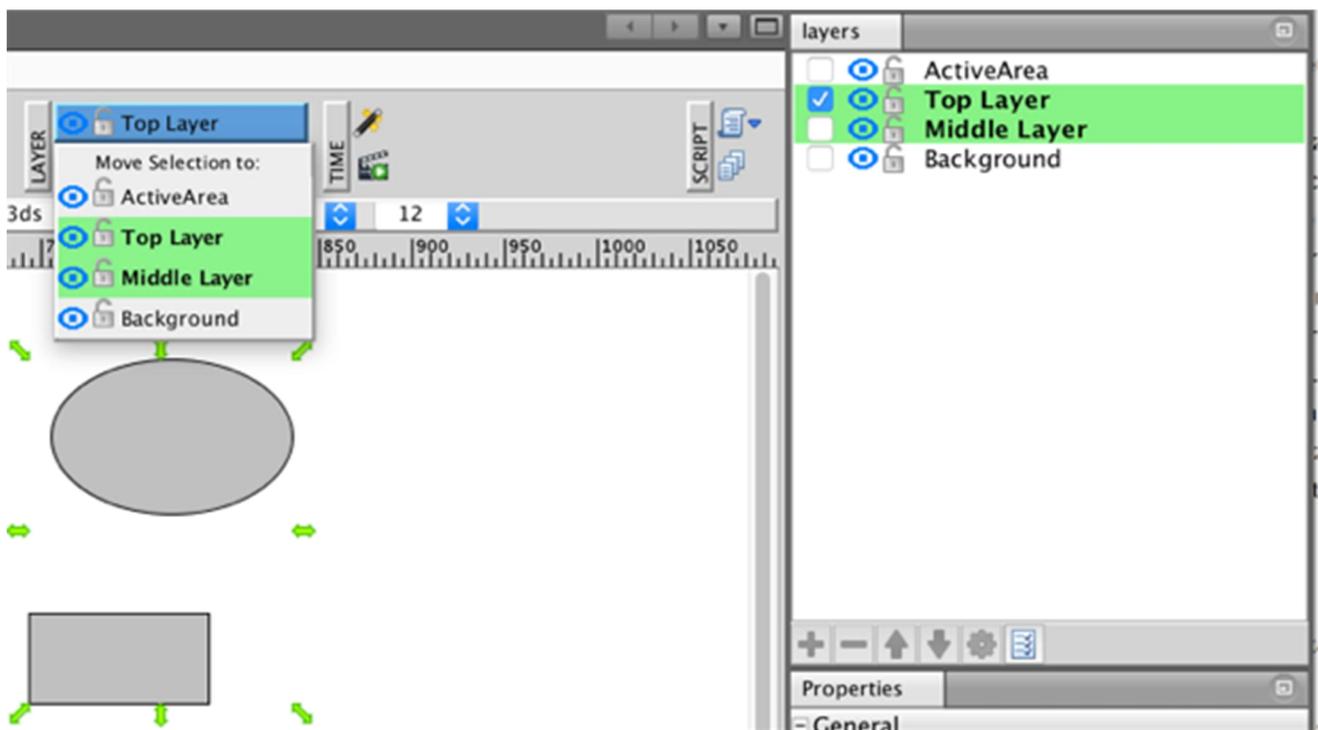
. 1 The selected object is on the layer named "Middle Layer". You can see the name of the layer in the Layer Combo control panel. Note that the name is in bold.



. 2 The selected object is on the layer with the name "Middle Layer." This is shown in the Layer Window by a green highlight and bold font.

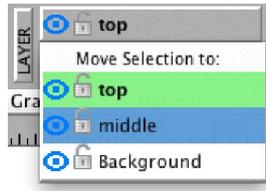
If you select multiple objects belonging to different Layers, the Layer controls will highlight all corresponding layers in green.

Note: The Layer Combo box will show the name of the top layer in the selection; its name will not be in bold.

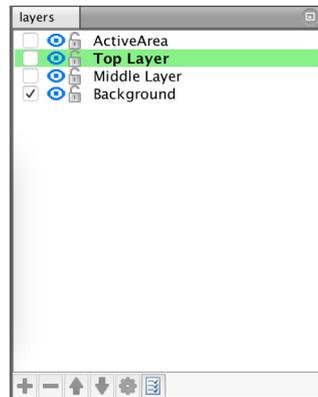


### Moving Objects to different Layer

If you want to move objects to a different layer, select the objects on the canvas and then select the desired layer. The selected objects are all moved to the desired layer.



A green highlight shows which layer contains your selected objects.  
To move the selection to a different layer, just click on the layer name.

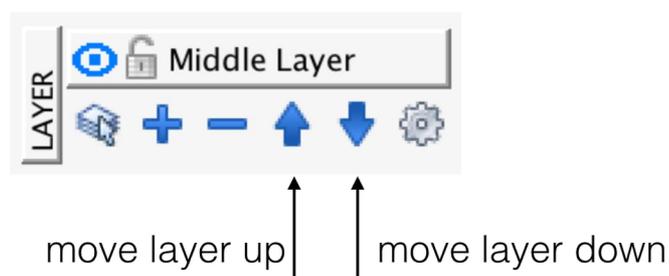


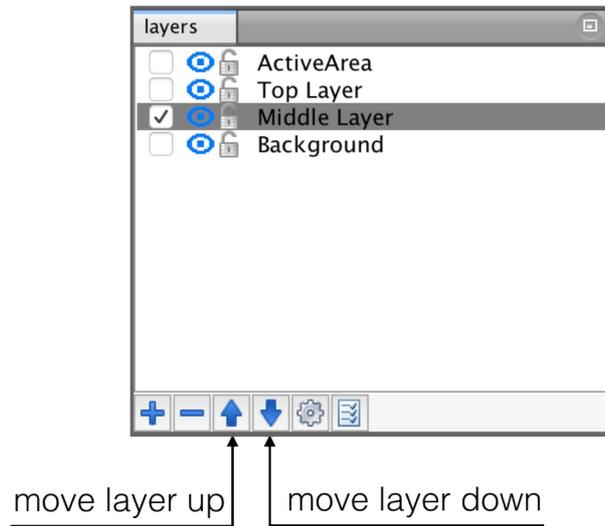
A green highlight shows which layer contains your selected objects.  
To move the selection to a different layer, just click on the layer name.

### Changing Order of Layers

Changing the order of Layers is quite simple. Just select a layer you want to move and then click on the Up or Down arrow to move it.

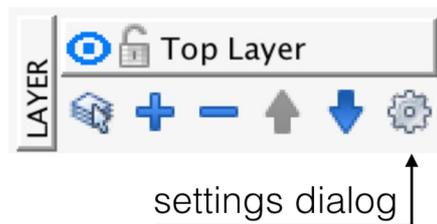
Layers at the top of the list overlap the layers at the bottom of the list. Therefore, if you move a layer up, it will overlap all the layers below it in the list.



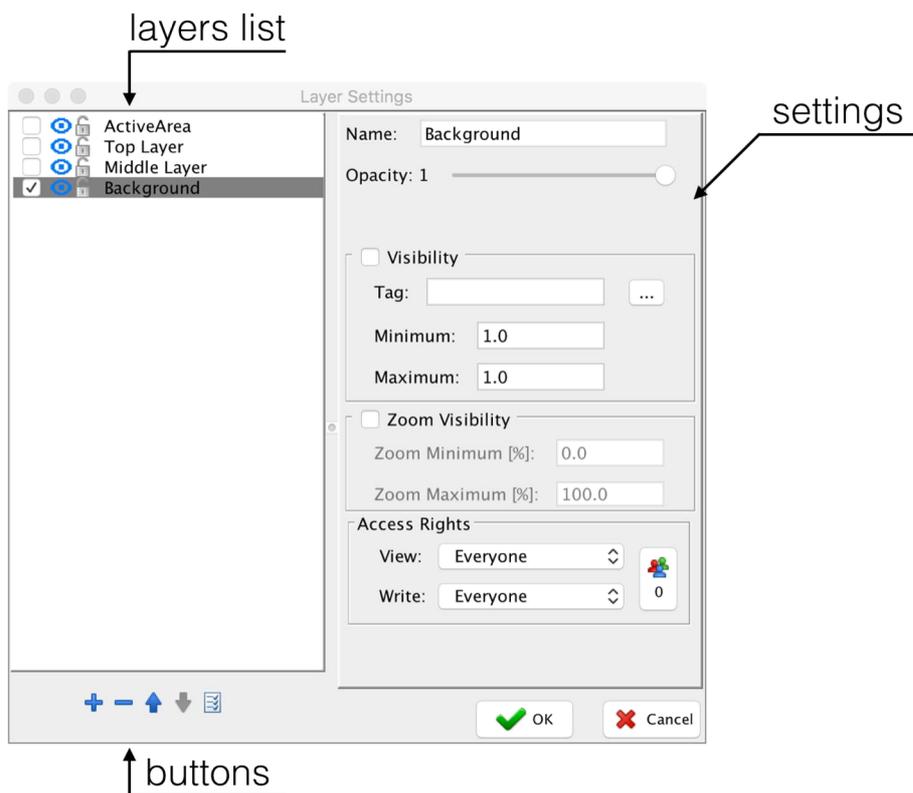


### Changing Layers Properties

To change layer properties, press the setting button.



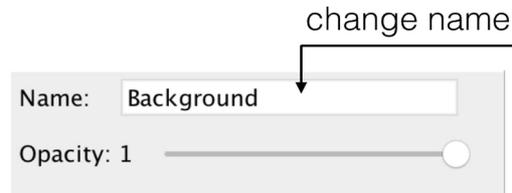
You will be presented with the Layer Settings dialog.



The layer settings dialog has two parts; on the left side, there is a list of layers. You can add, delete, change position, and merge layers with the buttons at the bottom of the list. On the right side, there is a settings pane for the selected layer.

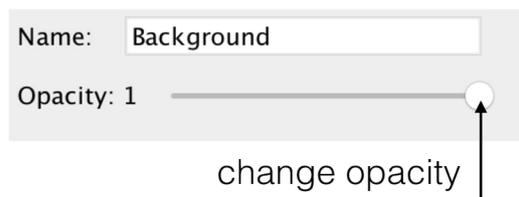
### Changing Layer Name

To change a layer's name, click on the layer in the list of layers to select it. Then you can write a new name in the Name box.



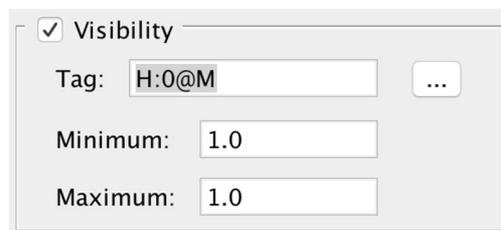
### Changing Layer Opacity

To change a layer's opacity, click on the layer in the list of layers to select it. Then drag the Opacity slider to change it to a new value.



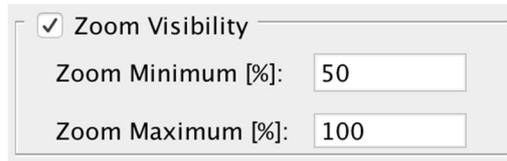
### Changing Layer Visibility

To change a layer's visibility, click on the layer in the list of layers to select it. Then, fill in the conditions for visibility. E.g., enter tag and minimum and maximum values. If the tag value is within the range of the specified minimum and maximum values, the layer will be visible.



### Changing Layer visibility based on Zoom level

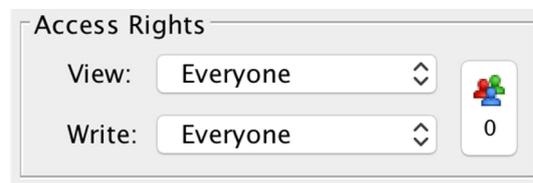
To change a layer's zoom visibility, click on the layer in the list of layers to select it. Then check the zoom visibility check box. Now you can specify the minimum and maximum zoom level when the layer will be visible.



For more information about zoom visibility animation, please refer to the chapter [Animations](#).

### Changing user access rights for given layer

You can limit who can view the objects in a given layer. In addition, you can limit which users will have access rights for objects in given layer. To change a layer's access rights, click on the layer in the list of layers to select it. Then you can change the access levels for read and write access as well as access group levels.



For more info about User Access Rights, please refer to the chapter [User Accesses](#).

## 6.16 Copying and Pasting Elements

### Paste to Same Location

This function is used for pasting objects being copied to the same location on the canvas; this is best utilized to specify more parameters for one visible object, such as a button.

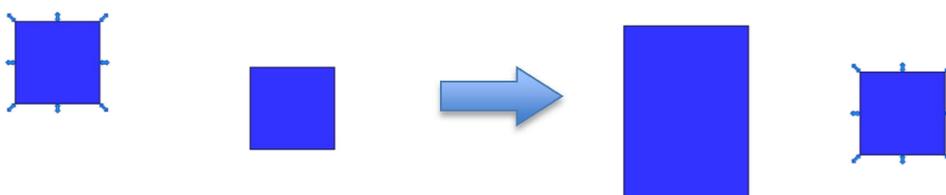
Begin by creating an object and click on *Edit -> Copy*. Now paste the object being copied by clicking on *Edit -> Paste to same location* (you can also right-click on the object and select this function). This operation can be repeated as many times as you need.

*Note: You can verify that an object has been pasted successfully by moving it aside and then moving back by clicking on *Edit -> Undo*.*

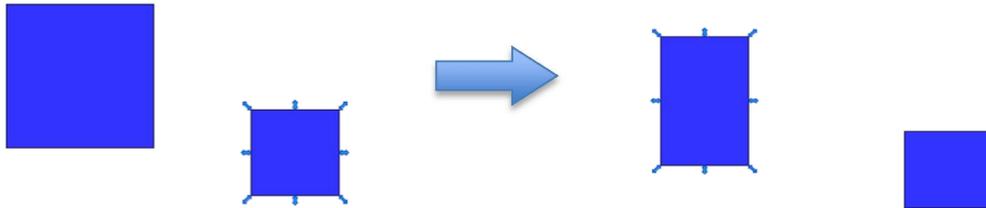
### Paste Dimensions

This feature allows you to paste specific dimensions of currently selected objects. Select an object whose dimensions you want to copy. Then select the object to which you want to apply the dimensions by clicking on *Edit -> Paste dimensions*.

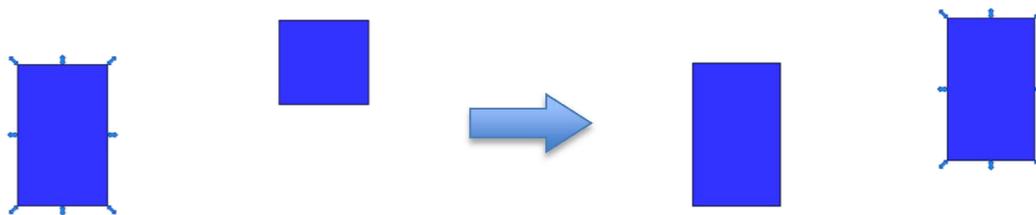
**Paste size** - this function will paste all dimensions



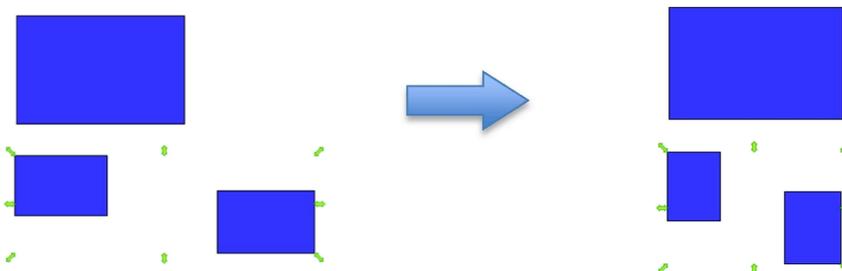
*Paste width* - this function will paste only the width of an object



*Paste height* - this function will paste only the height of an object



When the function *Paste dimensions* is being applied to multiple selected objects, the dimensions are transferred to the selection bounding box and the space around the objects is accordingly scaled.

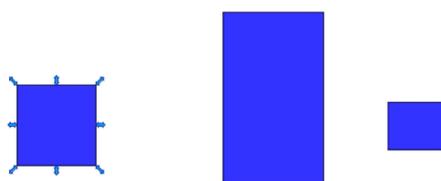


### *Paste size separately*

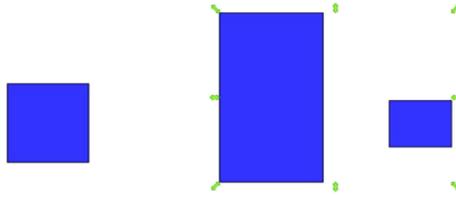
This function allows you to apply dimension to multiple objects at one time.

In the following example, we will copy the square on the left and then resize the rectangles:

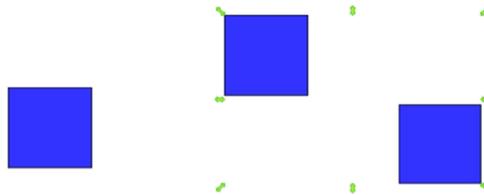
- 1) Create multiple objects and then copy the size of the object you want to apply to the remaining objects.



- 2) Select the remaining objects.



- 3) Open the *Edit* menu and select *Paste dimensions - Paste size separately* (you can also right-click on the object and select this function). All objects are now the same size.



#### *Paste width separately*

This function allows you to change the width of multiple objects at once, as described in the previous example.

#### *Paste height separately*

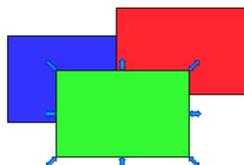
This function allows you to change the height of multiple objects at once, as described in the previous example.

## 6.17 Object's Order

Graphics objects use a hierarchical order. If two objects overlap, the one located higher overlaps the one located below it. You can change the order of the objects by raising them up to the top or lowering them down the bottom of the layer.

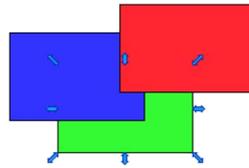
#### *Example:*

- 1) Create a new object and use *Edit -> Copy* and *Edit -> Paste on same location* twice. Now you have three rectangles of the same size. For easier visualization, fill each object with a different color (function Properties - Fill - Color) and drag the objects so they partially overlap.



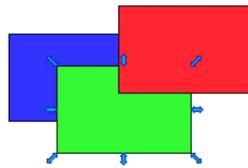
- 2) Now click on *Transforms -> Order* to change the objects' order and use one of the functions below:

*Lower to the background* - moves a selected object into the background so that all other objects are on top of it



*Raise to the top* - moves a selected object to the top so that all other objects are beneath it

*Raise* - moves a selected object one level up

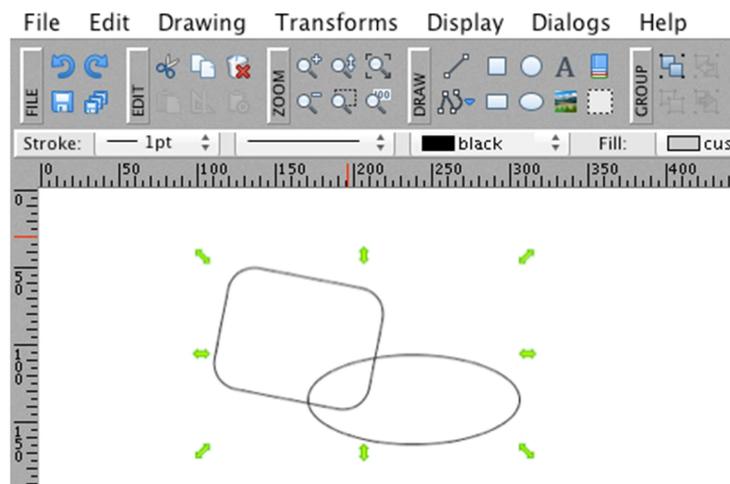


*Lower* - moves an object down one level down

## 6.18 Grouping

### Grouping Elements

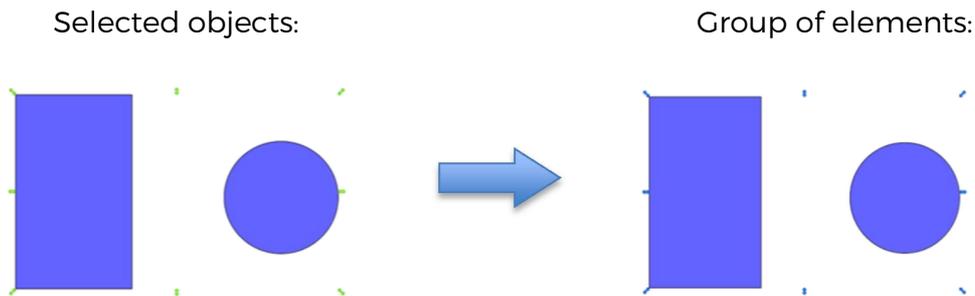
You can compose your graphics from multiple elements. Moving or copying these graphics might be difficult, but to simplify these operations you can group multiple objects together so you can use them as a single unit.



	Group - groups selected objects together
	Ungroup - ungroups selected objects

	Enter group - allows changes to objects inside the group without ungrouping
	Exit group - ends enter group mode

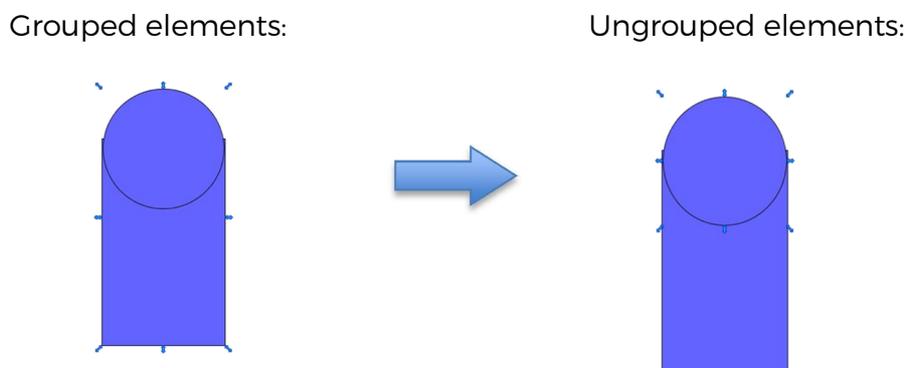
Select the objects to be grouped and click on the icon  in the toolbar (or select it from the menu *Edit*). The objects are now grouped and behave as one object; the group can be moved, resized, rotated, skewed, etc.



### Ungrouping Elements

This function is used to divide grouped objects back into independent elements.

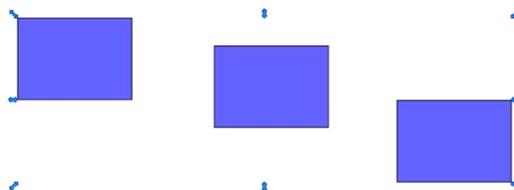
Select the group of objects and click on the icon  in the toolbar (or select it from the *Edit* menu), and the object will divide into its elements.



### Enter Group

With this function, you can work with individual objects that are grouped without needing to ungroup them first. Therefore, you can work with each object individually without ungrouping.

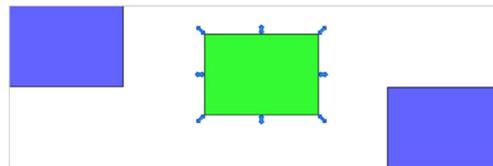
- 1) Select a group of objects.



- 2) Click on *Edit* (shortcut **Ctrl+E**) and select *Enter group* (or use the appropriate icon in the GUI Toolbar).



3) Now you can work with any object in the group individually.



### Exit Group

Once you have finished working with the objects in the group, you can leave the group by clicking on *Edit -> Exit group* (or shortcut **Ctrl+Shift+E** or by using the appropriate icon in the GUI Toolbar).

## 6.19 Repeated Actions Mode



This mode allows you to continue drawing the same type of objects without having to re-select the object type or shape each time you draw it.

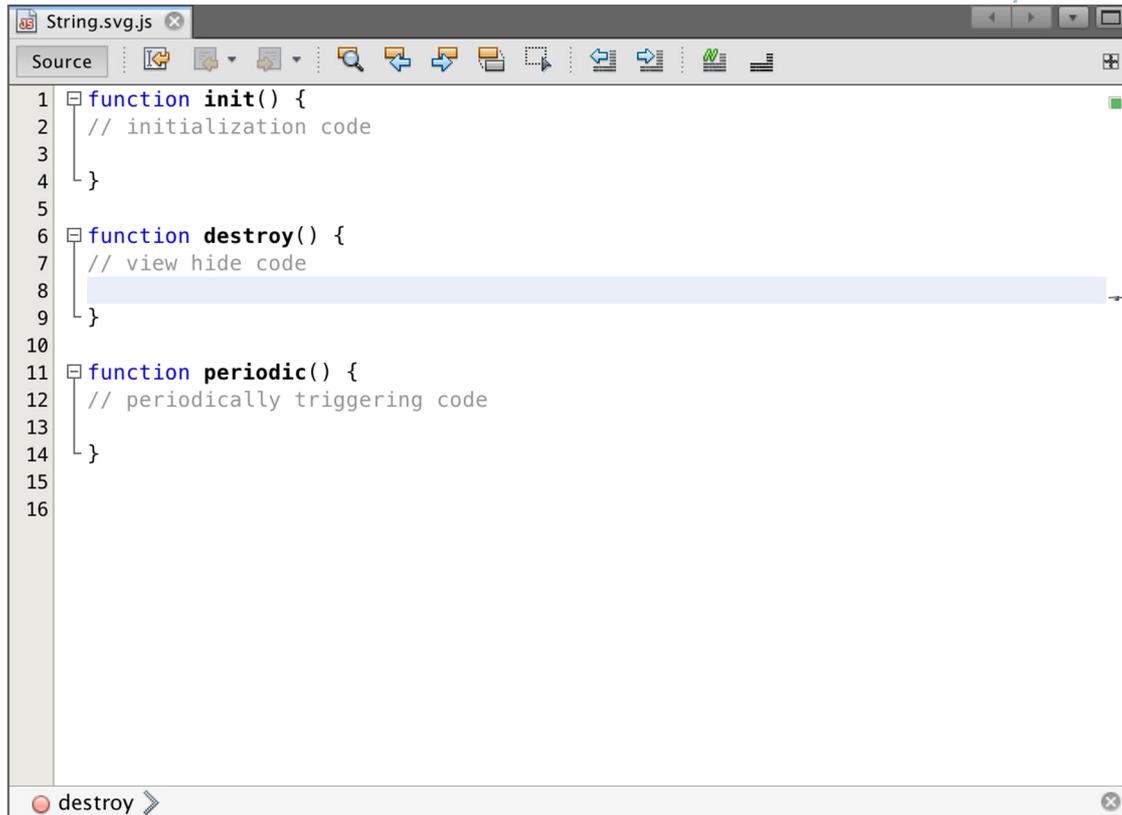
Click on the **Repeated actions mode** icon in the lower options bar and select an object to draw.

## 6.20 View Scripts



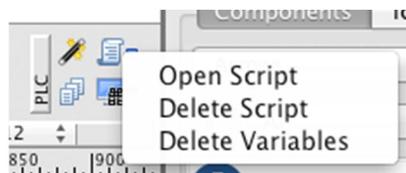
Click on the **Scripts** icon in the toolbar to open a new tab with the script editor. The script editor will open:

Watch video describing this functionality: <https://www.youtube.com/watch?v=t7BtSyaPVss>



```
1 function init() {
2   // initialization code
3
4 }
5
6 function destroy() {
7   // view hide code
8
9 }
10
11 function periodic() {
12   // periodically triggering code
13
14 }
15
16
```

The **Script** icon has a context menu with which you can open a current view script in the editor, delete an existing script, or clear all defined script variables. Use the right-click to activate the menu.

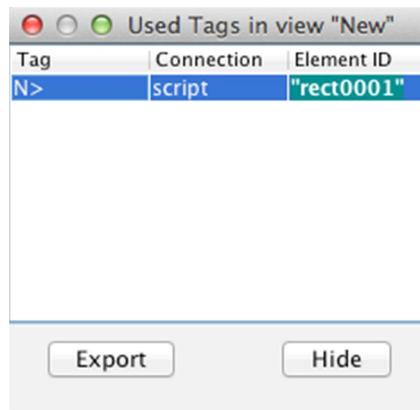


A detailed description of View Scripts can be found in the corresponding chapter: [View Scripts](#).

## 6.21 Used Tags



This function lists all tags used in an active view.



If you click on the item in the *Element ID* column, the graphical object with this ID will be automatically selected.

## 6.22 Zoom on Zone



This function zooms in on a selected area within the whole workspace. First, click on this icon in the toolbar, then click and drag the mouse on the area on which you want to zoom.

## 6.23 Undo and Redo



The **Undo** button allows you to revert your last action. You can find it in the toolbar or in the *Edit* menu. You can also right-click on an object and select this function.

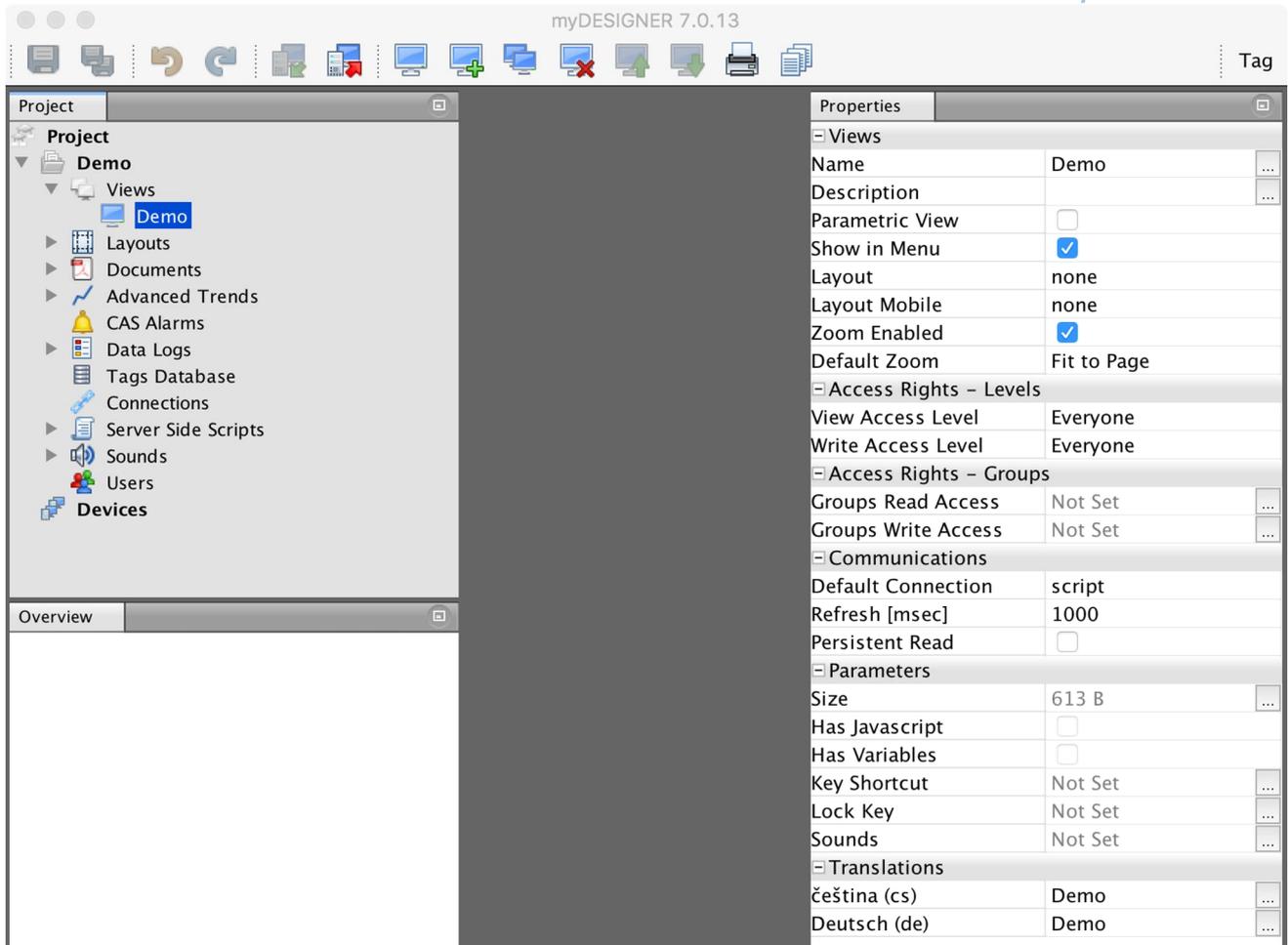


The **Redo** button allows you to redo an action after using the *Undo* function. You can find it in the toolbar or in the *Edit* menu. You can also right-click on an object and select this function.

*Note: If these buttons are grayed out, then it is not possible to use them at that moment.*

## 6.24 View Properties

If you select a view from the project tree, you will see its properties in the *Properties window*.



Now look at the properties window. It has several sections.

## View Properties

**Name:** *Name of the View.*

**Description:** *View Description.*

**Parametric View:**

Option to create a parametric view - use parametric views when you need multiple views different only in a data source; such sources are specified by the received index when a parametric view is prompted (see *Commands - Open command* and *Parametric window*).

**Show in menu:** Whether to show this view in menu

**Layout:** *Active layout for this view. For more info, please look at section [Layout Views](#).*

**Layout Mobile:** *Active layout for mobile devices. For more info, please look at section [Layout Views](#).*

**Zoom Enabled:** *Enable/Disable zoom options for this view*

**Default Zoom:** Default zoom for this view. Options are Fit to Page, Full Width, and Original Size

## Access Rights Levels

- View Access Group:*** minimal level of access for viewing. Enter a level between zero and nine: None, 0, 1,..., 9. For more info, please look at the section [UserAccesses](#).
- Write Access Group:*** level of access for writing to the PLC. Enter a level between zero and nine: None, 0, 1, ..., 9. For more info, please look at the section [UserAccesses](#).

## Access Rights Groups

- Groups Read Access:*** List of user groups to allow view access. For more info, please look at the section [UserAccesses](#).
- Groups Write Access:*** List of user groups to allow write access. For more info, please look at the section [UserAccesses](#).

## Communications

- Persistent Read:*** If this option is enabled, mySCADA will read all the data from the PLCs, even if the view is not being viewed by anyone. This option is useful for fast view refresh when loaded; however, heavy usage can lead to a slower overall reaction of the system as more data is read from the PLCs continuously.
- Connection:*** List of PLC connections defined in the current project. This is a default connection - if you enter the tag and do not specify the connection, this connection will be assumed as *default*.
- Refresh:*** *Refresh rate of the View in milliseconds.*

## Parameters

- Size:*** Displays the view size in kilobytes [kB].
- Has JavaScript:*** Checks if the view contains script functions.
- Has Variables:*** Checks if the view has view script variables.
- Key Shortcuts:*** Allows use of key shortcuts in the view (see the chapter [Commands](#))
- Lock Keys:*** If enabled, set commands will be active only if the lock key is pressed.
- Sounds:*** Specifies the sound animations used in the view (see the chapter [Commands](#))

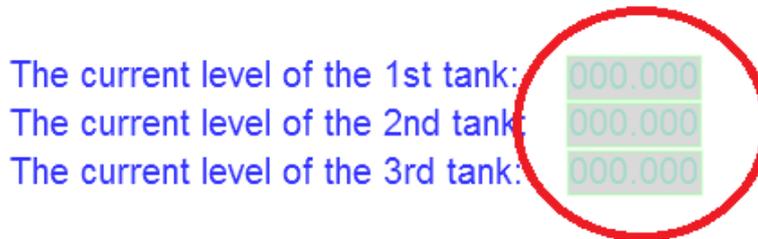
## Translations

- Language** *You can provide language translations for the view name and description. To see this option, you must activate at least one language in the Languages option in your project. For more details, please see chapter [Multilanguage support](#)*

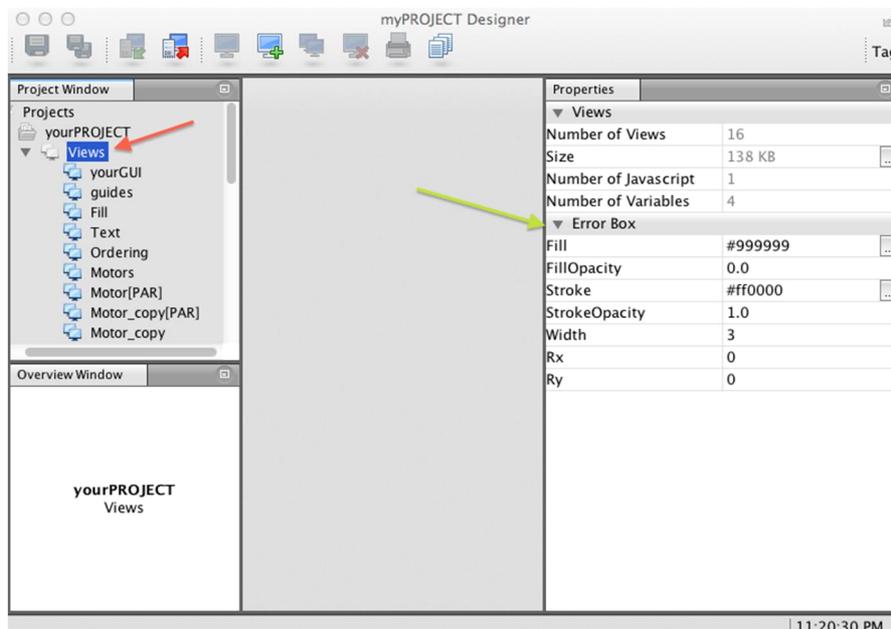
## Error Box Properties

You can define a graphic representation of an “offline” error for objects tied with the tags for animation. When such tags cannot be read from the PLCs, *mySCADA* will display an error box around such objects.

In the *Error Box* properties, you can define the look of the error boxes in your project.



Select the *Views* folder and its properties will appear in the *Properties* window:



Set the fill color, fill opacity, frame color, frame thickness, and opacity of the *Error Box*. With the parameters *Rx* and *Ry*, you can change the softness of the *Error Box* corners.

Error Box	
Fill	#999999
FillOpacity	0.75
Stroke	#ff9900
StrokeOpacity	0.75
Width	3
Rx	0
Ry	0

For example, if you fill the *Error Box* properties with the parameters below, the communication error will be displayed as follows:

Error Box	
Fill	#3333FF
FillOpacity	0.85
Stroke	#FF0000
StrokeOpacity	0.85
Width	6
Rx	3
Ry	3



The current level of the 1st tank: 

The current level of the 2nd tank: 

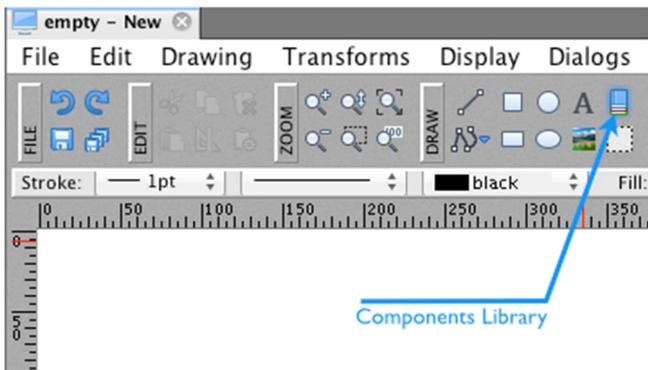
The current level of the 3rd tank: 

# 7 Components

The *Components* library is a collection of high-quality artwork and ready-made graphic symbols that you can simply edit and integrate into your views. Each component represents a fully functional graphical object with easily editable properties, including the ability to be directly connected to the PLC tags without the need to use animations.

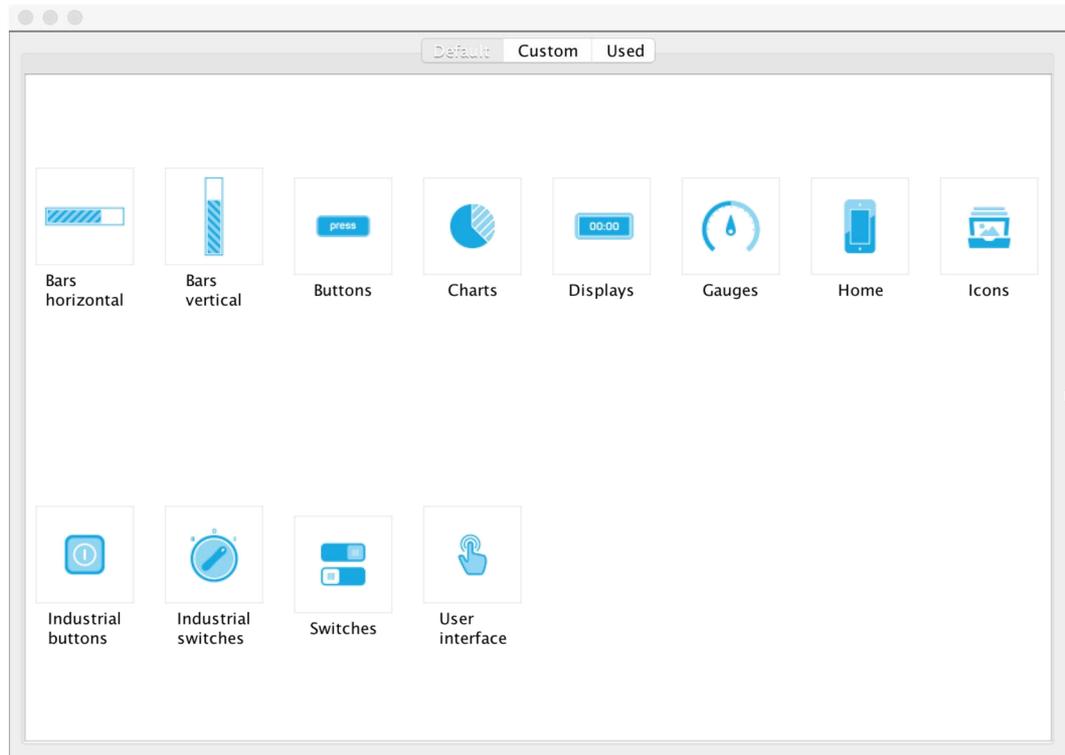
Watch video describing this functionality: <https://www.youtube.com/watch?v=tYh4jD-2f2M>

The **Components** library icon is located in the DRAW section of the main toolbar.



Upon opening the *Component* library, you will see a dialog window consisting of three main section tabs:

- Default** - contains original factory components that you can choose from and customize to your needs
- Custom** - contains all your already customized components
- Used** - contains all the components you have used in your project (i.e. you can change the component properties for all used components in the project at one time)

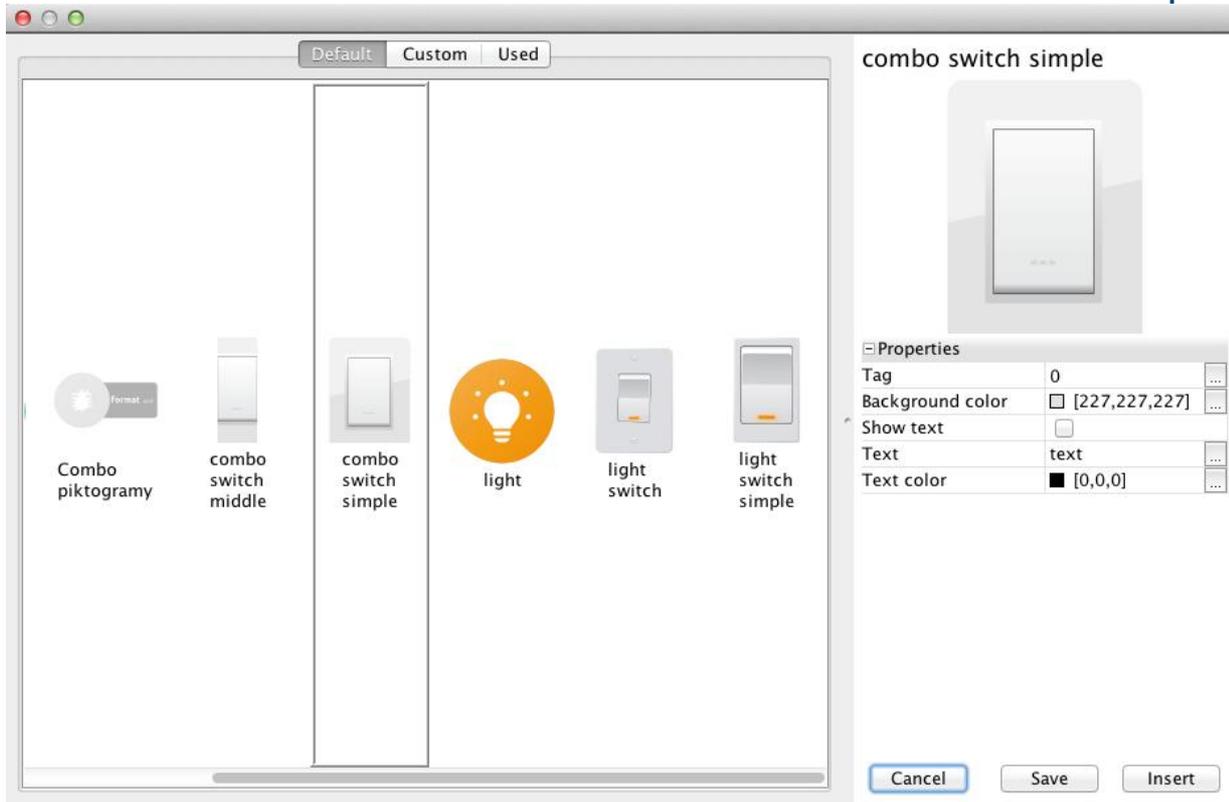


## 7.1 Default Components

This section contains 10 groups of basic and the most common factory components.

Click on the components group of your interest and select the component you need for your visualization.

After you have selected your component, you will see its preview on the right side of the dialog window. Below the component preview, there are editable properties of the component such as *Tag*, *Min* and *Max* limits, background color, pointer color, text size, etc. Note that each component has slightly different properties to edit.

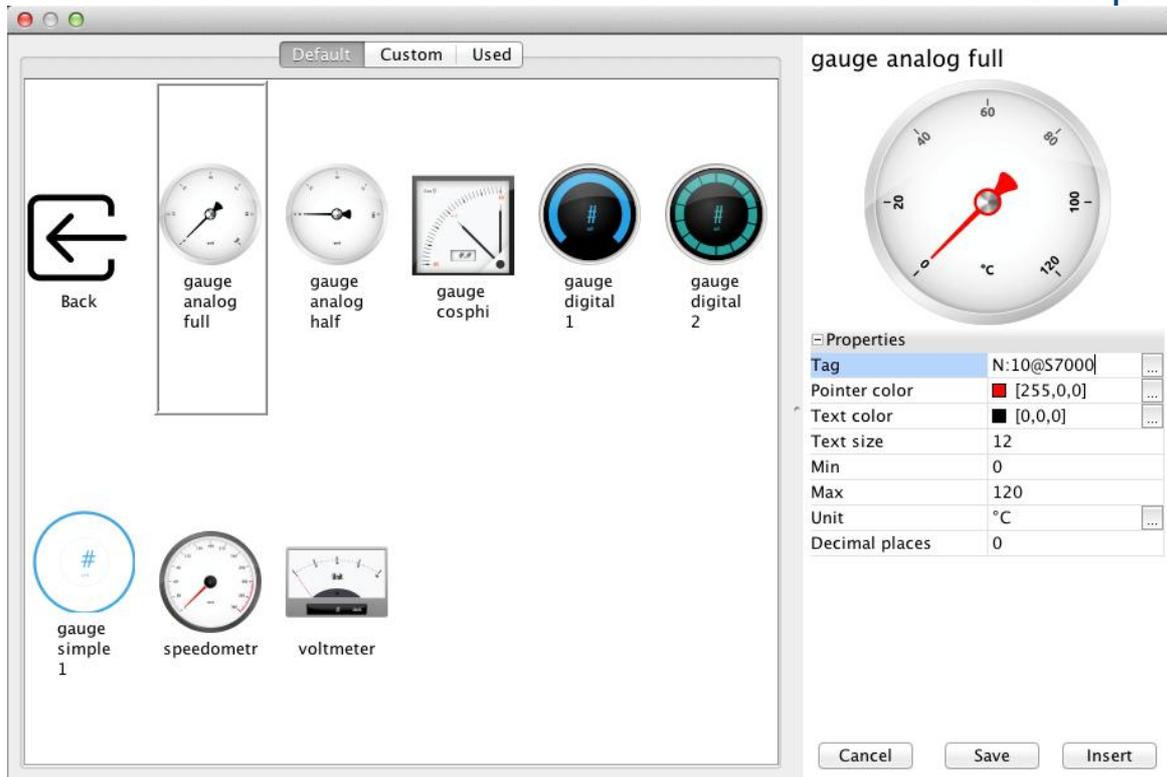


If you navigate to the bottom right corner of the dialog window, you will see three buttons:

- **Cancel** – closes the component dialog window without saving the changes
- **Save** – saves the customized component in the user's **Custom** section
- **Insert** – inserts your component into the View

#### *Example:*

- 1) Let us navigate to the *Gauges* group and select the analog gauge called *Gauge Analog Full*, and then navigate to the *Properties* section in the right part of the dialog window.



- 2) Customize the component properties to your needs and click on:
- Insert** – to put the component straight into your view (without saving)
  - Save** – if you wish to save your component settings for later use

Another dialog window will appear:



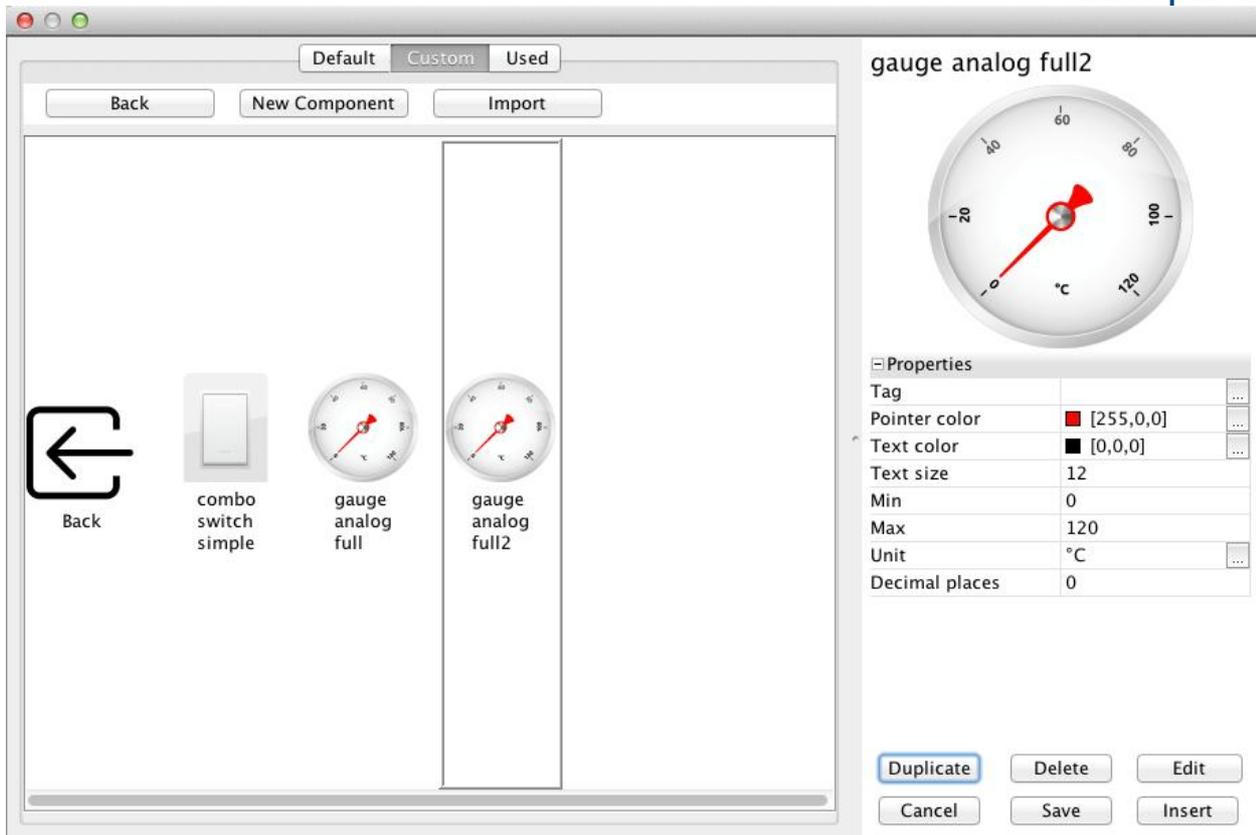
Write the name of the new category in the *Category* field, fill in the name of your customized component, and click on **OK**.

Your component will be saved and stored in the **Custom** section.

## 7.2 Custom Components

Watch video describing this functionality: <https://www.youtube.com/watch?v=tYh4jD-2f2M>

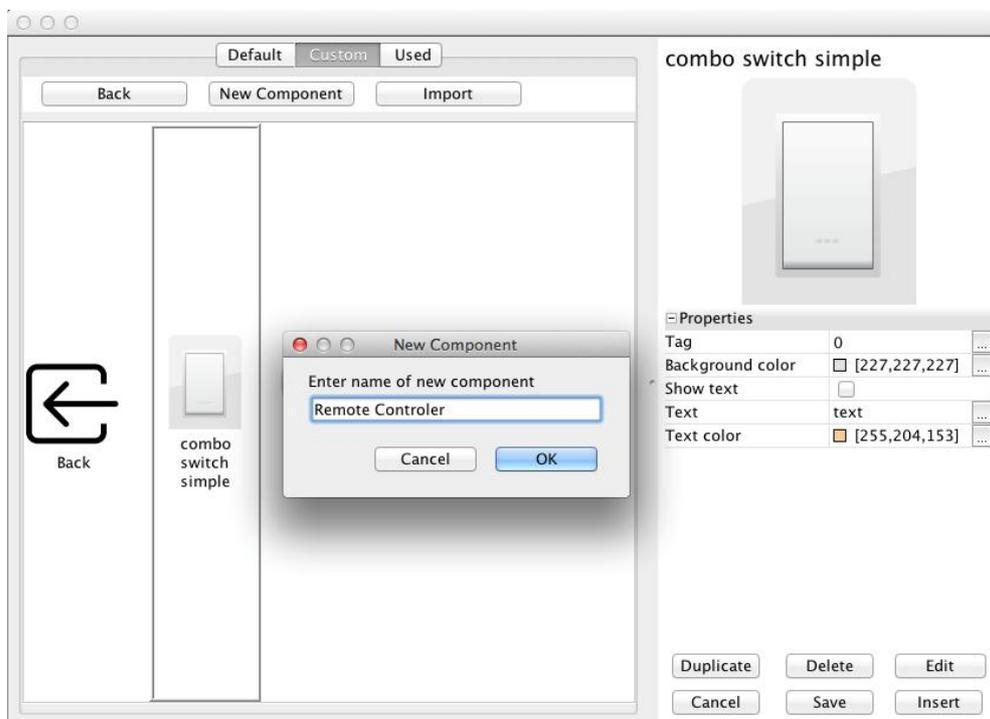
This section contains all your custom components. Here you can find all of your saved components, create new components, or import new ready-made components from SVG files.



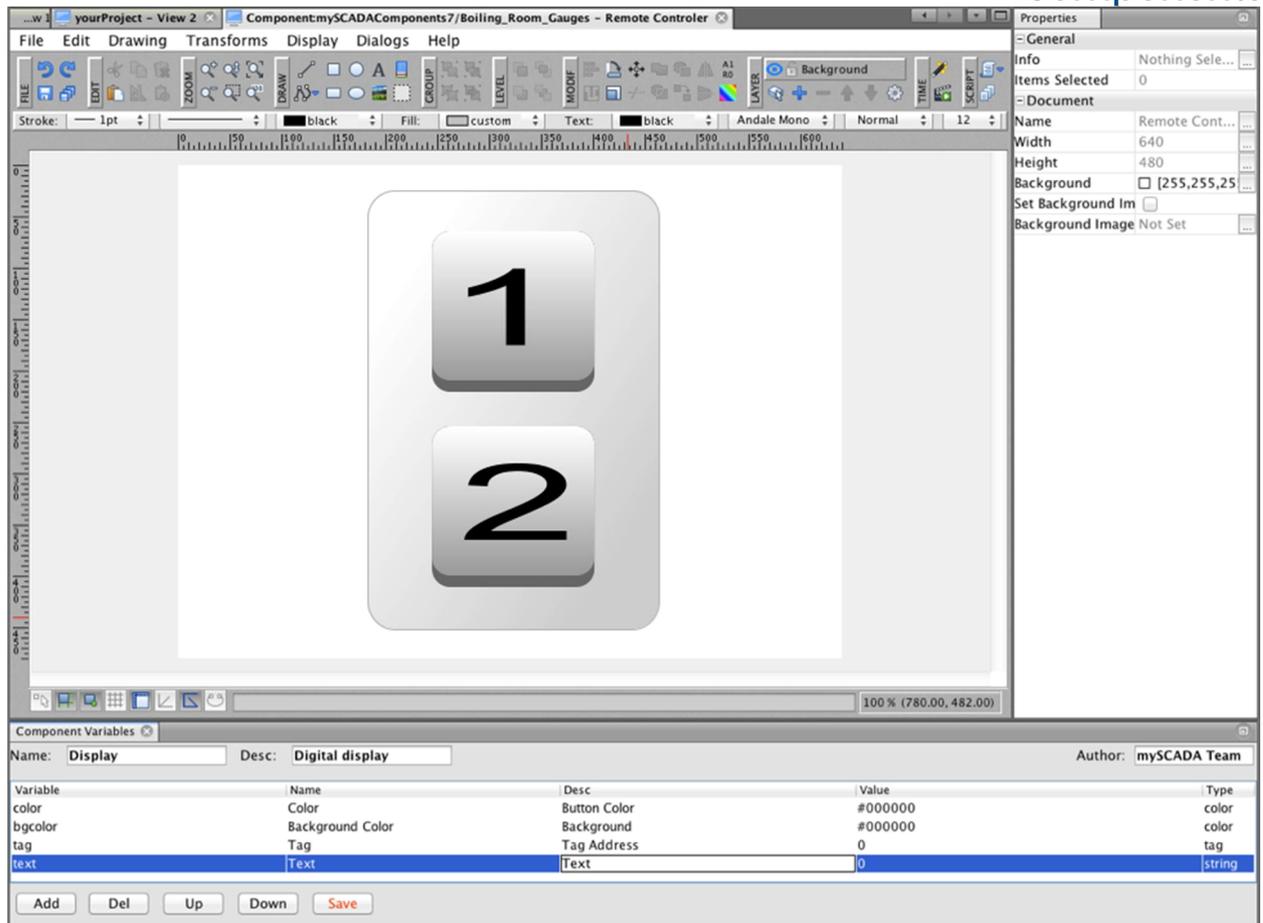
## Creating Components

This feature allows you to create, customize, and implement your own components.

- 1) Click on the **New Component** tab and enter the name of your new component, then click on **OK**.

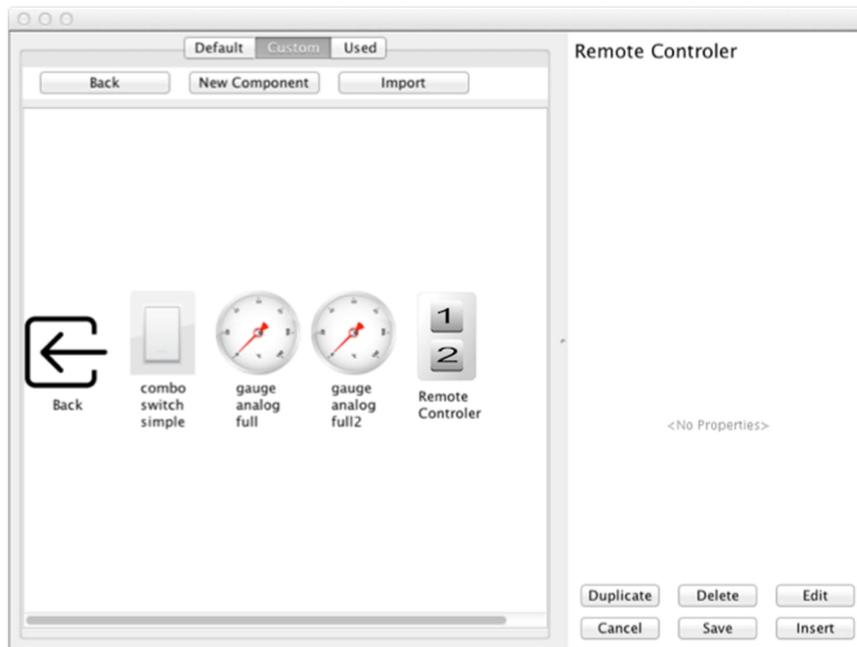


- 2) Select the new empty component from the list and click on the **Edit** button to prompt a new edit window (*identical to the drawing canvas, see more in the chapter [Drawing](#)*).



In the very bottom part of the editing window, you can set the component variables, which will later become its settable properties (i.e. tag, background color, min & max values, etc.)

- When you have finished creating your component, click on the **Save** button at the bottom and close the editing window. Your component will now be stored in the **Custom** section. All components in the **Custom** section can be duplicated, deleted, or edited later with the use of the buttons located in the right bottom corner of the *Components* dialog window.



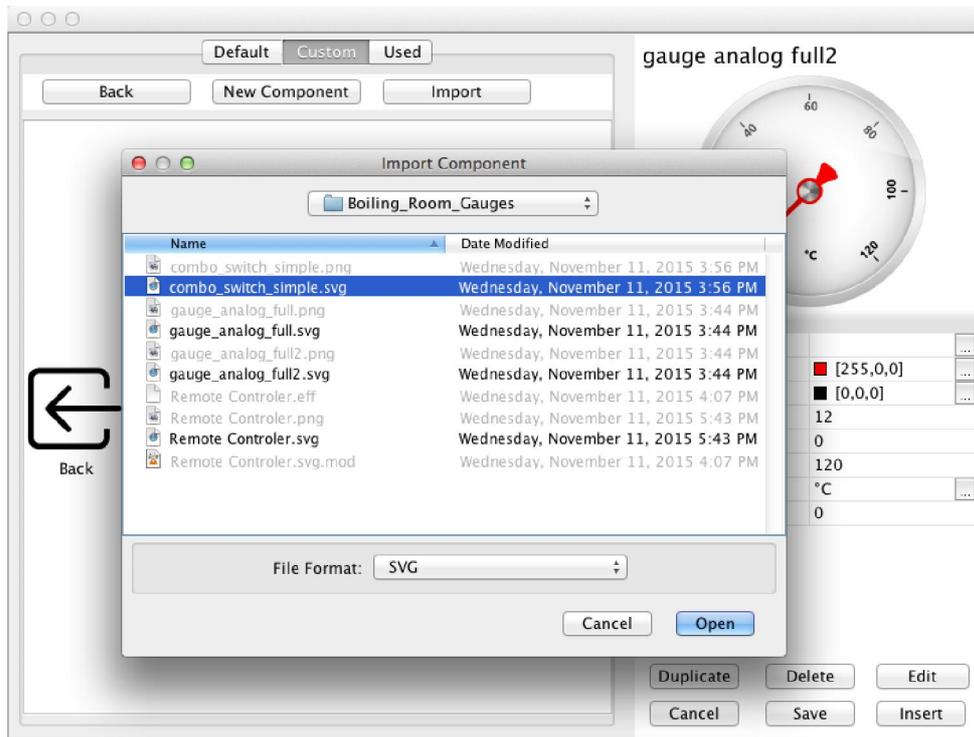
*Note:*

*All created and edited components in the Custom section are stored as SVG files, and you can find them in your user folder on the hard-drive.*

## Importing Components

The components can also be stored as individual SVG files, so you can import new components from outside resources. (They have to be in SVG format!)

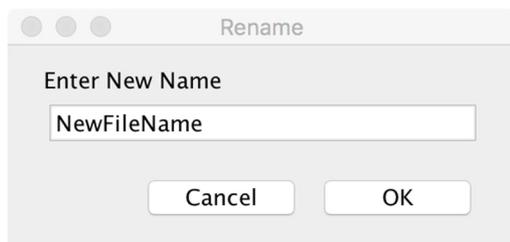
- 1) Navigate to the **Custom** tab and click on **Import**, which will prompt the dialog window *Import Component*.



- 2) Select the SVG file from the source folder and click on **OPEN** to integrate the new component into to your custom components list.

## Renaming Components

- 3) Select the component you want to rename. Click on the Rename button. A new dialog will appear:



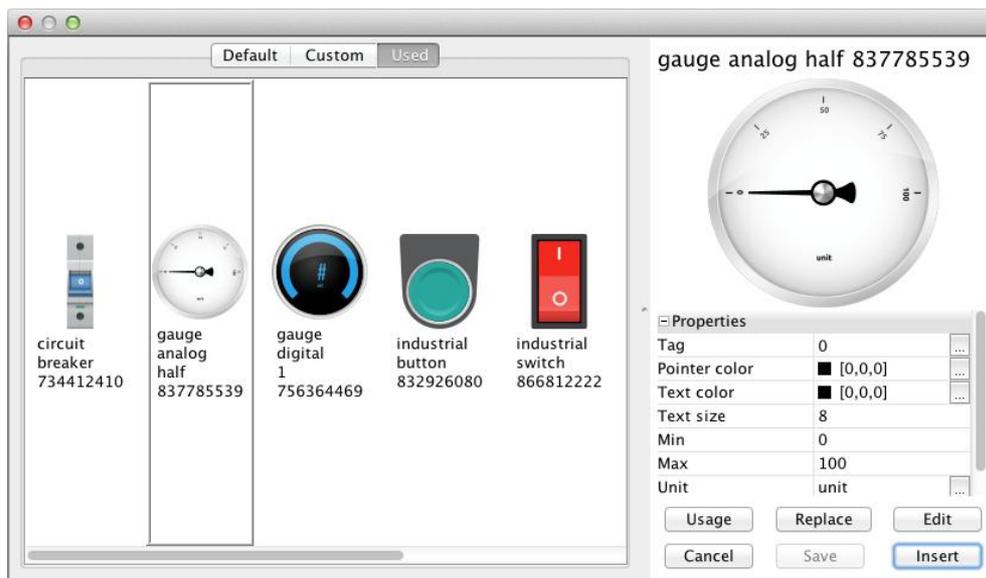
Enter a new name and hit OK.

## 7.3 Used Components

Watch video describing this functionality:

<https://www.youtube.com/watch?v=jeJNHUKfWEk>

This section shows all master components used in the whole project. If you change properties of the master components, then all subordinate components will change too.



If you navigate to the bottom right corner of the dialog window, you will see these buttons:

- **Usage** – shows details about a selected component, including its location, view name, id, used colors, min & max values, units, and format

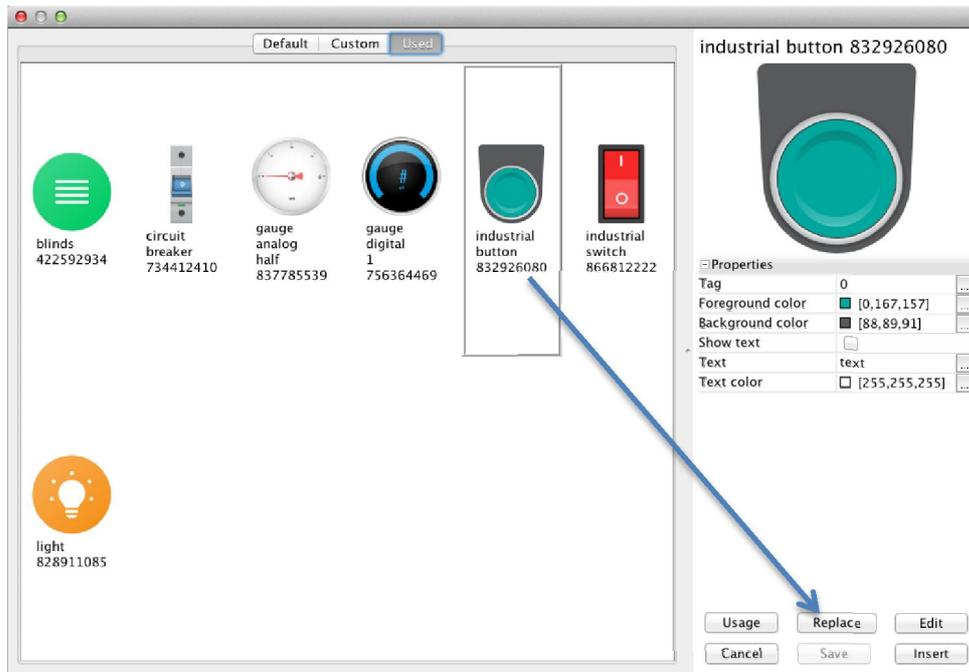
View	ID	Pointer color	Text color	Text size	Min	Max	Unit	Decimal pla...
Boilroom1	Comp90944...	Red	Blue	8	0	100	unit	0
Boilroom2	Comp46489...	Black	Black	8	0	100	unit	0

A 'Cancel' button is located at the bottom right of the dialog.

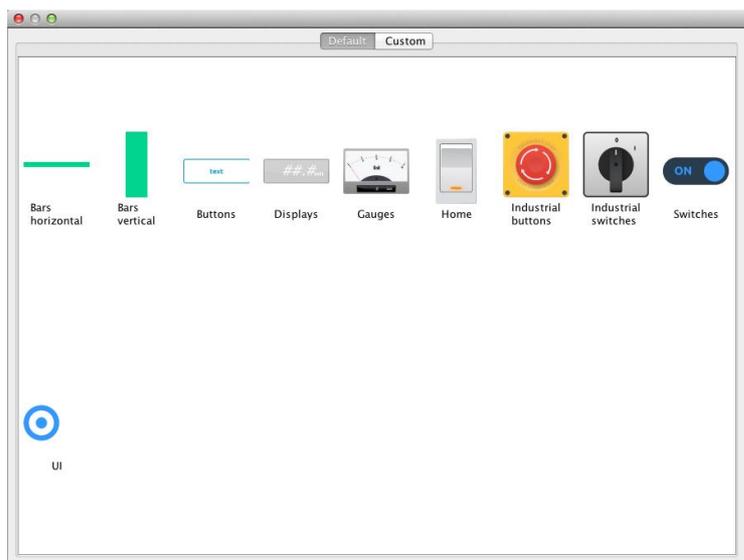
- **Replace** - with this feature you can replace master components with components with similar properties (i.e. buttons to buttons, gauges to gauges, etc.)
- **Edit** - opens up the component editor

## Replacing Components

- 1) In the *Component Library* navigate to the tab **Used**, select the master component you want to replace, and click on **Replace**.

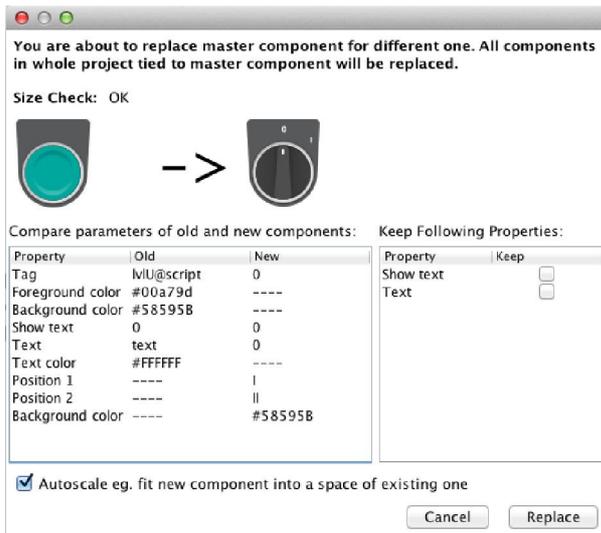


2) In the next dialog window, select the component to which you want to change.

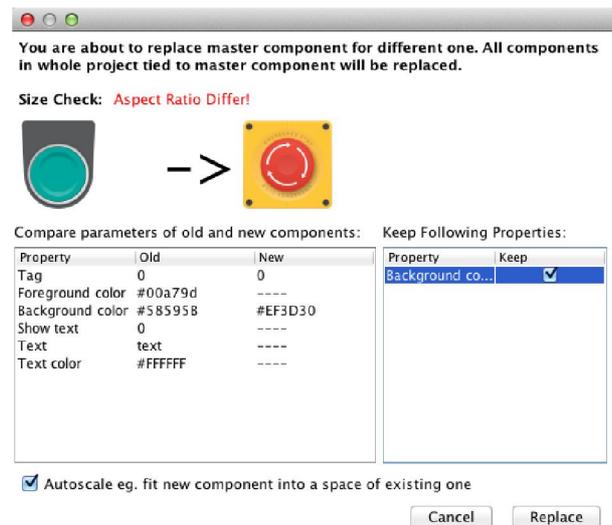


4) Once you have selected the replacement component, *myDESIGNER* will check if the aspect ratio matches for both components:

Sizes match



Sizes do not match

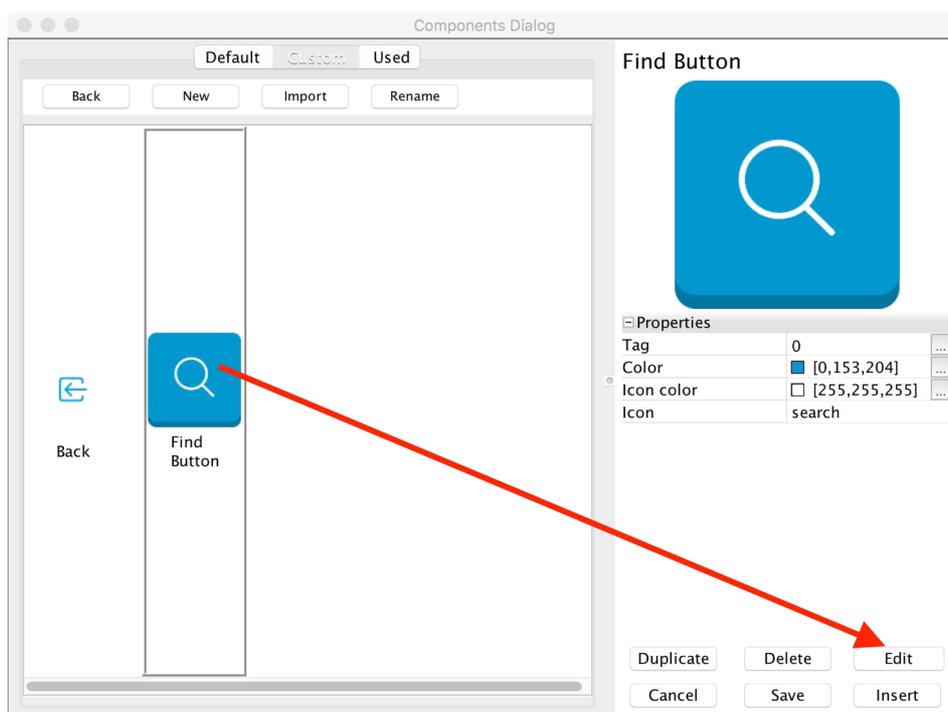


If the sizes do not match, the system can automatically scale the size of the replacement component to the size of the component being replaced.

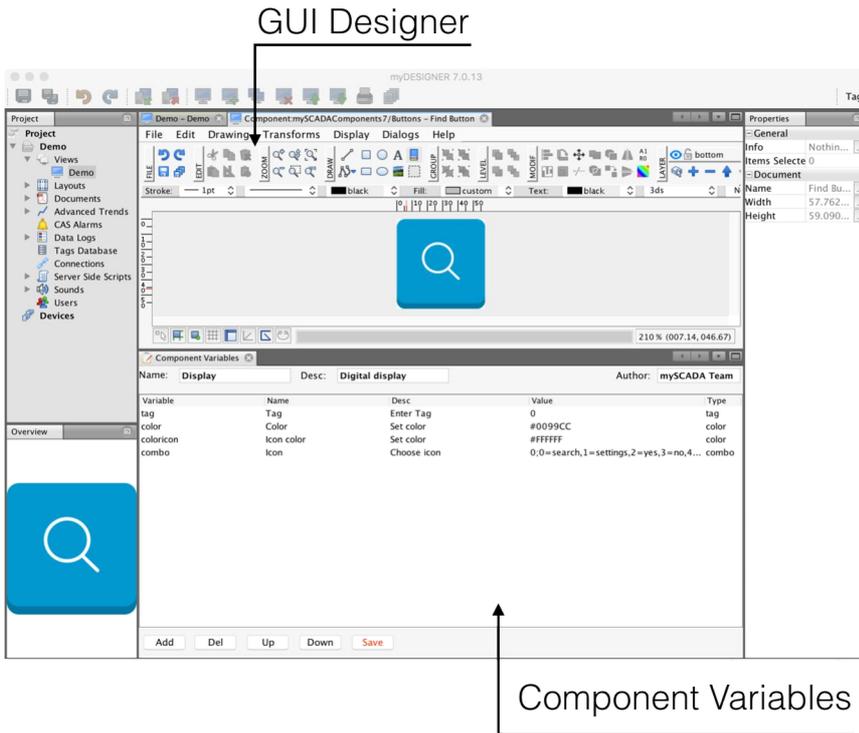
- In the properties below you can also select the parameters to remain unchanged and the ones to be changed. When you are satisfied with the settings, click on the **Replace** button.

## 7.4 Editing Components

When you create a new component or you would like to modify an existing one, you can simply edit it. To do so, please select the component you would like to edit and click on the edit button.

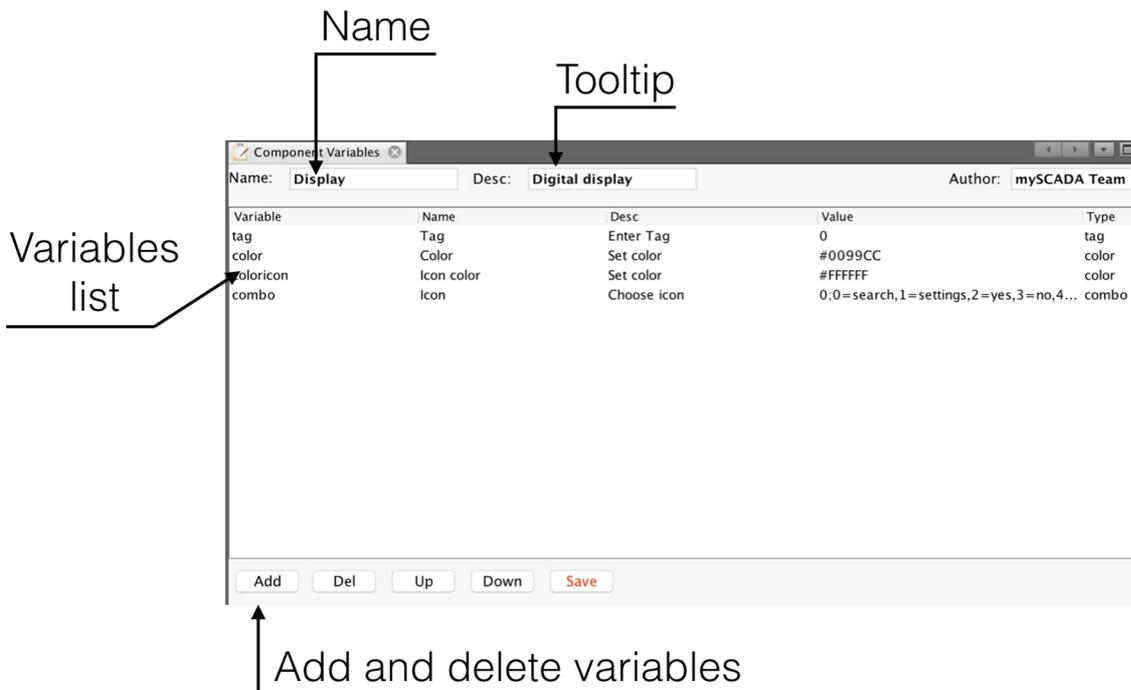


The Graphical designer opens, and you can design your component.

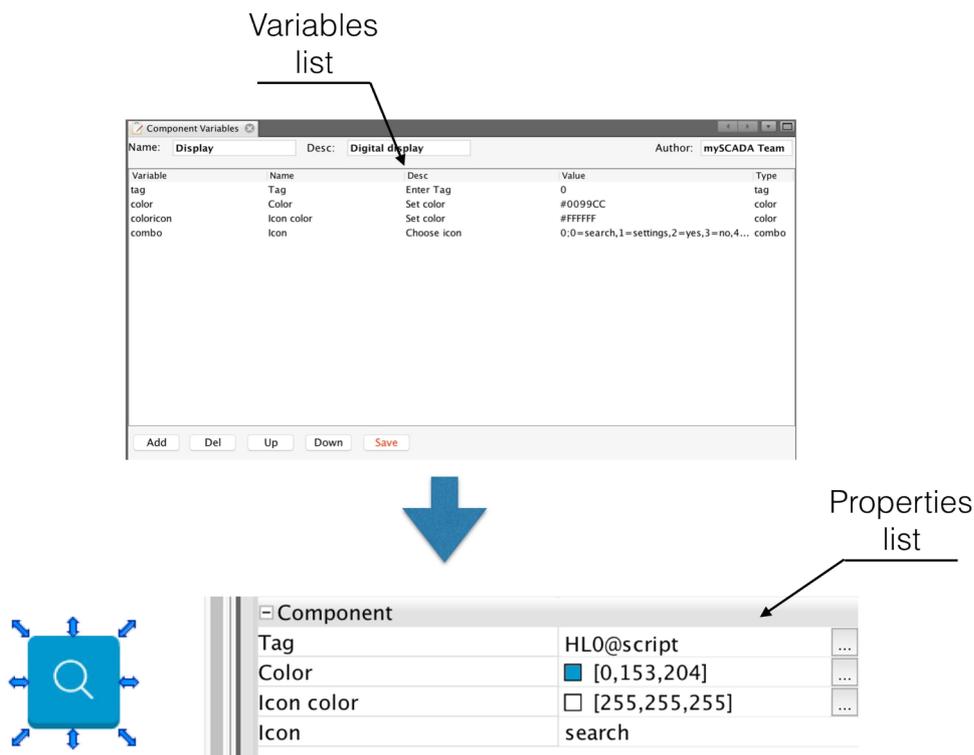


As you can see, the GUI Designer is exactly same as when you design your view. You can use all the functions you are used to. There is a new window: it is the *Component Variables Window*.

### Component Variables



This window contains all variables that control the visual appearance and behavior of the component. All variables you create here will be presented to the user as properties.



In the component variables window, you define all variables by specifying a name, short description, and type for each variable. These variables can contain tag values read from the PLC, color, and fill values for different elements, calculation results, etc.

Each component can have as many local variables as you need. You have to save your progress after the addition of each new variable by clicking on **Save** in the tab toolbar. The **Color** button opens the color palette when an appropriate type of variable is selected.

Each local variable has following properties:

Property	Description
Variable	Name of the variable as internal value: it is used as a reference in animations and properties of your component.
Name	Name of the variable as seen by the user: this will be shown in the component properties.
Desc	Description of the variable: description is visible as a tooltip when the user hovers over the name of the variable.
Value	Variable default value: for tag type variable, please leave empty.

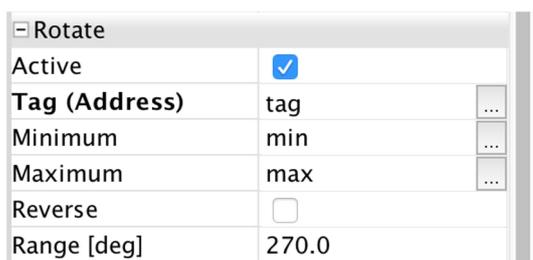
Property	Description
Type	<p>Supported types of variables are:</p> <ol style="list-style-type: none"> <li>1. <i>Tag</i></li> <li>2. <i>String</i></li> <li>3. <i>Int</i></li> <li>4. <i>Double (Float)</i></li> <li>5. <i>Color</i></li> <li>6. <i>Boolean</i></li> <li>7. <i>Object</i></li> <li>8. <i>Combo Box</i></li> </ol>

You can refer to the local variable anywhere in the animations, effects, or properties across the edited component.

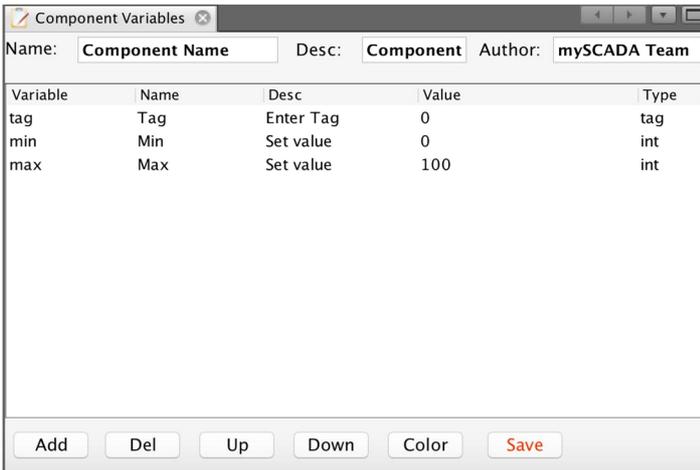
### Example

To explain how to use Component Variables, we will present you with a simple example. We will animate a handle of a gauge.

- Select a handle of the gauge
- Fill in the variables for rotate animation as follows:

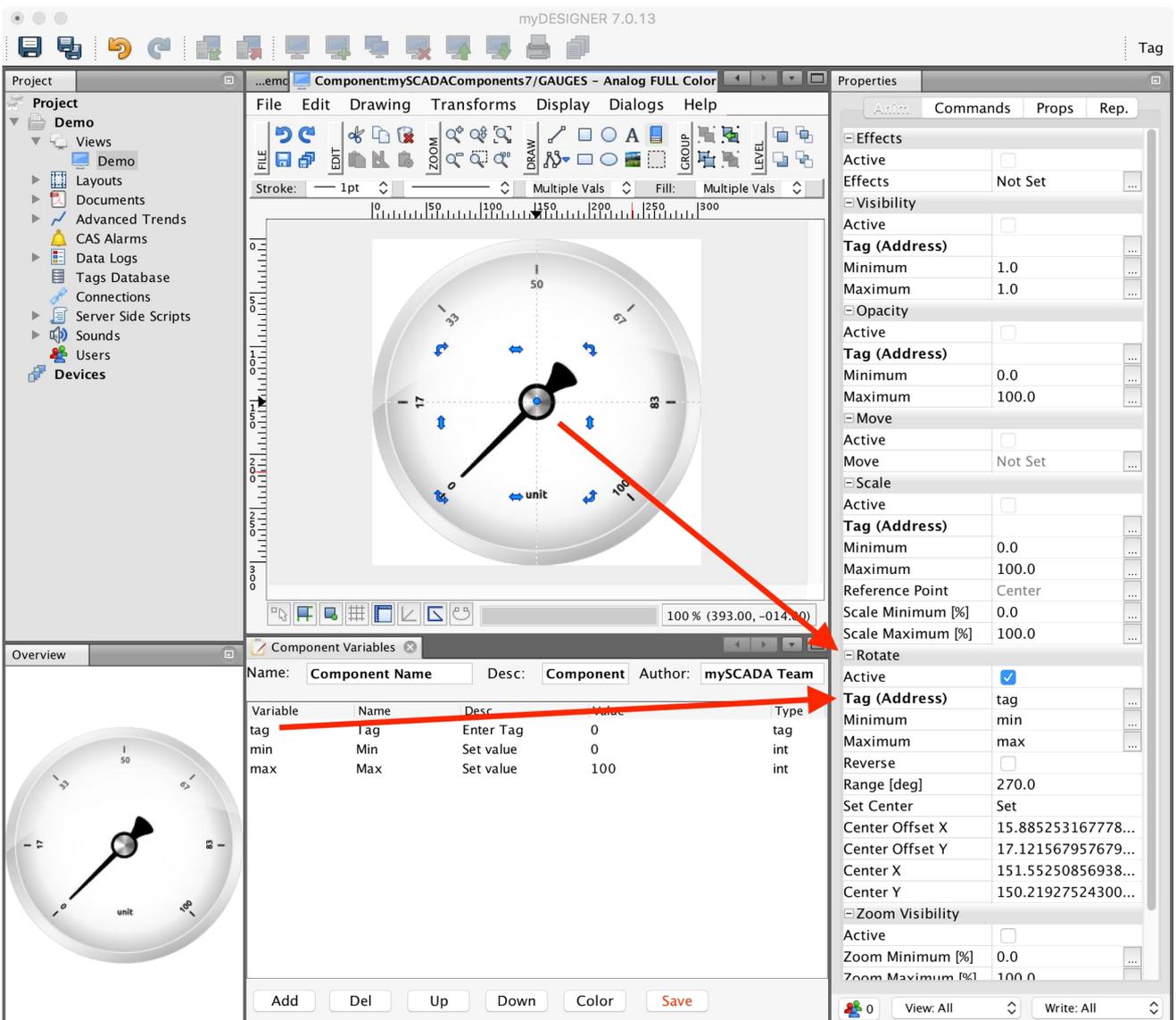


- Create the corresponding variables in the Component variables window



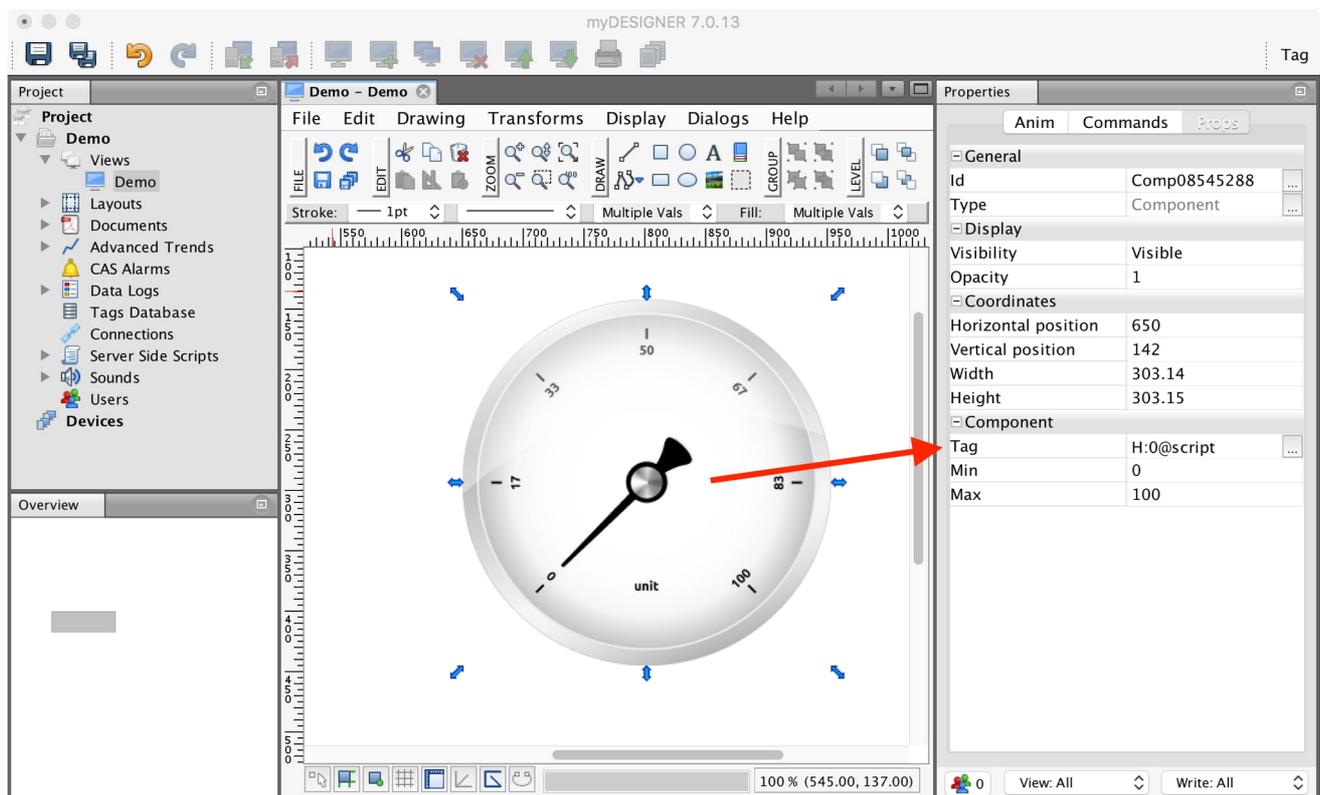
- Save

The complete process is shown in the following picture. You can see how we used the component variables directly in the Animation fields.



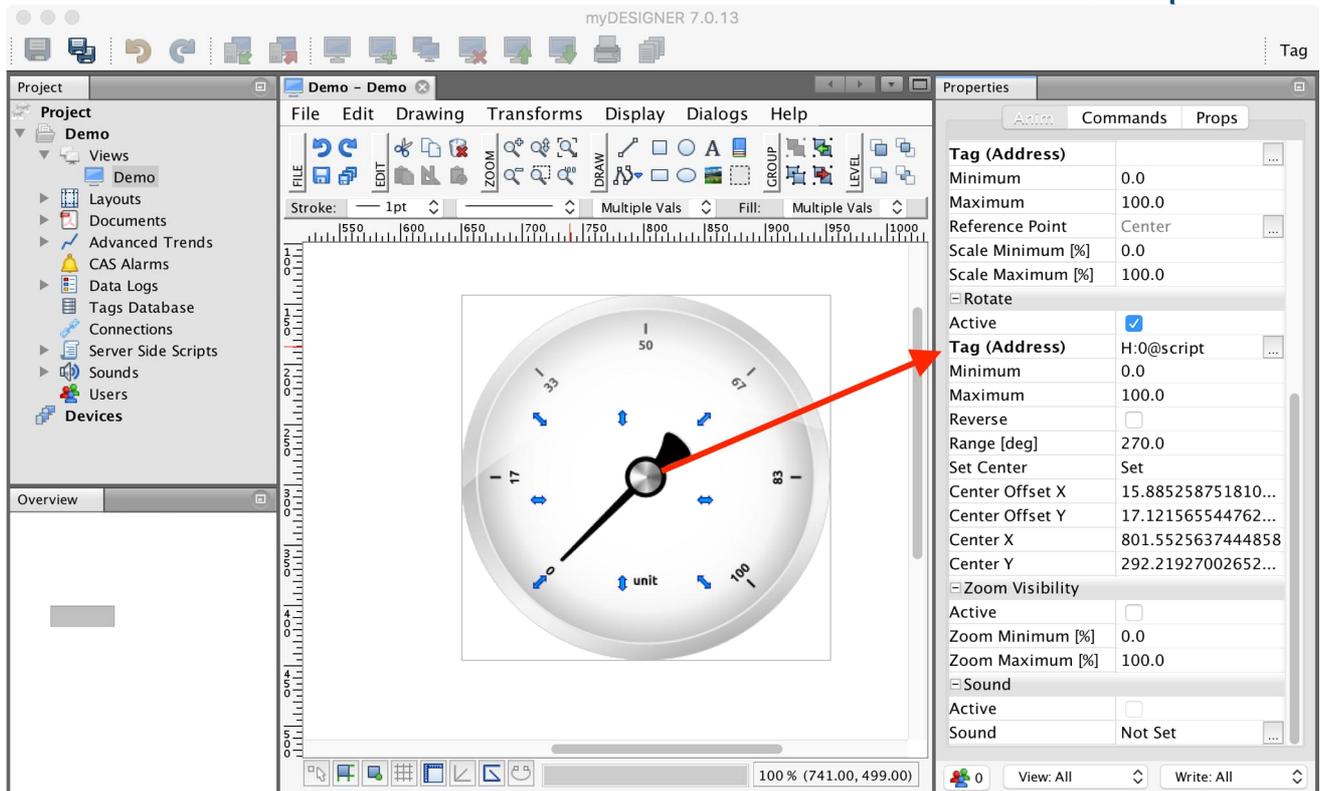
## Components

Now return to the view and test the new component. To do so, open your view, click on the Components Icon in the toolbar, and select your component. After pressing the Insert button, your component should be added to the view.



In the *Properties window* you can see the parameters of the component (e.g. the variables you have defined in the component variables window) - the name of the component variable "Tag", *Min* and *Max*.

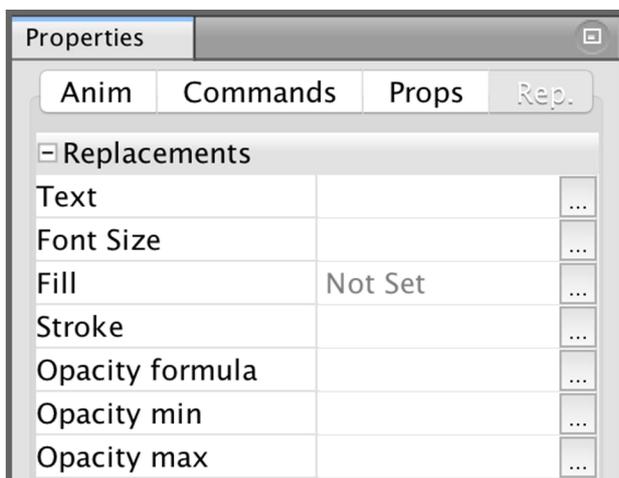
*TIP: To check if the tag address in the animation has been changed, right-click on the component and click on **Enter group**. Select the handle again, open its properties in the Properties window, and look at the Rotate animation.*



You can see that the tag address used for this animation has changed to the requested value.

## 7.5 Replacements

This feature should ease replacement of a predefined text in a component. You can change the text, font size, color, stroke, or opacity of the text element.

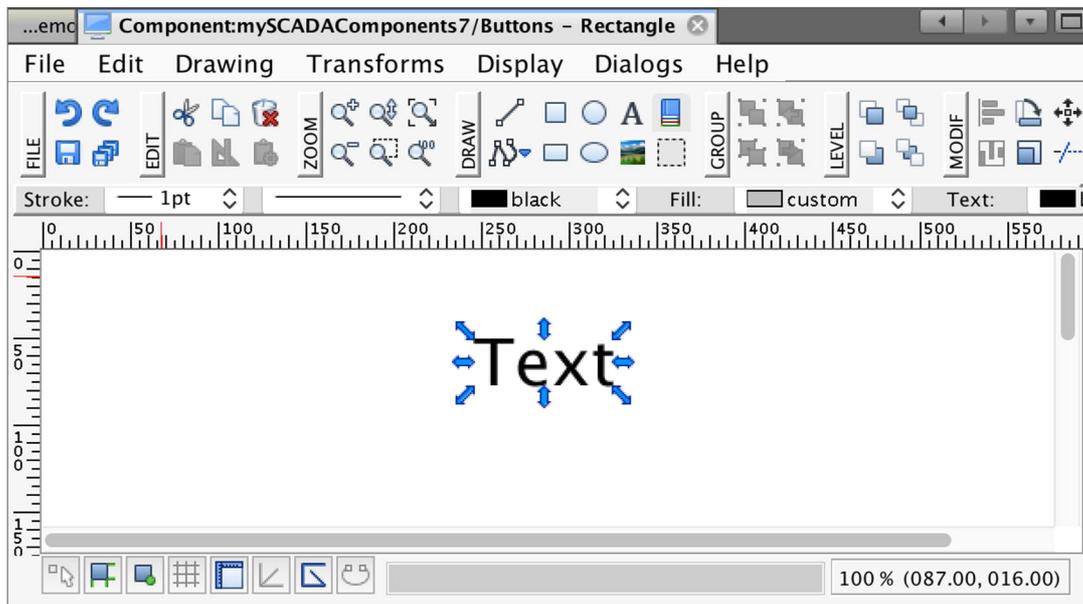


1) Start with the component variables again and create the following variables:

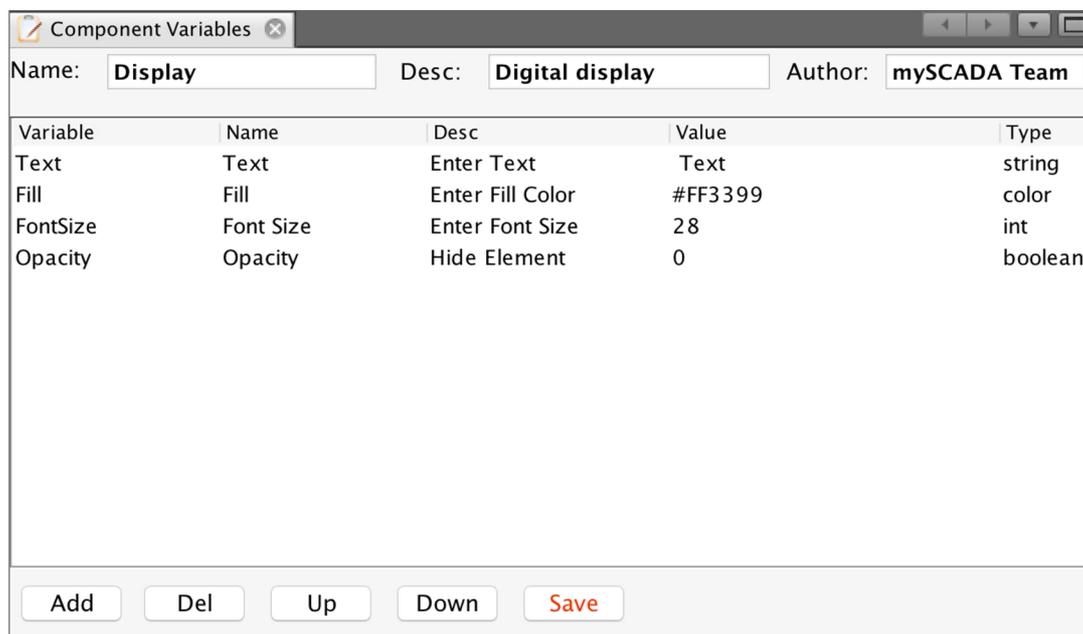
- Variable Text with type *String*
- Variable Font Size with type *Int*
- Fill with type *Color*
- Opacity with type *Boolean*

The *Opacity* here is similar to *Opacity* animation, which means that the text will be visible if the value of *Opacity* formula is within the *Opacity* min/max range.

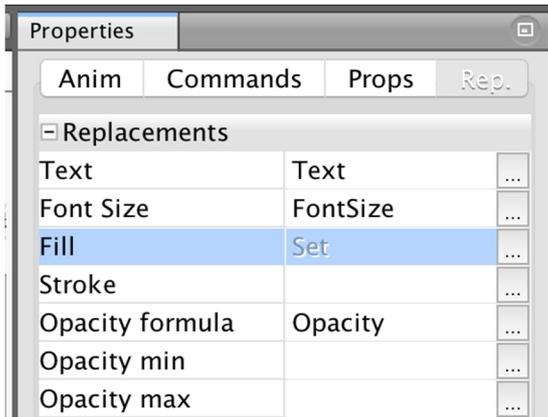
- 2) Now create a text element:



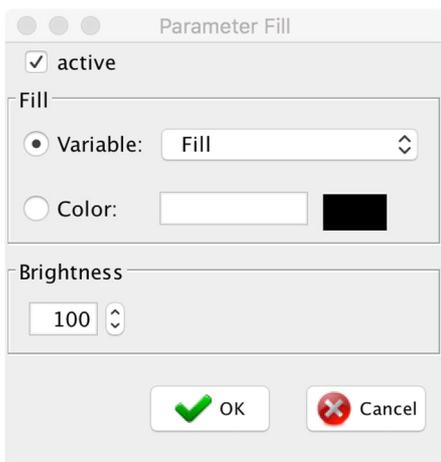
- 3) Fill in the *Component Variables*.



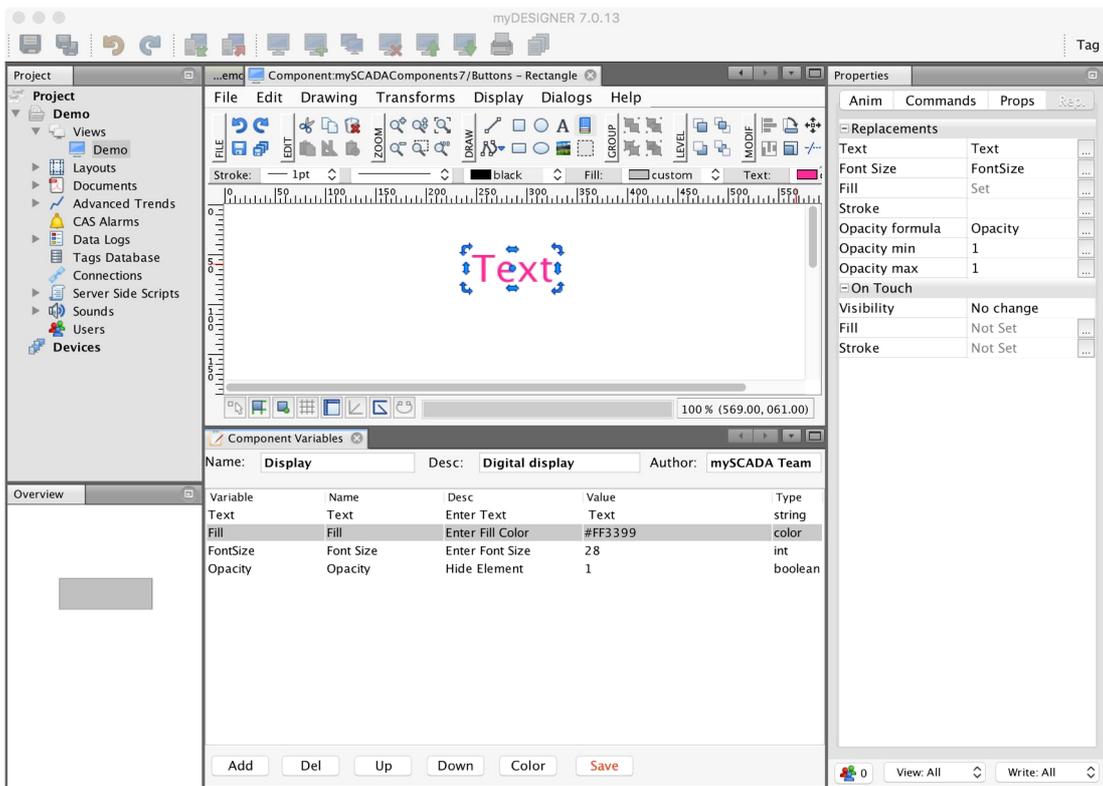
- 4) Fill in the *Replacement* -> *Text* in the *Properties* window. Note: The variable names are case sensitive.



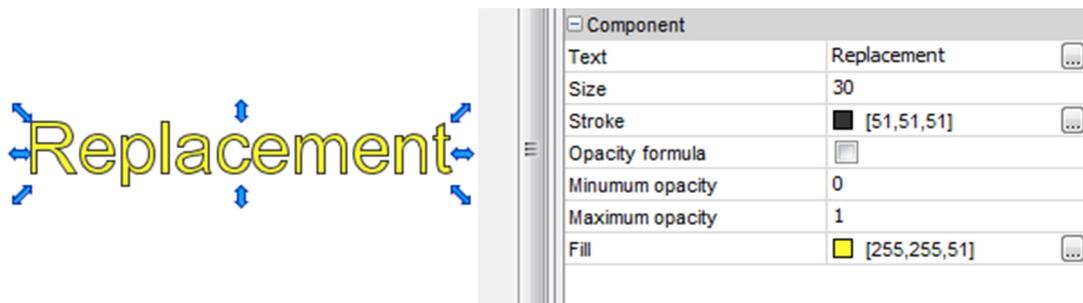
To set the Fill Replacement, you are presented with the dialog:



5) When you have filled in all properties, save your component.



- 6) The text will change according to the value of an attached component variable. After saving the component, import it into your view.

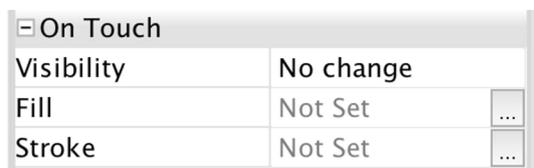


Replacements are important for complex components where you can replace the component text (button text, for example) with just one click.

## 7.6 On Touch Actions

On Touch Actions are used to modify the visual appearance of your component when a user touches it. Usage is very simple: when a user touches the component, the component can change its color or text to denote the action for the end user. When the user releases their touch, the component will return to its default look.

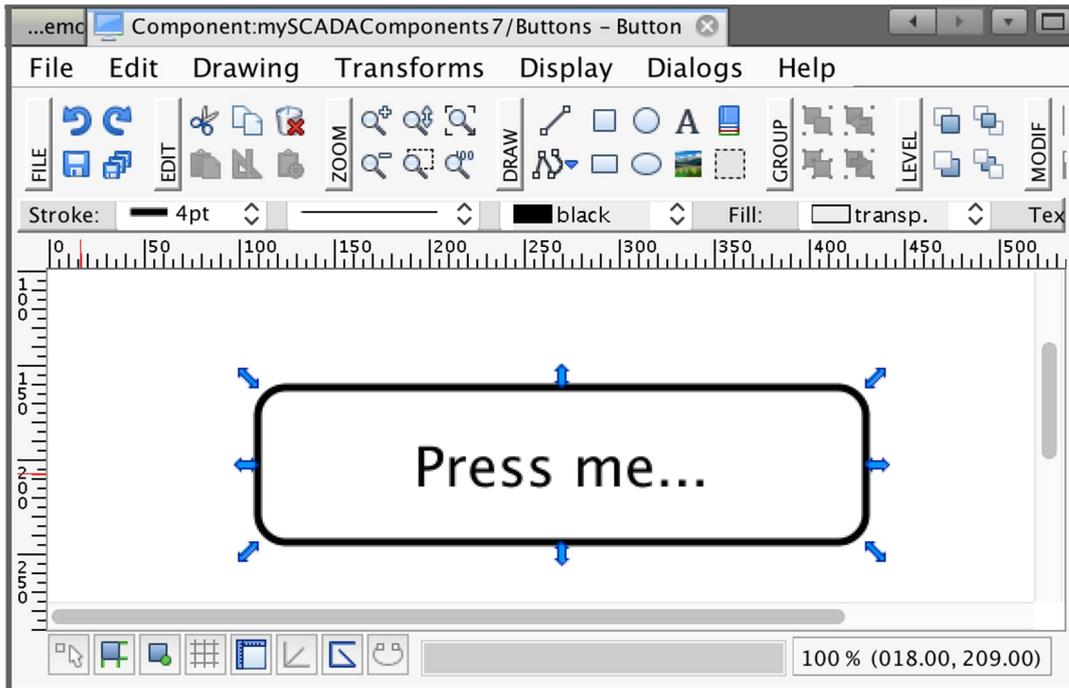
You can modify visibility, fill, and stroke color of any object in your component. Select the object you would like to modify based on user touch and, in Properties, fill in the On Touch section:



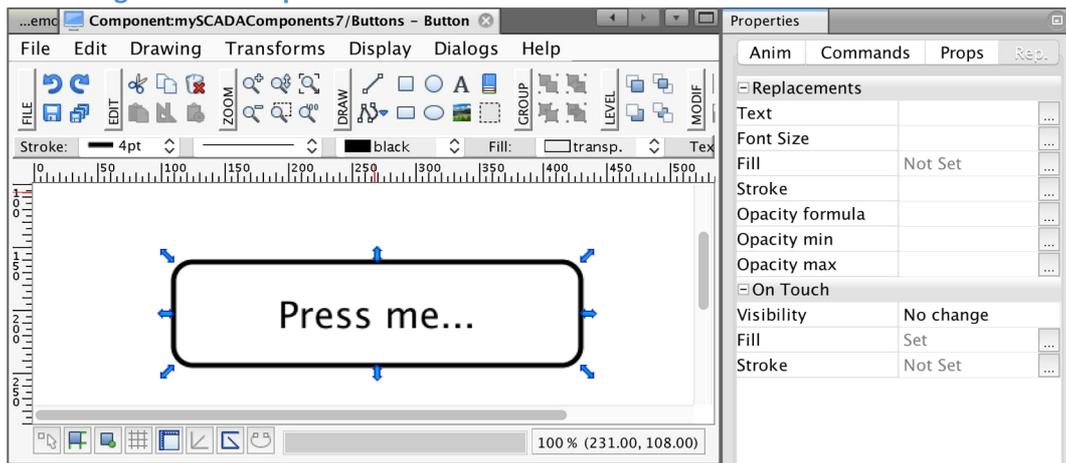
Example:

We will create a simple button that will change the background color on touch.

- Create a new component with a rectangle inside
- Enter in the text "Press me."



Click on the rectangle and in Properties set the color to red in On Touch -> Fill



When you set the fill color, a new dialog is presented:



Save the component and insert it into a view

Now, you can test your component.

When a button is not pressed, it looks like this:



When a user is pressing the button, it looks like this:



**Creating Complete Component:**

In the following example, we will create a simple slide bar showing the variable progress and its Min and Max values.

Create a new component and draw a rectangle, as seen in the picture below.

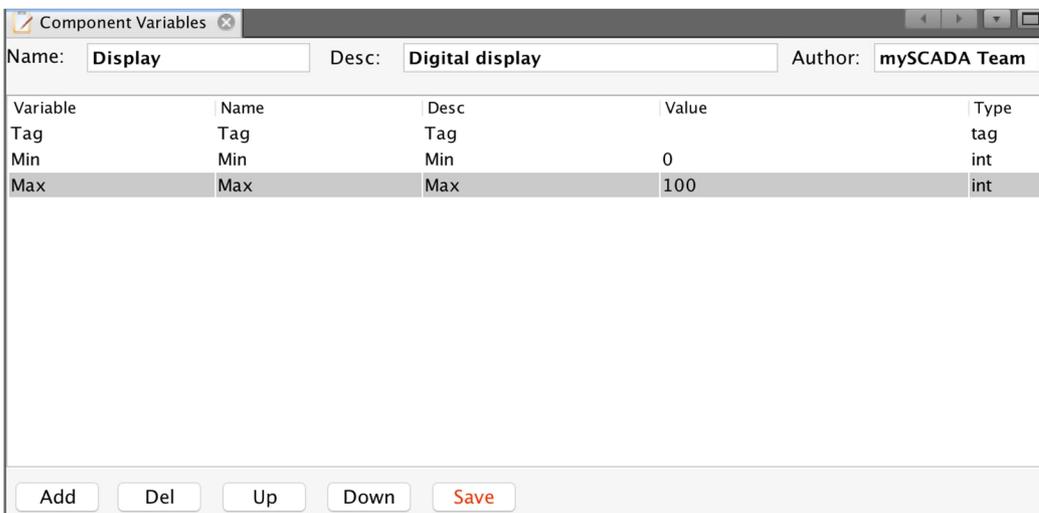
Copy the selected rectangle and apply *Paste on same location* to it.

Set the Fill property to “none” then put the element in the background with the function *Lower to the background*.  We have now created a frame of the scale.

Now create 3 text fields; your component will look like this:

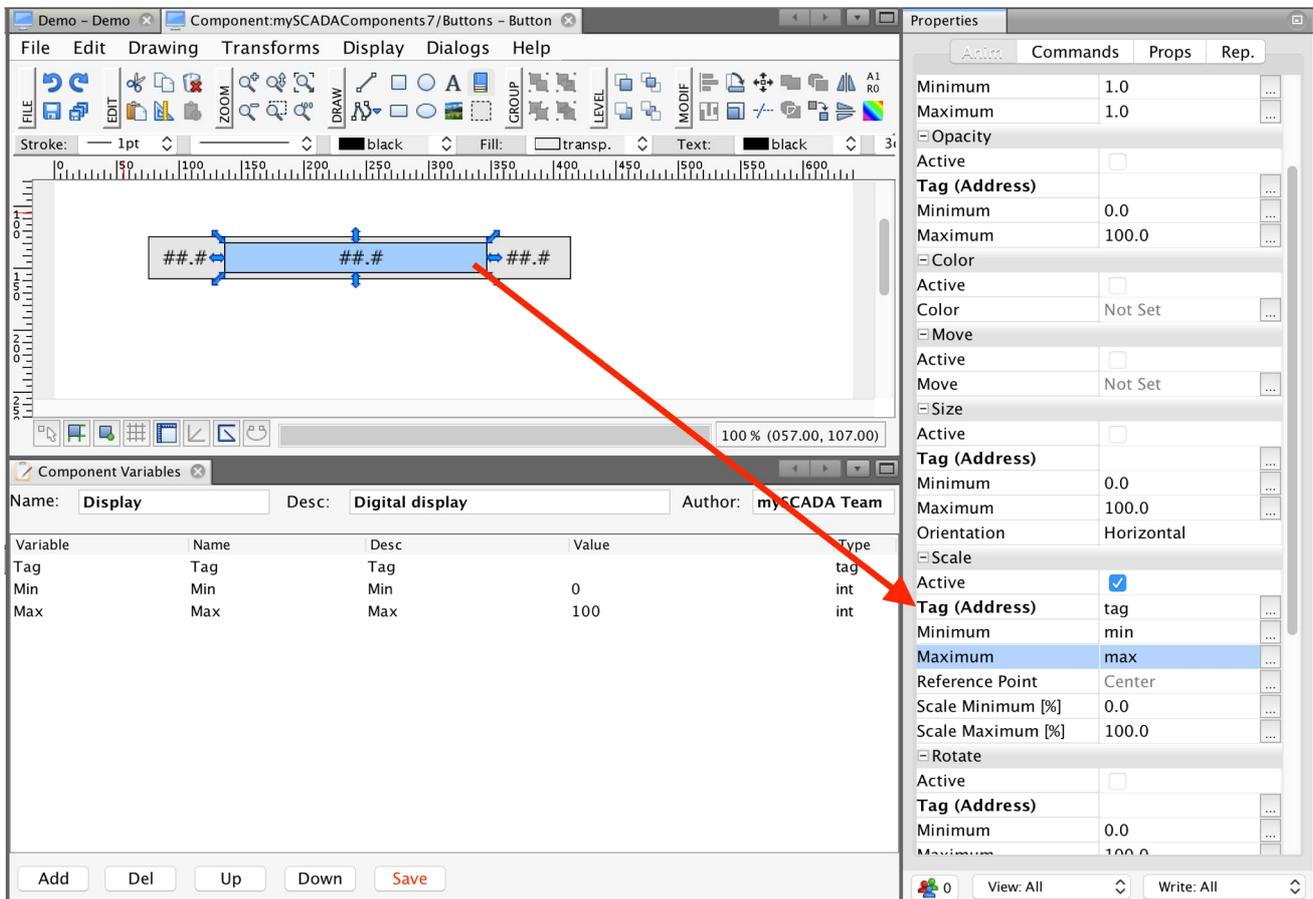


Now fill in the Component Variables as shown in the picture below:

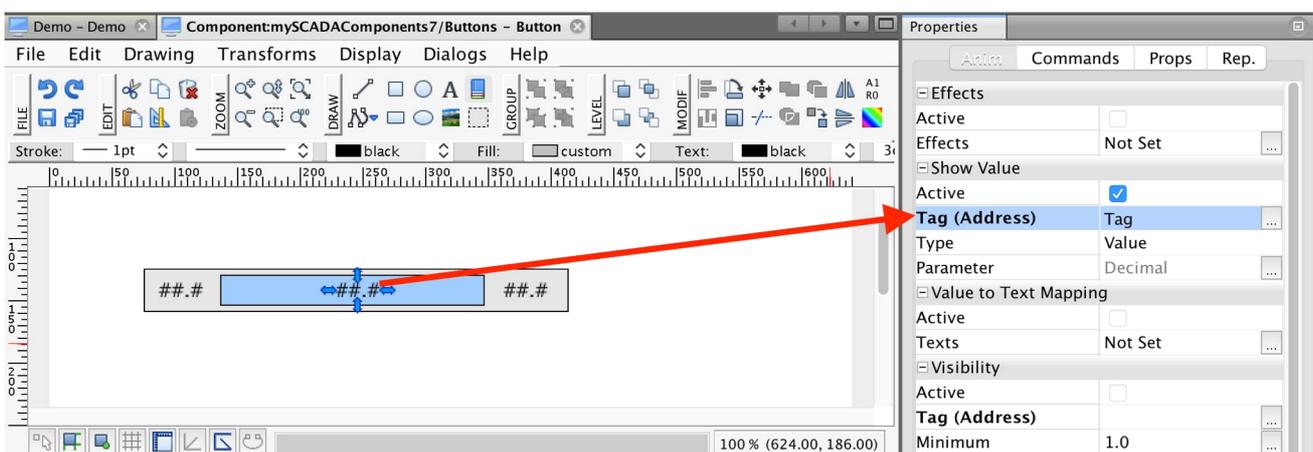


As you can see, we have created 3 local variables. For the tag value, you have used a variable tag; for specifying the minimum and maximum values, you have used the min and max variables.

You can make indicator out of the rectangle by resizing it according to the value read from the PLC; set the *Scale animation* as shown in the following picture:



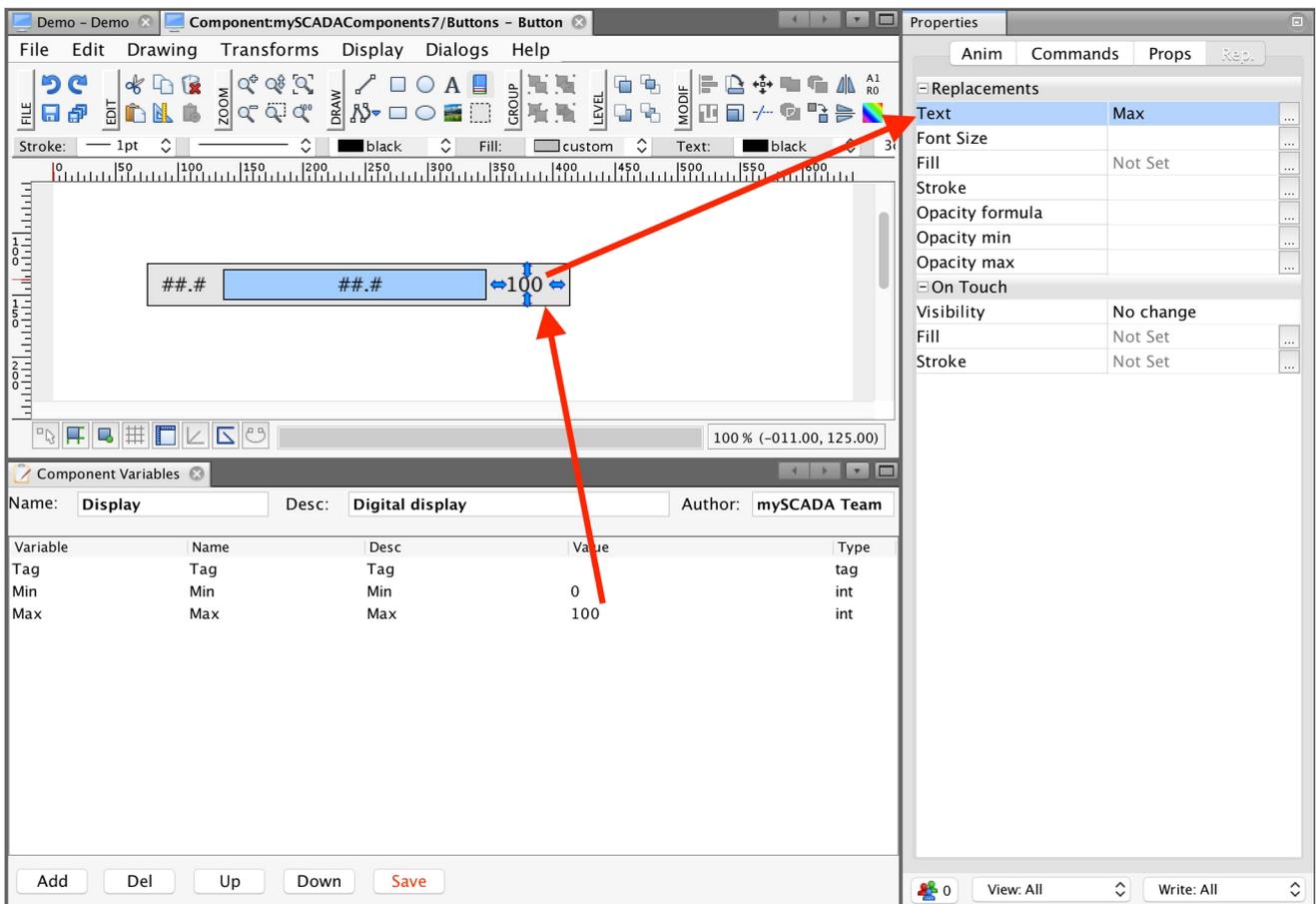
We also want to show the actual value in the middle of the slide bar. Click on the text field in the middle of the rectangle, select the *Anim* tab in the *Properties* window and click on the *Tag (Address)* field in the *Show Value* section. Then, fill in the *Tag* field with "tag," which corresponds to our local variable name.



## Components

As you can see, we have used one of our local variables. You can use the same variable (especially the tag type) for multiple animations/effects to achieve a better effect.

We will also show the *Min* and *Max* limits of our tag. Click on the right text field, then select the *Rep.* tab from the *Properties* window and set "max" for *Text* in the *Replacements*.



7) Now do the same for the other text field but place the *min* variable there.

The final component should look like this:

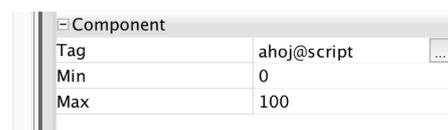


8) Save the component and test its functionality.

## Testing Component

Create a new view and open it, then navigate to the *Library*. Select your component and press the *Insert* button.

Now click on the component and look at the *Properties* window:

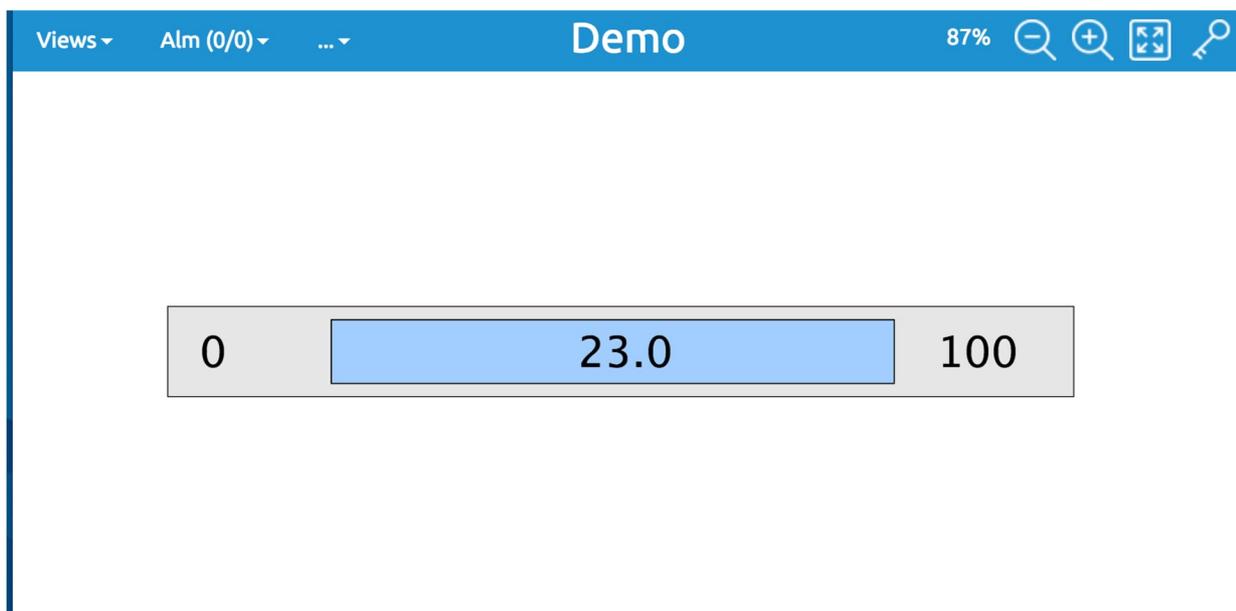


In the *Component* section, you can see all the local variables that you have defined.

Set the **Tag (address)** and the **Minimum** and **Maximum** values:

Component	
Tag	N100:0
Minimum	1.0
Maximum	36.0

Now, save the view and send to a supported device; the screen should look as follows:



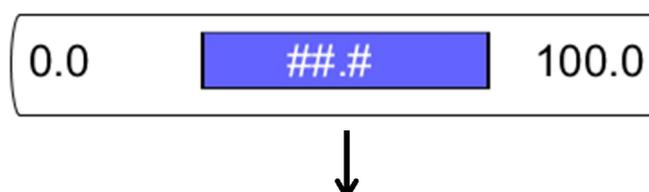
## 7.7 Entering Advanced Functions (Equations)

To create complex and feature rich components, you need to enter complex formulas instead of simple variables. You can use functions anywhere you like: inside animations, instead of constants, or as a text replacement in the text fields.

Formulas are evaluated with JavaScript, so you can enter any formula that is a valid JavaScript expression. You can also include any of the Component Variables that you have defined.

To show the functionality, we will extend our example by implementing the scale along the slide bar we have just created.

- Open the slide bar created in the previous chapter and adjust as follows:

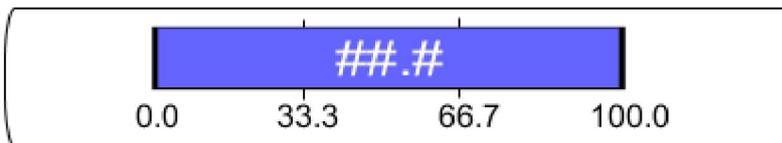




- Delete the *Min* and *Max* fields and stretch the blue rectangle, including the frame. After that, insert additional lines with the *Drawing-Line* tool to divide the rectangle into thirds. With *Transforms -> Order -> Lower to background* send both lines and the frame into the background



- Now insert additional text fields.



For each text field we need to specify the *Text* in the *Replacement* in the **Rep.** tab:

1. Left text field:  $(min).toFixed(1)$
2. Second text field (from left):  $((min+max)/3+min).toFixed(1)$
3. Second text field (from right):  $(2*(min+max)/3+min).toFixed(1)$
4. Right text field:  $(max).toFixed(1)$

The left text field will show the value of the variable  $min.toFixed(x)$ , which is a function that rounds the result to x decimal places. In our case, the value will be formatted to show one decimal place.

The second text field from the left should show  $1/3$  of the scale; therefore, we have used the formula  $min+(min+max)/3$  to get  $1/3$  of the scale. We will use the  $toFixed(1)$  function to round the result.

The other two text fields are similar to those two; we can add units and a label to our component to finish:

Component Variables

Name: **Display** Desc: **Digital display** Author: **mySCADA Team**

Variable	Name	Desc	Value	Type
Tag	Tag	Tag		tag
Min	Min	Min	0	int
Max	Max	Max	100	int
label	label	label	Label	string
unit	unit	unit		string

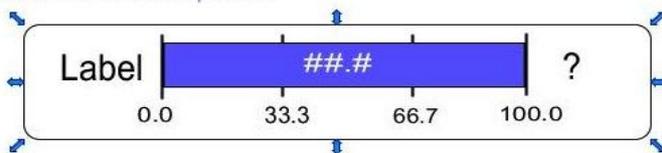
Add Del Up Down Save

To use the Label and Units text fields, we need to create two new variables. Both will be of the type 'string'. The first variable will be named *label*, and the second will be named *unit*. For the text field label, add Text Replacement equal to the variable label. For Units text field, add Text Replacement equal to variable units. Finally, click on the **Save** button to see the changes

Our Simple Component:



Our Advanced Component:

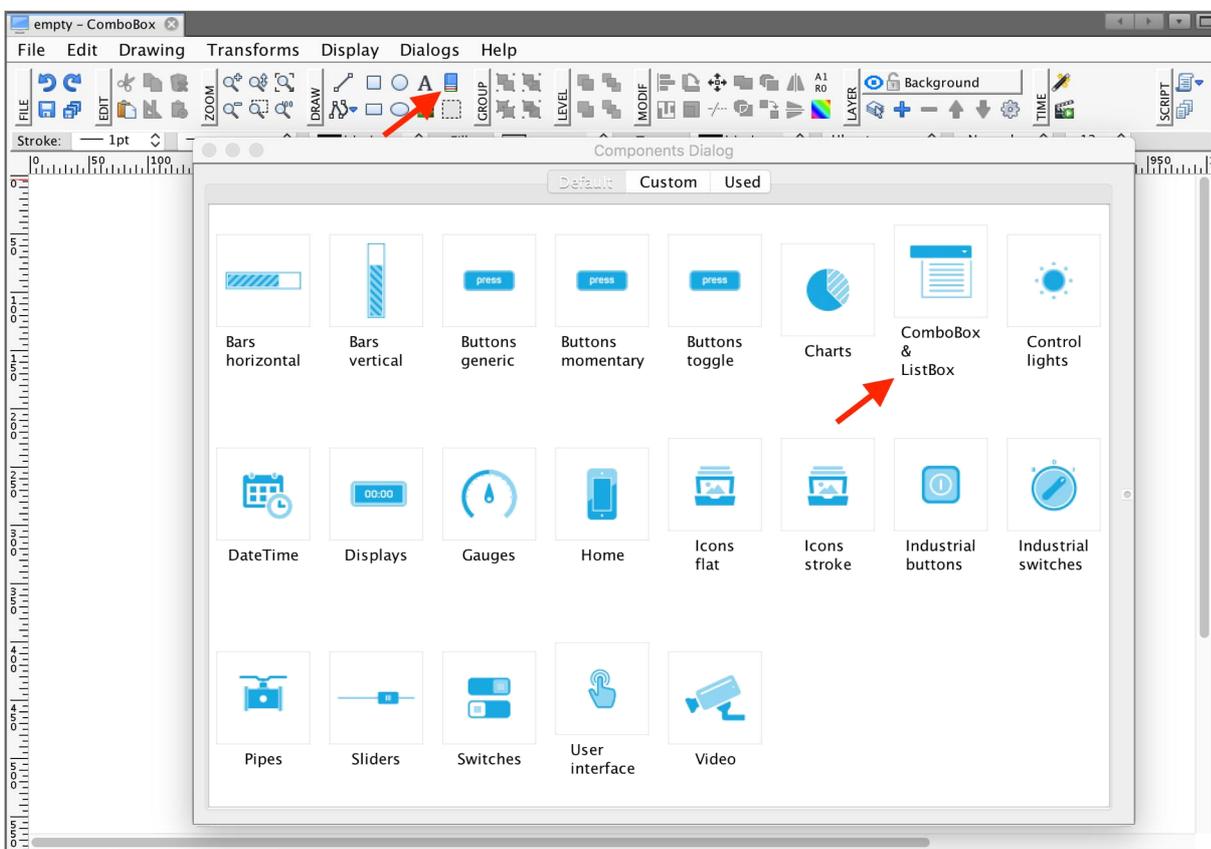


## 8 Combo Box, List Box and Text Box

Combo Box, List Box and Text Box elements are essential components when designing your GUI with forms or multiple entry. mySCADA provides these components to use in two different scenarios. One is to use the components along with the View Script to achieve the desired functionality. The second usage is as standalone components connected directly to a PLC or database.

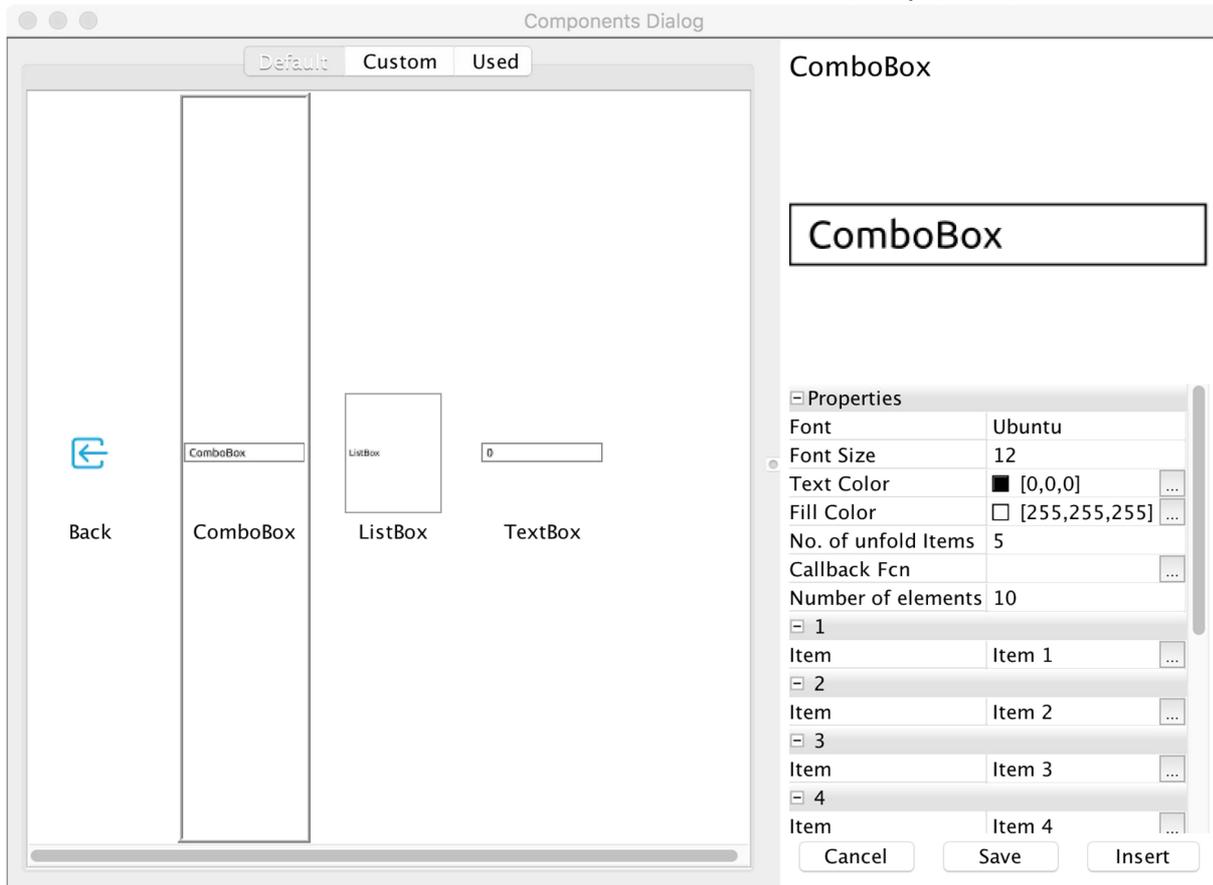
### 8.1 Inserting Into View

To insert a Combo box, list box or Text box into your view, please click on the Library icon and select the Combo box, list box and Text box category.

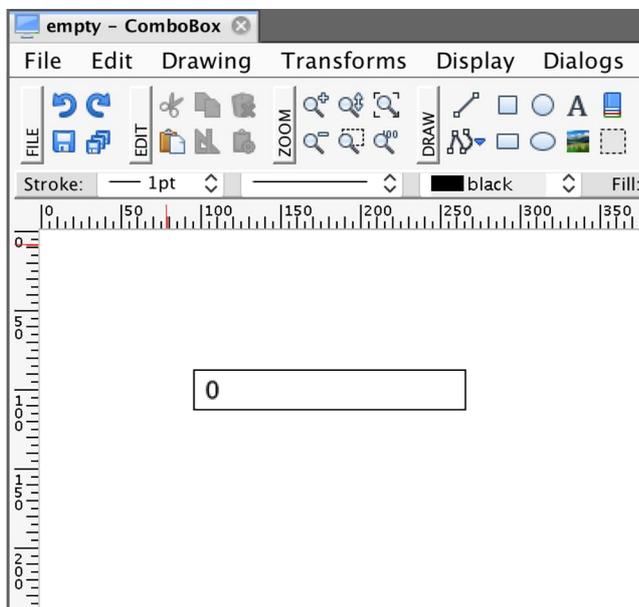


Then select a component, fill in the parameters (such as the number of elements) and click on insert.

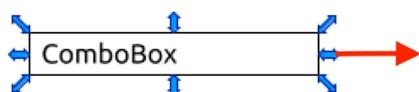
## Combo Box, List Box and Text Box



Now position your component to the view.



TIP: change the size of the component by dragging it to the desired size. Please use the horizontal and vertical sliders only.



## 8.2 Components parameters

### Combo Box

[-] Properties	
Font	Ubuntu
Font Size	12
Text Color	■ [0,0,0] ...
Fill Color	□ [255,255,255] ...
No. of unfold Items	5
Callback Fcn	...
Number of elements	1
[-] 1	
Item	Item ...

Parameter	Explanation
Font	Font name
Font Size	Size of the text
Text Color	Text Color
Fill Color	Fill Color
No. of unfold items	When is combo box unfolded, specify how many items you want to see.
Callback Fcn.	When user selects an item, you can run a custom function. Define a function in the View script and write here its name.
Number of elements	Total number of elements in the Combo box.

### List Box

## Combo Box, List Box and Text Box

Properties	
Font	Ubuntu
Font Size	12
Text Color	■ [0,0,0] ...
Text Anchor	Left
Text Border	<input type="checkbox"/>
Text Border Color	■ [0,0,0] ...
Fill Color	□ [255,255,255] ...
Border	<input checked="" type="checkbox"/>
Border Color	■ [0,0,0] ...
Callback Fcn	...
Number of elements	1
1	
Item	Item ...

Parameter	Explanation
Font	Font name
Font Size	Size of the text
Text Color	Text Color
Text Anchor	Text align, to the left, center or to the right.
Text Border	Select to show border around each item.
Text Border Color	Color of the border around an item.
Fill Color	Fill Color
Border	Select to show border around whole component.
Border Color	Color of the border around whole component.
Callback Fcn.	When user selects an item, you can run a custom function. Define a function in the View script and write here its name.
Number of elements	Total number of elements in the Combo box.

## Text Box

Properties	
Type	String
Max Length	100
Decimal Pieces	2
Lower Limit	0.0
Upper Limit	100.0
Regex for custom	<input type="text" value=""/>
Text	0
Font	Ubuntu
Font Size	12
Text Color	<input type="color" value="#000000"/> [0,0,0]
Fill Color	<input type="color" value="#FFFFFF"/> [255,255,255]
Border	<input checked="" type="checkbox"/>
Round Corneer	0.0
Callback Fcn	<input type="text" value=""/>

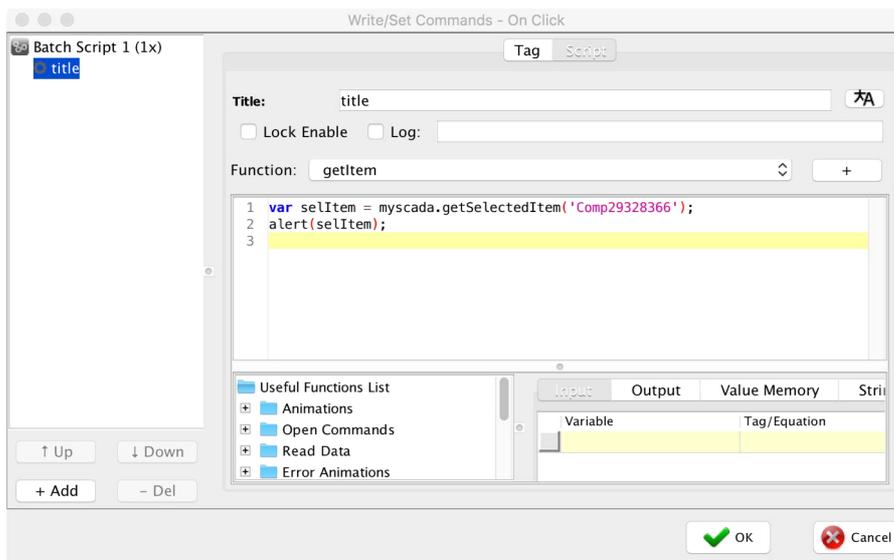
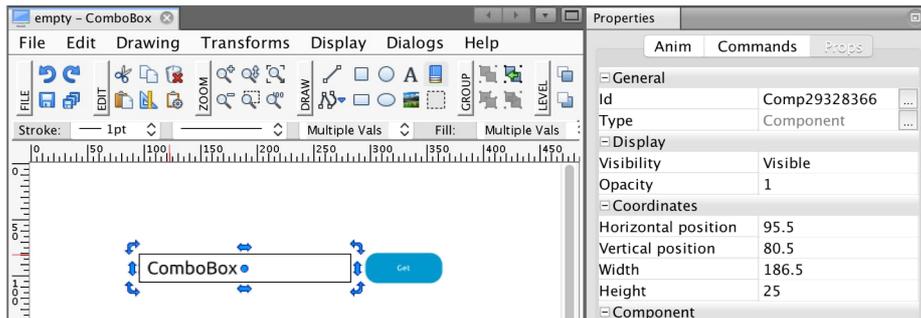
Parameter	Explanation
Type	Specify if user should enter only numbers, strings, or custom values into the text box. If you specify number, please fill also the Limit and decimal places values. If specifying custom, please fill in the regex function for input check.
Max Length	Maximum number of characters entered
Decimal Pieces	Maximum number of decimal pieces entered.
Lower Limit	For numbers only: minimum number allowed
Upper Limit	For numbers only: maximum number allowed
Regex for custom	For custom type only: enter regex by which is user input checked.
Text	Default text or number.
Font	Font name
Font Size	Size of the text
Text Color	Text Color
Fill Color	Fill Color
Border	Select to show border around whole component.
Round Corner	If you want rounded corners, please specify a radius.
Callback Fcn.	When user selects an item, you can run a custom function. Define a function in the View script and write here its name.

## 8.3 Using Components In View Scripts

You can easily use the List box, combo box and text box in view scripts.

We will start with a simple example showing how to use the combo box programmatically:

1. We will create the combo box and add it to the view.
2. We will create a button next to the combo box, on its press we will call view script function `getItem`

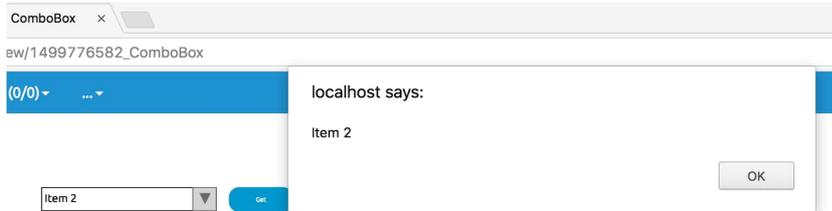


3. In `getItem` function, we will get value of selected component

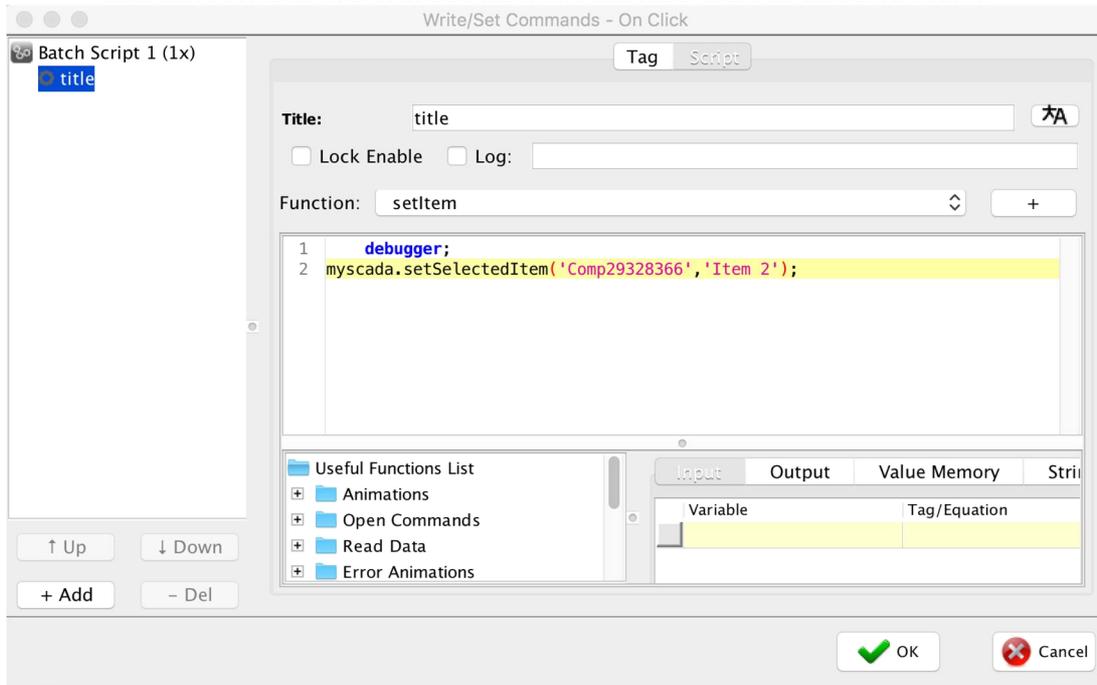
```
var selItem = myscada.getSelectedItem('Comp29328366');
```

'Comp29328366' is an ID of our Combo box. You can find ID in properties window upon clicking on your Combo box.

Once we get the selected item, we will show it by calling `alert(selItem);`



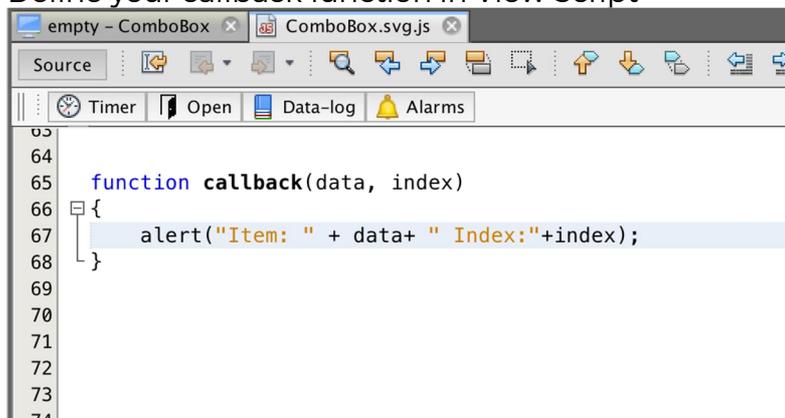
- Now we will create a new button. On its press we will call view script function *setItem* where we will set item in combo box to second item.



## Callback function example:

If you want to be notified, when user selects an item in the combo box or list box, or presses and enters into the text box, set up a callback function.

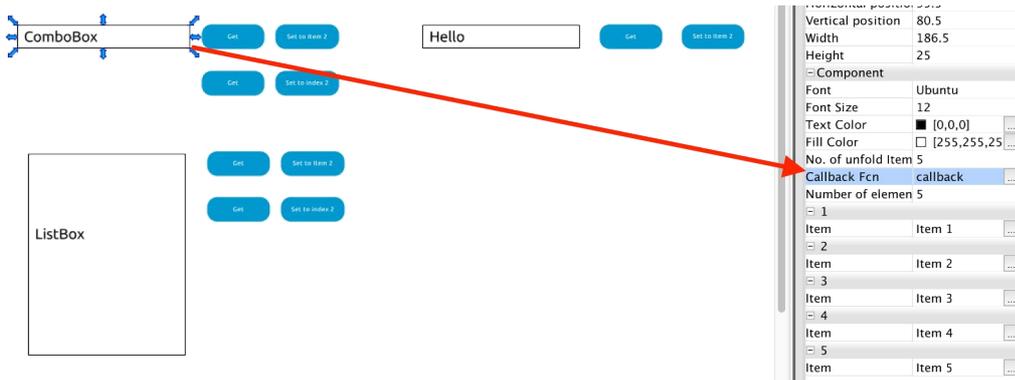
- Define your callback function in View Script



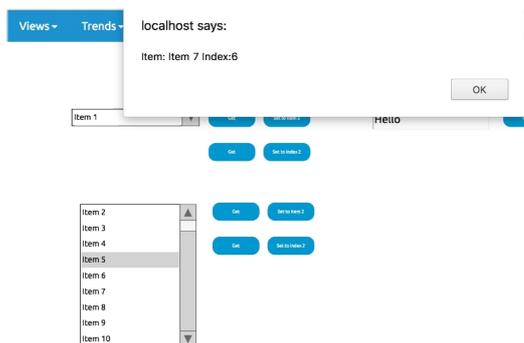
As you can see, into your callback function, you will get 2 arguments, first is a selected item and second is an index of the selected item.

- Click on your component in view and fill in the callback function name

## Combo Box, List Box and Text Box



3. When user selects an item, your callback function will be called



## Setting new values into Components Programmatically

You can easily change the items in your component programmatically. To do so, you can use function `setItems`:

```
var items = ['item 1','item 2'];  
myscada.setItems('id',items);
```

## Coloring items in List Box

List box allow you to fine tune appearance of each item by specifying the colors and bold face of the font. To do so, use function `setItems`. Passed items in array will not be strings but objects with additional properties. Simple Example:

Following script

```

75
76
77 function ColorMe() {
78
79     var items = new Array(
80         {key:"item 1 regular"},
81         {key:"item 2 fill blue",fill:"blue"},
82         {key:"item 3 color red",color:"red"},
83         {key:"item 4 bold",fill:"green", bold: true},
84         {key:"item 5 all custom",fill:"#a4cfff",color:"#ff881c",bold:true}
85     );
86     myscada.setItems('Comp26354109',items);
87
88 }
89

```

Will produce following output



## 8.4 Supported functions

Function	Combo Box	List Box	Text Box
var item=myscada.getSelectedItem(id);	Yes	Yes	Yes (returns value)
myscada.setSelectedItem(id,('item');	Yes	Yes	Yes (sets value)
var index =myscada.getSelectedIndex(id);	Yes	Yes	No
myscada.setSelectedIndex(id,(index);	Yes	Yes	No
var object = myscada.getObject('id');	Yes	Yes	Yes
var items = ['item 1','item 2']; myscada.setItems('id',items);	Yes	Yes	Yes (sets first value in array)

# 9 Active Area

## 9.1 Introduction

Active areas are the regions in views where you can show dynamic content. It can be another view, trend, alarm window, datalog, or even an external HTML page or a live video stream. Active areas act as an active container for all those multiple options. Aside from *mySCADA's* specific functions (views, trends, etc.), you can use the active areas to show any content that conforms to HTML 5 standards. That way you can show an external web page or code an HTML page directly in *mySCADA*. You can also combine the active area with view scripts (JavaScript based) to achieve dynamic interaction. That way you can, for example, show *Google Maps* or create your own HTML component with SCADA options.

Watch video describing this functionality:

- Active Areas [part1] [https://www.youtube.com/watch?v=g\\_V91pm8C1c](https://www.youtube.com/watch?v=g_V91pm8C1c)
- Active Areas [part2] <https://www.youtube.com/watch?v=uq-hHTbrwrs>

With active areas, you can show:

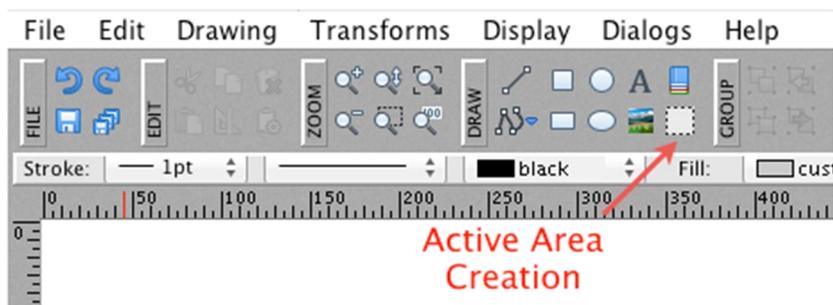
- Window (Arbitrary View, Parametric View)*
- Advanced Trends*
- Data-log Views*
- Online alarms*
- Historic alarms*
- External HTML page / Internal HTML code*
- Combine DIV with view scripts*

*Note: One of the main benefits is that you can change the active area content at any time using the Open command (for more, see the Commands section).*

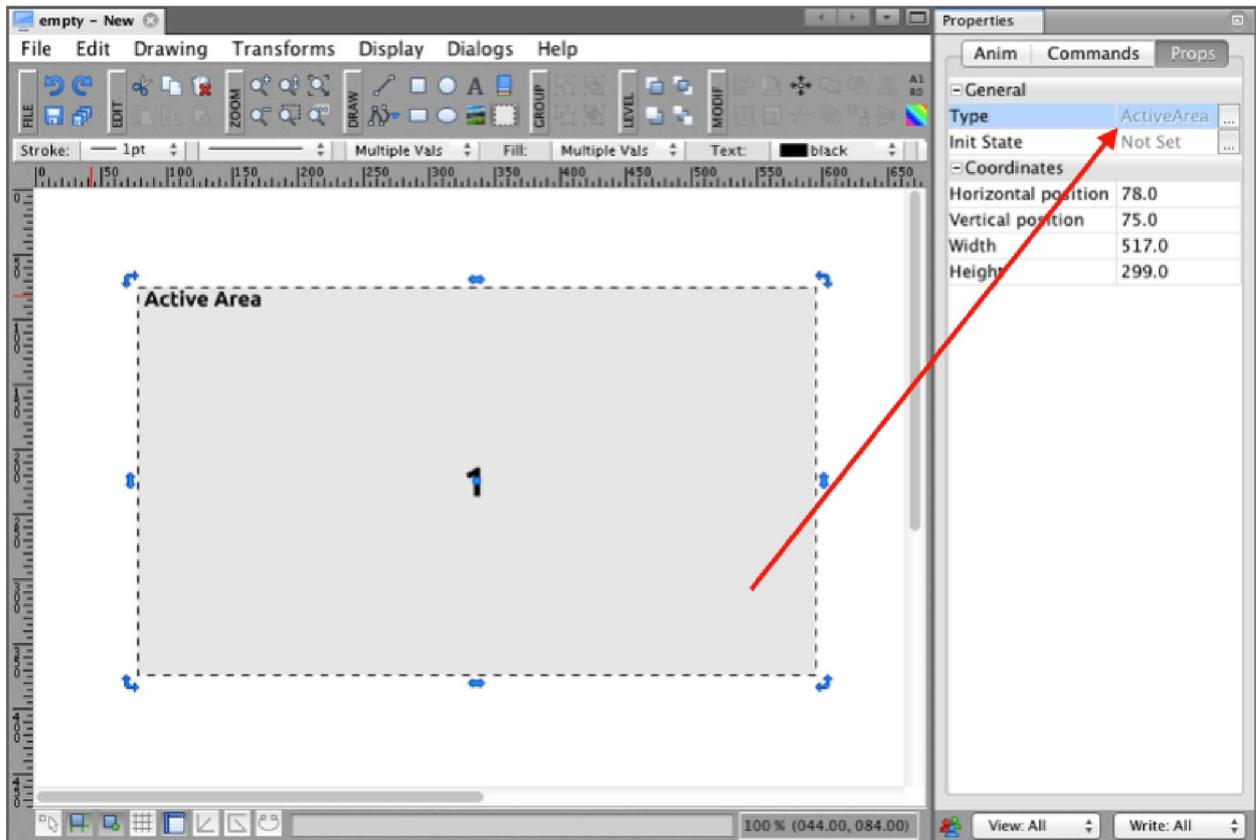
## 9.2 Creating Active Areas

Active Areas (or multiple active areas) can be created in any view.

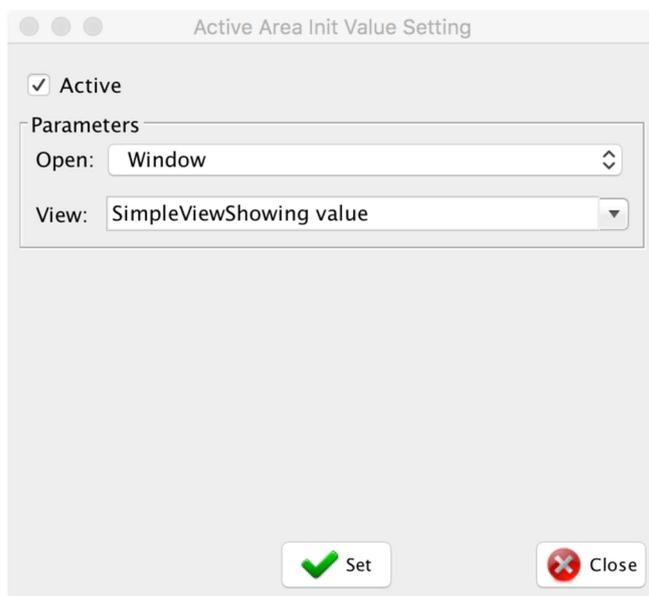
- 1) Click on the Active Area icon in the GUI Toolbar.



- 2) Draw the Active Area:



- 3) Click on the **Init State** in the *Properties* window and select the content you wish to display.

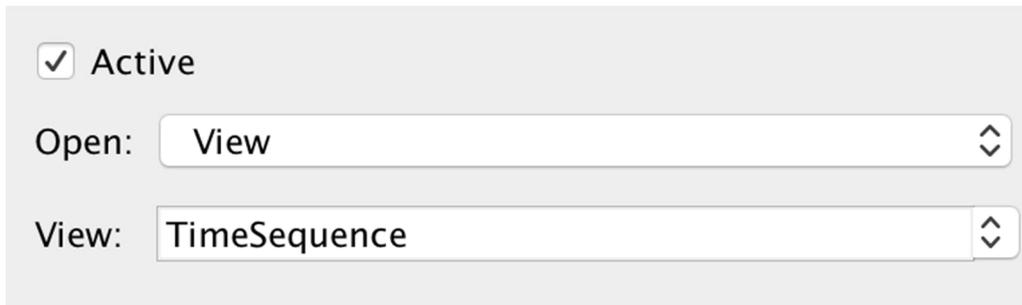


In this dialog, you can set up the type of Active Area. In other words, you can set up how the Active Area will be initialized when the view is shown.

Note: If you do not activate the initial state (e.g. you leave the *Active* check box unchecked), the Active area will not be visible. You can set a new state later when using open command.

## View type

If you select this option, you can specify the view to be shown in the active area. You can display any view you have previously designed. For best results, use a window with the same resolution or at least the same aspect ratio as your active area. If you need to change the active area size to an exact width and height, you can do so in the *Properties* window.

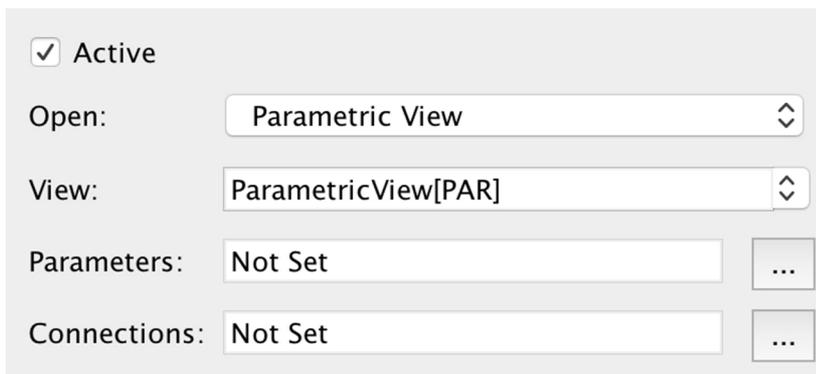


Active  
 Open: View  
 View: TimeSequence

*TIP: If you specify the option "Previous View," open command will jump back to previously open window.*

## Parametric View type

If you select this option, you can specify the parametric view to be shown in the active area. You can display any view you have previously designed. For the best results, use a view with the same resolution or at least the same aspect ratio as your active area. If you need to change the active area size to an exact width and height, you can do so in the *Properties* window.



Active  
 Open: Parametric View  
 View: ParametricView[PAR]  
 Parameters: Not Set ...  
 Connections: Not Set ...

Aside from the Parametric View name, you can also specify the Indexes and Connection indexes.

For more details on how to deal with parametric views, please see chapter [Parametric Views](#).

## Document type

You can show a PDF document linked to the project in an active area. The PDF document will be automatically scaled to the size of the active area.

Active

Open:

Document:

Page:

Select a PDF document. You can also specify on which page to open the document.

### Alarms type

This feature can show online alarms and alarm history in the active area. There are several parameters you can use to set up the specific visual appearance of the alarm table that will be shown in the active area. You can also limit shown data using filters.

Active

Open:

Alarms:

Toolbars:  Top  Bottom

Parameters:

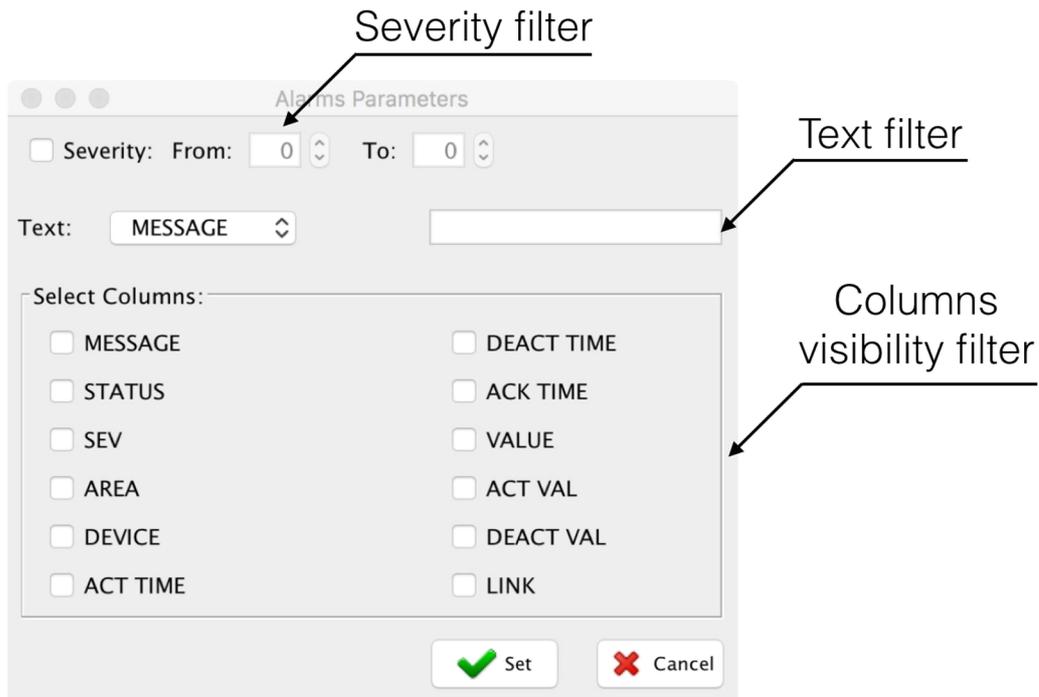
Time Scale [min]:

Background Color:  Transparent  Color

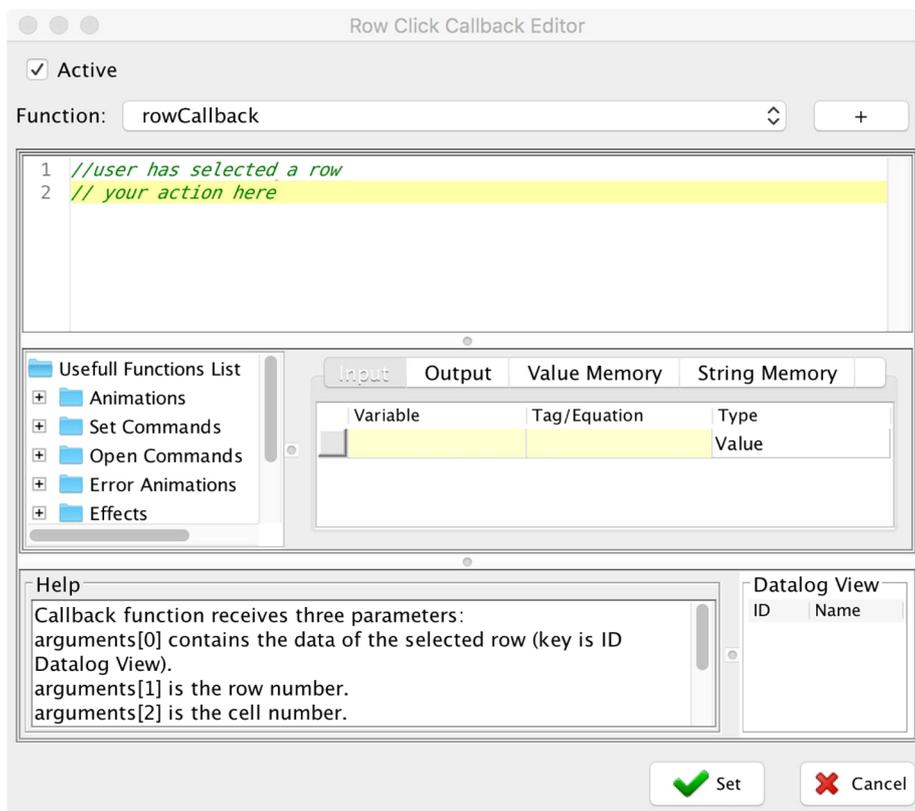
Row Click Callback:

Enable on Lock element or Lock Key

- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters and alarm settings.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g., the date selection controls).
- *Parameters* allow you to specify which columns will be visible in the alarm table. In addition, you can set a filter for severity and text filters for the message, area, and device.



- *Time Scale [min]* parameter specifies the time interval that will be shown in the alarm history table. Units are in minutes.
- *Background Color* can be set to transparent or to a specific color. If you set this parameter to color, you can specify which color will be shown as the background.
- *Row Click Callback* is a neat feature: when a user selects a row in a table, you will get a callback in the specified JavaScript function. This function must be defined in View Script.



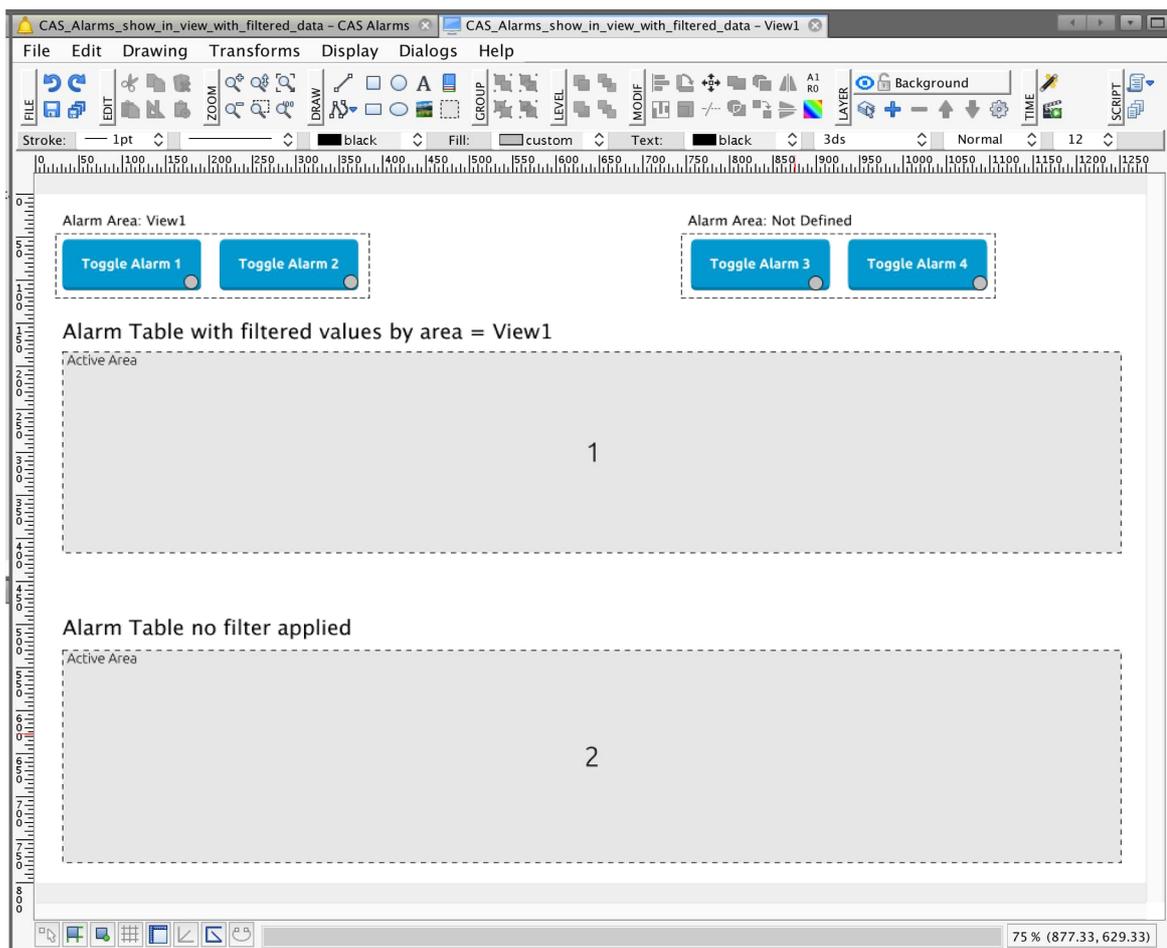
## Example on showing filtered Alarm data

Suppose you want to show the alarm table on the screen, but only show data in respect to the given view.

1. Create alarm definitions in CAS Alarm Windows. To designate which alarms belong to the view you plan to show, specify the name of the view in the Area column.

ID	Tag@Conn/*Alias	Sev	Area	Message
1	alm1@script	0	View1	Alarm1
2	alm2@script	0	View1	Alarm2
3	alm3@script	0		Alarm3
4	alm4@script	0		Alarm4
new		0		on

2. Create a new view and insert an Active area into it. (If you want to show filtered and unfiltered data in one view, insert two active areas.)



## 3. Set up its init state.

Active Area Init State Settings

Active

Open: Alarms

Alarms: OnLine

Toolbars:  Top  Bottom

Parameters: Set

Time Scale [min]: 1

Background Color:  Transparent  Color

Row Click Callback: Not Set

Set Close

## 4. Set up data filter as follows:

Alarms Parameters

Severity: From: 0 To: 0

Text: AREA View1

Select Columns:

MESSAGE  ACK TIME

STATUS  VALUE

SEV  ACT VAL

AREA  DEACT VAL

DEVICE  LINK

ACT TIME

DEACT TIME

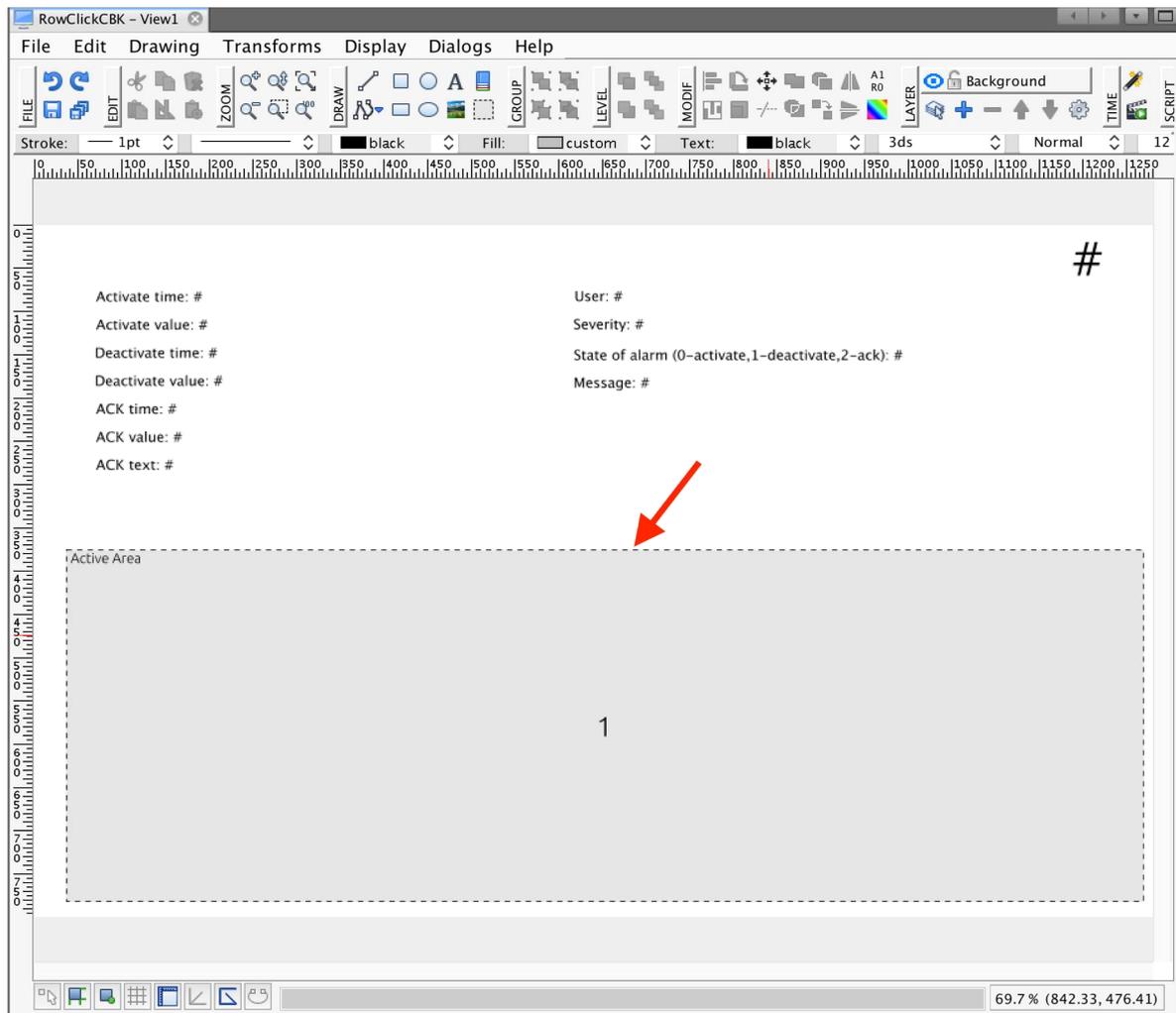
Select All Deselect All Set Cancel

## 5. Test your view.

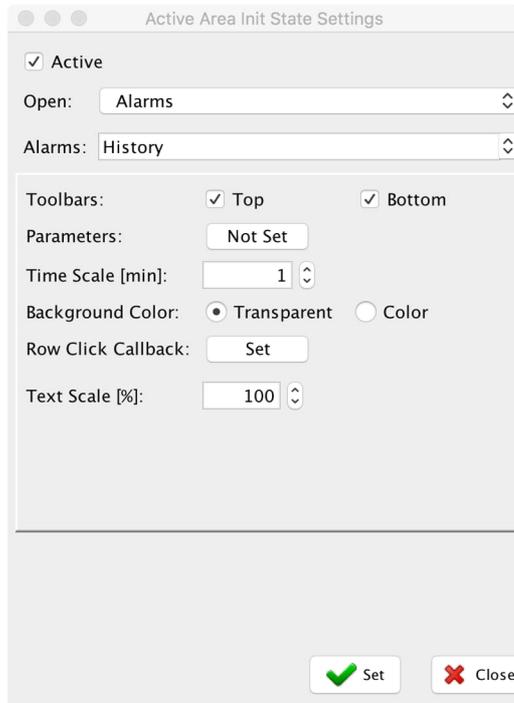
### Example of using Row Click Call-back

Suppose you have an alarm table shown in the Active area, and you want to detect when a user selects a row in the table.

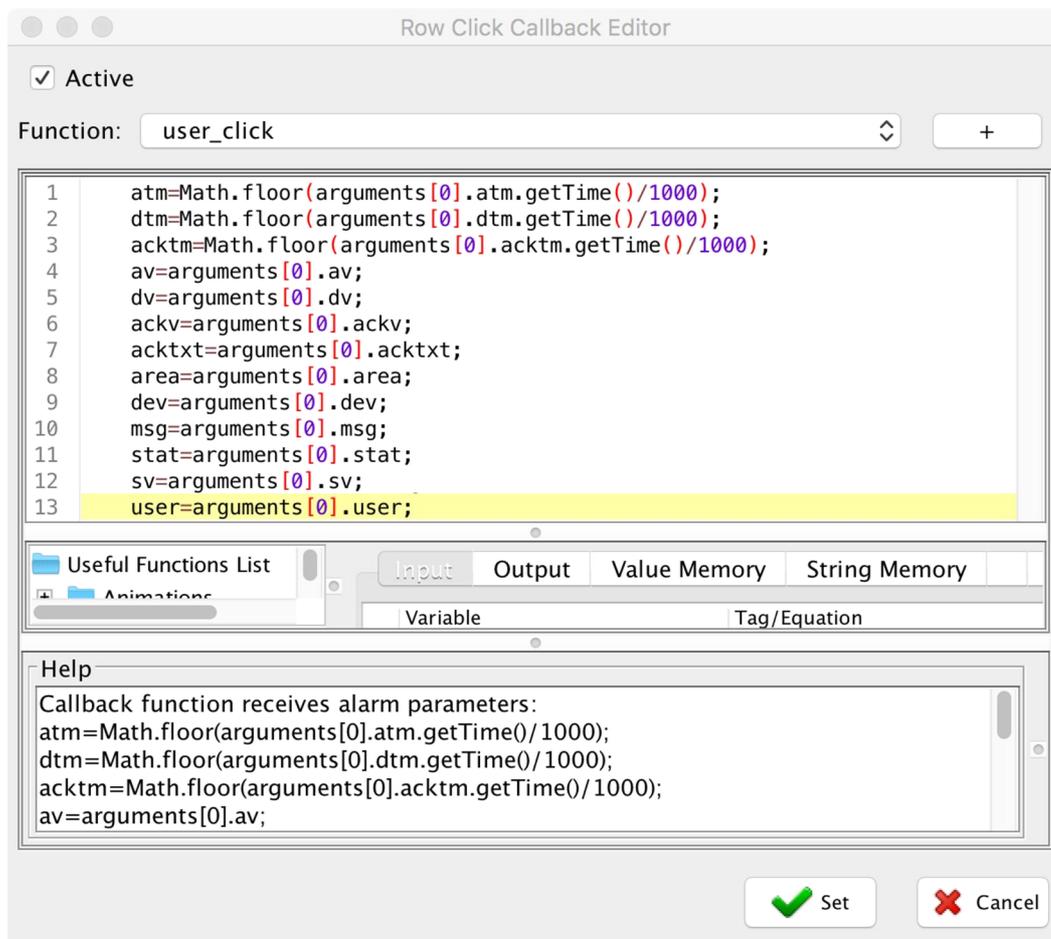
1. Create a new view and insert an Active area into it.



2. Set up its init state.



- Set up row click callback as follows:



(variable *arguments[1]* is the row number and variable *arguments[2]* is the column number). In addition, you will get all values in the selected row; they are in the variable *arguments[0]*.

5. Now test your view.

6

Activate time: October 9, 2016 23:06:59  
 Activate value: 5  
 Deactivate time: January 1, 1970 01:00:00  
 Deactivate value: 0  
 ACK time: January 1, 1970 01:00:00  
 ACK value: 0  
 ACK text:

User: system  
 Severity: 0  
 State of alarm (0-activate,1-deactivate,2-ack): 1  
 Message: Alarm ON

LIMIT: 10000   ACT   DEACT   ACK   SUP   UNS   SEVERITY   TEXT   Export

#	MESSAGE	STATUS	ACT TIME	ACT VAL
1	Alarm ON	ACT	October 9, 2016 23:06:59	5.0
2	Alarm ON	DEACT	October 9, 2016 23:06:59	5.0
3	Alarm ON	ACT	October 9, 2016 23:07:09	5.0
4	Alarm ON	DEACT	October 9, 2016 23:07:09	5.0

October 9, 2016 22:07:18   October 9, 2016 23:07:18   1 hour

DOWNLOAD DEMO PROJECT HERE:

[http://nsa.myscada.org/projects/example/Row\\_Click\\_Callback\\_Alarms\\_History.mep](http://nsa.myscada.org/projects/example/Row_Click_Callback_Alarms_History.mep)

### Advanced Trend type

You can show trends, as defined in the *Advanced Trends* section. The size of the trend is automatically adjusted to the size of the active area.

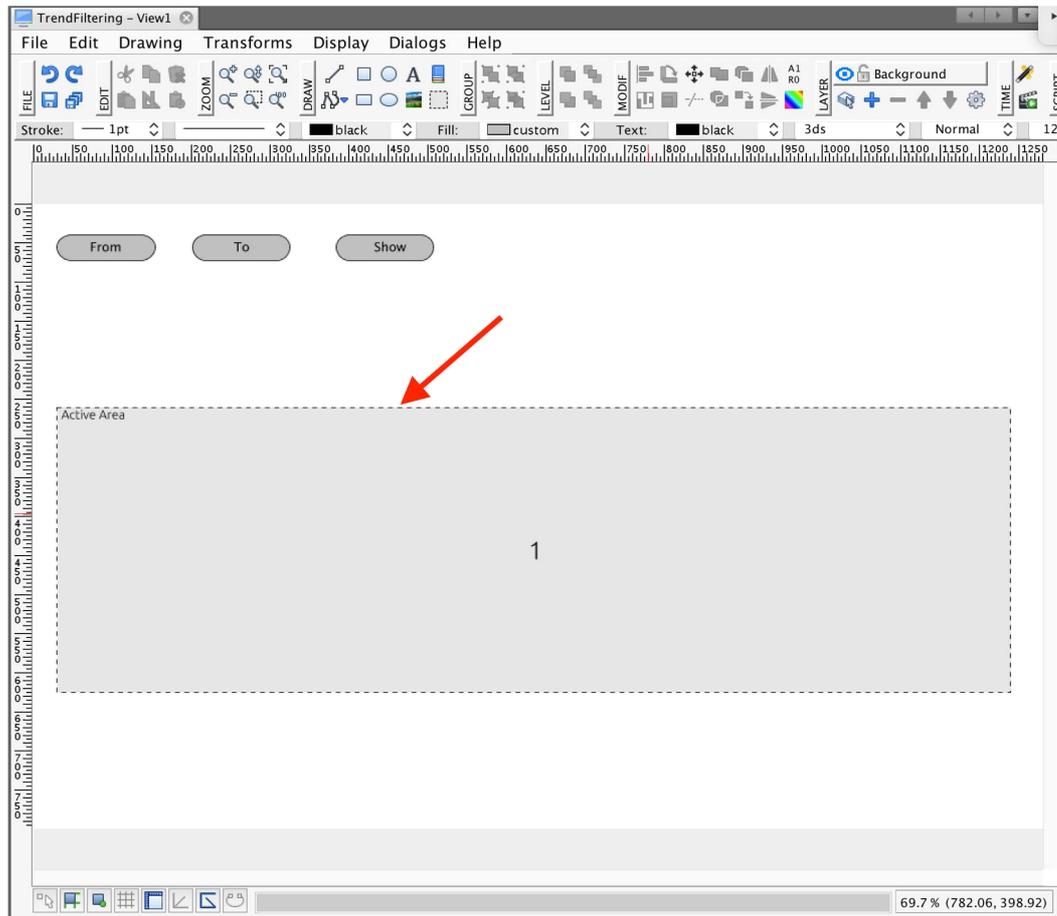
- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters and advanced trend settings.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g., the date selection controls).
- Filters allow you to set a time range for the advanced trend shown. You can specify *Time From* value (in UNIX UTC time; e.g. seconds since the year 1970), *Time To* value, or *Time Scale* value. You can use constants, or you can use tags from PLCs or variables from View Scripts.  
If you use variables or tags in the bellow fields, you can change the time interval of the advanced trend dynamically.

- *Time Scale [min]* parameter specifies the time interval that will be shown by the Advanced Trend. Units are minutes.
- *Background Color* can be set to transparent or to a concrete color. If you set this parameter to color, you can specify the color that will be shown as the background.
- *Row Click Callback* is a neat feature. When a user selects a row in a table, you will get a callback in the specified JavaScript function. This function must be defined in View Script.

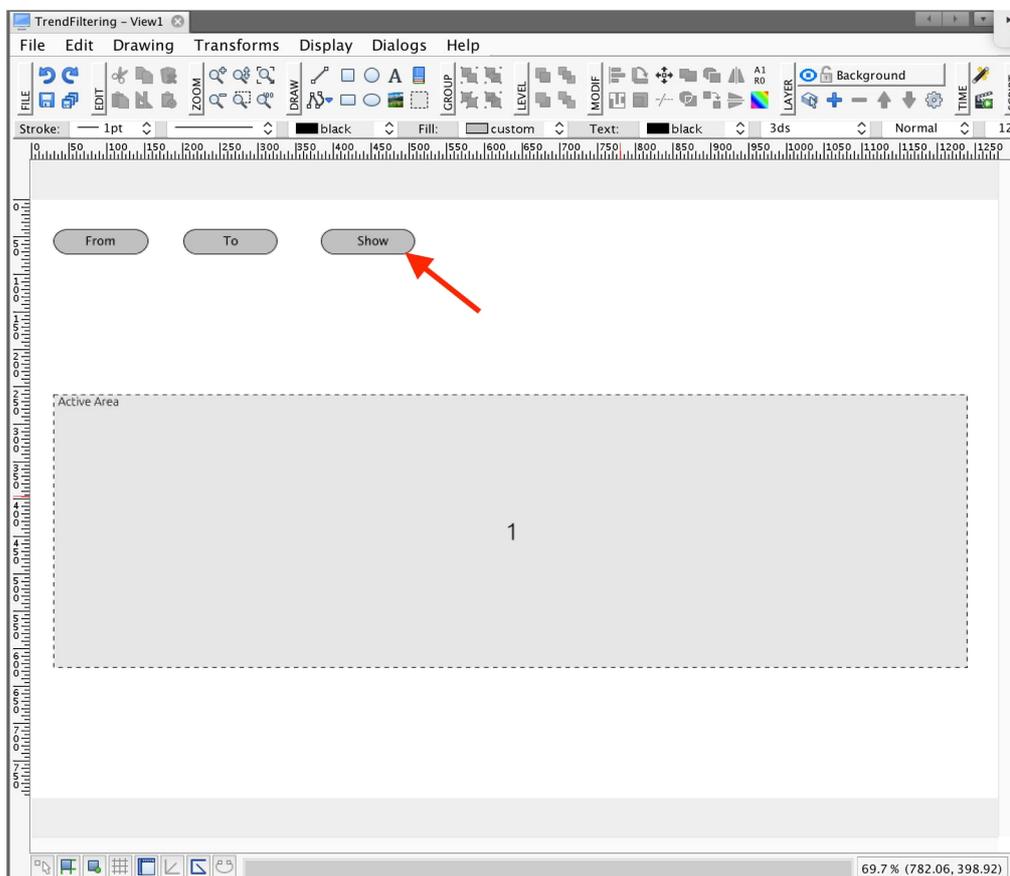
### Showing Trend in active area with time interval specified by the user

This example demonstrates how to use the dynamic filters in Advanced trend shown in an active area. The goal is to create the simple view with Advanced Trend shown. Users can specify the interval shown by the Advanced Trend using multiple buttons.

1. Create View and insert Active Area into it.



2. Create buttons as shown in the following picture:

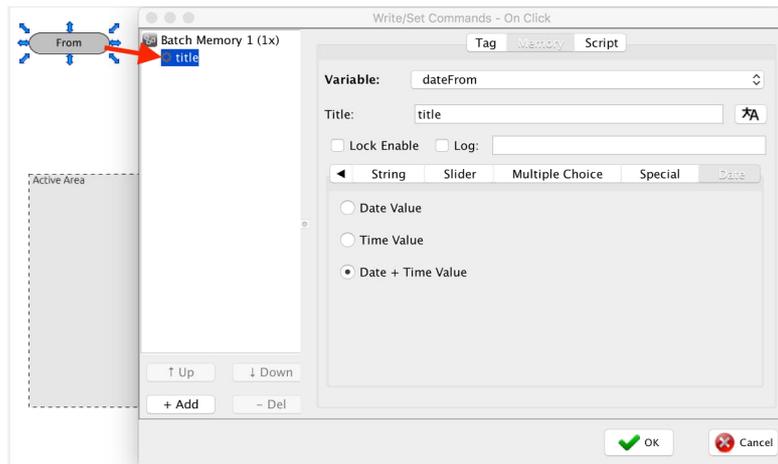


3. Now open View Scripts and create the following variables:

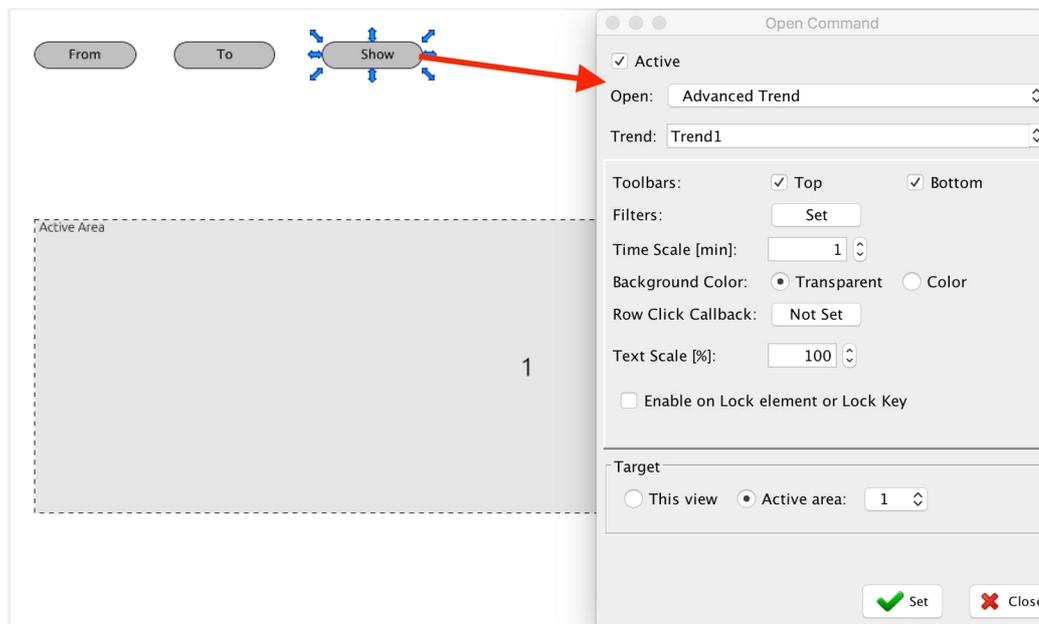
*dateFrom*  
*dateTo*

Variable	Value (Double)
dateFrom	0
dateTo	0

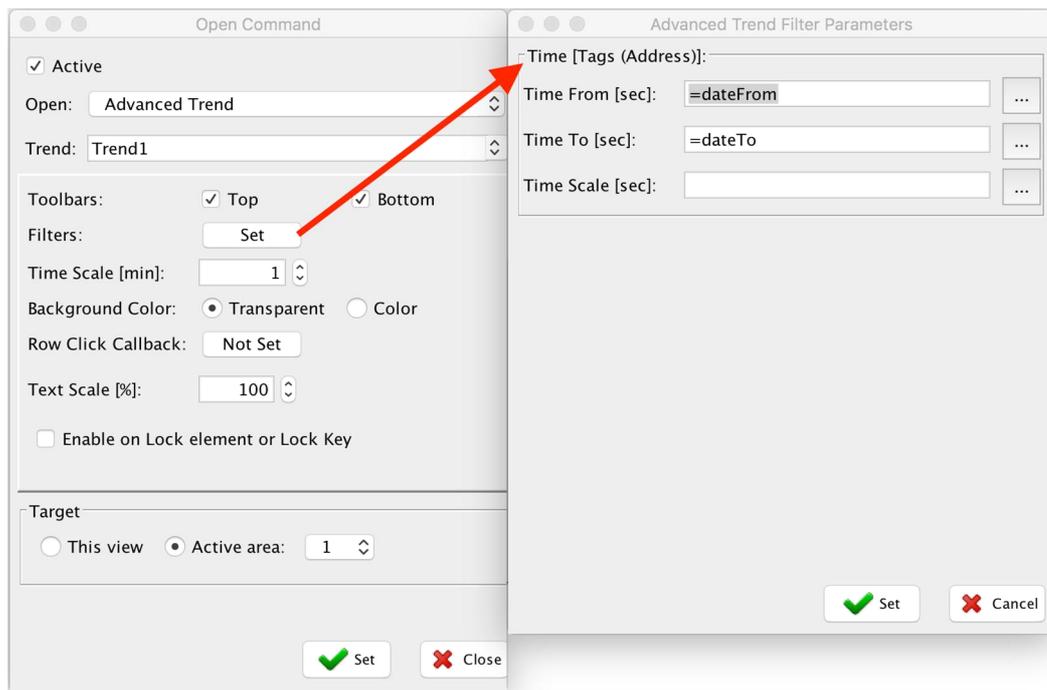
4. Each button will set the corresponding variable *dateFrom* and *dateTo* to specify the Advanced trend interval.



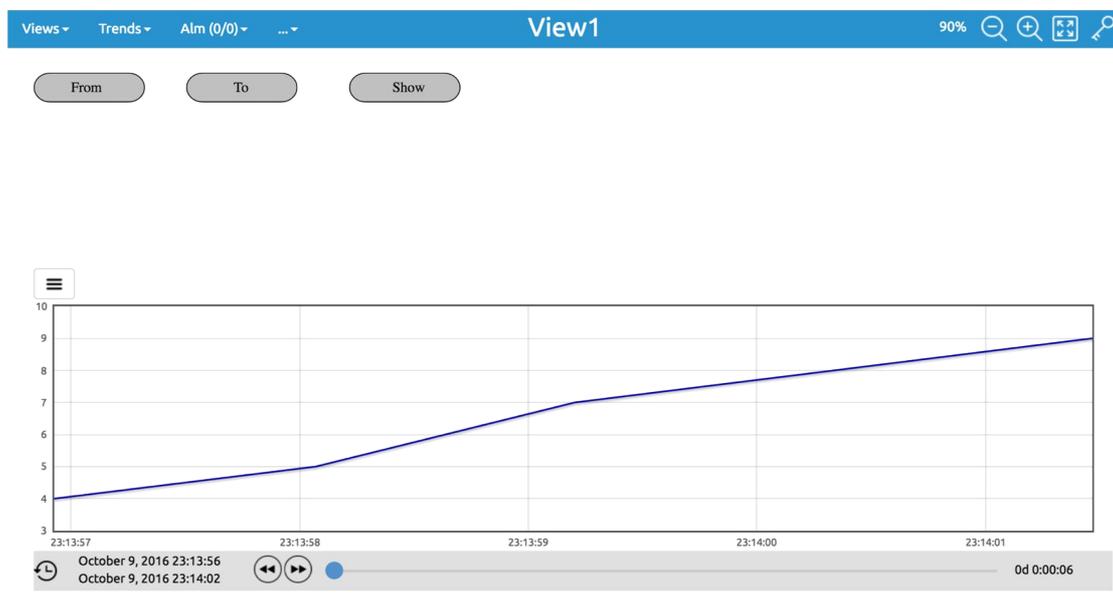
5. Now set up the SHOW button to open the Advanced trend.



6. The Filter will be set as follows:



7. Now when the user clicks on the button, the new time interval is shown in the Advanced trend.

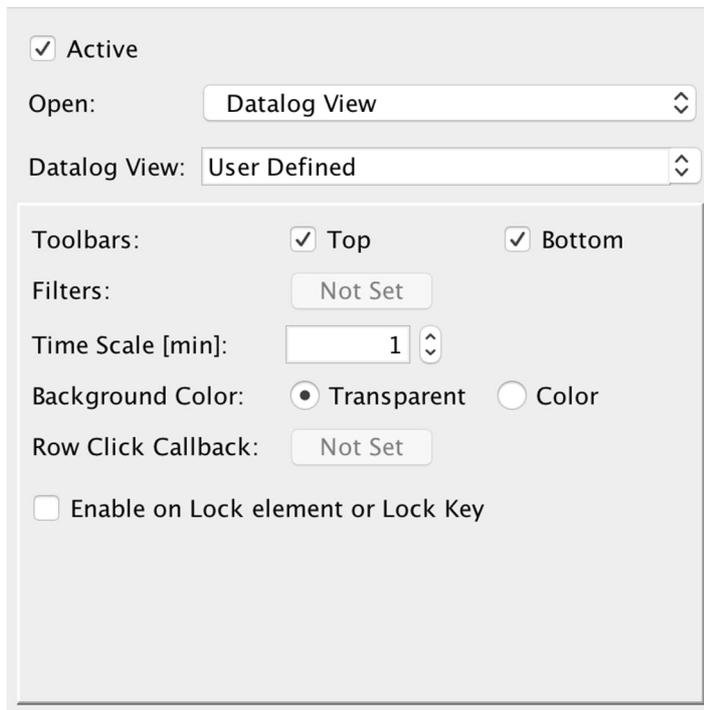


DOWNLOAD DEMO PROJECT HERE:

[http://nsa.myscada.org/projects/example/Trend\\_active\\_area\\_with\\_time\\_interval.mp](http://nsa.myscada.org/projects/example/Trend_active_area_with_time_interval.mp)

## Data-log View type

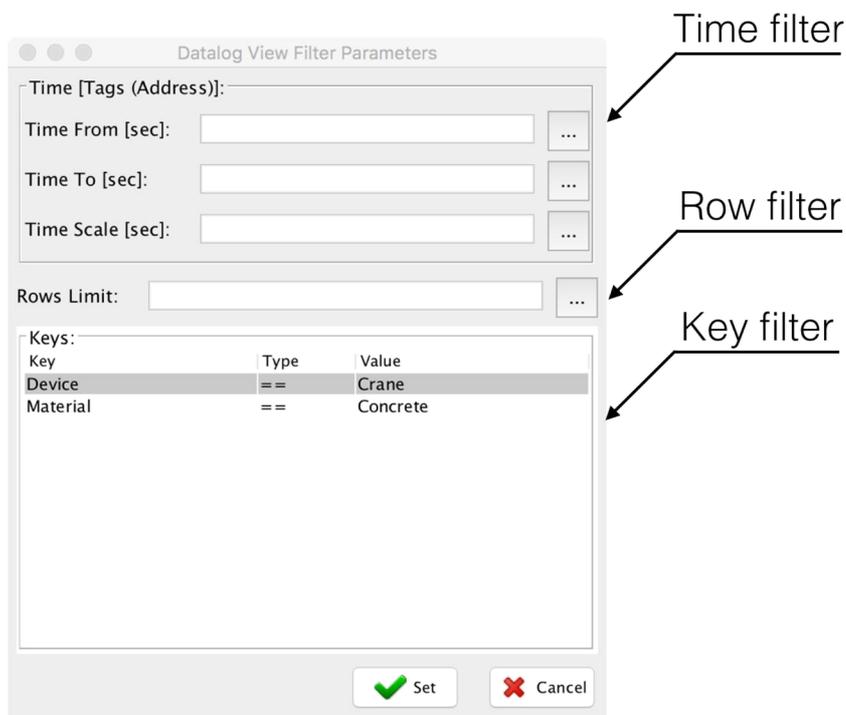
You can show data-log view, as defined in the *Data-log* section. The size of the data-log view will be automatically adjusted to the size of the active area.



The screenshot shows a configuration panel for the Data-log View. It includes the following settings:

- Active
- Open: Datalog View
- Datalog View: User Defined
- Toolbars:  Top,  Bottom
- Filters: Not Set
- Time Scale [min]: 1
- Background Color:  Transparent,  Color
- Row Click Callback: Not Set
- Enable on Lock element or Lock Key

- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters and data-log view settings.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g. the date selection controls).
- Filters allow you to limit the data shown by time, number of records, or by parameters.



**Time filter** allows you to set time ranges for records shown. You can specify a *Time From* value (in UNIX UTC time; e.g. seconds since the year 1970), *Time To* value, or *Time Scale* value. You can use constants, or you can use tags from PLCs or variables from View Scripts.

1. Using constants: just write the value in seconds in the text field
2. Using Tag: enter the tag value or use “...” button to specify a tag.
3. Using Variable: enter an = followed by the variable name.

**Row Filter** limits the total number of records shown. Again, you can use constants, or you can use tags from PLCs or variables from View Scripts.

**Key Filter** can limit the data shown by specifying filters for data items. Keys must be enabled in data-log definition to show up in the filter. For each defined key, you can specify the value. To specify a value, you can use constants, or you can use tags from PLCs or variables from View Scripts.

- *Time Scale [min]* parameter specifies the time interval that will be shown by the Advanced Trend. Units are minutes.
- *Background Color* can be set to transparent or to a specific color. If you set this parameter to color, you can specify which color will be shown as the background.
- *Row Click Callback* is a neat feature: when a user selects a row in a table, you will get a callback in the specified JavaScript function. This function must be defined in View Script.

### Example on showing filtered datalog data

Suppose you want to show an on-screen data-log view table, but only show data filtered by user input.

1. Create data-log and data-log view. In the data-log tag definition, check the Key check box to enable filtering using this tag value.

FilteredDataLogData/Datalog - default

Connection:

Day:  Hour:Min:Sec.Msec:  :  :  .

Log rate:  Read Refresh:  :  :  .

Triggered logging

Log single value  Continuous logging

Number of samples to log

Before event:  After event:

Alarm IDs:  ...

Id	Name	Description	Tag	Unit	Format	Hysteresis	Delta Abs	Type	Parameter	Alignment	Key
1	output		output@s...		#. #	<input type="checkbox"/>	0 Value	Decimal		left	<input checked="" type="checkbox"/>
	new				###	<input type="checkbox"/>	0 Value	Decimal		left	<input type="checkbox"/>

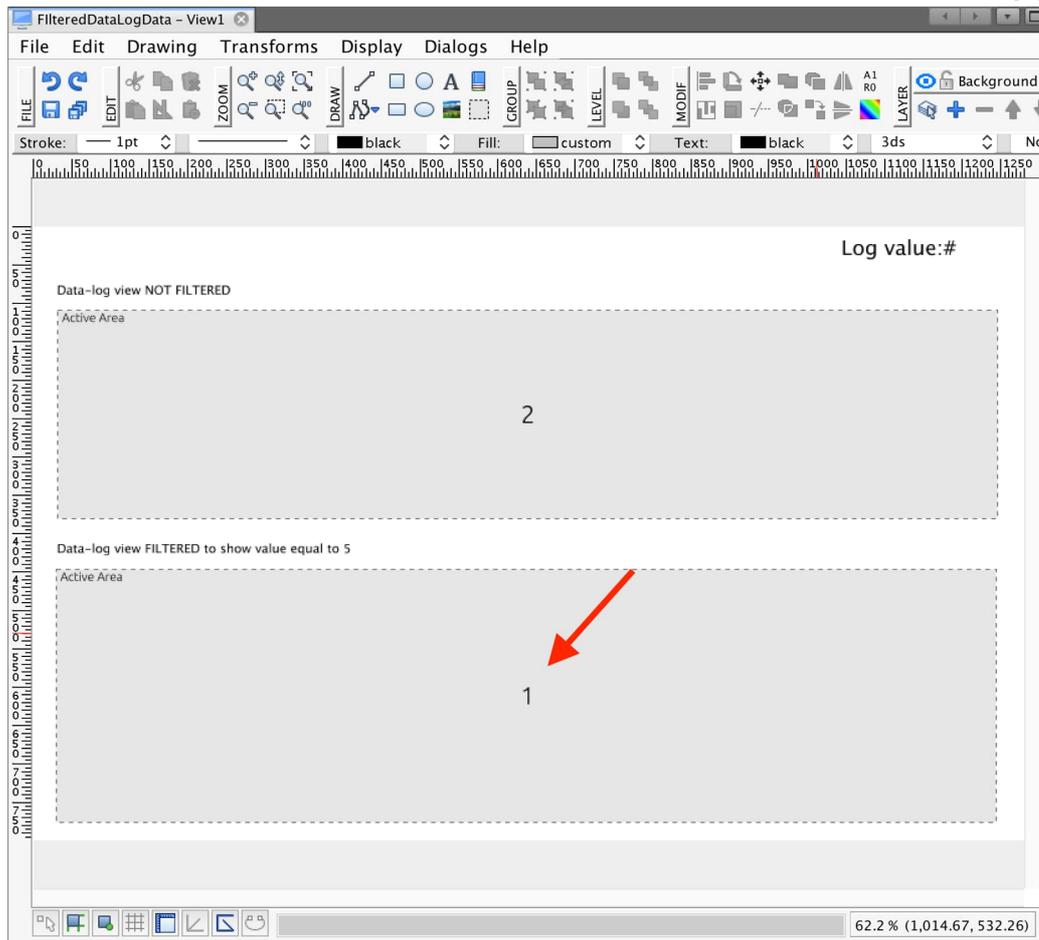
Down ↓ Up ↑

Export CSV, PowerBI  Enable Data Filter

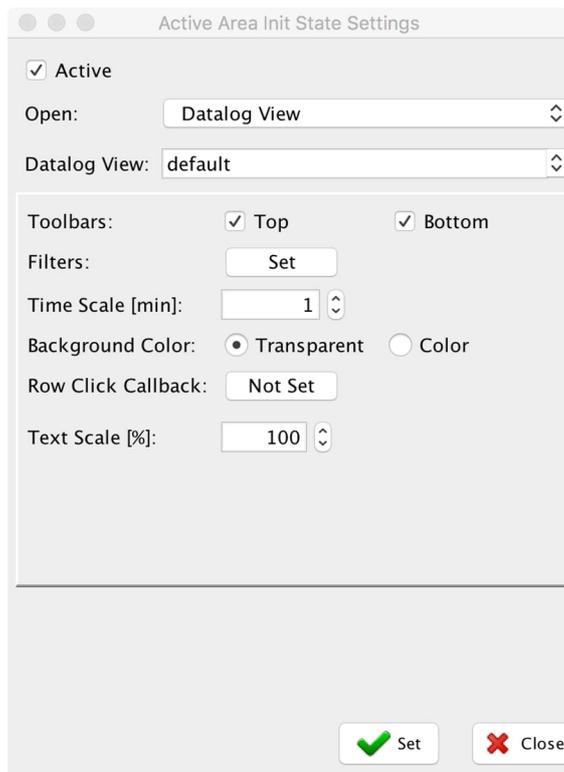
Data-log Views

Name	Description	Data Points (IDs)	Show in Menu	Hide ID	Hide Date	Accesses
default		1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Set
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Set

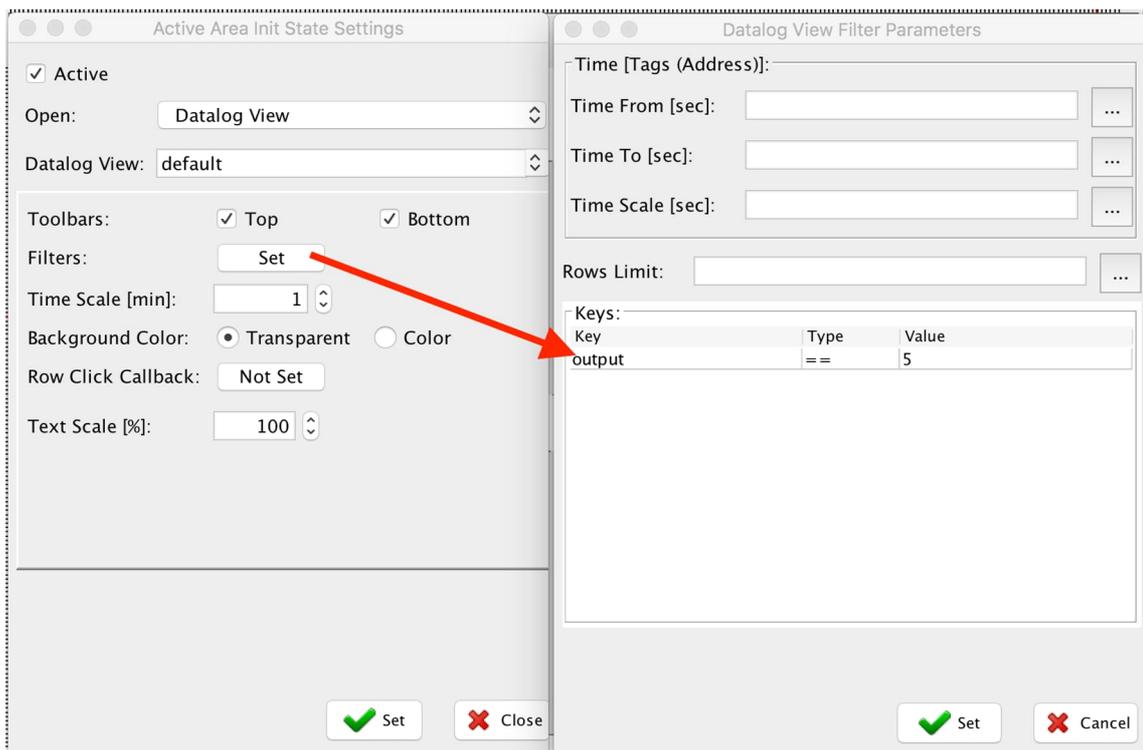
2. Create a new view and insert Active area into it.



### 3. Set up its init state.



## 4. Set up data filter as follows:



## 5. Test your view.

Views Alm (0/0) ... View1 50%

Log value:4

Data-log view NOT FILTERED

LIMIT: 10000 FILTER TIME ON Export

#	TIME	output
32	October 10, 2016 20:10:24	2.0
31	October 10, 2016 20:10:22	9.0
30	October 10, 2016 20:10:20	7.0
29	October 10, 2016 20:10:18	5.0
28	October 10, 2016 20:10:16	3.0
27	October 10, 2016 20:10:13	1.0

5 minutes

Data-log view FILTERED to show value equal to 5

LIMIT: 10000 FILTER TIME ON Export

#	TIME	output
3	October 10, 2016 20:10:18	5.0
2	October 10, 2016 20:10:07	5.0
1	October 10, 2016 20:09:48	5.0

5 minutes

DOWNLOAD DEMO PROJECT HERE:

[ftp://nsa.myscada.org/history/projects/example/Showing\\_filtered\\_data\\_log\\_data.mep](ftp://nsa.myscada.org/history/projects/example/Showing_filtered_data_log_data.mep)

## User Actions type

You can show the history of all user actions. The size of the user actions view will be automatically adjusted to the size of the active area.

The screenshot shows a configuration dialog box for 'User Actions'. It has a title bar with a checked 'Active' checkbox. Below the title bar is a dropdown menu labeled 'Open:' with 'User Actions' selected. The main content area contains several settings: 'Toolbars:' with checked checkboxes for 'Top' and 'Bottom'; 'Time Scale [min]:' with a numeric input field set to '1'; 'Background Color:' with radio buttons for 'Transparent' (selected) and 'Color'; 'Row Click Callback:' with a text input field containing 'Not Set'; and a checkbox for 'Enable on Lock element or Lock Key' which is currently unchecked.

- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g. the date selection controls).
- *Time Scale [min]* parameter specifies the time interval that will be shown in the alarm history table. Units are minutes.
- *Background Color* can be set to transparent or to color. If you set this parameter to color, you can specify which color will be shown as the background.
- *Row Click Callback* is a neat feature: when a user selects a row in a table, you will get a callback in the specified JavaScript function. This function must be defined in View Script.

## SQL Table

The SQL Table feature allows you to show values from the SQL database in a table. To get data from the database, specify your connection first. In the connection settings, you can modify the SQL selection to suit your requirements.

*TIP: You can create a new SQL connection specifically for use with this table.*

### Parameters:

Once is your connection set up, you can use parameters to modify your SQL query. In the Parameters, you can put multiple parameters. Each parameter can be a constant or you can use tag or variable from View Scripts.

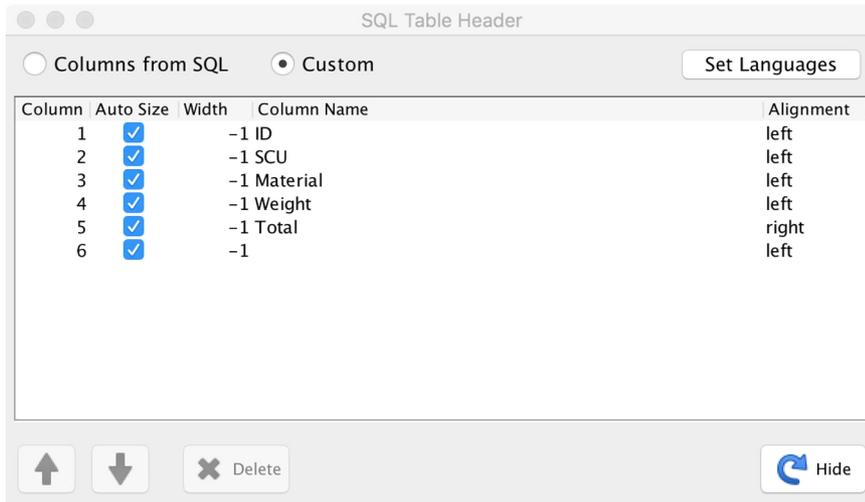
- using constants: just write the value into the text field
- using Tag: enter the tag value or use the “...” button to specify a tag.
- using Variable: enter an = followed by the variable name.

### Connections:

By passing the connection parameters, you can override the connection that will be used to show the SQL table. This option is especially useful if you are working with parametric views. To set your connection parameters, click on the “...” button next to the Connections field.

### Show Header

This option enables you to show the header row. If you select this option, you can fine-tune the parameters by pressing the “Customize” button.



The default option is “Columns from SQL.” If you use this option, your table will show a top row with a description of columns taken directly from the SQL.

You can also set custom columns to set up names of columns manually. In addition, with this option, you can specify the Width of the column in pixels. If you leave with=-1, column width will be set to auto-scale. The final option is Alignment, where you can specify whether a given column should be aligned on the left, right, or in the center.

## Background Color

The background Color can be set to transparent or to a given color. If you set this parameter to color, you can specify which color will be shown as the background.

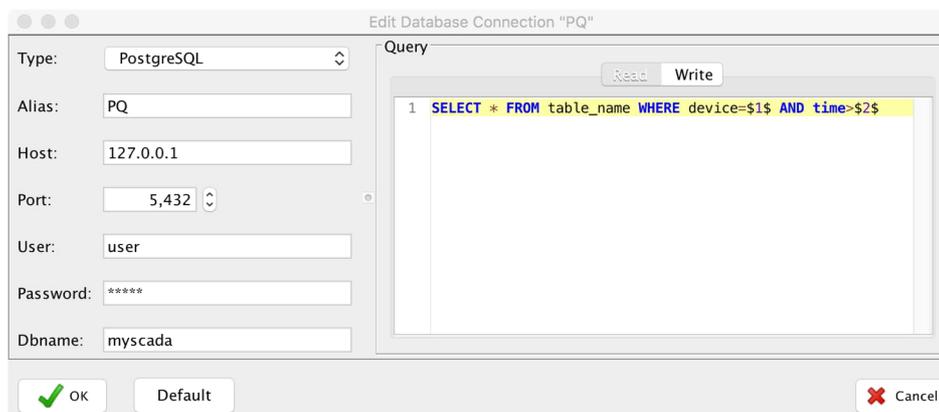
## Row Click Callback

Row Click Callback is a neat feature: when a user selects a row in a table, you will get a callback into the specified JavaScript function. This function must be defined in View Script.

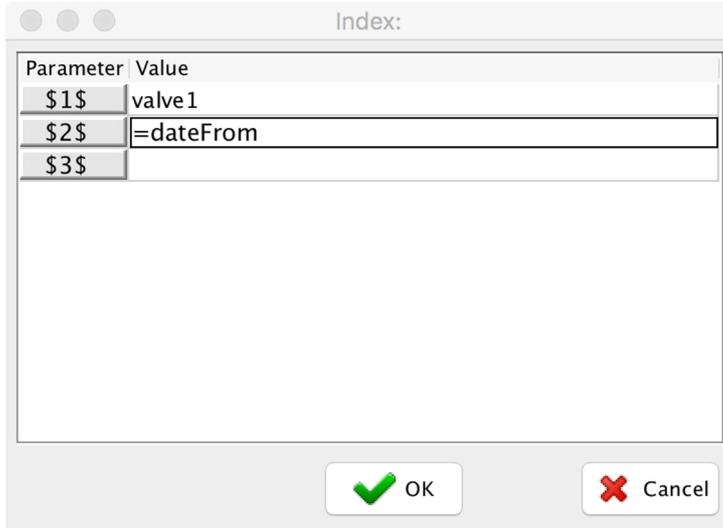
## Modifying Query

Specifying parameters enables mySCADA to modify an SQL query and insert the specified parameters into the SQL query during communication. To do so:

1. Create a connection to be used with the SQL Table. Put \$1\$, \$2\$, ... into the SQL query whenever you want to replace the parameter.



- Now set up Parameters and pass those parameters to the query.



The screenshot shows a dialog box titled "Index:" with a table containing three rows of parameters and their values. The first row has parameter "\$1\$" and value "valve1". The second row has parameter "\$2\$" and value "=dateFrom". The third row has parameter "\$3\$" and an empty value field. Below the table are two buttons: "OK" with a green checkmark icon and "Cancel" with a red X icon.

Parameter	Value
\$1\$	valve1
\$2\$	=dateFrom
\$3\$	

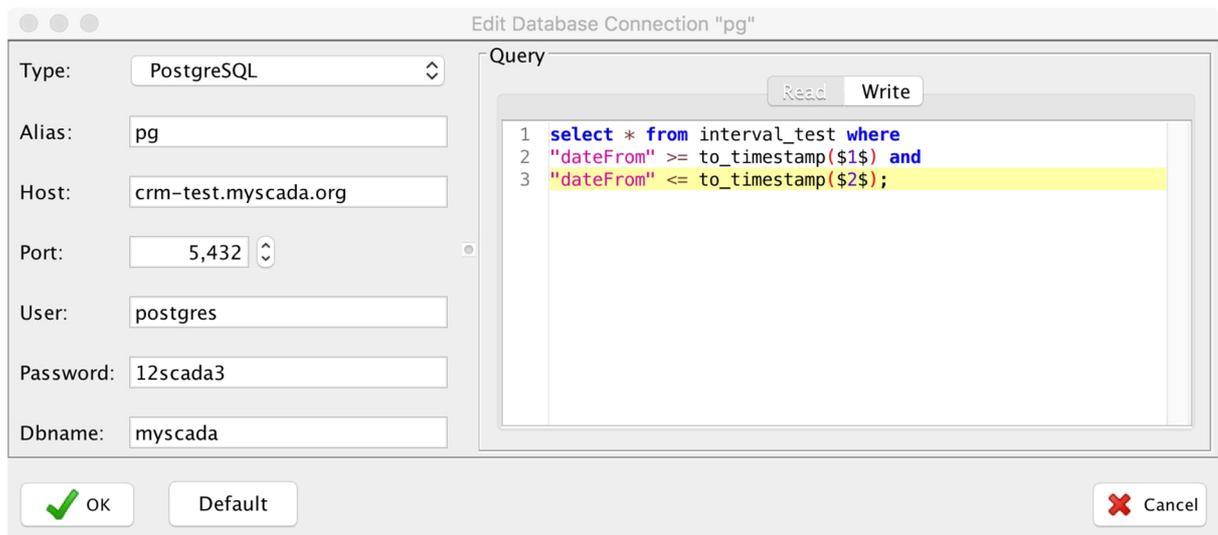
- When mySCADA processes the SQL Query, it will replace **\$1\$** for **valve1** and **\$2\$** for the value of the variable **dateFrom**.

### Example:

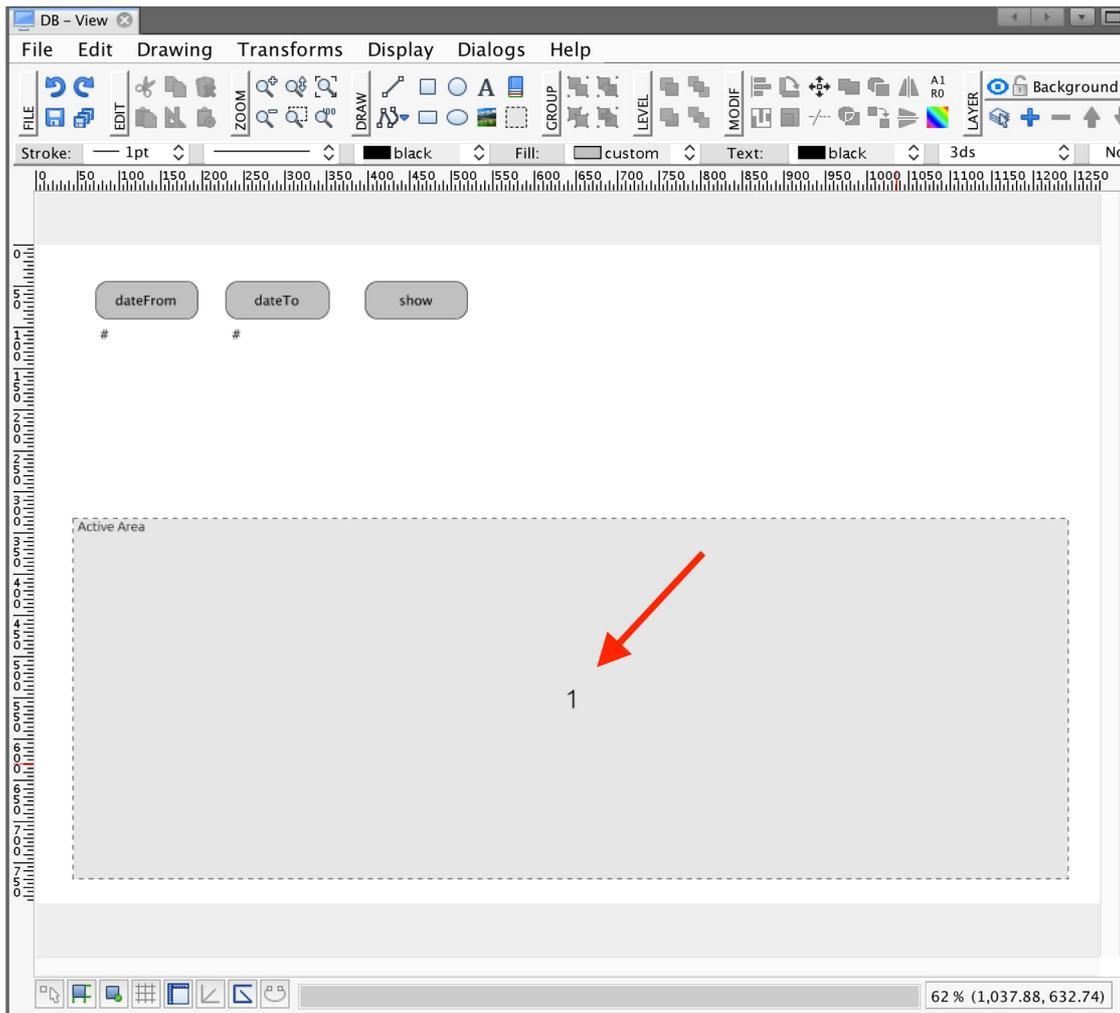
Let's say you need to read data from a database in a given time interval. The time range is specified by the View Script variables `timeFrom` and `timeTo`. The SQL command could look like this:

```
SELECT * FROM table_name WHERE  
"dateFrom" >= to_timestamp($1$) and  
"dateFrom" <= to_timestamp($2$);
```

1. This SQL command should be specified in the Connection definition.



- Now create a new view. Insert it into the active area.



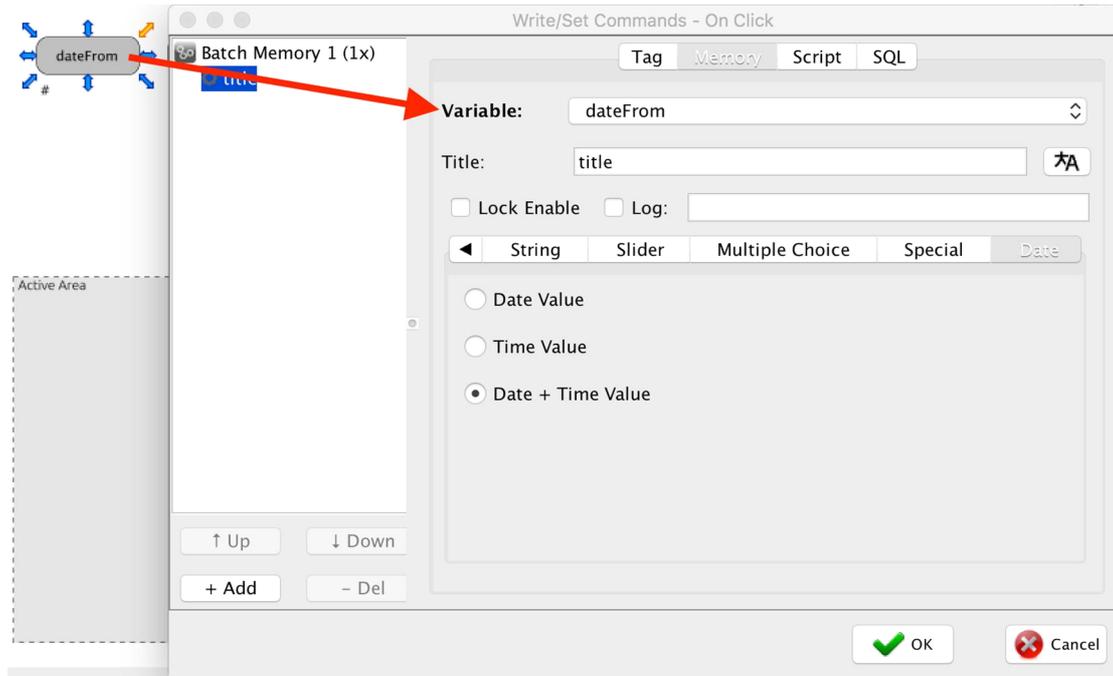
- Now create DateFrom, DateTo, and show buttons:



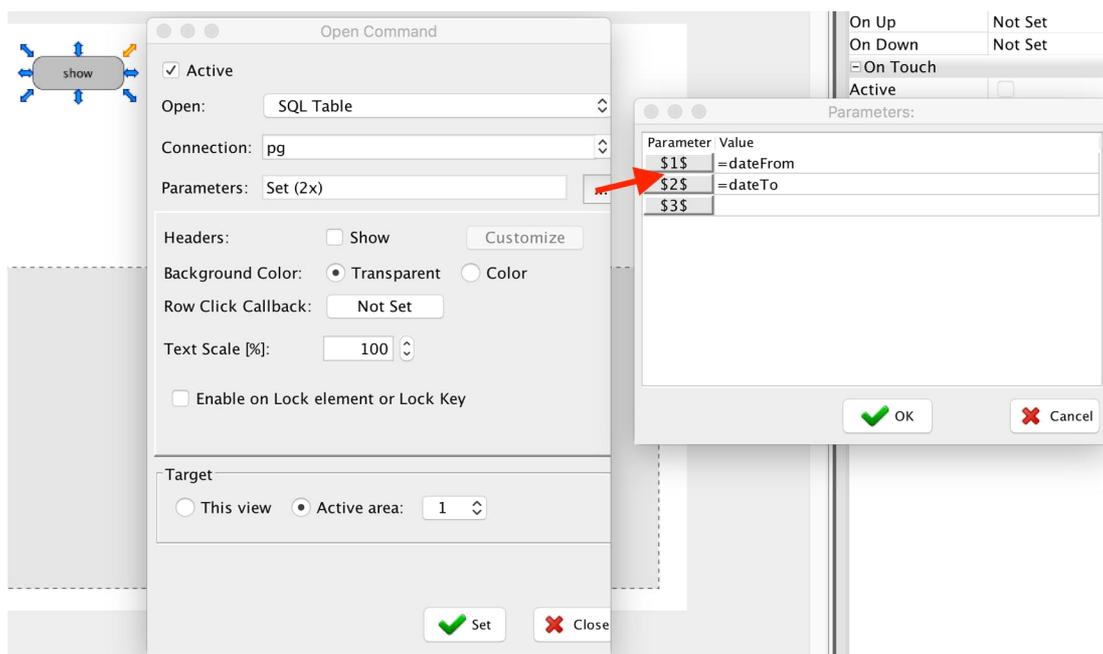
- Now open the view script and create two variables dateFrom and dateTo:

Variable	Value (Double)
dateFrom	0
dateTo	0
	0

- Create a set command for the dateFrom and dateTo buttons. We will set the date into the view script variables dateFrom and dateTo:



6. Now use the show buttons to open SQL table into Active Area:



As you can see, we are using two parameters that are automatically placed into SQL before the readout.

7. Now test your view:

DOWNLOAD DEMO PROJECT HERE:

[ftp://nsa.myscada.org/history/projects/example/Database interval data.mep](ftp://nsa.myscada.org/history/projects/example/Database%20interval%20data.mep)

## 9.3 External Web Page in Active Area

To show an external web page in the active area, select *HTML -> External*. Enter the HTML address you want to open in the *Link* field.

The screenshot shows the 'Active Area Init Value Setting' dialog box. It has a checked 'Active' checkbox. The 'Open:' dropdown is set to 'HTML'. The 'HTML:' dropdown is set to 'External'. The 'Link:' text field contains the URL 'http://www.myscadatechnologies.com'. There is a three-dot menu button to the right of the link field.

## 9.4 HTML Code

To show a custom HTML code in the active area, select *HTML -> HTML Code*. Enter your HTML code in the *Code* section. If you press the “...” button, you can open the HTML editor to enter your code.

The screenshot shows two overlapping dialog boxes. On the left is the 'Active Area Init Value Setting' dialog box. Its 'Active' checkbox is checked. Under 'Parameters', the 'Open:' dropdown is 'HTML' and the 'HTML:' dropdown is 'HTML Code'. The 'Code:' text field contains '<h1>TITLE</h1>'. A three-dot menu button is to the right of the code field. On the right is the 'HTML Code Editor' dialog box. It has a text area with two lines of code: line 1 is '<h1>TITLE</h1>' and line 2 is 'Your own HTML snippet'. A red arrow points from the three-dot menu button in the left dialog to the text area in the right dialog. Both dialog boxes have 'Set' and 'Close' (or 'Cancel') buttons at the bottom.

## 9.5 DIV Type

The *Active Area* element can also work as a placeholder for your web component. In the runtime, *mySCADA* will create an empty DIV component in the location and size of your active area. This DIV component can be manipulated in *View Scripts*.

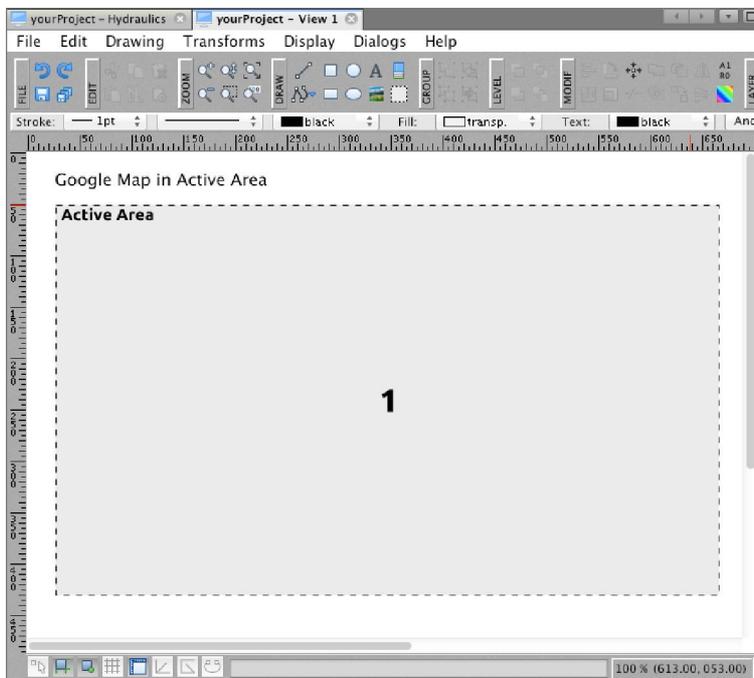
The screenshot shows the 'Active Area Init Value Setting' dialog box. It has a checked 'Active' checkbox. The 'Open:' dropdown is set to 'HTML'. The 'HTML:' dropdown is set to 'Div Element'. The 'ID:' text field contains the value 'DivElement0001'. There is a three-dot menu button to the right of the ID field.

## Active Area

Select *HTML -> Div Element*. Fill in the *ID* of your DIV (active area) component. With this *ID* you can access your component in the *View Scripts* using the function `document.getElementById ("ID")`.

### Example:

In the following example, we will use the *Active Area* to display a *Google* map. First of all, create a view and insert it into the active area. Then set the type to *HTML -> Div Element* in the *Init Value*. Type in 'googlemap' into the *ID type* field.



Now all we need to do is to initialize the *DIV* element with *Google* maps. This can be done in *View Scripts*. Open the view script and enter the code for initialization into the *Init function* field.

```

1 function periodic() {
2
3 }
4
5 function init() {
6     window.gmap_draw = function(){
7         var mapProp = {center:new google.maps.LatLng(50.100195, 14.390853),zoom:14,
8             mapTypeId:google.maps.MapTypeId.ROADMAP};
9         map=new google.maps.Map(document.getElementById("googlemap"),mapProp);
10
11         var marker=new google.maps.Marker({
12             position:new google.maps.LatLng(50.100195, 14.390853),
13             animation:google.maps.Animation.BOUNCE
14         });
15         //var map=document.getElementById("aaa");
16         marker.setMap(map);
17     };
18
19     var script = document.createElement('script');
20     script.src = "http://maps.google.com/maps/api/js?sensor=true&callback=gmap_draw";
21     document.head.appendChild(script);
22 };
23

```

PLC Variables Tables Window

Variable	Tag/Equation	Type
		Value

There are two interesting parts in this code:

- 1) This part dynamically loads external *JavaScript* from the Internet - in our case Google map API:  

```

var script = document.createElement('script'); //create script element
script.src
    =
    "http://maps.google.com/maps/api/js?sensor=true&callback=gmap_
    draw"; //url to the script
document.head.appendChild(script); //load and append js

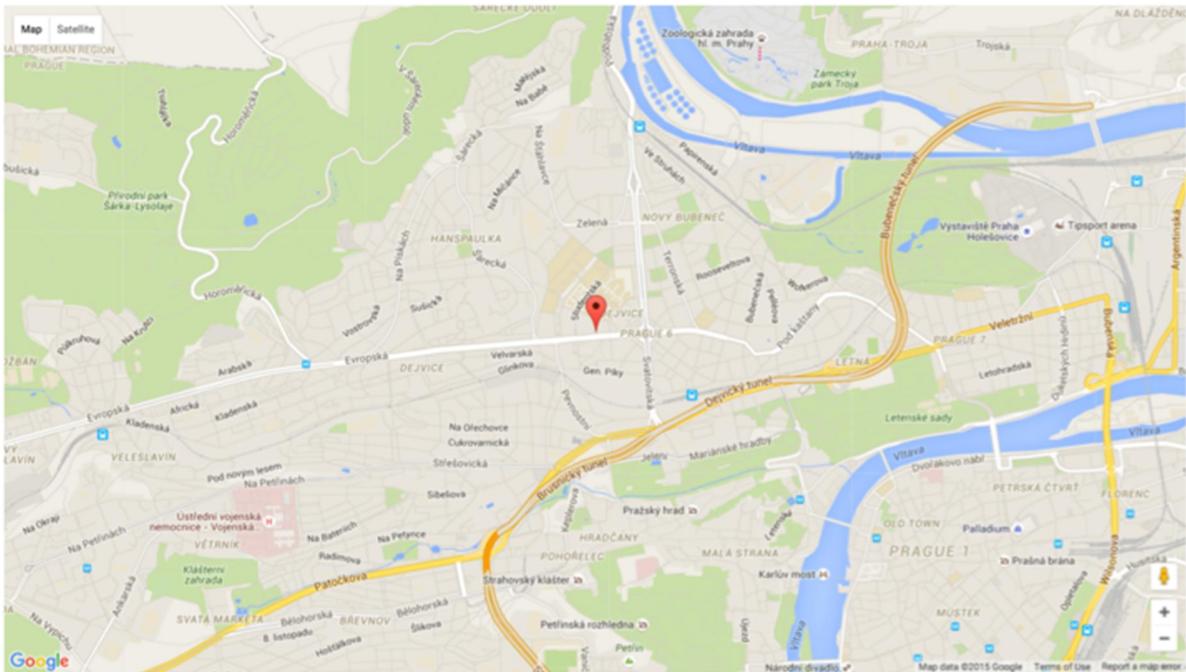
```
- 2) This piece of code will find our DIV element (e.g. Active area) with googlemap id  

```

document.getElementById("googlemap")

```
- 3) And finally the result:

## Google Map in Active Area



DOWNLOAD DEMO PROJECT HERE:

[ftp://nsa.myscada.org/history/projects/example/DIV\\_Google\\_maps.mep](ftp://nsa.myscada.org/history/projects/example/DIV_Google_maps.mep)

# 10 Layout Views

---

Watch video describing this functionality:

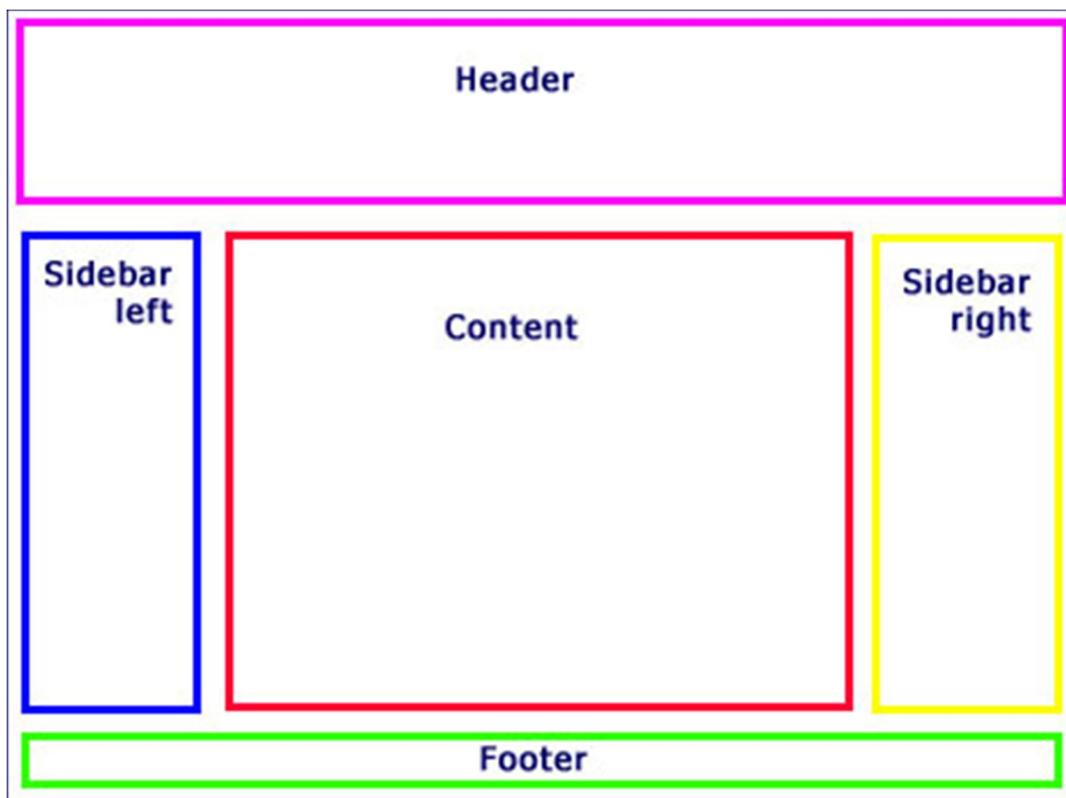
<https://www.youtube.com/watch?v=sVvFPCMYUZ8>

## 10.1 Page Layout

The *Layout* page defines the arrangement and style of the page content. In *myDESIGNER*, you can create multiple user-defined layouts. Each view that you create can use different layouts.

*Note: You can create new layouts at any time during the creation of your project.*

You can see all the options that your layout can have in the following picture:



- Header** top section that can be used, for example, to display a logo, name, logged user, main menu, etc.
- Main Content** section displaying your views; it sits prominently in the middle of the page
- Left/Right Sidebar** columns on both sides of the main content section; useful for displaying additional menus, pictures, etc. or can be used for control

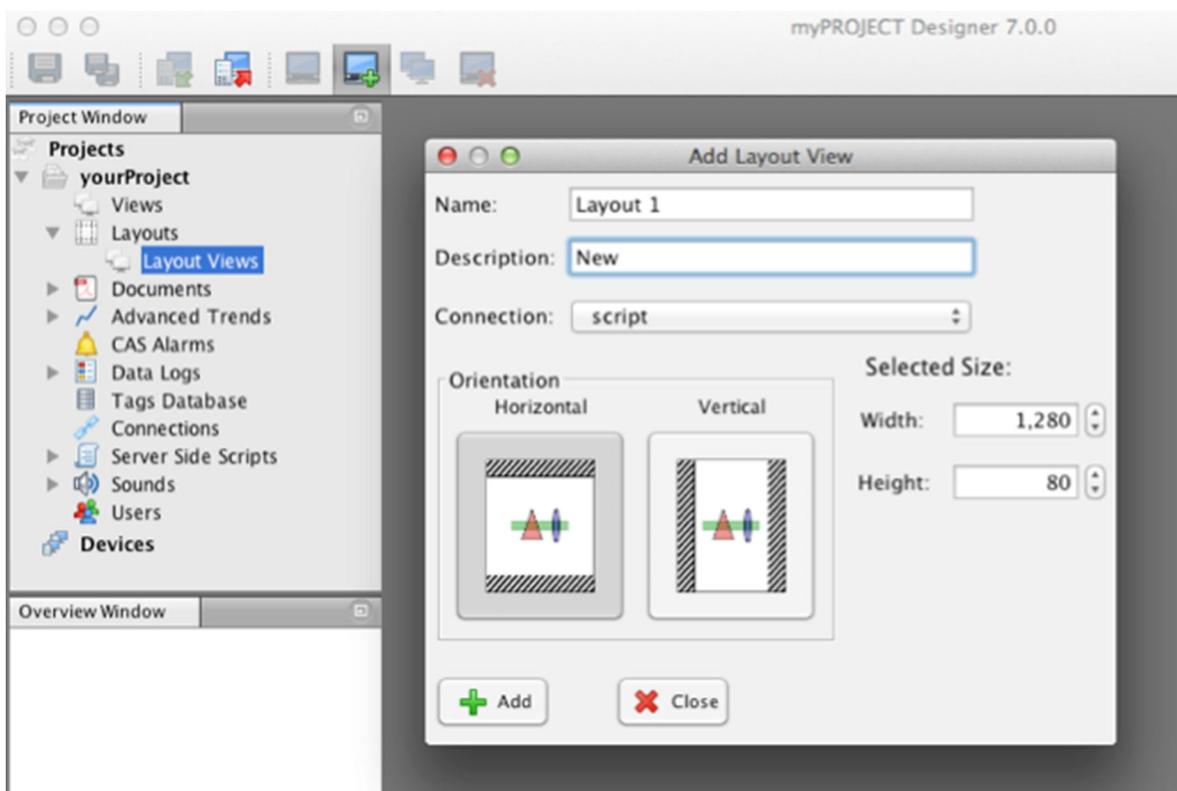
buttons and gauges. Sidebars can be visible or shown upon user action.

**Footer** bar spanning the bottom of the page

*Note: To use the Layouts, first you have to create Layout Views. The Layout Views behave exactly the same as regular views; therefore, you can apply any functionality to them, such as animations, effects, or view scripts.*

## 10.2 Adding Layout View

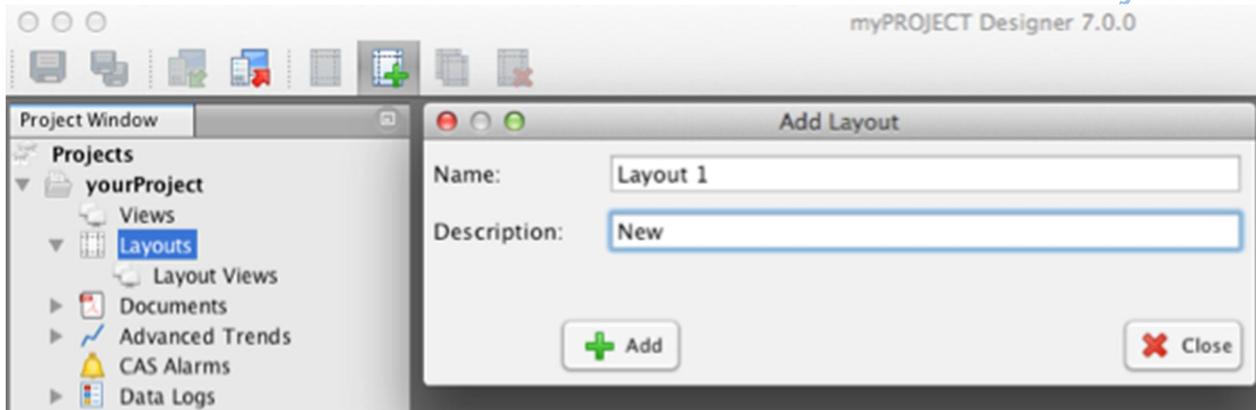
- 1) Select *Layouts* -> *Layout Views*
- 2) Click on the **Add Layout View** icon in the main toolbar
- 3) Enter the name and description
- 4) Define the orientation and select the layout size



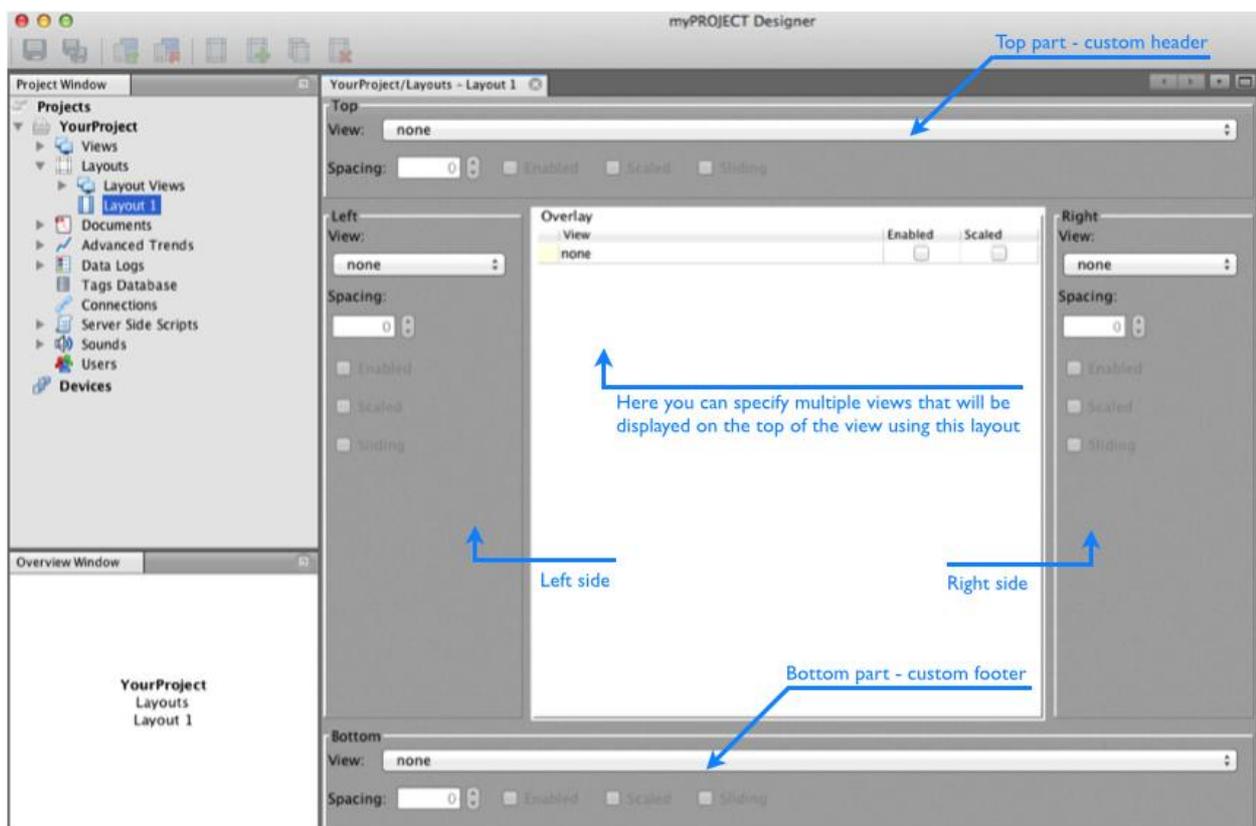
## 10.3 Creating New Layout

After you have created layout views, you can use them in the layouts:

- 1) Select *Layouts* in the *Project Window*
- 2) Click on the **Add Layout** icon
- 3) Enter the name and description and click on **Add**



After creating a new layout, double-click on it to fill in its properties:



For each section, you can select a corresponding *Layout View*. After you have selected a required view, do not forget to check the *Enabled* box to activate it.

Options for each section:

- *Spacing* – creates spaces between the sections; defined in pixels
- *Enabled* – checkbox for enabling particular sections
- *Scaled* – scales the section accordingly to the rest
- *Sliding* – hides the section, i.e. the user can open it by clicking on the tab in the center

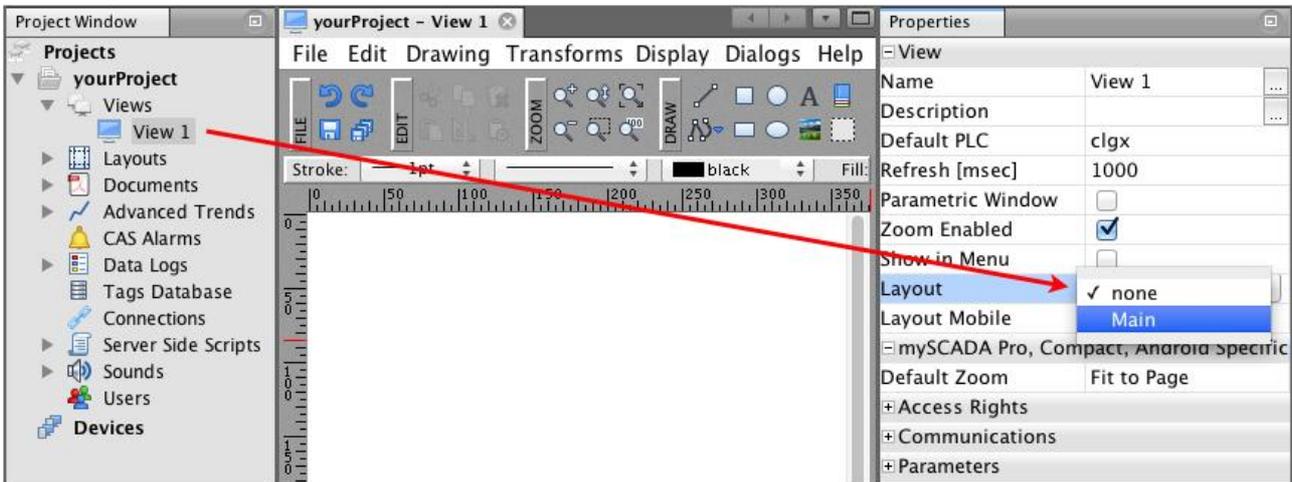
### Overlay Section

In this section, you can put one or more layout views above existing ones. This can be useful for showing messages or images tied to a visibility animation – instead of copying

these messages or images into all views, you just add them to a layout view and use it as an overlay for all of the views.

### Using Layouts in Views

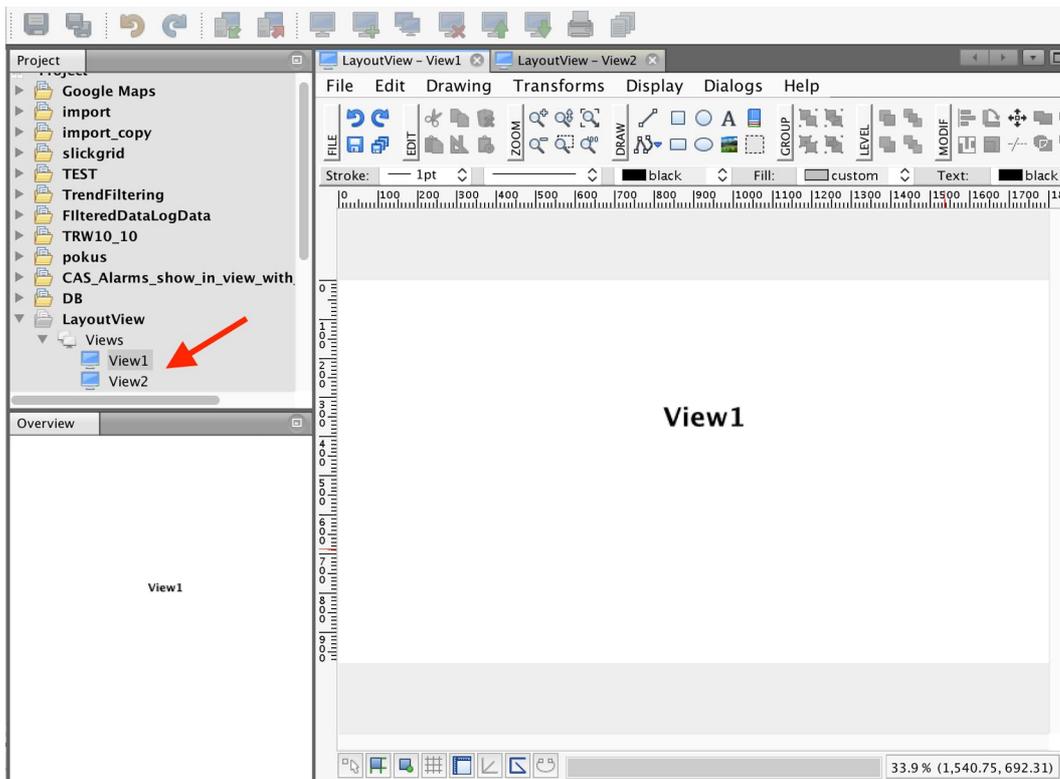
Select the view you want to apply a layout to and select *Layout* from the drop-down menu in the *Properties* window.



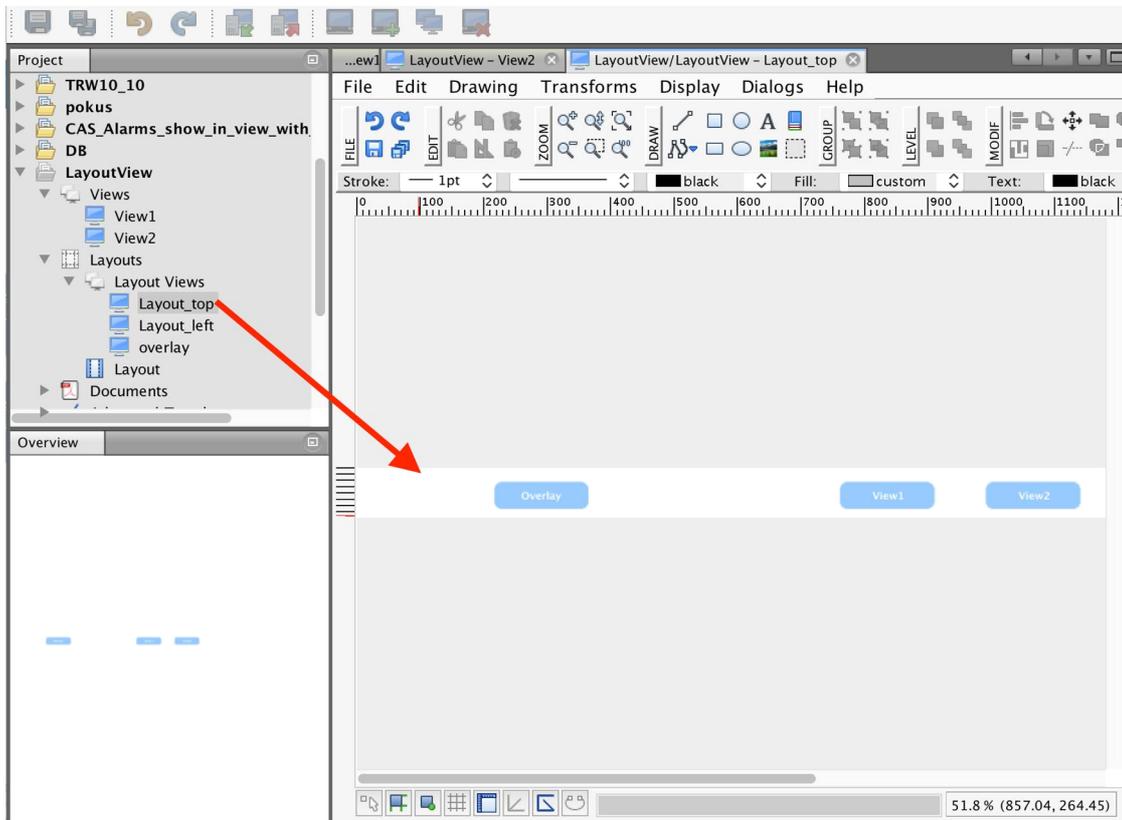
### Layout Views Example

In the following example, we will show you how to add a permanent menu above the views, how to show a message over the views, and how to show a sliding left side menu.

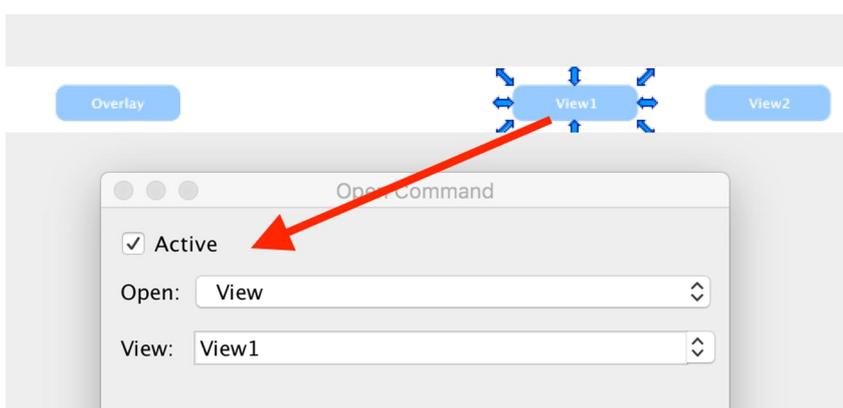
1. Create View1 and View2 in your project.



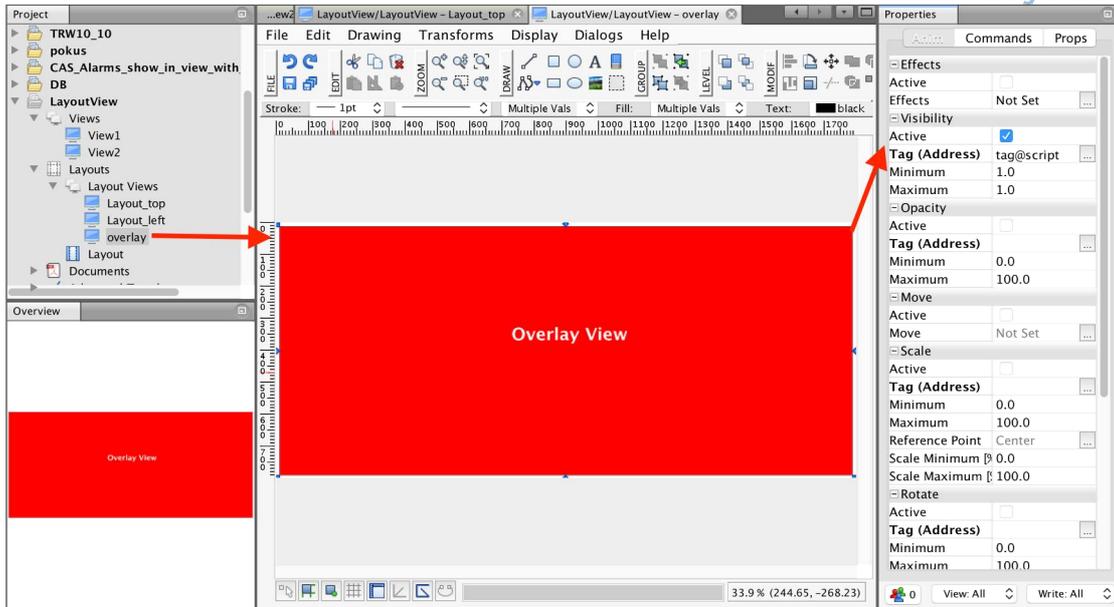
2. Create a horizontal view in Layout views; it will be used for our top menu. Now add buttons “View1.” “View2,” and “Overlay.”



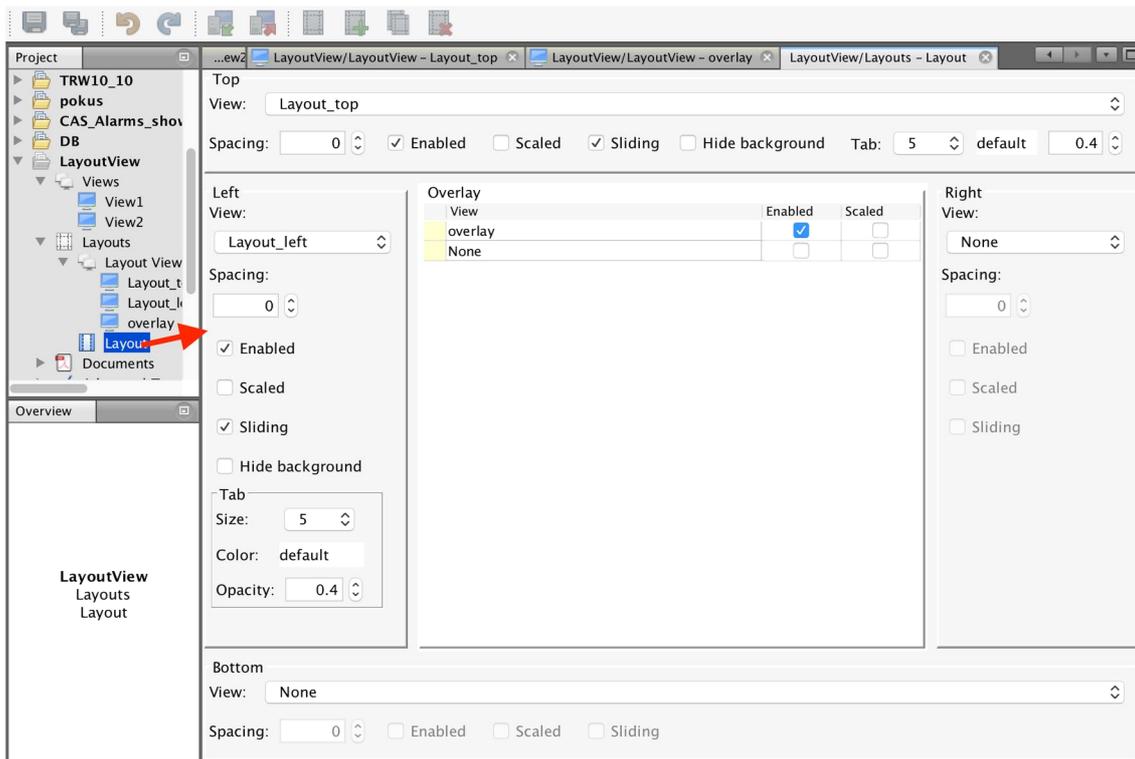
3. On the button “View1” add the open action to open view1. Do the same for the other button.



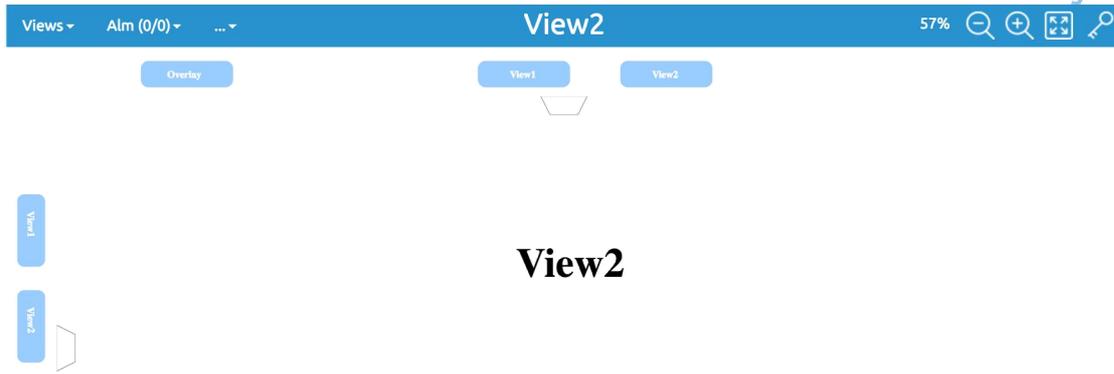
4. Now create the left side view in Layout Views. Again, add two buttons to it. Create them the same way as you did in the menu view.
5. Now create the overlay view in Layout Views. Make it the same resolution as your views. Add a text message to this view. Set the visibility animation for the text message.



- Now create your layout. For the top section, select the menu view. For the left section, select the left side view and check the sliding check box to add the overlay menu.



- Now, test your view.



DOWNLOAD DEMO PROJECT HERE:

[ftp://nsa.myscada.org/history/projects/example/Layout\\_view.mep](ftp://nsa.myscada.org/history/projects/example/Layout_view.mep)

# 11 Entering Tags and Math Expressions

## 11.1 Entering Tags

Before linking your visualizations with PLCs, first, you have to enter the name or address of the tags from which you wish to read/write data.

Watch video describing this functionality:

<https://www.youtube.com/watch?v=XNrNiTUPFDk>

The tag syntax depends on the PLC type that you want to access.

Tag syntax: `tag@connection_alias`



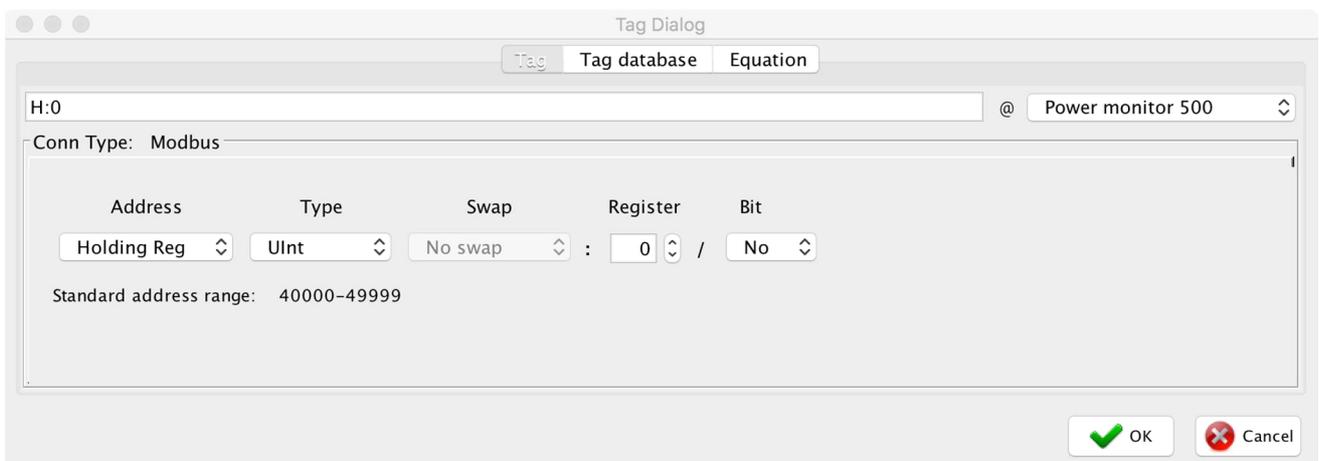
*Note: You don't need to enter full tag syntaxes all the time. Instead, you can use a simplified link to your tag, called an Alias. The Aliases can be defined in the Tag Database.*

Alias syntax: `*alias`

You can type the tag directly into the tag edit field:

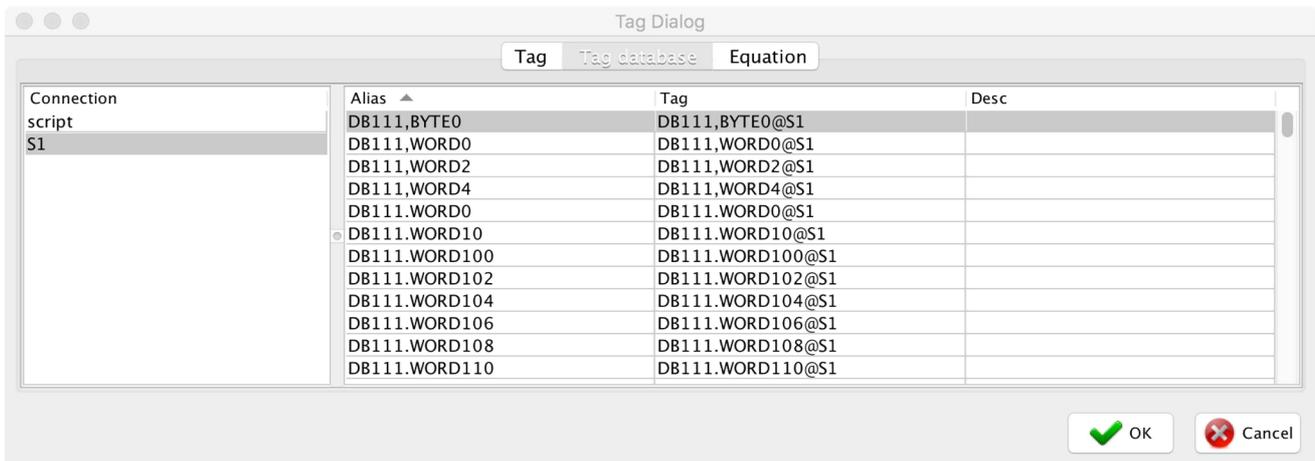
Tag (Address) `tag@PLC`

You can call the tag editor by clicking on the ... button on the right side of the tag edit field - a new dialog window will show up:



The *Tag Dialog* will guide you through entering your tag and will check if the syntax is correct.

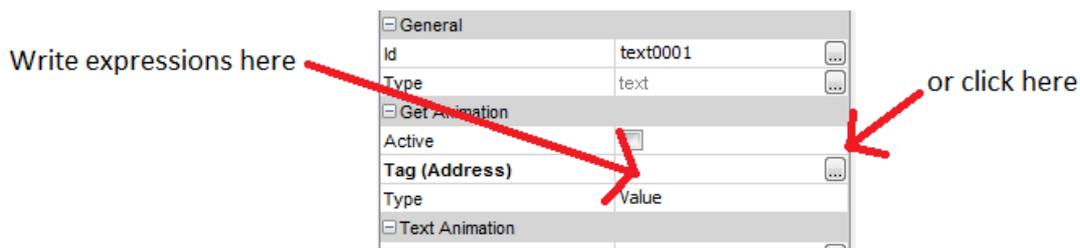
You can also choose your tag directly from the tag database. To do so, please click on the *Tag database*. The dialog will present you with all the tags entered in the tag database.



On the left side of the dialog, you can select connection. After you select it, you can select a tag on the right side of the window.

## 11.2 Entering Mathematical Expressions

Instead of writing the tag name, you can enter mathematical expressions. This way you can scale and offset the values read from the PLCs or create more complex data processing. These expressions can be entered either directly or through the dialog window.



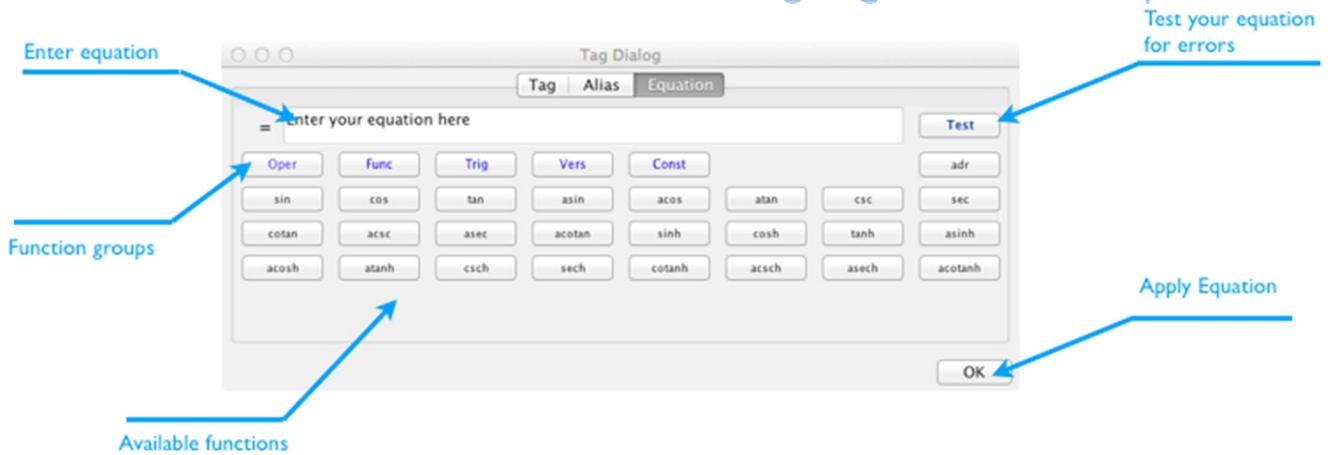
Equation syntax:  $=2*adr(HI00@wago)+alias(offset)+sin(...)$

Annotations:

- denotes equation (points to the equals sign)
- referencing a tag (points to adr)
- referencing a tag by alias (points to alias)

In the equations you can use operators + - \* / and common mathematical functions such as *sin*, *cos*, *exp*, etc... You can also do a binary comparison and much more. To get the complete list of options, call the *Tag Dialog* and click on the tab **Equation**.

## Entering Tags and Math Expressions



### Formatting and Limits

- Tag name is entered by the *adr()* function
- Tag name can be also entered as an alias using the *alias()* function
- You can use only supported functions and operators

The supported functions and operators are listed under the input box and are broken down into five groups: **Oper**, **Func**, **Trig**, **Vers**, and **Const** – they will be inserted into the box if selected.

You can use multiple tags in an expression.

#### Example:

- 1) We read a value from Modbus H:0
- 2) Let's scale this value by 10 and offset it by 0.5
- 3) Formula to enter is:  $=10*adr(H:0)+0.5$

### Supported Functions and Operators:

Operator	Function	Description
<b>Standard Operators</b>		
+	add	addition and unary positive
- r -	subtract and negative	subtraction and negation
o	multiply	multiplication
r÷	divie	division
	percnt	modulus and percentage of a value
!	facioial	factoria
**	po	exponential
	deg	converts values to radians
<b>Bitwise Operators</b>		
&	an	btise and
	or	bitwise or
^	xor	bitwise xor

~	not	bitwise not
<<	lshift	bitwise left shift
>>	rshift	bitwise right shift
<b>Comparison Operators</b>		
=	_eq	equal
!=	_neq	not equal
<	_lt	less than
>	_gt	greater than
<= or ≤	_toe	greater than or equal
>= or ≥	_gtoe	greater than or equal
<b>Logical Operators</b>		
&& or ∧	_and	logical and
or ∨	_or	logical or
! or ¬	_not	logical not

### *Degree operator °*

The degree operator (°) is very useful when converting user input. Because all of the trigonometric functions require their parameters to be in radians, the degree operator will convert its operand into radians. Thus,  $45^\circ$  is equivalent to  $dtor(45)$ .

### *Percentage sign %*

When the percent sign is interpreted as the *modulo*, then:

$10 \% 3 \dots$  evaluates to 1 (the remainder after 10 is divided by 3); however, if you flip the switch to make the % sign stand for percentage, then it becomes:

$250 + 10\%$

By default, the percentage sign % is usually shorthand for "/100". In other words, 42% becomes 42/100 or 0.42.

However, if the % term on the right-hand side of *subtraction* or *addition* (such as in "250 + 10%"), then the percent is evaluated as a percentage of the left-hand side (i.e. "250 plus 10% of 250").

If you choose to interpret the percent sign as the modulo operator, you can still request a percentage by using the function name directly:

$(10 \% 3) + \text{percent}(50) = 1.5$

### *Factorial and Logical Not (!)*

Differentiating between factorial (!) and logical NOT (!) is more difficult.

Logical NOT is interpreted as factorial (!) only if:

- it is the first token

- it is preceded by a binary operator
- it is preceded by a right associative unary operator

Otherwise (!) is always treated as factorial and the negating token - as logical NOT.

### Supported Functions

#### *Functions using more than one parameter:*

- **sum()** - returns a sum of the passed parameters
- **count()** - returns the number of passed parameters
- **min()** - returns the minimum of the passed parameters
- **max()** - returns the maximum of the passed parameters
- **median()** - returns the median of the passed parameters
- **stddev()** - returns the standard deviation of the passed parameters
- **average()** - returns the average of the passed parameters
- **random()** - returns a random integer. Can take 0, 1, or 2 parameters. The first parameter (if given) is the lower bound of the random integer. The second parameter (if given) is the upper bound of the random integer.
- **nthroot()** - returns the  $n^{\text{th}}$  root of a number; for example, *nthroot(27,3)* returns the cube root of 27, or 3.

#### *Functions using one parameter:*

- **sqrt()** - returns the square root of the passed parameter
- **log()** - returns the base 10 log of the passed parameter
- **ln()** - returns the base e log of the passed parameter
- **log2()** - returns the base 2 log of the passed parameter
- **exp()** - returns e raised to the power of the passed parameter
- **ceil()** - returns the passed parameter rounded up
- **floor()** - returns the passed parameter rounded down

#### *Trigonometric functions:*

- **sin(), cos(), tan()**
- inverses (**asin, acos, atan**)
- reciprocals (**csc, sec, cotan**)
- reciprocals of the inverses (**acsc, asec, acotan**)
- hyperbolic variations (**sinh, cosh, tanh, asinh, acosh, atanh, csch, sech, cotanh, acsch, asech, acotanh**)
- versine functions (**versin, vercosin, coversin, covercosin, haversin, havercosin, hacoversin, hacovercosin, exsec, excsc, crd**)
- **dtor()** - converts the passed parameter from degrees to radians
- **rtod()** - converts the passed parameter from radians to degrees

#### *Functions using no parameters ("constant functions"):*

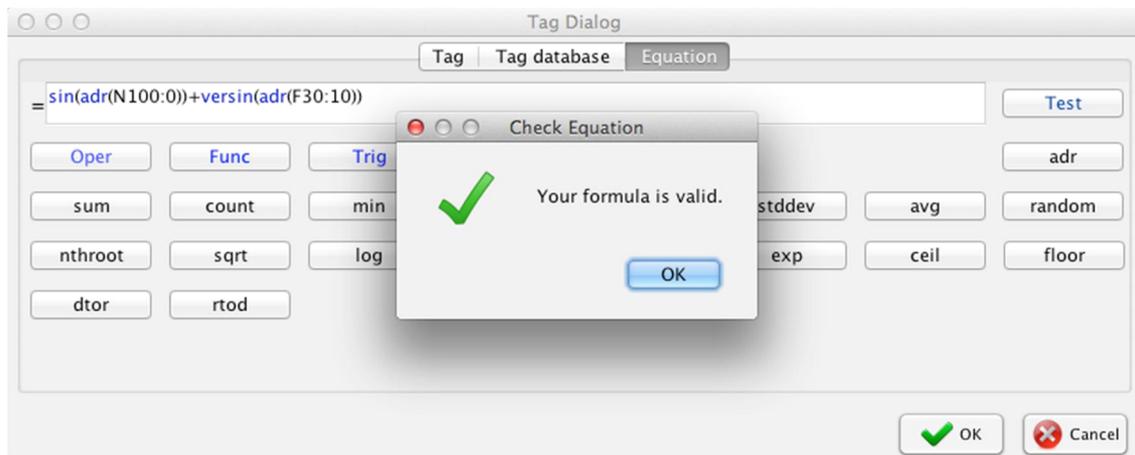
- **phi()** - returns the value of  $\phi$  (the Golden Ratio), also recognized as  $\phi()$
- **pi()** - returns the value of  $\pi$ . Also recognized as  $\pi()$
- **pi\_2()** - returns the value of  $\pi/2$
- **pi\_4()** - returns the value of  $\pi/4$
- **tau()** - returns the value of  $\tau$ . Also recognized as  $\tau()$
- **sqrt2()** - returns the value of the square root of 2
- **e()** - returns the value of e
- **log2e()** - returns the value of the log base 2 of e
- **log10e()** - returns the value of the log base 10 of e

## Entering Tags and Math Expressions

- **ln2()** - returns the value of the log base e of 2
- **ln10()** - returns the value of the log base e of 10

The parentheses are used for grouping sub-expressions and setting the order of execution, and they can be nested to any depth. All computation is carried out with a double precision floating point. In the case of error, the editor shows a warning, and the error expressions will not be evaluated.

You can always check the validity of the entered expression Clicking on the **Test** button.



Click on **OK** after completion to see the expression in the **Tag(Address)** field.

### *Examples of valid expressions:*

- `adr(N100:0)*adr(F30:10)`
- `sin(adr(N100:0))+cos(adr(F30:10))`
- `median(adr(N100:0),adr(F30:10),adr(N20:5),adr(F10:10))`

*Note:* Multiple arguments should be separated with commas  $2 \geq 1$ . Logical operations always return binary result (0 or 1).

# 12 Tag Database

This feature is very useful for management of all tags and connections. You can start creating your project by creating the tag database and then design the visualizations afterward.

	Alias	Tag@Conn	Description	Unit	Format	Eng. Unit	Usage Count
1	valve A1	valveA1@CLGX	Valve WasteVater A1	O/C	#	Not Set	0
2	valve A2	valveA2@CLGX	Valve WasteVater A2	O/C	#	Not Set	0
3	valve A3	valveA3@CLGX	Valve WasteVater A3	O/C	#	Not Set	1
4	T1	I:10@MOD	Motor temperature...	deg C	##	Not Set	0
5	T2	I:11@MOD	Motor temperature...	deg C	##	Not Set	1
6	T:3	I:12@MOD	Motor temperature...	deg C	##	Not Set	0
0					##	Not Set	0

## 12.1 Changing tag value

The most important function of the tag database is the ability to change the used tags all in one place. If you click on the Tag cell in the table, you can simply change the tag (address), and the change is propagated throughout the whole project. Everywhere the tag is used, the old value is replaced with the updated one.

## 12.2 Engineering units

Every tag in your project can be scaled or even have a mathematical equation applied to it. This is extremely useful when you need to get an engineering value from raw values. To enable scaling on a given tag, please select the corresponding row and click on the Eng. Unit cell. A new dialog is shown:

Engineering Unit Settings

Not Set Scale Equation

Scale and offset tag value:

1 \* Tag / 1 + 0

Set Cancel

There are three options you can set:

1. Not Set - no scaling is applied
2. Scale - you can scale and offset the tag value

- Equation – you can apply a mathematical equation for more complex situations

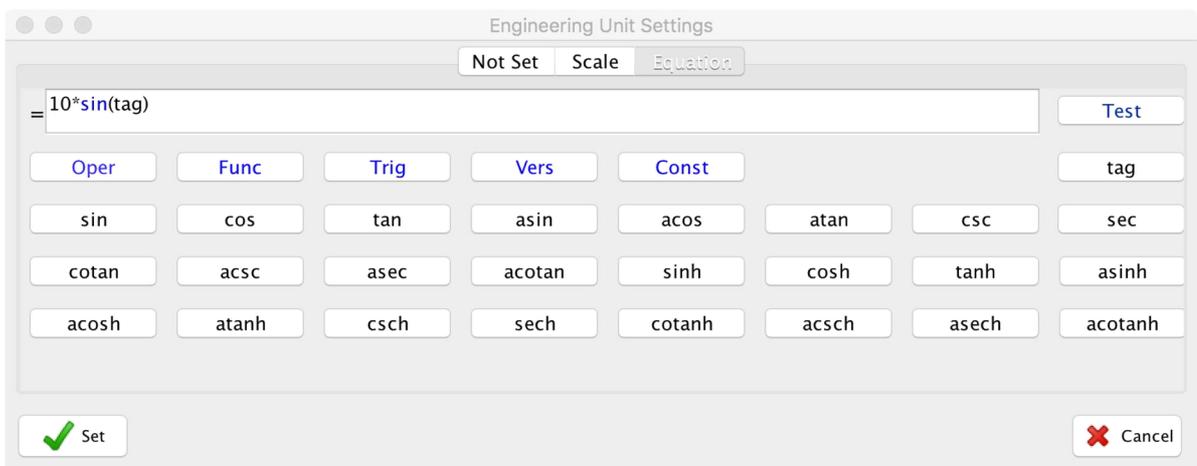
## Scaling tags

To scale a tag, please fill in the equation in the dialog. You can multiply the tag by a constant, divide it, and set the offset. The equation is as follows:

$$\text{Value} = \text{multiplier} * \text{tag} / \text{divider} + \text{offset}$$

## Applying equation

To apply the equation to your tag, select the Equation tag. Here you can use multiple mathematical operations in your formula. The tag value should be inserted into the equation as the word **tag**. The following dialog shows how to compute the sinus value of the tag multiplied by 10.

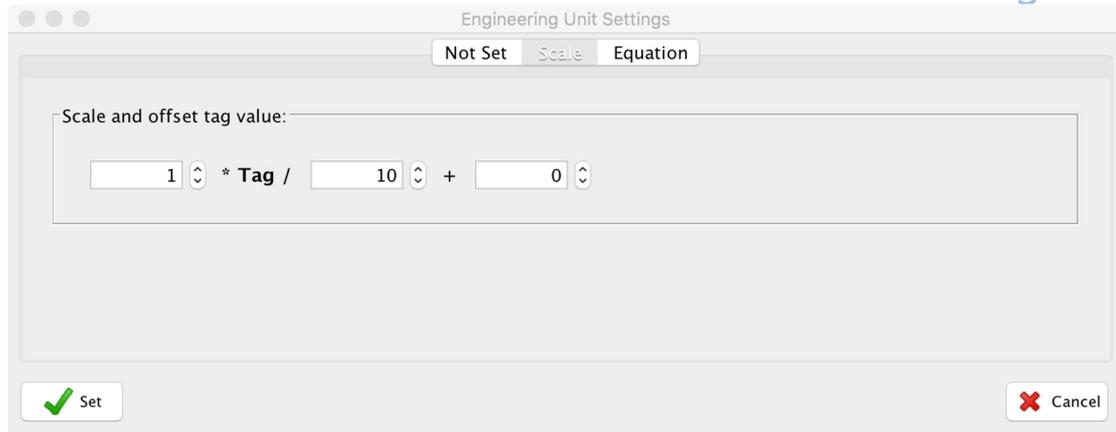


*TIP: To test if the formula is valid, use the "Test" button.*

## Simple example of scaling a tag

A very common situation is a representation of floating values by integers in PLCs (especially on Modbus devices). In our example, we will scale the value read from a Modbus device to achieve a floating-point value.

- Select your tag in the tag database and click on the Eng. Units cell
- Set divider in the dialog to 10



- Now your value will be automatically scaled by 10. So if you will read, for example, the value 235, it will be represented as 23.5

## 12.3 Filtering data

As your project grows, the tag database can get large and hard to manage. To simplify the search for a tag, you can use data filtering. There are two types of filters in the Tag Database:

Data Filter

Filter

	Alias	Tag@Conn	Desc	Unit	Format	Eng. Unit	Usage Count
1	H:0	H:0@script			##	Not Set	0
2	DB111.WORD0	DB111.WORD0@S1			##	Not Set	0
3	DB111.WORD2	DB111.WORD2@S1			##	Not Set	1
4	DB111.WORD4	DB111.WORD4@S1			##	Not Set	1
5	DB111.WORD0	DB111.WORD0@S1			##	Not Set	1
6	DB111.BYTE0	DB111.BYTE0@S1			##	Not Set	1
7	DB111.WORD2	DB111.WORD2@S1			##	Not Set	1
8	DB111.WORD4	DB111.WORD4@S1			##	Not Set	1
9	DB111.WORD6	DB111.WORD6@S1			##	Not Set	1
10	DB111.WORD8	DB111.WORD8@S1			##	Not Set	1
11	DB111.WORD10	DB111.WORD10@S1			##	Not Set	1
12	DB111.WORD12	DB111.WORD12@S1			##	Not Set	1

Import Show References

Connection Filter

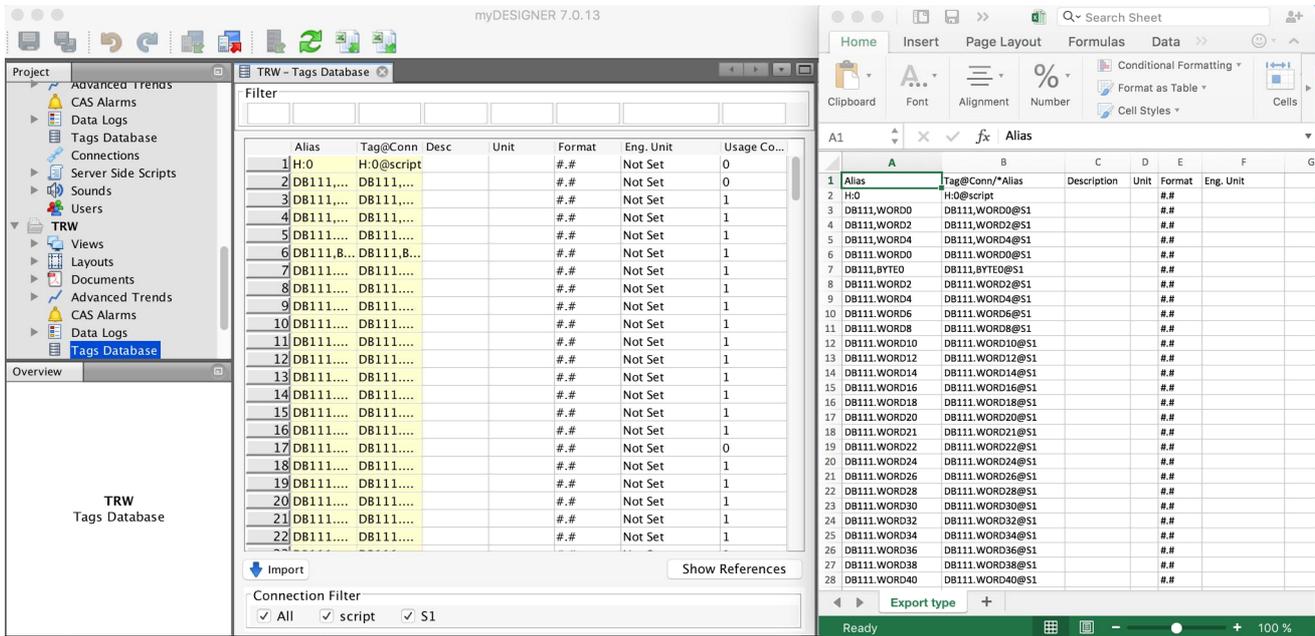
All  script  S1

connection filter

The connection filter lists all connections in your project. You can specify which connections you want to see in the table.

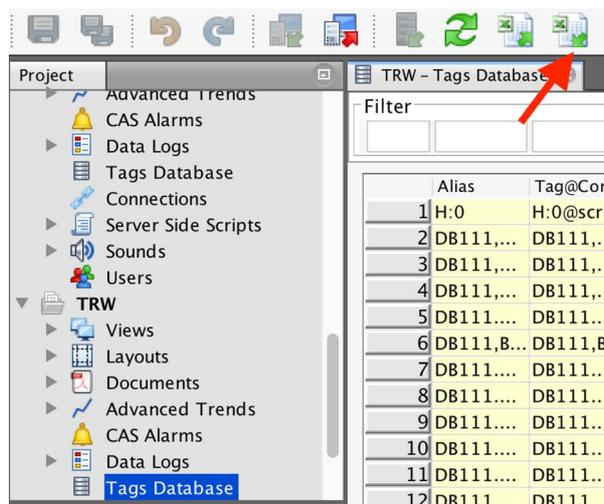


Once the export file is created, you can open it in MS Excel.



### Tag Database Import

The tag database can be imported from an MS Excel file. It can be useful to keep the tag database in an MS Excel file and import it into myDESIGNER or you can use the MS Excel file as a bridge between your PLC programming software and myDESIGNER to transfer the tag database. To import the tag database from MS Excel, click on the Import button in the



main toolbar:

Select a file for import and confirm. The MS Excel file should have all tags in the first sheet in a plain table. The table should have columns with the following names:

Alias	Tag@Conn/*Alias	Description	Unit	Format	Eng. Unit
-------	-----------------	-------------	------	--------	-----------

Once you import the tags, myDESIGNER will go over your existing tag database. If you import a tag with the same alias as one that is already in the tag database, it will be overwritten.

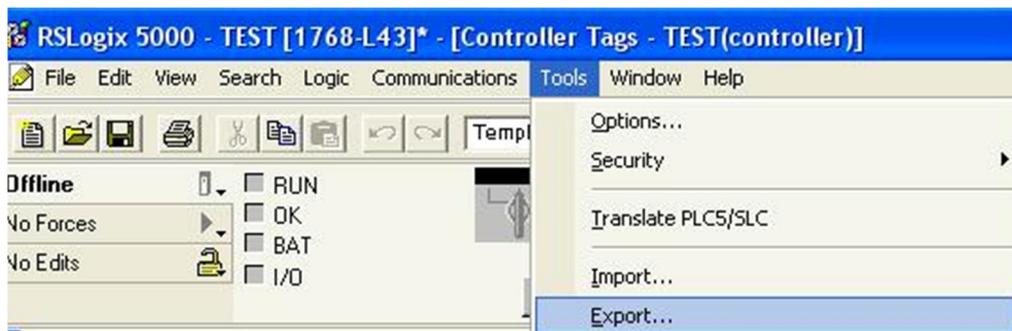
## 12.6 Tag Import

You can also import tags directly from the programming software from Rockwell Automation RSLogix 5000 and Siemens TIA Portal.

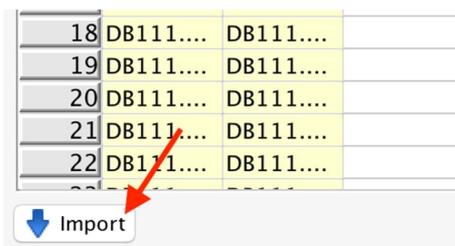
### Importing from RSLogix 5000

To import tags from RSLogix/Studio 5000, do the following:

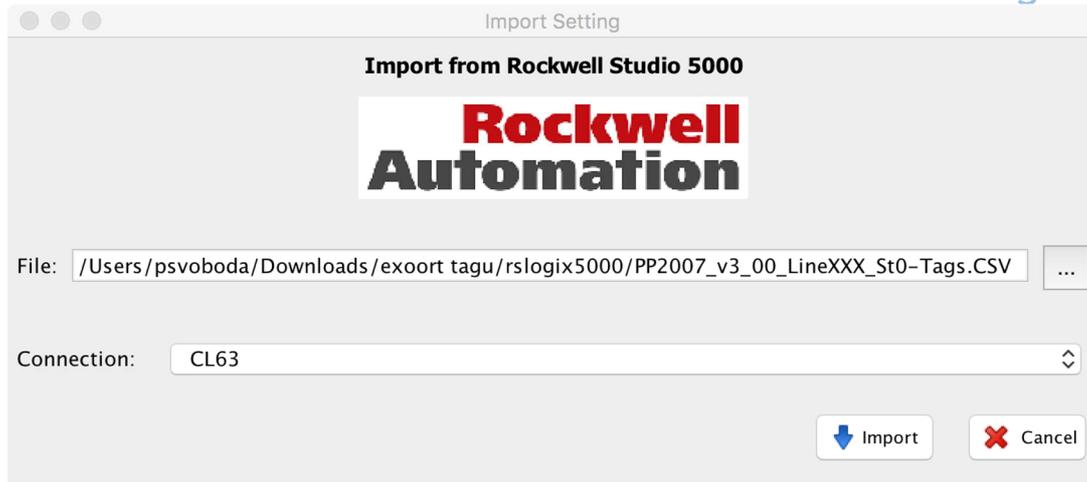
1. Export tags from RSLogix/Studio 5000. To do so, Select Tools and Export:



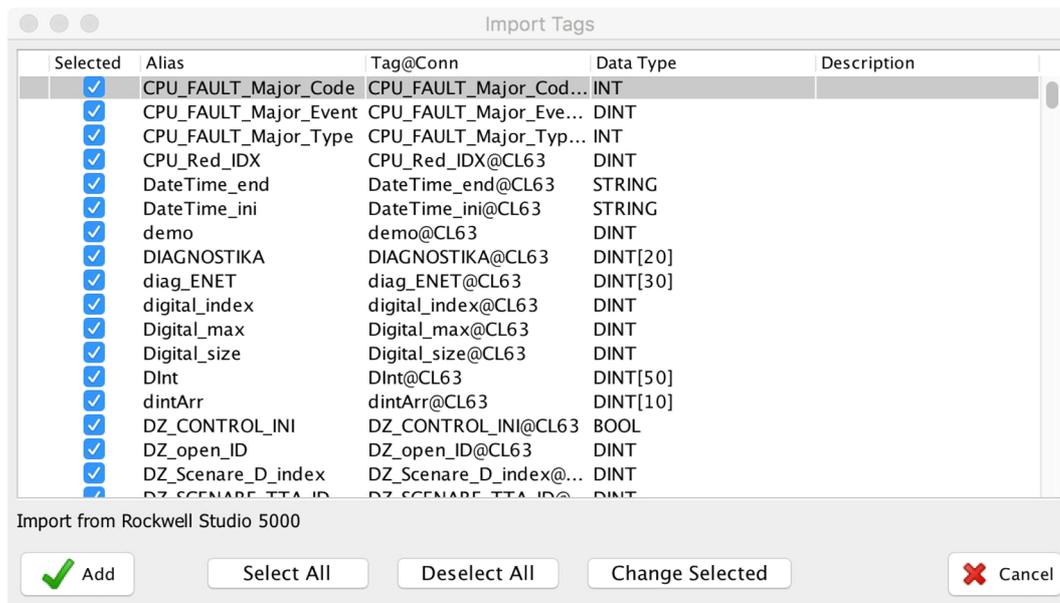
2. Click on the Import button in the Tag Database window. Select Rockwell Automation import.



3. In the following dialog, select the .csv file you exported in step 1. In addition, choose the connection you wish to associate the imported tags with:



4. A new dialog showing all the tags you can import is shown:



*TIP: In the dialog, you can select multiple rows/tags; click on Change Selected to select them.*

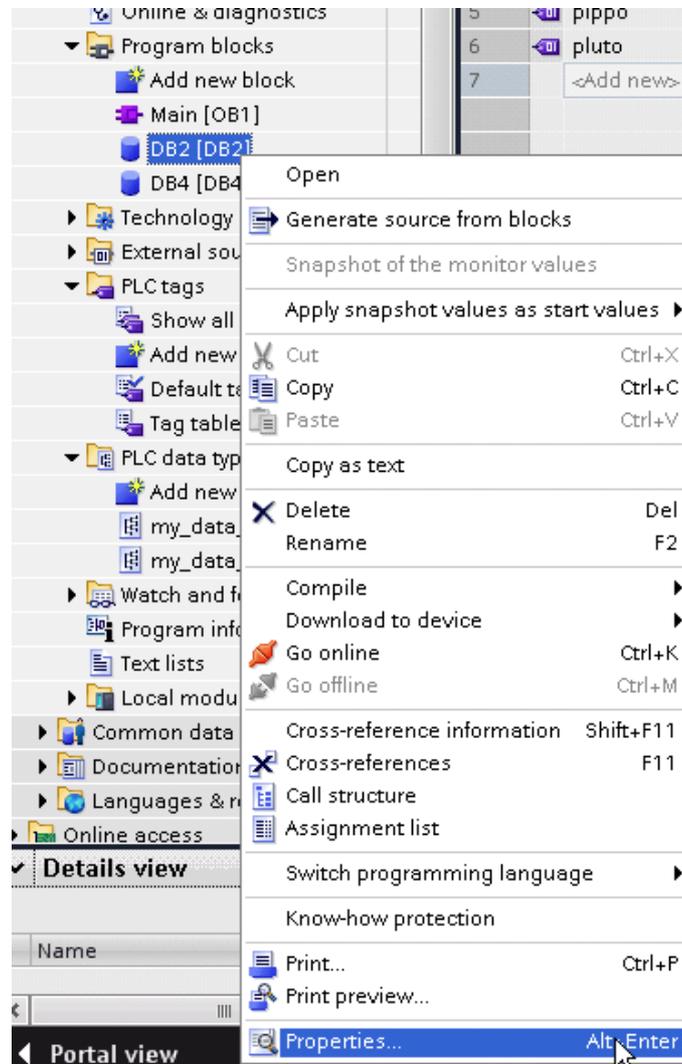
5. Select the tags you wish to import and confirm.

### Importing from Siemens TIA Portal

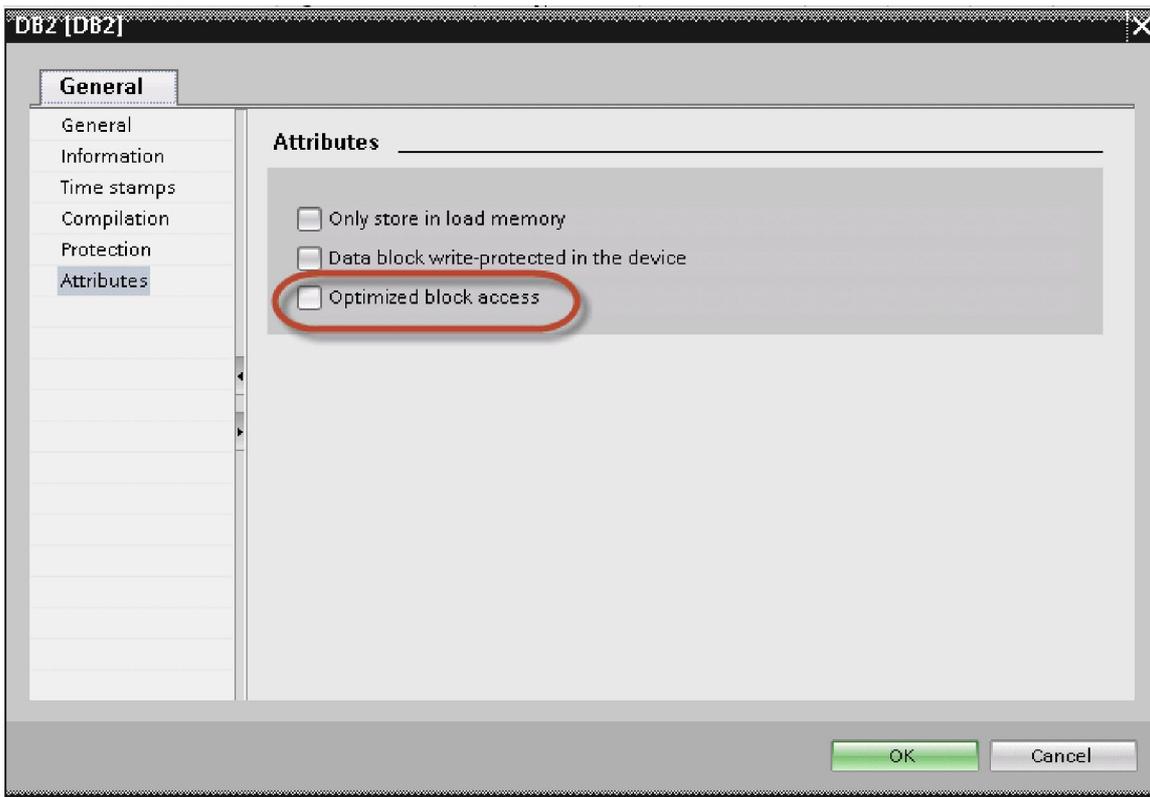
To import tags from TIA portal, you must first export them from TIA Portal. Unfortunately, TIA portal does not have a single export button to get all your tags into one file in one step. You must export PLC tags and DB tags separately. Please follow this procedure to export the data:

## Step One: Exporting From TIA Portal

First of all, to be able to export variables defined as "Program blocks," you need to configure DB as "Not optimized." To check or change DB optimization, you need to enter DB Properties:



Then in General > Attributes uncheck "Optimized block access" as shown below:



If checkbox "optimized block access" is grayed out, the given DB is an "instance DB" linked to an "optimized access FB."

Now, before exporting tags, you need to compile your project to ensure that the TIA Portal calculates tags offsets:



### Exporting DB tags

For every Program block, you need to perform the following steps:  
Double click on the DB name:

Project tree: Tia\_project\_Luca [V11] > PLC\_1 [CPU 1211C DQ/DQ/DC] > Program blocks

Devices: DB2

	Name	Data type	Offset	Start value	Re
1	Static				
2	tag1	Bool	0.0	false	
3	tag2	Char	1.0	''	
4	tag3	Byte	2.0	16#0	
5	Static_1	Array [0..1] of Bool	4.0		
6	Static_2	CONN_ANY	6.0		
7	Static_3	CONN_OUC	8.0		
8	Static_4	CONN_PRG	10.0		
9	Static_5	CREF	12.0		
10	Static_6	Date	20.0	D#1990-01-01	
11	Static_7	DInt	22.0	0	
12	Static_8	DTL	26.0	DTL#1970-01-01-1	
13	Static_9	DWord	38.0	16#0	

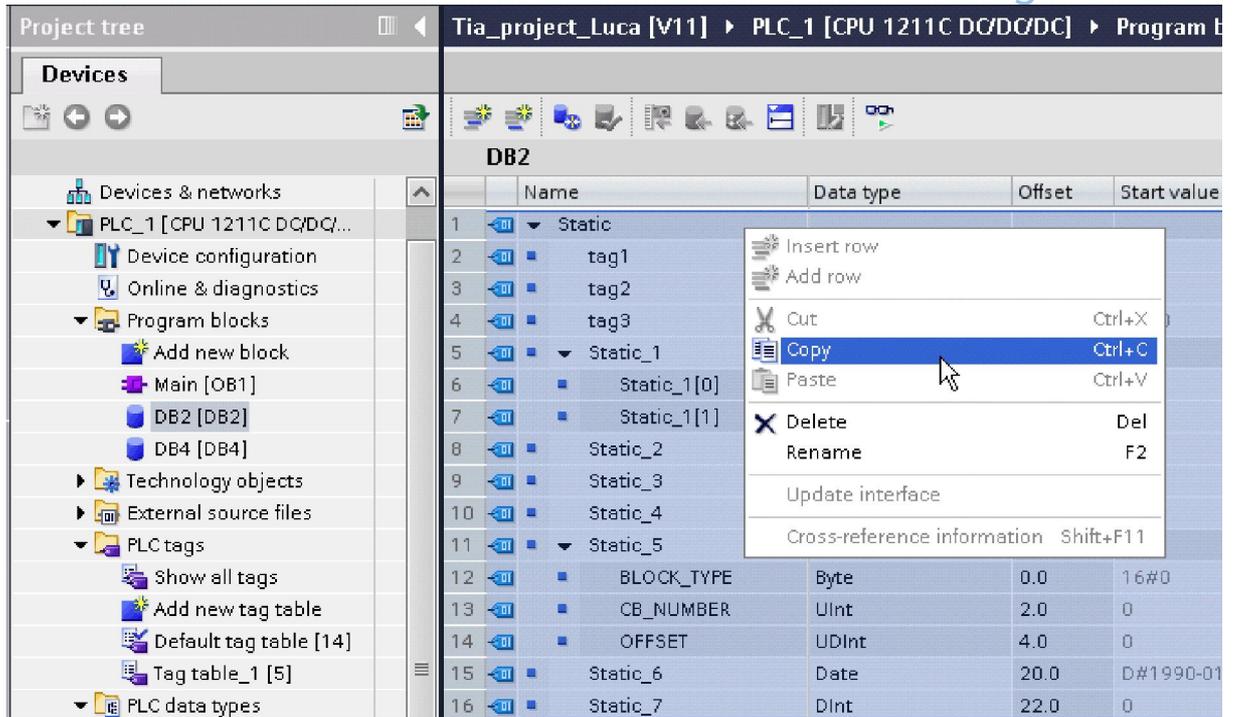
Expand the view of the selected program block:

Project tree: Tia\_project\_Luca [V11] > PLC\_1 [CPU 1211C DQ/DQ/DC] > Program blocks

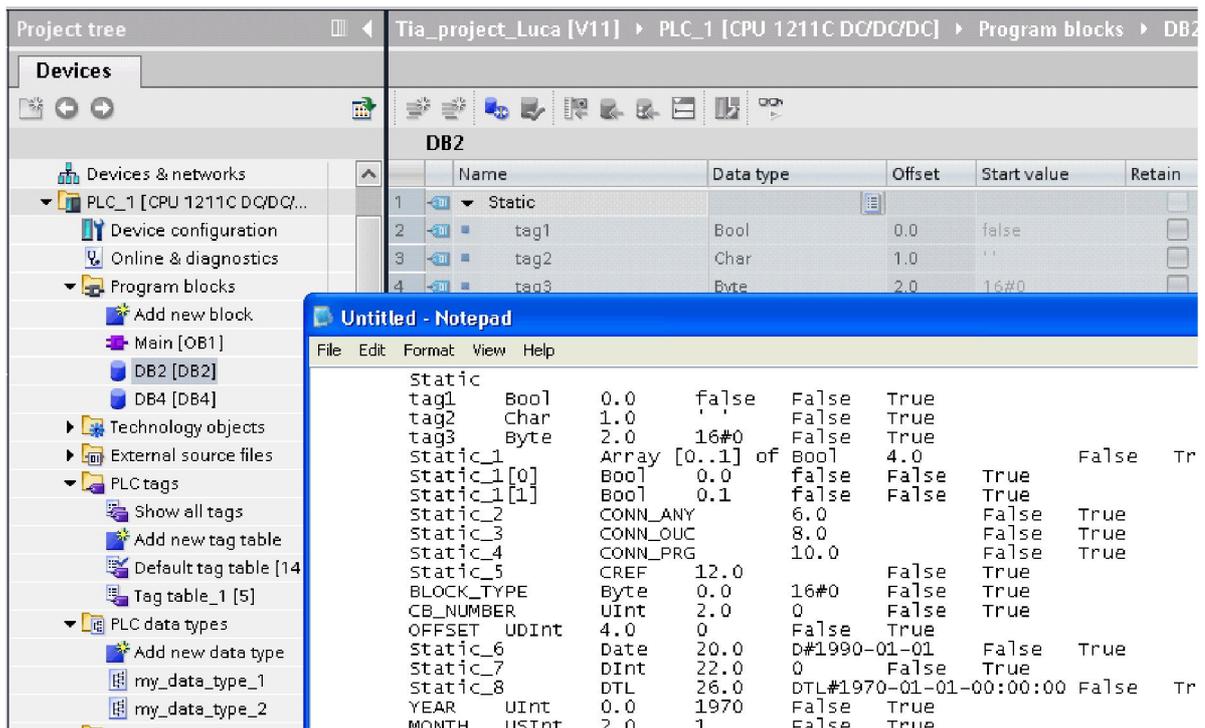
Devices: DB2

	Name	Data type	Offset	Start value	Re
1	Static				
2	tag1	Bool	0.0	false	
3	tag2	Char	1.0	''	
4	tag3	Byte	2.0	16#0	
5	Static_1	Array [0..1] of Bool	4.0		
6	Static_1[0]	Bool	0.0	false	
7	Static_1[1]	Bool	0.1	false	
8	Static_2	CONN_ANY	6.0		
9	Static_3	CONN_OUC	8.0		
10	Static_4	CONN_PRG	10.0		
11	Static_5	CREF	12.0		
12	BLOCK_TYPE	Byte	0.0	16#0	
13	CB_NUMBER	UInt	2.0	0	
14	OFFSET	UDInt	4.0	0	
15	Static_6	Date	20.0	D#1990-01-01	
16	Static_7	DInt	22.0	0	

Then highlight all row in this view (for example, using CTRL + A) and copy to clipboard:



Now open an empty text file (with notepad, for example) and paste the content.



Now save the file as "DBxxx.tia" where xxx is the number of DB.

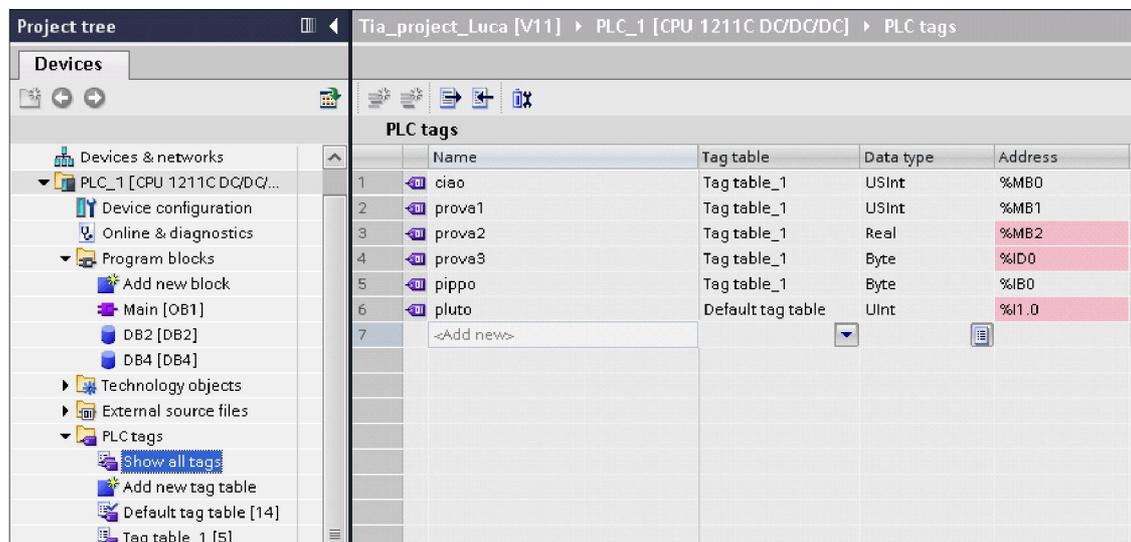
**Important:** Make sure your file is saved with extension .tia otherwise it might not be visible for import.

Repeat the above steps for all defined program blocks. For each DB, a new ".tia" file must be created.

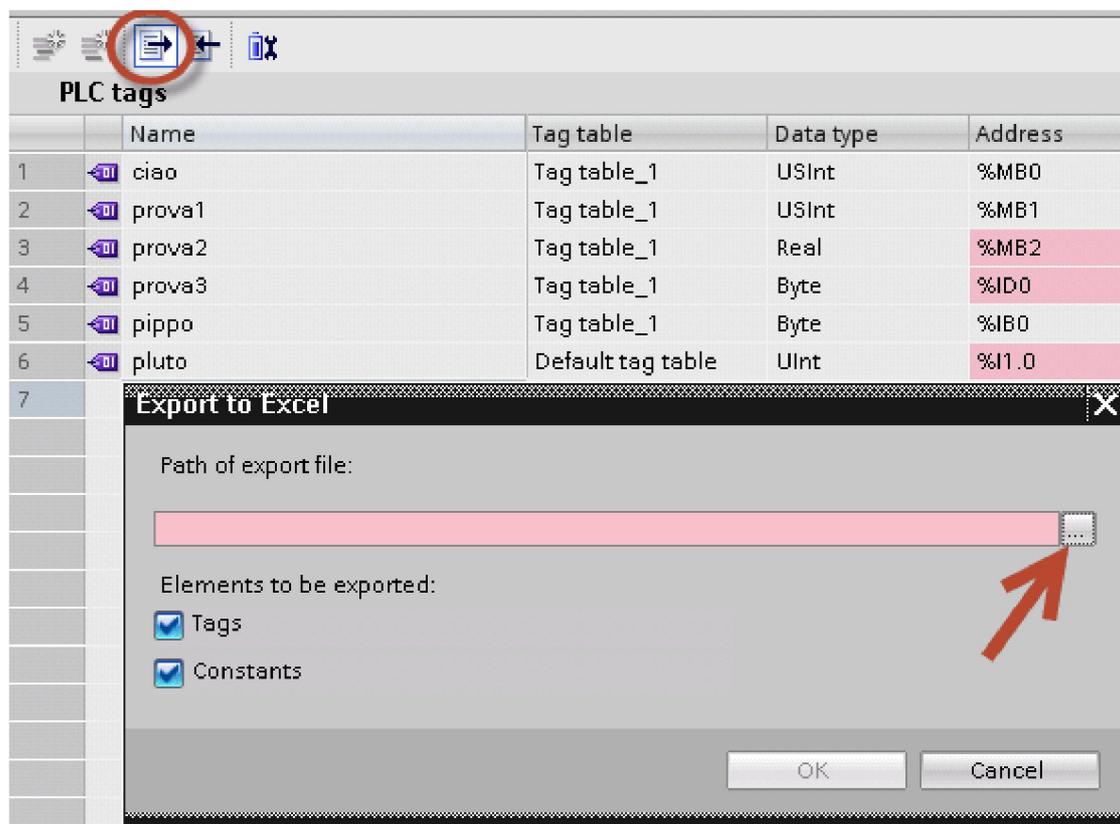
## How to export PLC tags

To export PLC tags, you need to proceed as follows:

1. Double click on "Show all tags" to open tag table:

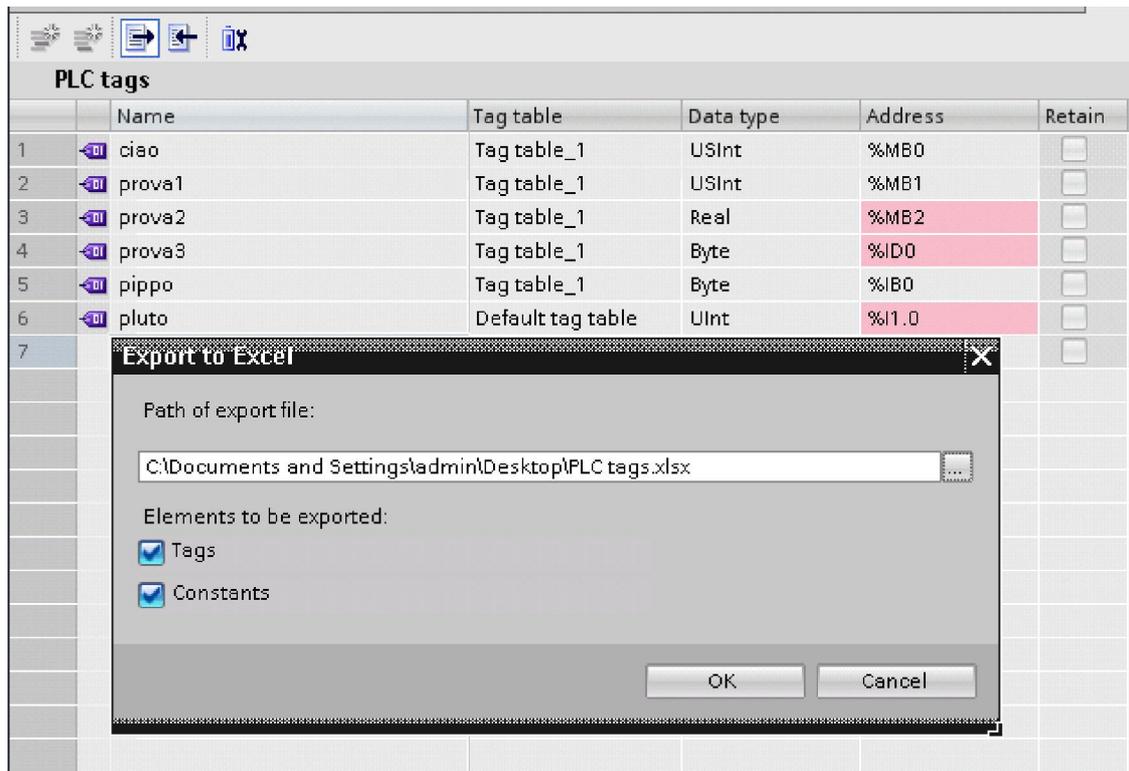


2. Then click on the "Export" icon and browse for path file:



3. Define the file name for export and confirm with "Save"

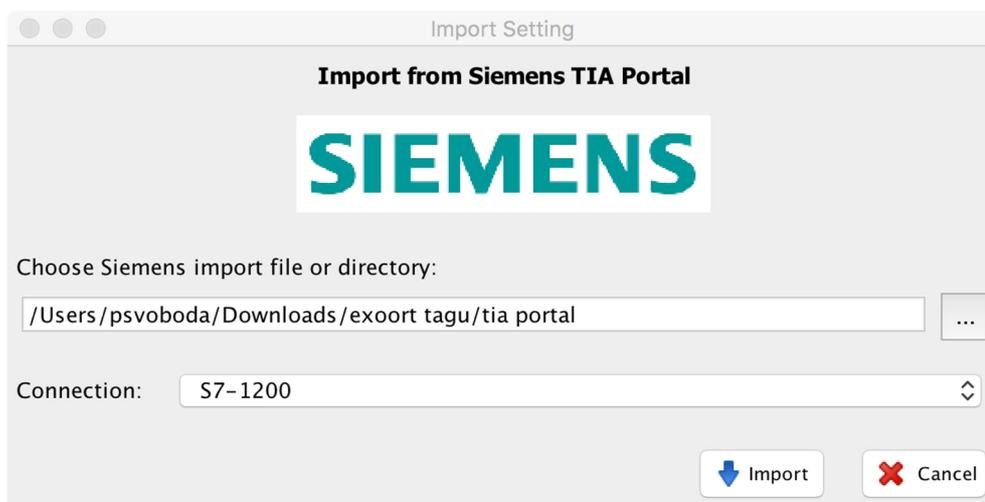
4. Finally, confirm export with "OK":



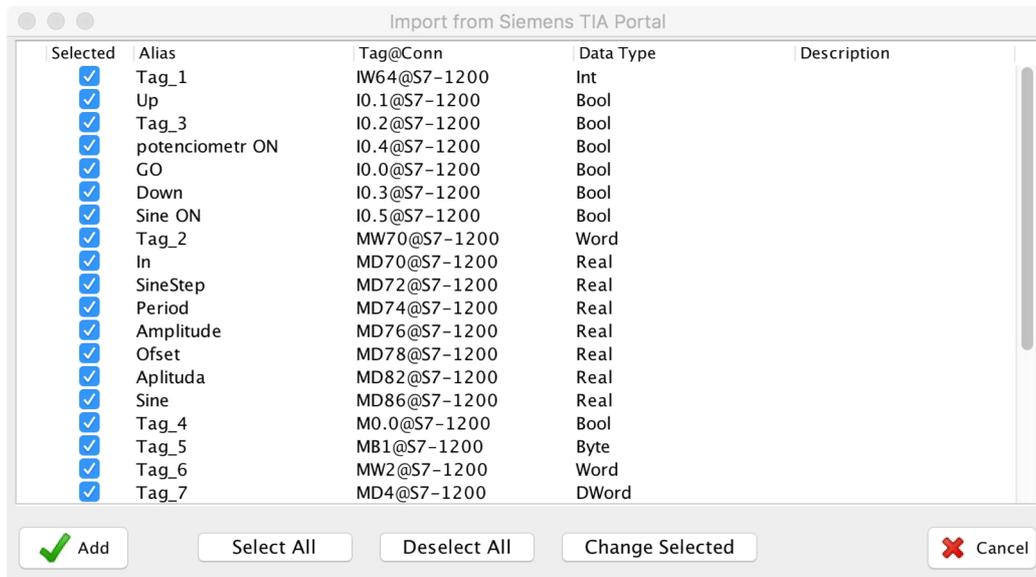
## Step Two: Importing into myDESIGNER

Once you have created your export files in TIA Portal, you can import them into the myDESIGNER Tag database. To do so, please follow these steps:

1. Click on the Import button and select Siemens.
2. Now select the directory where your exported files are located.



- After selection of the directory, click on Import. myDESIGNER will scan all files in the provided directory. It will process all .xlsx files for Program tags imports and all .tia files for DB tags imports. You will be presented with a table of tags to import:

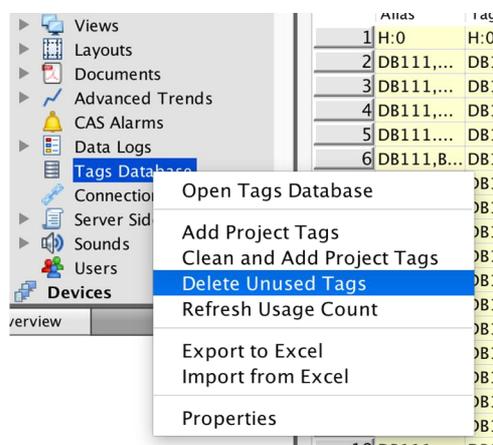


*TIP: In the dialog, you can select multiple rows/tags and click on Change Selected to select them.*

- Select the tags you wish to import and confirm.

## 12.7 Deleting unused tags

As your project grows, the tag database can also get larger. Some of the tags listed in the tag database may no longer be needed. You can delete unused tags simply by invoking the command Delete Unused Tags. To do so, right click on the Tag Database in the project. A popup menu is shown; now select Delete Unused Tags.



All tags with reference count = 0 (e.g. not used) will be removed.

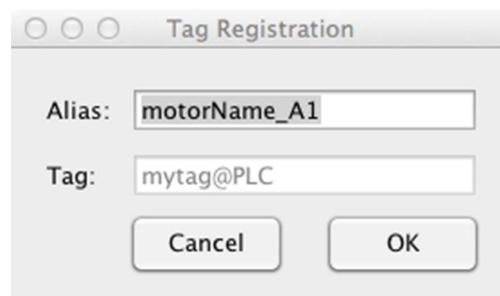
## 12.8 Restoring tag database

In rare cases, the tag database can become corrupted. If you experience strange behavior when dealing with tags, you can regenerate the tag database. To do so, you have two options:

- Add Project Tags: will scan your project and add all tags that are missing from the tag database.
- Clean and Add Project Tags: will delete all tags from the tag database, then it will scan your project and add all tags that are missing to the tag database.

## 12.9 Specifying a new tag during development

If you write a new tag anywhere in *myDESIGNER*, a dialog box will show up to add this tag to the *Tag Database*.

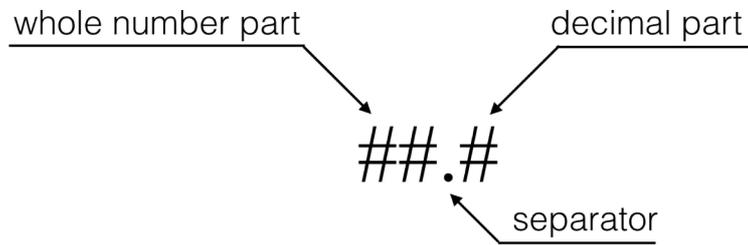


*Note: Instead of writing 'mytag@PLC' you can use the alias 'motorName\_A1' anytime.*

Click on **OK** to confirm adding the tag to the database. Once the tag is created, you can use its *Alias* as a reference.

# 13 Formatting Numerical values

Numerical values can be formatted according to a specified format. To specify a format, please use # as a formatter.



The following table will show you how values will be formatted according to a specified formatter. Value 23.89323232 will be formatted as follows:

Formatter	Result
#	23
###	023
#. #	23.9
##.##	23.89
###.##	023.89

# 14 Linking Views with PLC

## 14.1 Introduction

There are two options to link your visualization with the PLC:

### Animations

With animations, you link the visual appearance of graphic objects with real values read from the PLC. The visual change is reflected immediately, e.g. you can show PLC tag/variable values in a text element, change the fill and stroke color of an object, etc.

*Note: Multiple animations can be applied to the same object.*

**Watch video describing this functionality:** <https://www.youtube.com/watch?v=cPPysaTWKc>

### Effects

With effects, you add dynamic effects to graphic objects. An effect is the visual appearance of an object in a specified time sequence and can be triggered either by a tag/variable from the PLC or by a user's action, i.e., *finger tap* or *mouse click*.

**Watch video describing this functionality:** <https://www.youtube.com/watch?v=GTOFCwrRTPM>

Imagine you want to display a blinking motor on your visualization if the tag/variable read from the PLC is equal to 1.

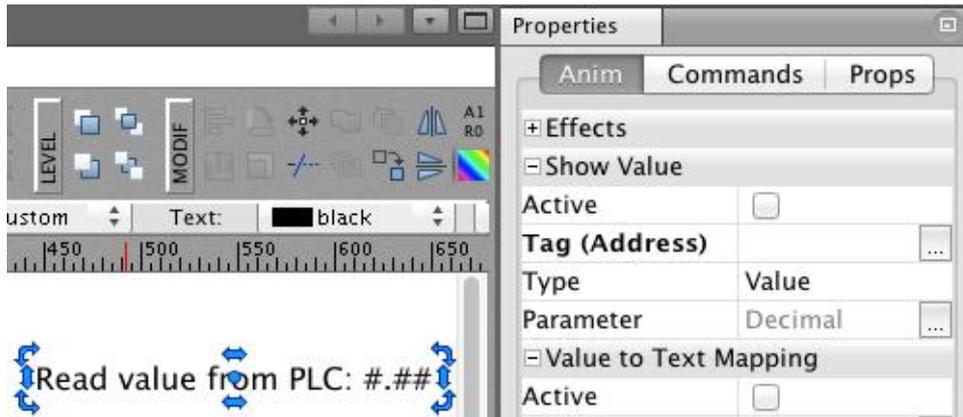
- 1) Set the *Blinking* effect on the motor by specifying the blinking speed and infinite repeat. This way the motor will be blinking as long as the tag read from the PLC is equal to 1.
- 2) If you use the *Color* animation instead, the motor will change its color if the value read from the PLC equals 1 and return back to normal when the value is 0.



## 14.2 Animations

**Watch video describing this functionality:** <https://www.youtube.com/watch?v=cPPysaTWKc>

If you select an object and navigate to the **Anim** section in the *Properties* window, you will find all available animations for that object.



### 14.3 Show Value Animation



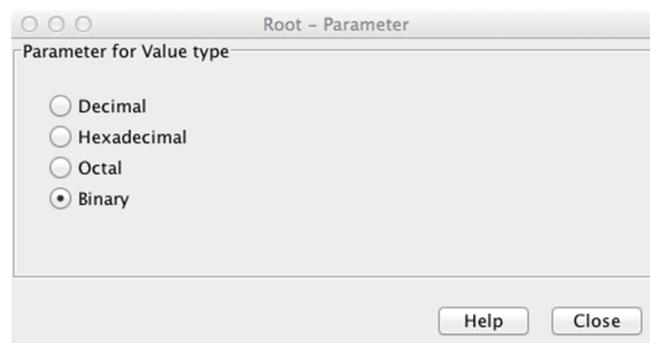
This animation can display tag values in a text element.

In the *Type* section, you can select how the data will be displayed. There are three options:

- *Value*
- *String*
- *Date*

#### Value

**Value** shows the read PLC values in the *Decimal*, *Hexadecimal*, *Octal*, and *Binary* numerical systems.



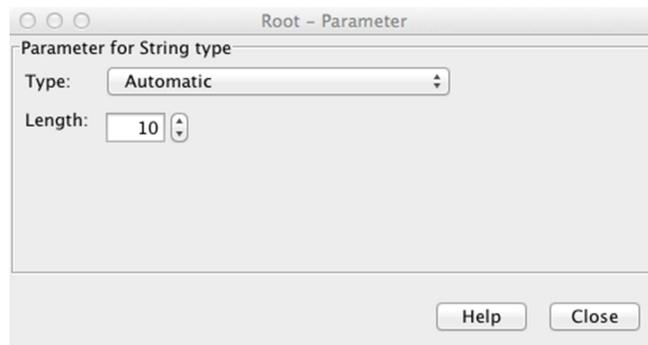
By default, Value type shows values in *Decimal* format. If you are showing a floating-point number, you can specify how the value will be presented by using formatting. For more info about formatting, please see [Formatting Numerical values](#).

### String type

String type reads string data type from the PLC and shows the resulting value as a string. You can specify special conditions such as maximum string length or string type.

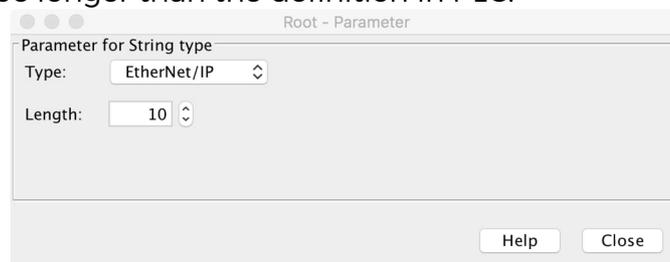
- **Automatic**

The PLC type used in the tag definition selects the string type automatically. You need to specify only the maximum string length.

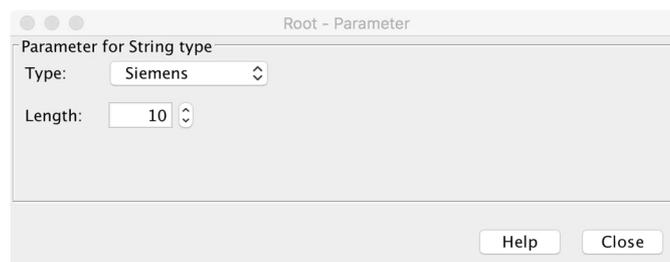


- **EtherNet/IP**

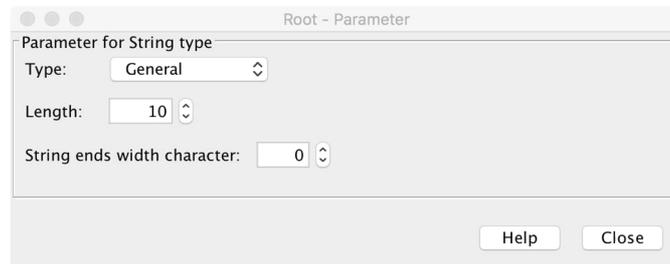
Rockwell family string type. The string is automatically encoded according to the RA standard. Please specify the maximum length of the string. The provided maximum length must not be longer than the definition in PLC.



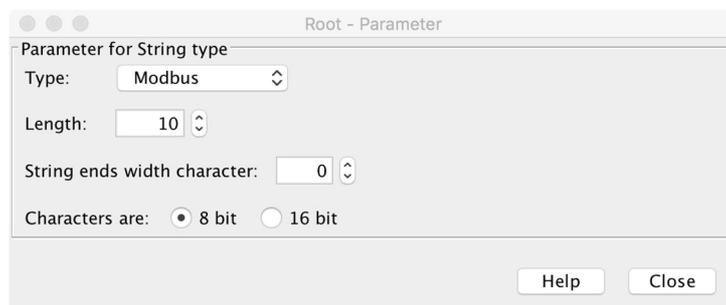
- **Siemens** - Siemens family string. Please specify the maximum length of the string. The provided maximum length must not be longer than the definition in PLC.



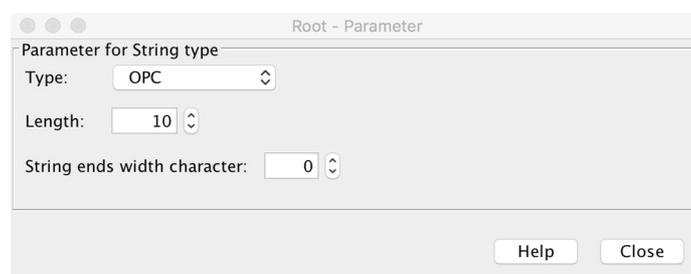
- General** - User defined. You have complete control of converting a PLC data array into a string. First of all, you should specify the maximum length of the string. The provided maximum length must not be longer than the definition in PLC. Then you can specify which character terminates the string or specify at which position the length of the string is encoded.



- Modbus** - Modbus protocol definition does not define the string type. Most of the PLC makers, however, allow the use of the string in Modbus protocol by providing the values in an array. As there is no given standard for use in Modbus protocol, the user has complete control over how the string is defined. It allows you to convert a PLC data array into a string. First of all, you should specify the maximum length of the string. The provided maximum length must not be longer than the definition in PLC. Then you can specify which character terminates the string or specify at which position the length of the string is encoded. If your PLC encodes two characters into one register, please choose 8-bit characters; otherwise, choose 16-bit characters.

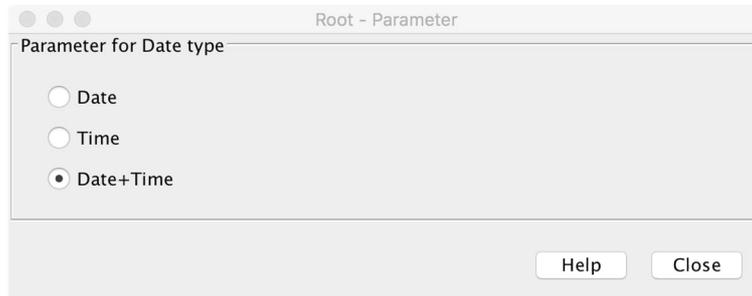


- OPC** - OPC UA family string. The OPC has defined a standard for reading strings. Please specify the maximum length of the string. The provided maximum length must not be longer than the definition in OPC UA server.



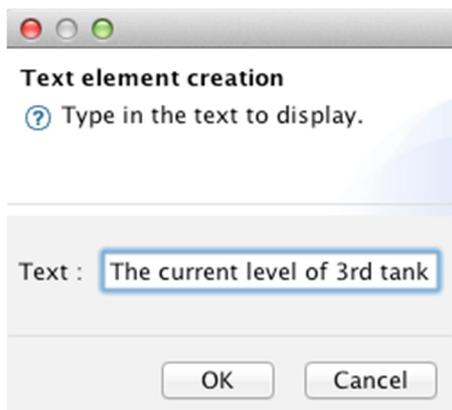
## Date type

Date type enables you to read date value from the PLC and convert it to the date. mySCADA assumes the date is in unix time – the number of seconds since 1 January 1970. You can choose to show only date, only time, or both date and time.



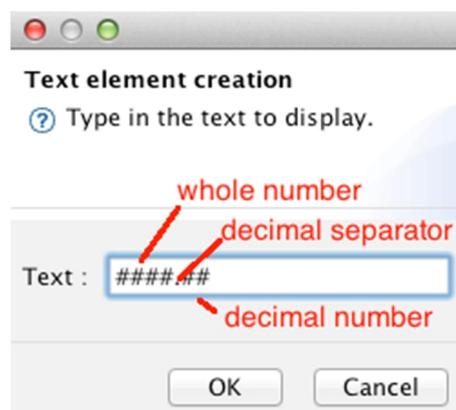
### Example of Showing Values from PLC:

- 1) Let's assume there are three tags in *ControlLogix* PLC, representing the water levels – *level1*, *level2*, and *level3*.
- 2) Create the text elements for these three water levels using the *Create Text Element* tool **A**.



The current level of 1st tank  
 The current level of 2nd tank  
 The current level of 3rd tank

- 3) Continue adding the text elements responsible for reading values from the PLC. This process is exactly the same as creating the text elements. The text written in the text element specifies how the value read from the PLC will be formatted on the screen. The format specification is described in the following figure:

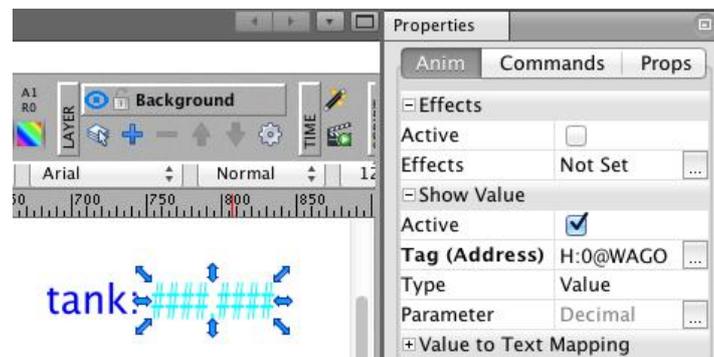


*Note: Any value received from the PLC that does not follow the format specification will be automatically transformed to a set format. For example, if the PLC returns a value of 3.47 and the visualization expects only one decimal place with the format (##.#), then the number displayed will be rounded to one decimal place, showing the value of 3.5.*

#### 4) Connect the text elements to specific PLC tags.

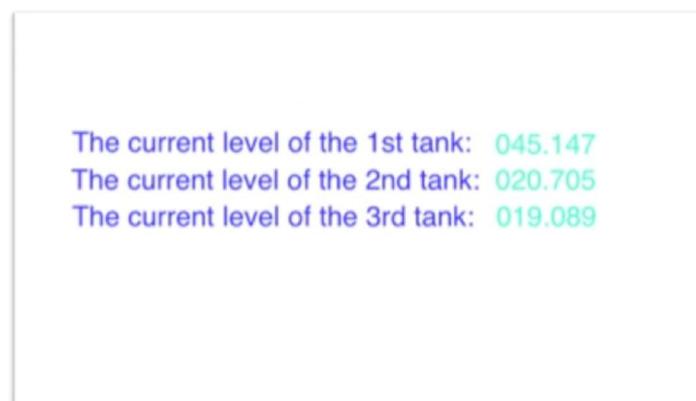
Click on the text element that should be animated, according to the data received from the PLC. You will see the animations properties in the *Properties* window. Select the **Anim** tab in the *Show Value* animation section; enter the specific tag address from which the text element will read.

*Note: If the entered tag address is invalid, the text will be marked in red.*



You can also combine a static text element with PLC values.

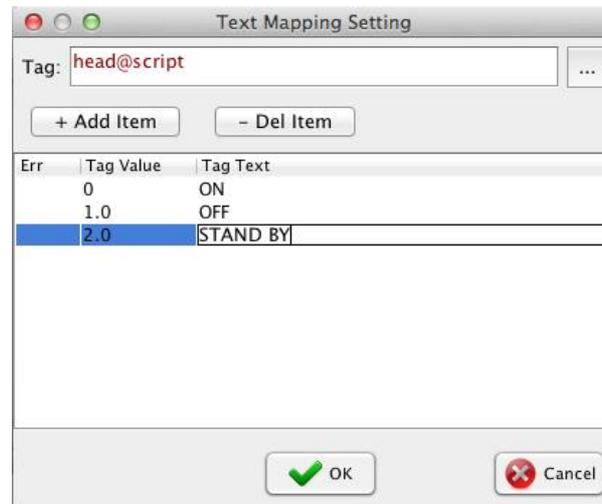
#### 5) Upload the view to your supported device - the screen should look like the following:



In the example above, we have used the tags from *ControlLogix*. You can generally use the tags for all supported PLCs, as long as you use the proper syntax (for a brief description of proper tag syntax, see the chapter *Basic Tag Syntax*).

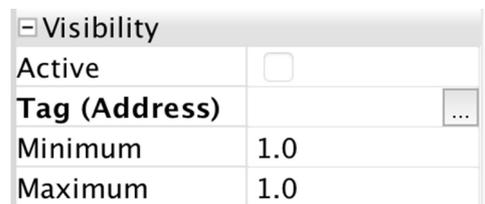
## 14.4 Value to Text Mapping Animation

This animation converts numerical values from the PLC into text values. The tag value is automatically converted to the string based on the definition table:



## 14.5 Visibility Animation

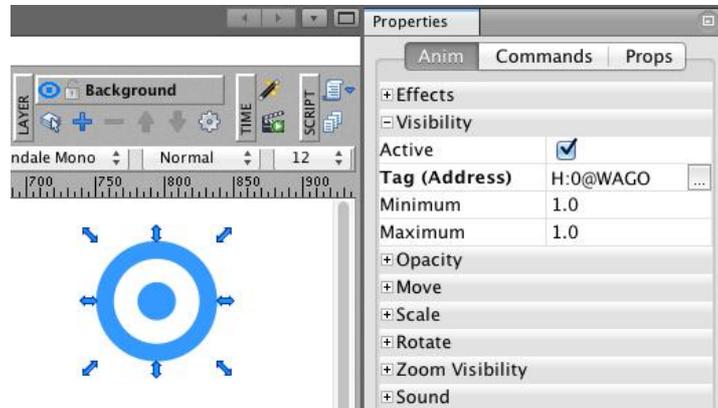
This animation allows you to control the visibility of an object. There are three inputs required for this animation: **Tag**, **Min**, and **Max**. When the value is within the min and max values range, the object will be visible; otherwise, it will be hidden.



To animate visibility based on the discrete (*Boolean*) value, set both **Min** and **Max** to 1; this way the object will become visible when the tag value equals 1 (*TRUE*).

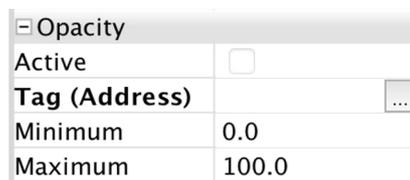
*Example:*

- 1) Click on the object that you want to animate (this will prompt the properties in the **Anim** tab of the *Properties window*) and navigate to the *Visibility* section.
- 2) Set the **Tag (Address)** and the **Minimum** and **Maximum**; if the tag value is within this range, the object will be visible, otherwise, it will be hidden.

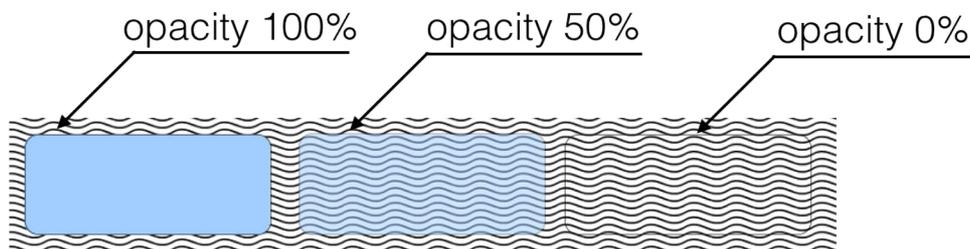


## 14.6 Opacity Animation

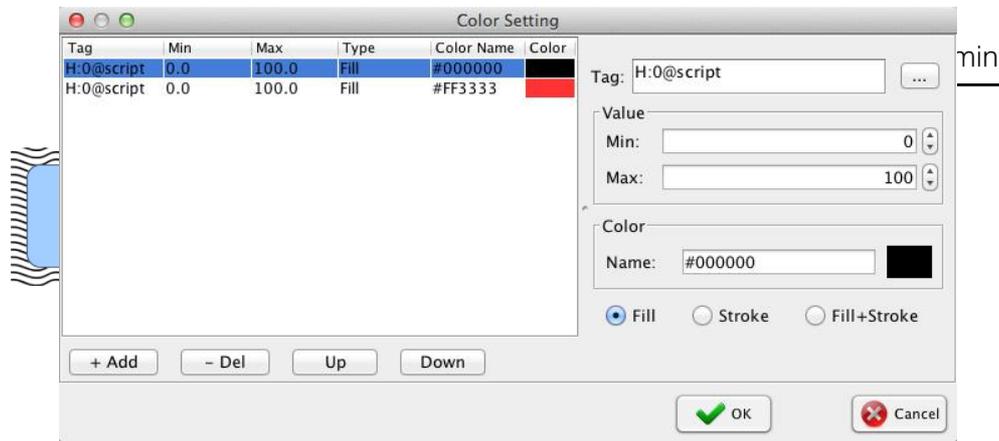
Opacity defines how transparent an object is. Set the tag that will control the opacity. Then set the minimum and maximum values. When the tag is at the defined minimum, the object will be fully transparent; when the tag is at the defined maximum, the object will be completely opaque.



The following picture shows different levels of opacity:



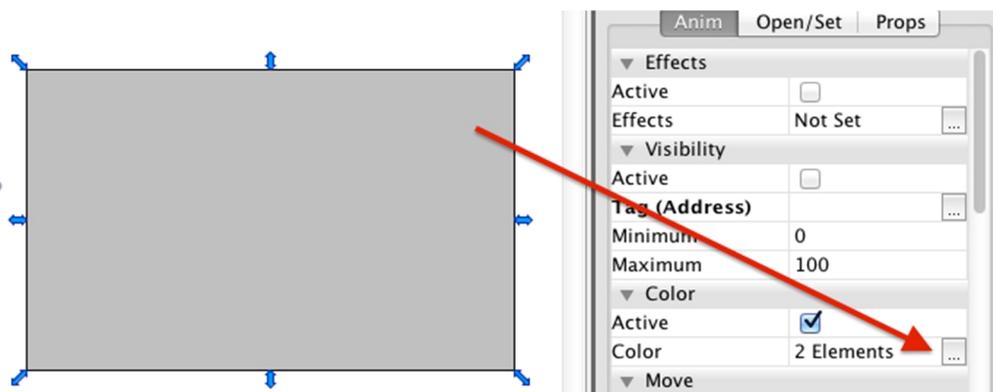
And these levels again, converted to the tag values:



## 14.7 Color Animation

This animation displays a certain color when the associated tag is within an acceptable value range. You can enter multiple conditions to achieve multiple color changes with one animation. The conditions are evaluated from top to bottom; therefore, the bottom condition has a higher priority.

To animate colors based on a discrete (*Boolean*) value, set **Min** and **Max** to 1. This way, an object will change its color when the linked tag value is equal to 1 (*TRUE*).



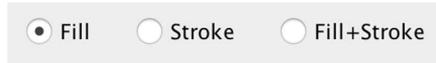
### Example:

- 1) Select the object you want to animate and click on the **Color** tab in the **Anim** section in the *Properties window* and then click on the ... button.
- 2) You will see a new dialog window; click on **Add** and fill in the **Tag**, **Min**, and **Max** values. You can define multiple conditions. If the tag value is 0, the object will be red; if the value is 1, it will be black. Click on **OK** to confirm.

*Examples of color animations:* Buttons, Traffic lights, Fans, Valves, Motors

## Specifying the Fill and Stroke.

For each condition (item in the table), you can specify if the condition applies to fill, stroke, or both.

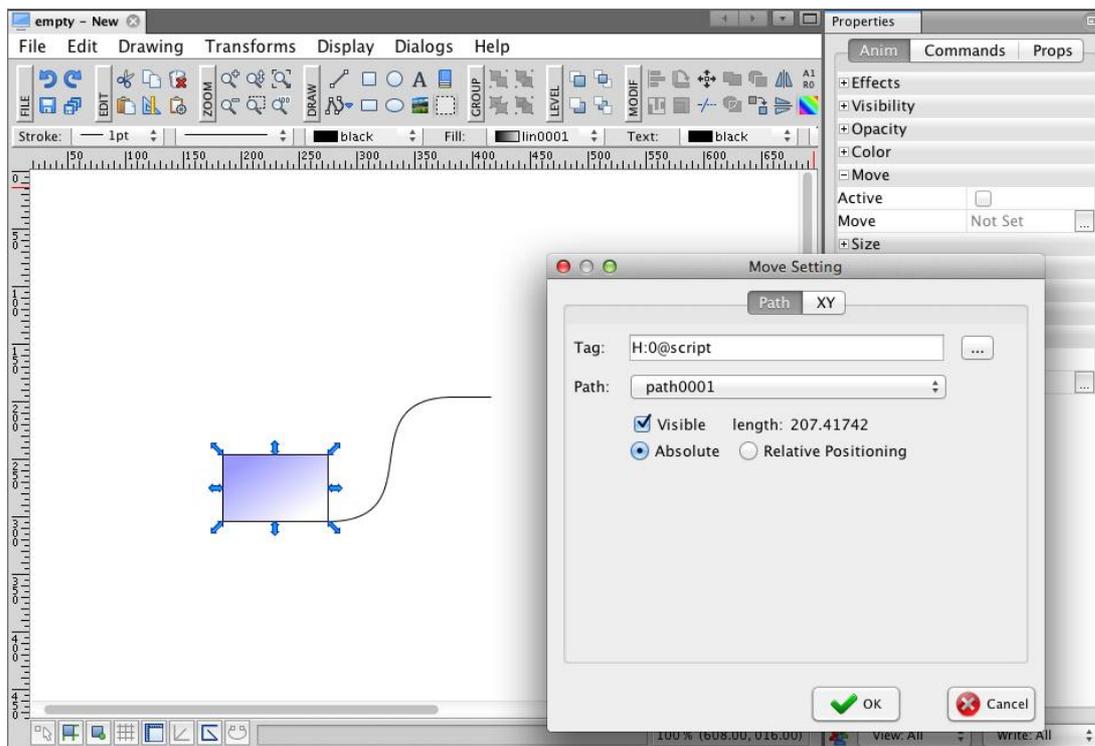


## 14.8 Moving Animation

This animation is a programmed motion of an object moving along a defined path or XY axis, based on the tag value.

### Moving along the path

- 1) Create an object and a path to move this object along. To add the *Move* animation, select the object and click on the **Active** button in the **Move** animation tab of the *Properties* window, as indicated in the picture below:



- 2) Fill the tag address in the dialog window and select one of the available paths.

The other available features are:

*Visible* – move path remains visible if this option is enabled

*Length* – represents the length of a selected path; it is displayed automatically

*Absolute* – this option tells the *mySCADA* engine to map the whole path length for the *Move* animation up to the tag value of 1:1

*Relative Positioning* – this option allows using only a certain tag value threshold. The **Min** value represents the starting animation point (the object

## Linking Views with PLC

moves only when the tag value reaches this point); the **Max** value represents the ending animation point (the object moves to the end of the path when the tag value reaches this point).

*Reverse* - reverts the movement from *start-end* to *end-start*

### Moving along X-Y axis

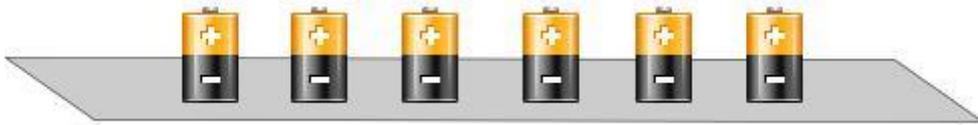
In this type of animation, you can set up movement along the X and Y axes. Check the appropriate axis that you want to move your object along and fill in the required parameters.



Start by filling the **Tag**, **Min**, and **Max** values. Then fill in the length of the movement in pixels. If you would like to reverse the movement, check the **Reverse** check box.



*Note: A typical use of the Move animation might be moving goods on a conveyor belt, etc.*



## 14.9 Size Animation

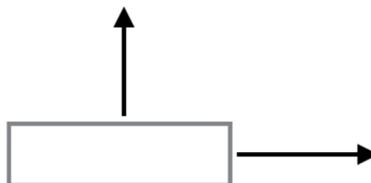
This animation type is used to change the size of objects. It is very useful for creation of an animated progress bar or for tank level animations.

Size	
Active	<input type="checkbox"/>
Tag (Address)	<input type="text" value="..."/>
Minimum	0.0
Maximum	100.0
Orientation	Horizontal

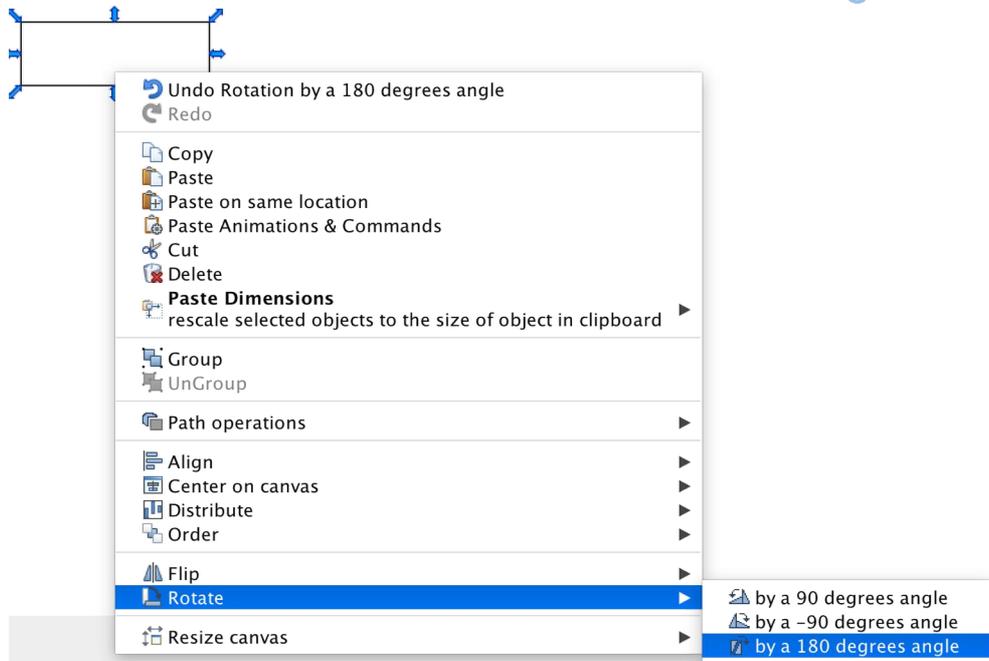
- 1) Select the object you want to animate; navigate to the *Properties* window, **Anim** tab, and section *Scale*.
- 2) Set the tag, Min, and Max limits.
- 3) As a last option, select one of the orientation options:
  - Horizontal
  - Vertical
  - Both

### Reversing the orientation

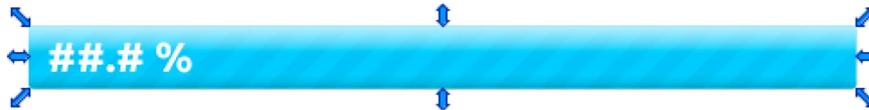
By default, horizontal orientation will change the location of the object to the right, e.g. the left side of the object will stay fixed. Similarly, vertical orientation will change the location of the object upwards, leaving the bottom of the object fixed.



If you want to change the orientation of the movement, simply rotate the object by 180 degrees. You can do so by right clicking on the object and selecting rotate -> by a 180-degree angle

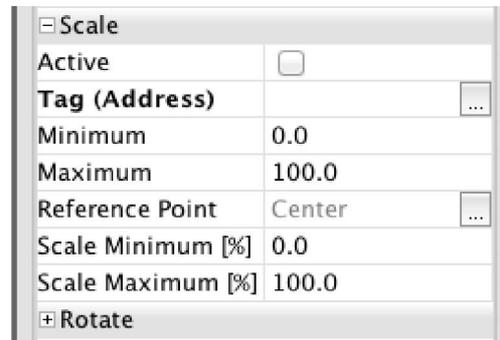


Size Animation Examples:

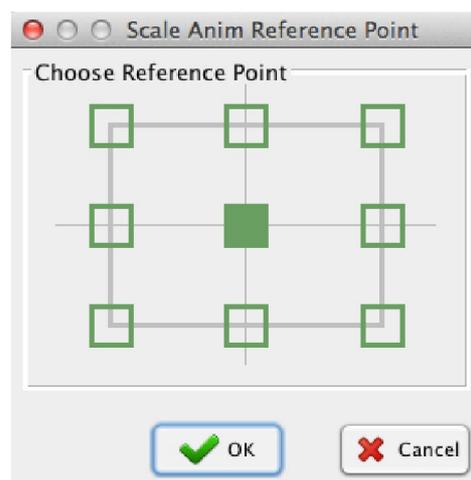


## 14.10 Scale Animation

This animation type is used to scale an object, corresponding to the entered value.



- 1) Select the object you want to animate; navigate to the *Properties* window, **Anim** tab, and section *Scale*.
- 2) Set the tag, Min, and Max limits; percentage of scaling; and the reference point.



If the tag value is at its minimum value, the object will be scaled to the defined scaled minimum. When the tag value is at its maximum value, the object will be scaled to the defined scaled maximum.

### Difference Between Size and Scale Animations

Size animation changes the real size of the object, while scale animation applies a scale transform to the given object. If you have a rectangle, for example, it is defined by its coordinates, width, and height. When we apply size animation, we are changing the width and height parameters of the rectangle. When we apply scale animation, we are setting the scale transform of the whole object.

## 14.11 Rotate Animation

With this type of animation, you can rotate an object around the set axis, according to a percentage of maximum and minimum values ranging from 0 to 360 degrees (defined by the *Range* parameter).

## Linking Views with PLC

Enter the tag name and the minimum tag value, which correlates to the 0<sup>th</sup> degree, and then enter the maximum tag value, which correlates to the 360<sup>th</sup> degree.

By default, the axis of rotation is set in the geometric center of an object or a group. You can change the rotation axis by double-clicking on the object and setting its axis, which is marked as a blue dot. Move the selected point to the desired position, as shown in the picture below:



Go to the *Rotate* section of the **Anim** tab in the *Properties window* and fill in the **Tag (Address)**, **Minimum**, and **Maximum** values.

Rotate	
Active	<input checked="" type="checkbox"/>
Tag (Address)	M:1@script
Minimum	0.0
Maximum	100.0
Reverse	<input checked="" type="checkbox"/>
Range [deg]	360.0
Set Center	Set
Center Offset X	0.0
Center Offset Y	0.0
Center X	358.0
Center Y	134.0

**Set Center** property can be set to:

- Clear Center – clears the custom rotation center and sets to the default (geometrical) center
- Cancel – closes the menu without any changes

**Center Offset X (Y)** shows the difference between the custom and geometric rotation center.

**Center X (Y)** shows the absolute coordinates of the selected rotation center.

**Range property** defines the total revolution angle, which is 360 degrees by default.

## 14.12 Circular Sector Animation

This animation is applicable to circular objects. This animation creates a circular intersection on circles and ellipses.

Circular Sector	
Active	<input type="checkbox"/>
Tag (Address)	<input type="text" value="..."/>
Minimum	0.0
Maximum	100.0
Angle From	0.0
Angle To	90.0
Reverse	<input type="checkbox"/>
Is Arc	<input type="checkbox"/>

To use this animation:

1. Select a circle or ellipse and navigate to the Anim -> Circular Sector
2. Fill in the **Tag, Minimum, and Maximum** values
3. Specify **Angle From** and **Angle To** values

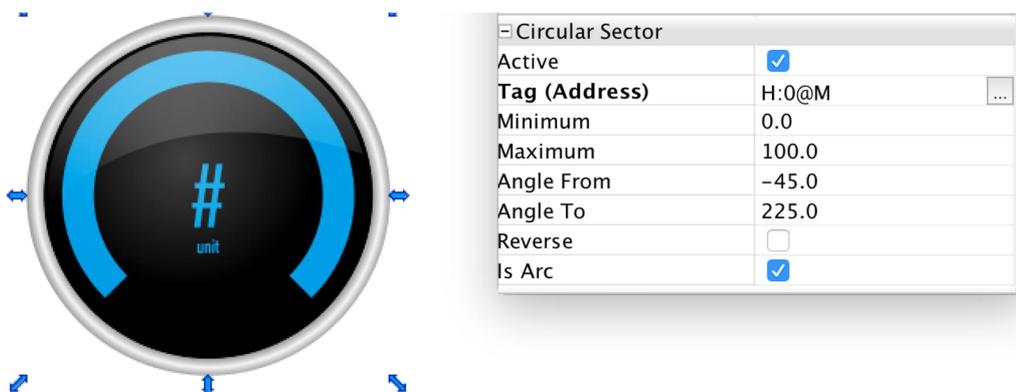
When the tag value is equal to the specified minimum, the object will be invisible, as the angle of the circular section will be equal to 0. As the tag value increases, the circular section will grow from the specified Angle From up to the Angle To. When the tag value is equal to the specified maximum, the circular sector will be between the Angle From and Angle To values.

### Other parameters

**Reverse:** reverse the orientation of the section.

**Is Arc:** if not selected, makes a circular section from the whole object. If selected, leaves only the Arc from the object.

### Circular Sector Examples



Circular Gauge in myDESIGNER



Circular Gauge in live view

## 14.13 Zoom Visibility Animation

Watch video describing this functionality: <https://www.youtube.com/watch?v=9BJjs-p9nfl>

With zoom visibility animation, you can show and hide objects or even layers based on the zoom level in runtime. This animation can be used in many scenarios. Imagine the following situation: you would like to give your user an overview of your complete technology, but it needs to fit on one screen. As the user zooms in, he will see more details. You can achieve this by using zoom visibility animation.

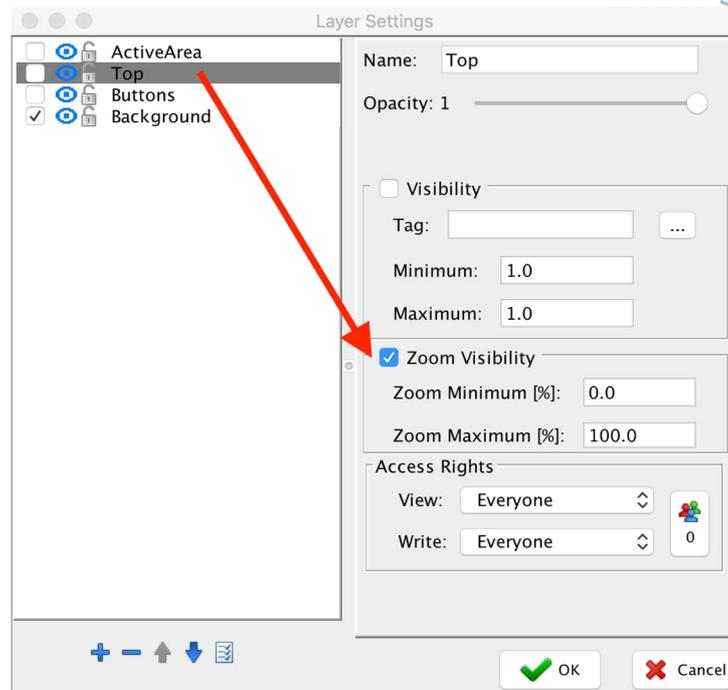
### Zoom visibility on object in view

1. Select your object and navigate to the Anim -> Zoom Visibility
2. Fill in the minimum zoom level where the object will be visible; accordingly, fill in the maximum zoom level where the object will be visible.

Zoom Visibility	
Active	<input type="checkbox"/>
Zoom Minimum [%]	0.0
Zoom Maximum [%]	100.0

### Zoom visibility on the whole layer

1. Invoke the Layer Settings Dialog
2. Fill in the minimum zoom level where the layer will be visible; accordingly, fill in the maximum zoom level where the Layer will be visible.

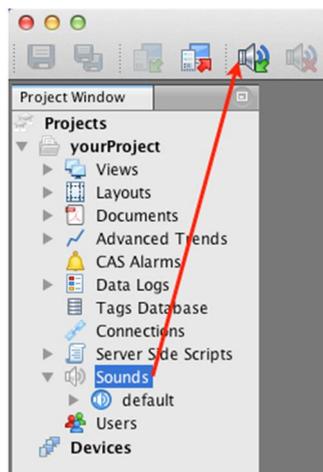


## 14.14 Sounds

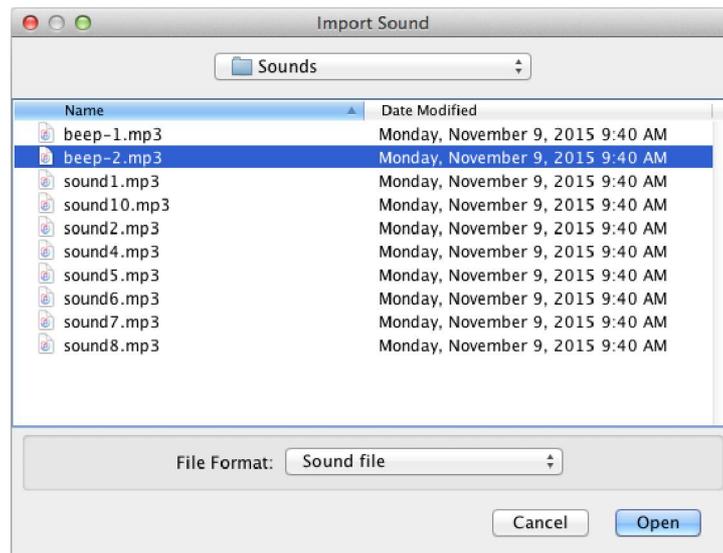
You can create a sound animation, which will play loaded MP3 files upon notification of non-standard situations or voice announcements. You need to import the sound files before you start creating the *Sound* animation.

### Sound import

- 1) Select the **Sounds** folder from the Project window and click on **Import** in the main toolbar or right - click on the folder and select *Import* from the context menu.



- 2) Select the MP3 file from the available folders in the *Import Sound* dialog. Please note the maximum size of the file is 3.5 MB!



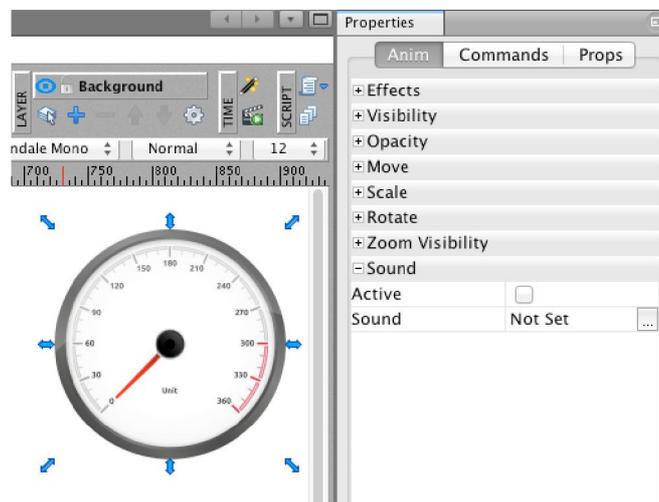
3) The sound file will be loaded into the *Sound* folder.

## Deleting Sound

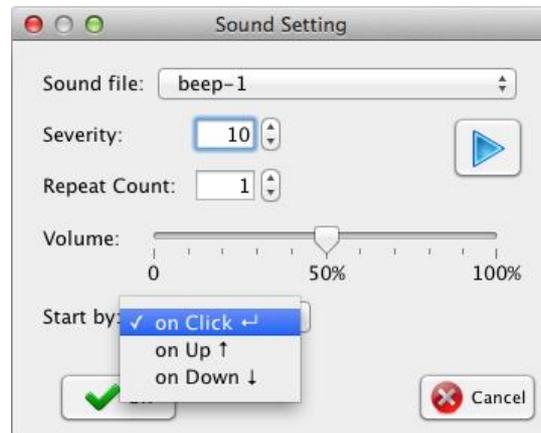
If you wish to delete a sound file, select it from the **Sound** folder and click on the **Delete Sound** icon in the toolbar or select the Delete command from the right-click menu.

## Applying Sound Animation

1) Select the object you wish to link a sound with, navigate to the *Sound* section under the **Anim** tab in the *Properties* window, and click on the '...' button.



2) In the dialog window you can set:



- *Sound file* - MP3 file to be loaded
- *Severity* (priority) - in case of simultaneous occurrence of multiple sound animations, only the sound with the highest numerical severity will be played
- *Repeat Count* - number of repetitions of a selected sound
- *Volume* - defines the relative volume of the played sound (related to the system maximum)
- *Start by* - sets the triggering action for the sound animation:



*on Click* - sound will play if you click on the object

*on Down* - sound will play if the object is pressed (click-and-hold)

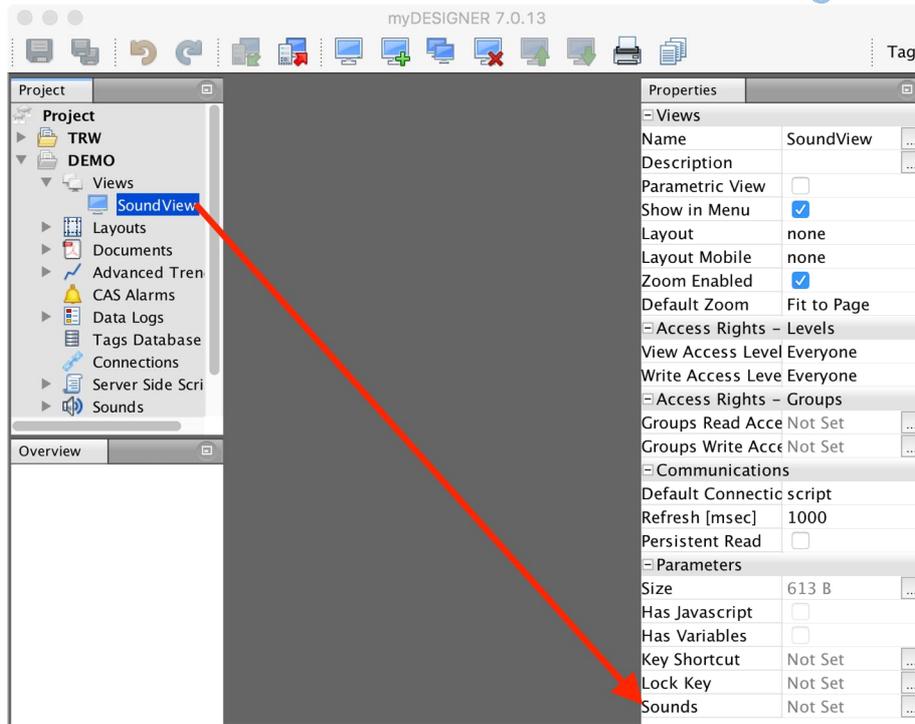
*on Up* - sound will play if the object is released after click or press

3) Click on **OK** once you have set all required parameters.

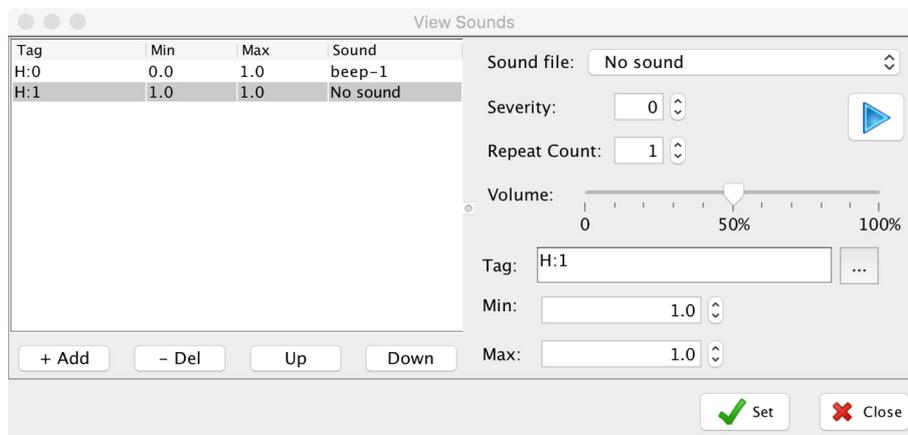
### 14.15 Sounds Triggered by Tag value

Apart from the animations, sounds can be also triggered by a certain tag value read from the PLC. They can be set for each view separately.

- 1) Select the view from the *Project* window in which you want to play the sounds.
- 2) Navigate to the *Parameters* section in the *Properties* window and click on the '...' button.



- 3) In the dialog window, click on the button **+Add** to add the triggering tag, and then select the sound file you want to play.



- 4) Set *Severity*, *Repeat Count*, *Volume*, and *Min & Max* tag range and click on the **Set** button to confirm.

## 14.16 Effects



With effects, you can add dynamic effects to your graphic objects, as described in the previous chapter (*see the difference between Animations and Effects*).

Watch video describing this functionality:

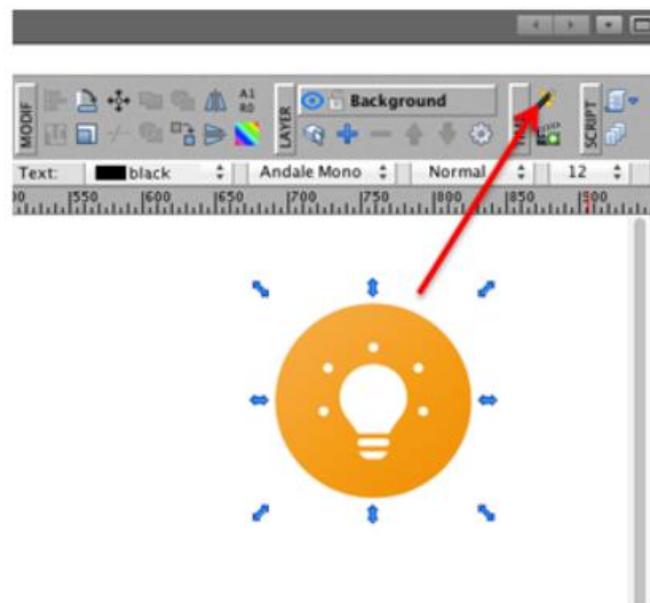
<https://www.youtube.com/watch?v=GTOFCwrRTPM>

## Linking Views with PLC

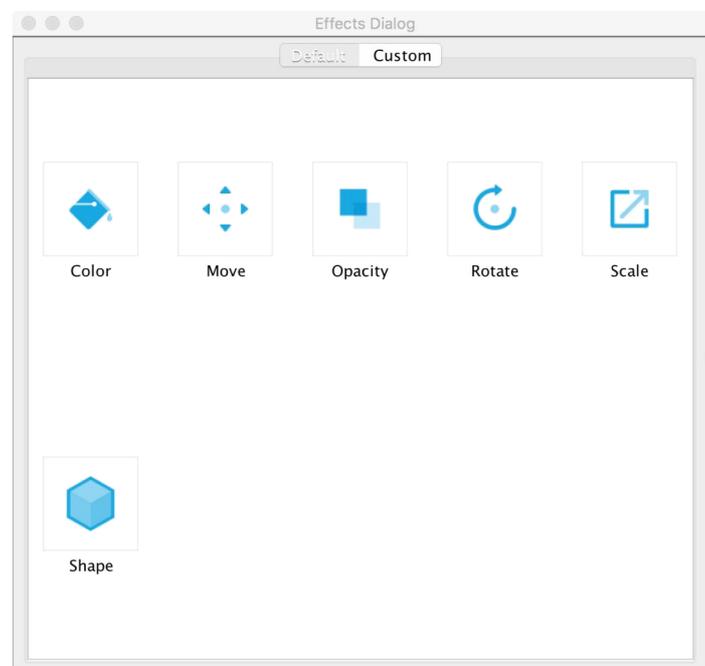
Effects make the graphic visualization less static, and their function is similar to animations. However, they do not reflect the real state of a technological process. We advise using the animations for visualization of a precious technological state and use the effects to visualize the real-time change of objects.

The states of objects during animating are based on actual values obtained from the PLC. For example, the motor rotation visualized through the *Animations* behaves according to the values obtained from the PLC, but the rotation visualized with the *Effects* is based on the values set by you and will use the “hard” data only as a trigger.

- 1) Select the object you want to set effects on and click on the **Effects** button in the GUI toolbar.



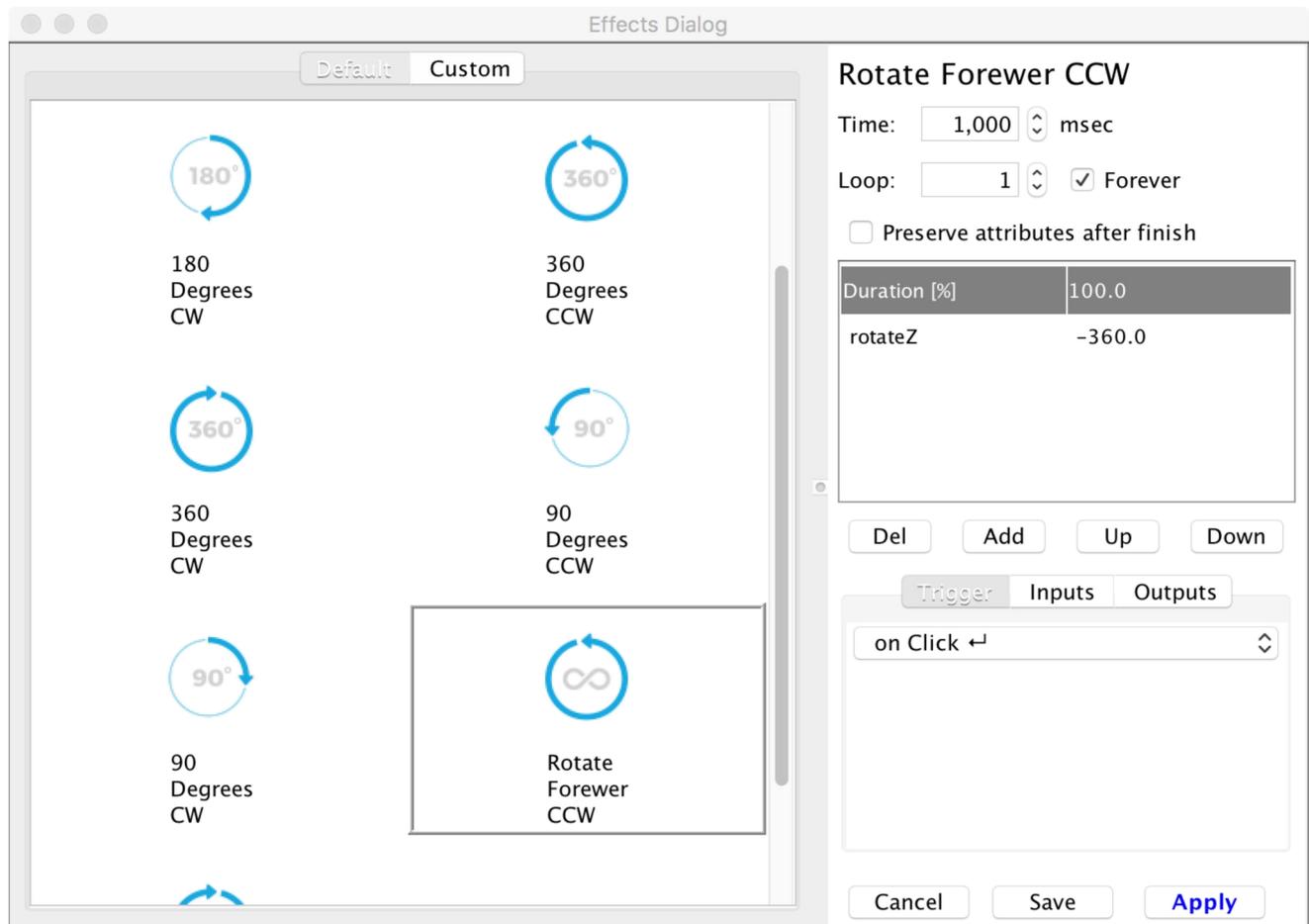
- 2) A new dialog window will show up.



This window has two sections:

- **Default** – here you can find useful effects provided by mySCADA Team
- **Custom** – here you can create your own effects or modify default ones

Effects are grouped together into categories. When you open the effects window, you are presented with the selection of those categories. Click on a corresponding category to see all effects under this category.



### Selecting an effect.

To select an effect, simply click on it. When you click on effect, it is highlighted and you are presented with effects' properties on the right side of the effects window:

Effects' Properties:

Effect name →

Timing →

Property change →

Trigger Action  
Input elements  
Output elements →

**Fill Simple**

Time:  msec

Loop:   Forewer

Preserve attributes after finish

Duration [%]	100.0
fill	

**Effect name:**

Each effect has its own name; you can change the name of an effect by double clicking on the effect icon.

**Timing:**

Effects' timing is controlled in the following section:

Time:  msec

Loop:   Forewer

Preserve attributes after finish

**Time:** overall duration of effect in milliseconds

**Loop:** how many times effect will repeat

**Forever:** check to repeat effect forever

**Preserve attributes after the finish:** when the effect ends, all attributes changed are reverted to the original state as it was before the effect started. If you would like to preserve attribute changes after the end of the effect, check this option.

### Property changes

The mySCADA effects engine enables you to animate different properties for any element. You can change, for example, color, size, move element, and much more. The properties table shows you which properties will be modified when the effect is run.

Duration [%]	100.0	← Section
fill		← property change

Del
Add
Up
Down
← Modification buttons

This properties table has just one section and one property change. When this effect is run, it will change the element color to red.

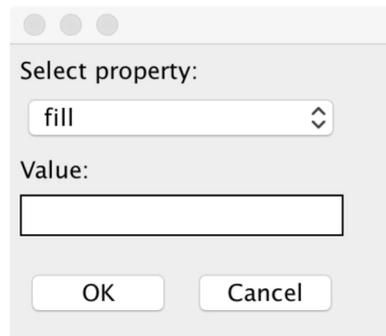
You can change multiple properties at once, or you can change different properties during a time. For example, you can create multiple entries per section (Duration), and you can have multiple sections. The following properties table will show how to change multiple properties at once and have multiple sections.

Duration [%]	50.0
fill	
stroke	
Duration [%]	50.0
rotateZ	90.0
Reset to:	
opacity	0.0

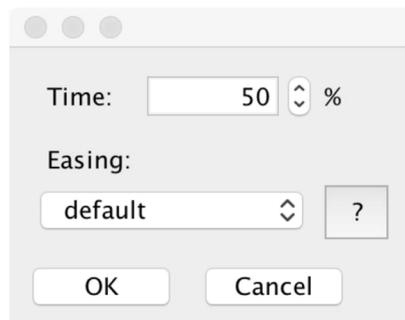
In this example, the fill and stroke color of output elements will be changed to red and yellow. The change will take time in first half of time (Specified by Duration 50.0 %). Then, the element will start to rotate and will rotate by 90 degrees. At the end of the effect, the

element will disappear; this is due to the “Reset” section where we have specified opacity = 0.0.

To modify a property, double click on it. The Modify Property Dialog will be shown:



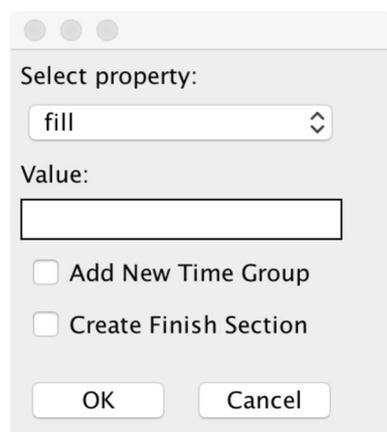
If you wish to modify a Time Section (Duration), double click on Duration label; a new Modify Duration Dialog will be shown:



Modify parameters:

- **Time:** this is the duration of the time section shown as a percentage. If you specify the overall time of effect to be 1000 milliseconds and then set the time duration to 20 %, the animated change will take exactly 200 milliseconds.
- **Easing:** specifies the rate of change of a parameter over time. You can choose from several easing functions. To see them all, click on the ? button next to the easing combo box.

To add a new property or section, click on **Add Button**, and a new dialog will be shown:



Select a property you would like to add and set its value.

If you would like to create a new Time Section (Duration), check “Add New Time Group,” and additional properties will be shown:

Select property:  
 fill

Value:  
 [Empty text box]

Add New Time Group

Time: 20 %

Easing:  
 default [?]

Create Finish Section

OK Cancel

Fill in the required parameters:

- **Time:** this is the duration of the time section shown as a percentage. For example, if you specify the overall time of effect to be 1000 milliseconds and then set time duration to 20 %, the animated change will take exactly 200 milliseconds.
- **Easing:** specifies the rate of change of a parameter over time. You can choose from several easing functions. To see them all, click on the ? button next to the easing combo box.

If you would like to create a Finish Time Section, check the appropriate checkbox:

Select property:  
 fill

Value:  
 [Empty text box]

Add New Time Group

Create Finish Section

OK Cancel

The Finish section is useful if you would like to set some properties for a given value at the animation end. If you use the Finish section, you should check “Preserve attributes after finish”.

## Trigger

A trigger specifies how an effect will be started.



In the context menu, you can set the effect trigger:

**on Click** – effect will be activated when the object is clicked

**on Down** – effect will be activated when the object is pressed

**on Up** – effect will be activated when the object is released (after on Down action)

**on Down Up** – effect will be activated when the object is pressed and stopped when the object is released.

**on Mouse Over** – effect will be activated when you hover over the object with the mouse pointer and stopped when the mouse pointer leaves the object

**Tag** – effect will be activated when selected tag value reaches given range and stopped when the value is out of range.

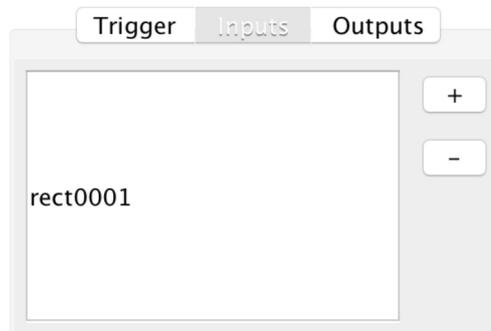
**Tag Trigger** – effect will be activated when selected tag value reaches given range, but the effect will NOT be stopped when the value is out of range.

If you select **Tag** or **Tag Trigger** from the trigger menu, set the tag address and the range of the **Min** and **Max** values.



## Inputs

The Inputs section specifies the elements that trigger the effect action. If you, for example, set the action to on Click and then specify multiple Input elements, any of them will act as a trigger effect when a user clicks it.



The "+" and "-" buttons allow you to add or remove Input elements from the list.

## Outputs

The Outputs section specifies the action elements to which the effect is applied. Usually, you want to apply an effect to the same element you have for the action trigger. In this case, you can leave the default settings as "ThisElement." If you want to have different or multiple elements, use the "+" and "-" buttons to select the corresponding elements.



## Saving Effect

If you have modified effect properties and would like to save them for later use, press the "Save" button.

## Applying Effect

If you would like to set/apply an effect, press the "Apply" button.

# 15 Time Sequence

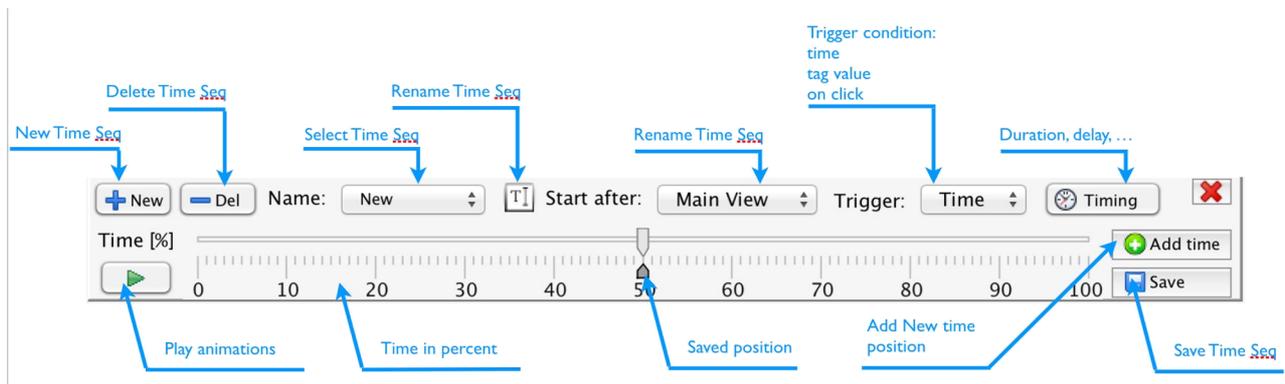
Time Sequence is an easy to use and extremely powerful function that adds timed animations to the views. The *Time Sequence* editor is part of the GUI editor and allows you to change graphic properties of your objects for specified time intervals. myDESIGNER automatically computes the transitions between these time intervals.

Watch video describing this functionality:  
<https://www.youtube.com/watch?v=VYeJfgMaERQ>

You can open the time sequence editor by clicking on the **Time Sequence** icon, located in the GUI toolbar.



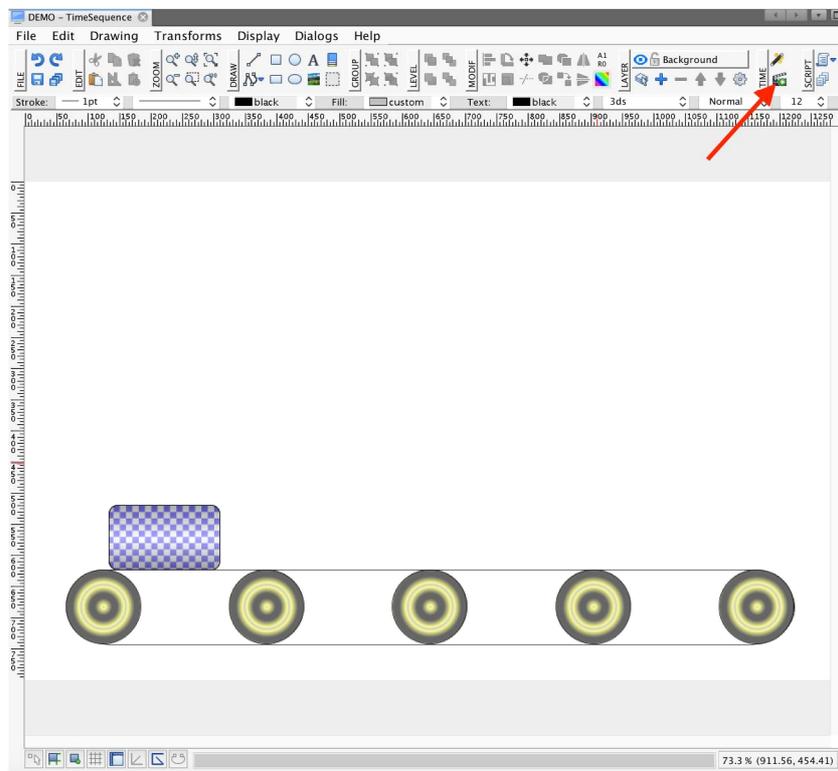
The GUI editor automatically adds the *Time Sequence* toolbar to the *Main Window*.



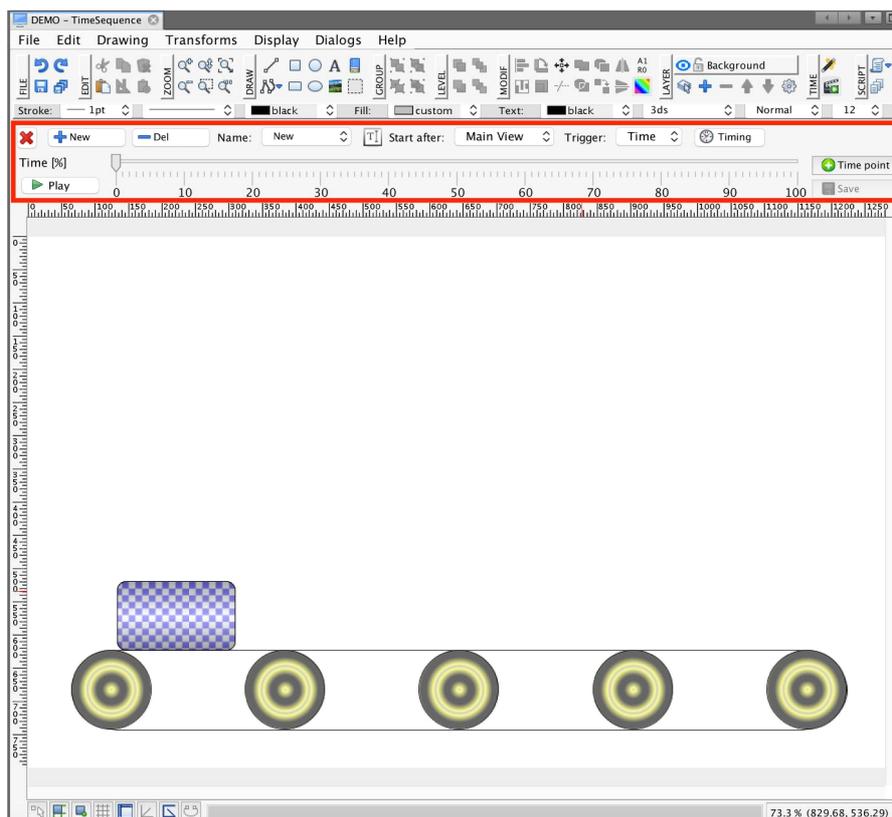
Creating a time-sequenced animation is very easy. You just move the time slider and modify the graphic objects for each time position, as you want them to appear at these time positions. The smart algorithm inside the mySCADA runtime will compute the animation so that it is time-fluent.

## 15.2 Example

1. Open the Time Sequence editor.



2. Now you are presented with the Time Sequence Editor. All of the controls are located in the red square.



3. Now move the time slider to the first time point.



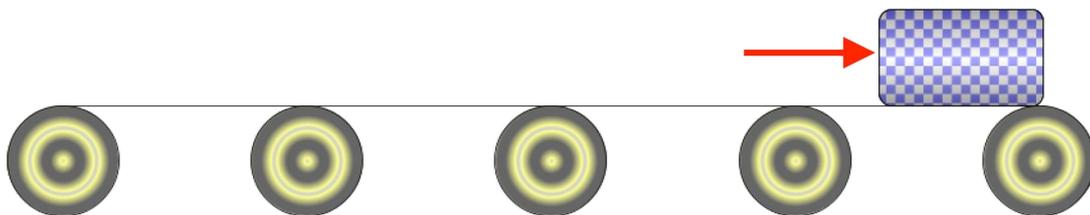
4. Now modify your objects in the way you want them to appear at this time point.

Allowed operations are:

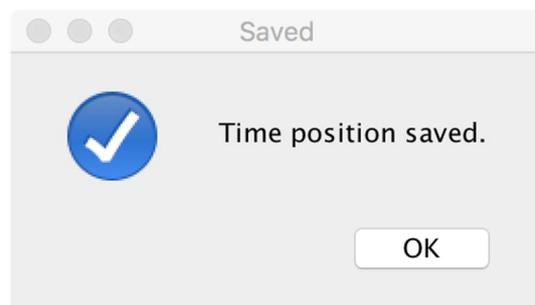
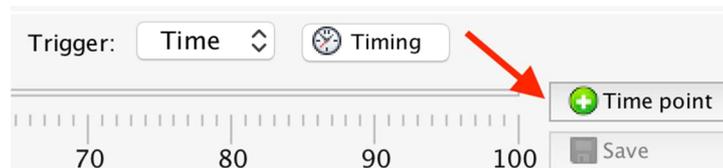
- Position change
- Size change
- Color change
- Opacity

Prohibited operations are:

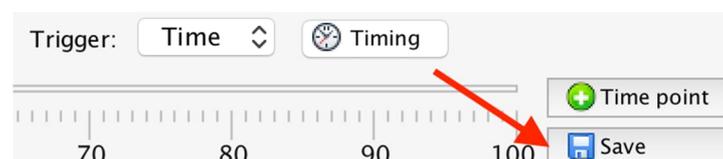
- New object creation (use opacity to show/hide objects)
- Rotation change



5. Save your time position.



6. Add more time positions and save the time sequence to complete it.



7. Now, if you play the animation, the object on the conveyor will move.



### 15.3 Triggering Time Sequences

Time sequences can be triggered by several actions:

- by time
- by tag value
- by user action

To set a trigger, select the required action in the Trigger combo box.

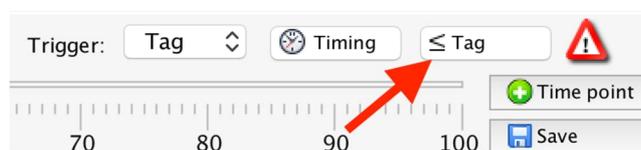


#### Trigger by Time

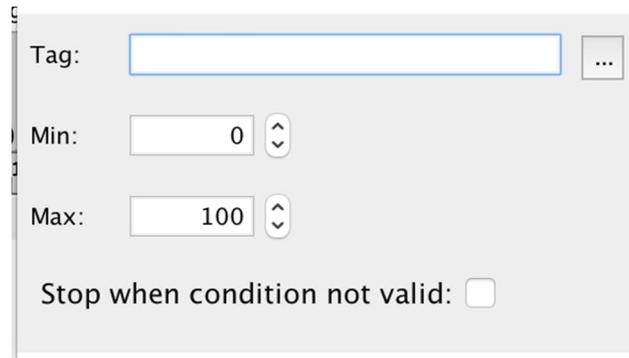
Trigger by time will play your time sequence after the view is shown. It can be delayed in the Timing settings.

#### Trigger by Tag

If you set trigger by tag value, you should specify the tag, minimum, and maximum values. If the tag is between the minimum and maximum values, your time sequence will be triggered. To modify the values, please click on the Tag button:



You will be presented with the Tag Settings dialog.

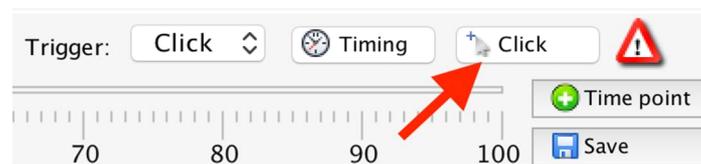


Once the tag value is between the specified **Min** and **Max** values, your time sequence will start playing. It will stop after the specified time duration (see Time Settings). If you want to stop your animation immediately after the tag is out of the Min/Max range, please check **“Stop when condition not valid:”**

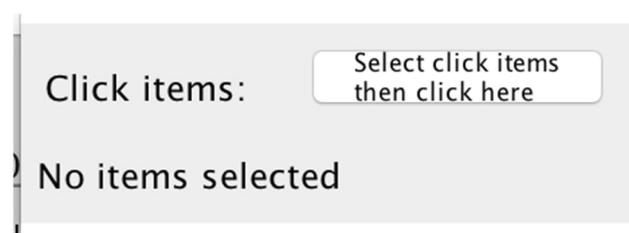
### Trigger by User Action - Click

A Time Sequence can be also started by a user action. You can specify which graphical objects can trigger your time sequence.

1. Select all the graphical objects you want to trigger your time sequence. Now click on the **“Click”** button.



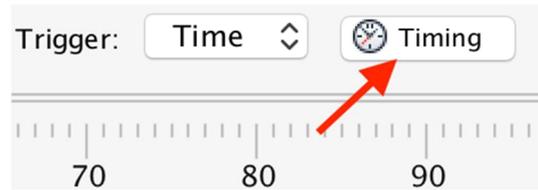
2. You will be presented with a setting dialog:



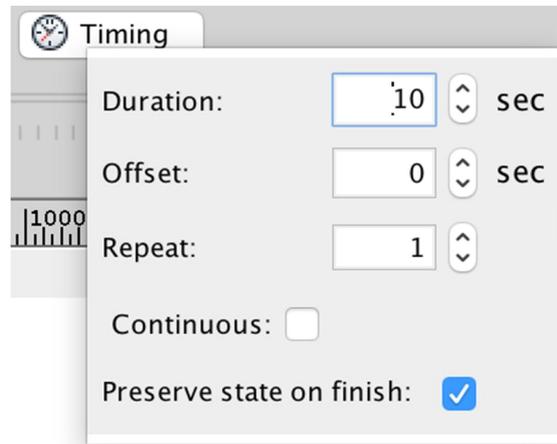
3. Click on the **“Select click items, then click here”** button.

## 15.4 Modifying Time Settings

To modify the time duration, offset, or repeat count of your sequence, press the Timing button:



You will be presented with the Time Settings dialog:



In the Time Settings dialog, you can specify the overall time duration (this is also the reason why the time scale is shown as a percentage). You can also specify the time offset. If the time offset is greater than 0, the time sequence will start after the specified number of seconds. Set the repeat count to repeat the series or click on Continuous to repeat it forever.

*Tip: Use the "Continuous" option in combination with the tag triggered option "Stop when condition not valid" to play your time sequence continuously when your tag value is within the limits.*

### Preserve state on finish

When the time sequence finishes, it can either leave all graphical objects as they were at the end of the animation or it can set your objects to the initial state (e.g. set them as they were at the beginning of time sequence).

Preserve state on finish activated: leave all graphical objects as they were at the end of animation

Preserve state on finish deactivated: reset your objects to the initial state (e.g. set them as they were at the beginning of the time sequence)

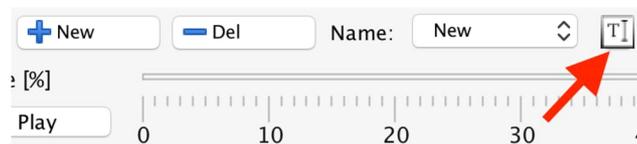
## 15.5 Adding New Sequences

You can easily add or remove time sequence by using the **New** and **Del** buttons:

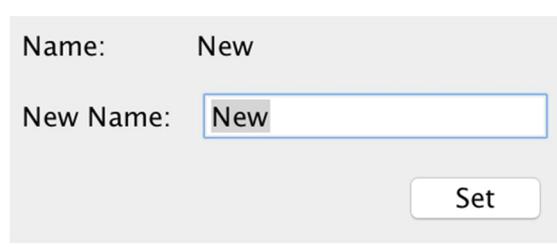


## 15.6 Renaming Time Sequence

To rename a time sequence, press the following button.



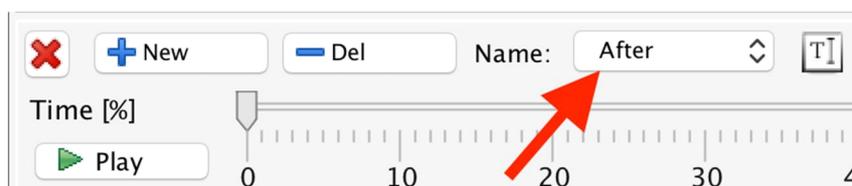
You will be presented with the rename dialog:



Put in a new sequence name and confirm using the **Set** button.

## 15.7 Switching Among Time Sequence

To change a time sequence to a different one, use the Name combo box:

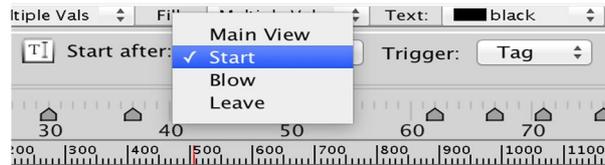


Once you change the time sequence, your view will change to the given time sequence, and you can start editing it.

## 15.8 Modifying Timing

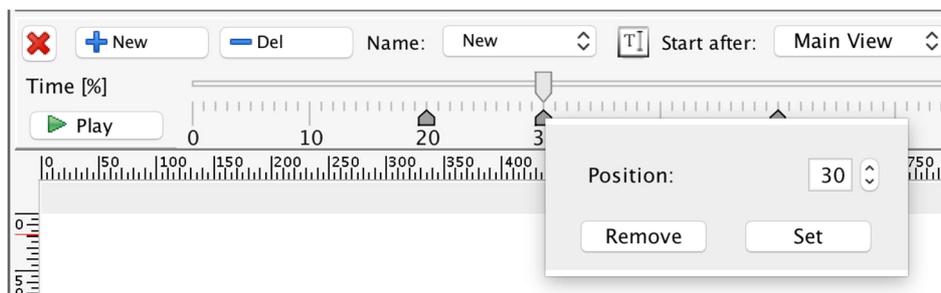
The Time Scale Slider shows all saved time points. These are the little arrows below the scale.





You can modify the timing by moving these time tags to the side. You can also specify the exact value by right clicking on the time tag. To delete the time tag, right click on it and select **Remove**.

## 15.9 Combining Multiple Time Sequences



You can combine multiple time sequences. Imagine you have a production line where you need to perform 10 different operations, and each operation must start one after another. This can be easily achieved by combining multiple time sequences. You can simply nest one animation after another. To do so, change the **Start after** option to your previous animation.

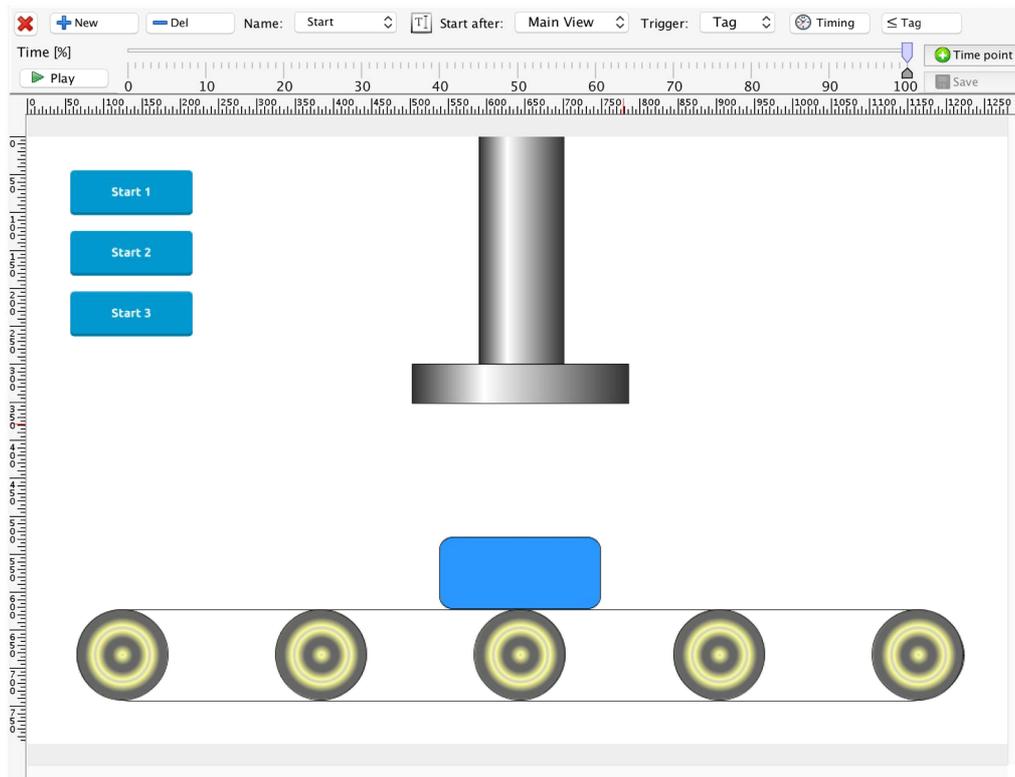
### Example:

Let's say you need to animate three consequent operations:

1. Object moving on the belt to the desired position
2. Object being pressed
3. Object leaving

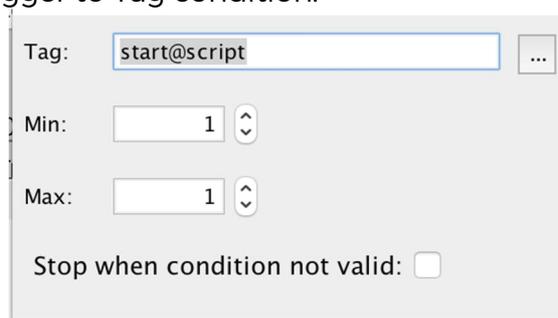
Each operation is triggered by the tag.

1. First of all, we will start by creating our view.
2. We will create the start sequence by creating the first time sequence.

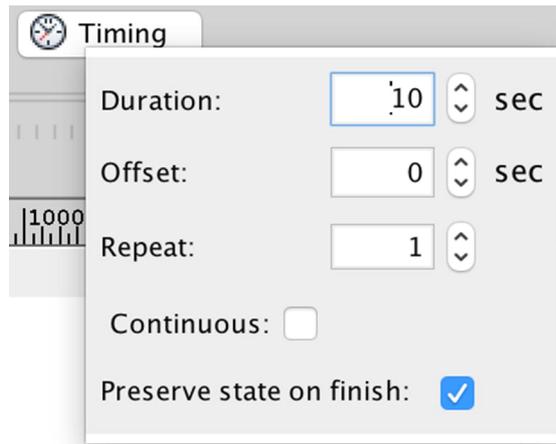


You can see that we have moved the time slider to the end. We have also moved the object under the press.

We have set the trigger to Tag condition:



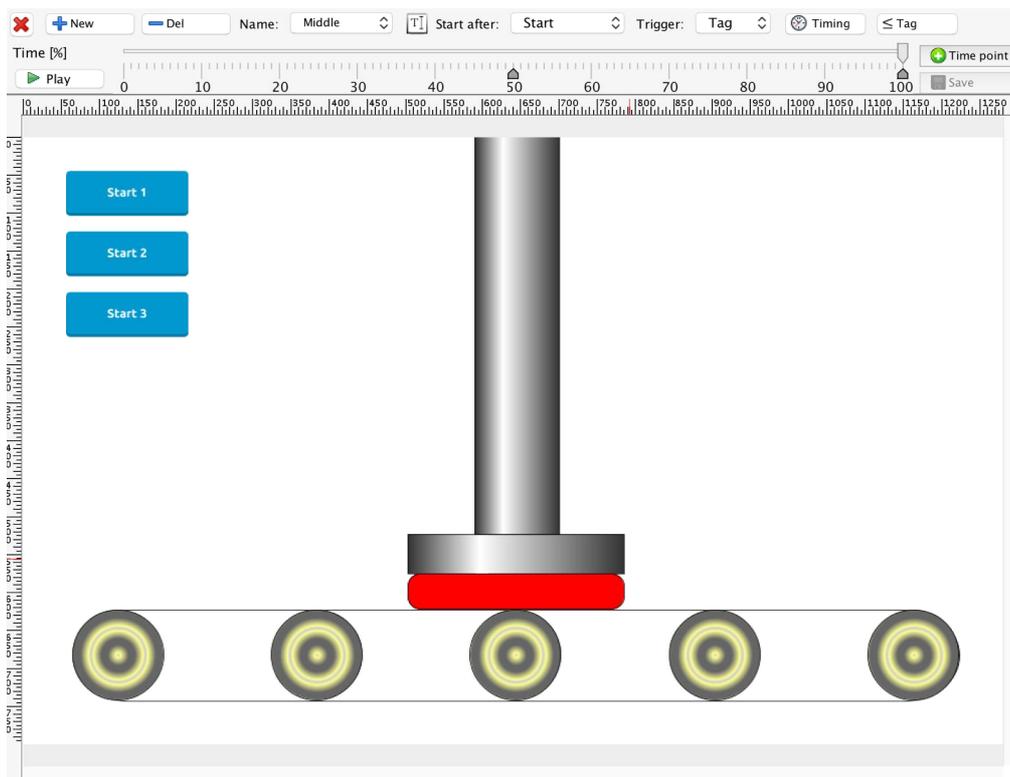
In the timing settings, we have checked Preserve State on finish.



- Now we will create the second time sequence. This one will start after the first one, so we will set the **Start after**:



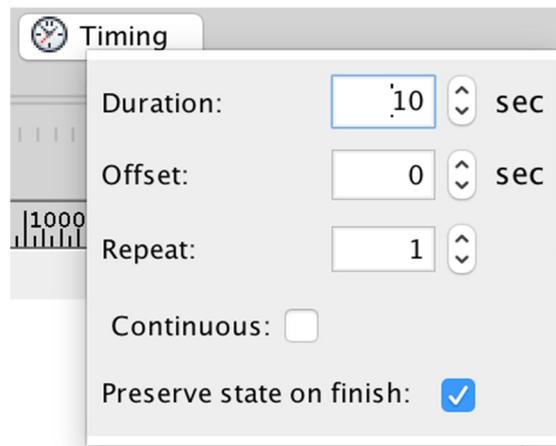
And our time sequence:



We have set the the trigger to Tag condition:



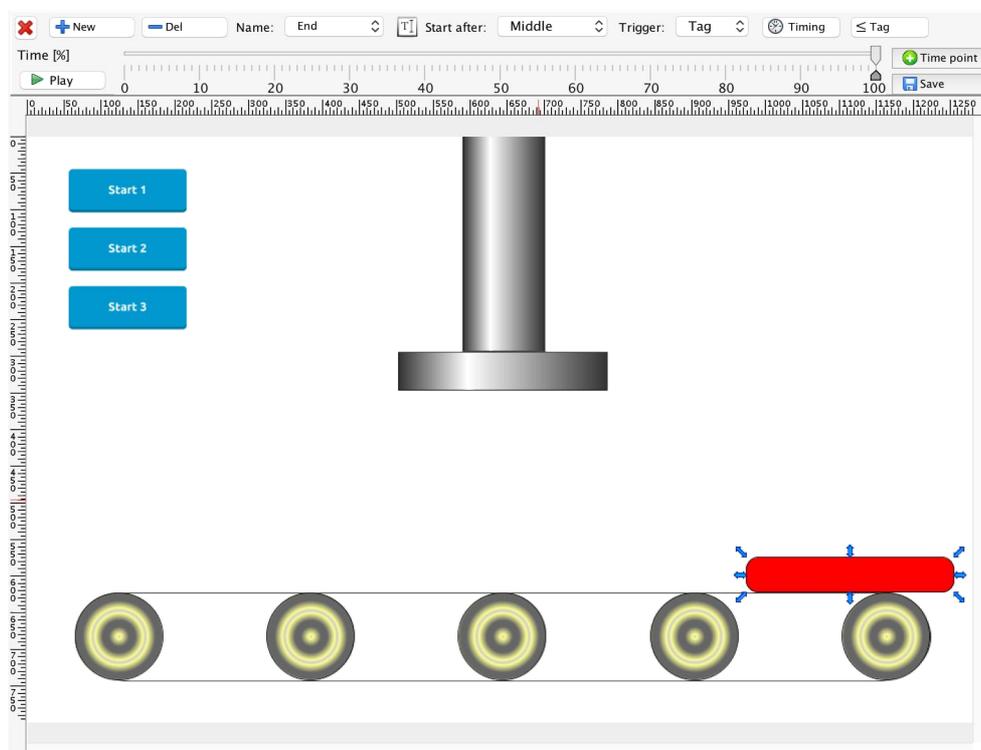
In the timing settings, we have checked Preserve State on finish.



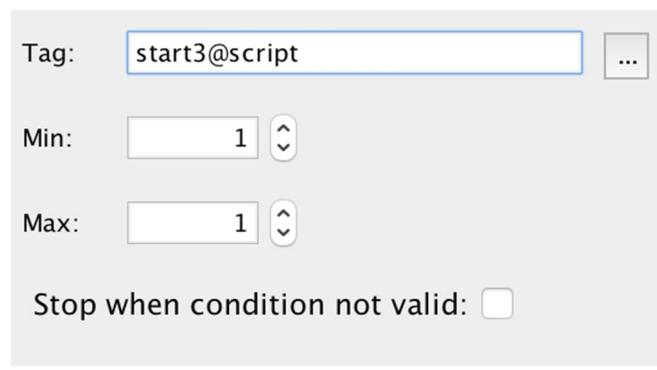
- And now we will create the final time sequence. This one will start after our previous one, so we will once again set the Start After:



And our time sequence:



We have set the trigger to Tag condition:



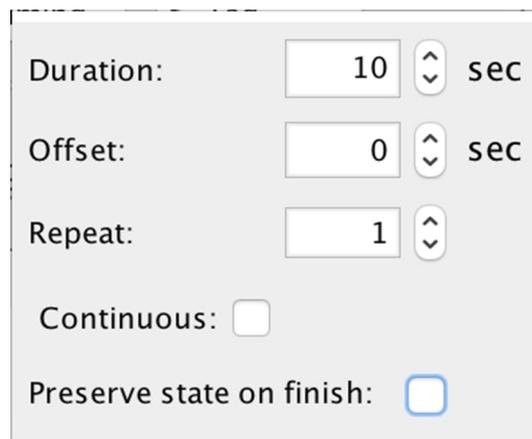
Tag:  ...

Min:  ^ v

Max:  ^ v

Stop when condition not valid:

In the timing settings, we have disabled Preserve State on finish. This will allow our view to return to its starting condition.



Duration:  ^ v sec

Offset:  ^ v sec

Repeat:  ^ v

Continuous:

Preserve state on finish:

# 16 Open Command

This command is used to navigate between the HMI screens, e.g. you can open other project views from currently open ones. In addition, you can use the open command to change the content of the Active Area. The Open Command can be applied to any graphical object.

Watch video describing this functionality:

[https://www.youtube.com/watch?v=7z\\_0\\_DcmvSQ](https://www.youtube.com/watch?v=7z_0_DcmvSQ)

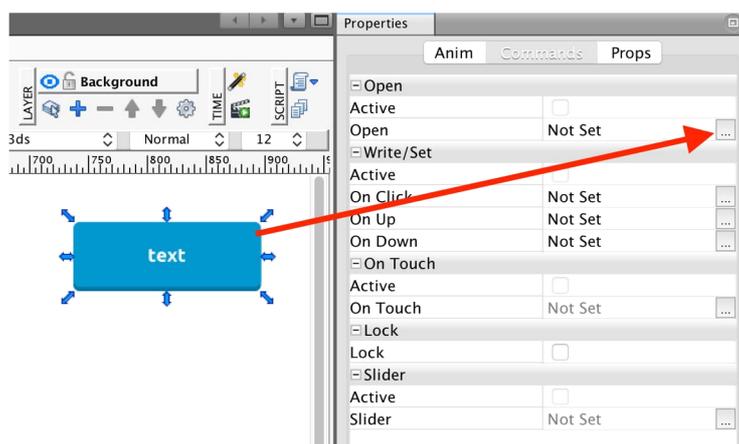
With the open command you can:

- Navigate to a different view
- Open a view in a new window (except on mobile devices)
- Change the content of the Active Area

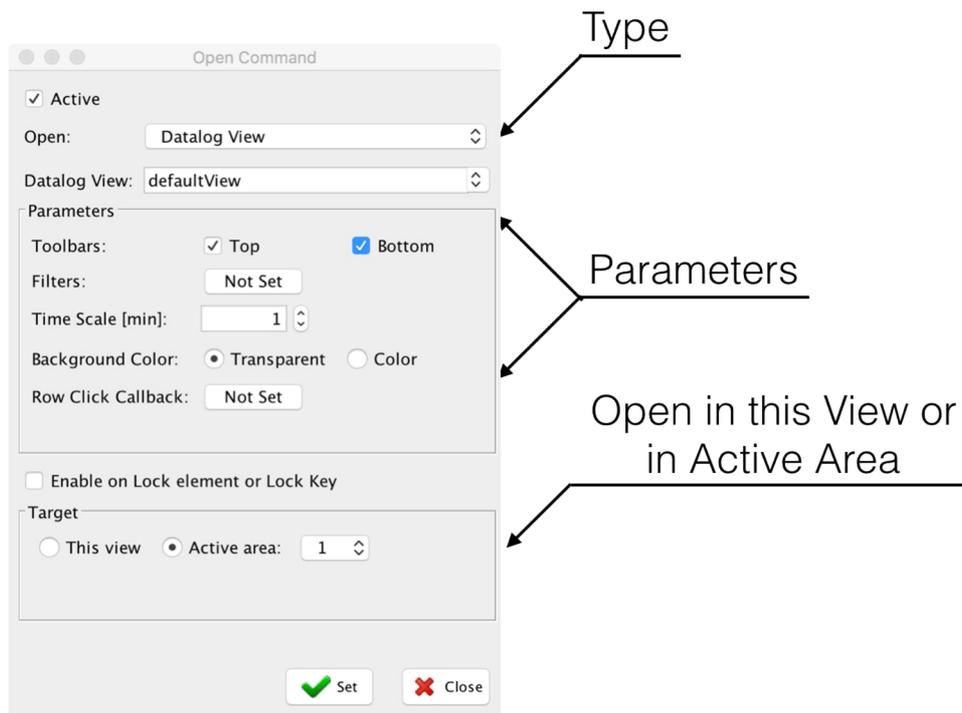
With the Open Command, you are able to open:

- Views
- Parametric View
- Advanced Trends
- Data-log views
- User actions
- External web-pages
- Any HTML5 content

To use the open command, please navigate to Properties -> Commands and click on the Open button.



You will be presented with the Open dialog:

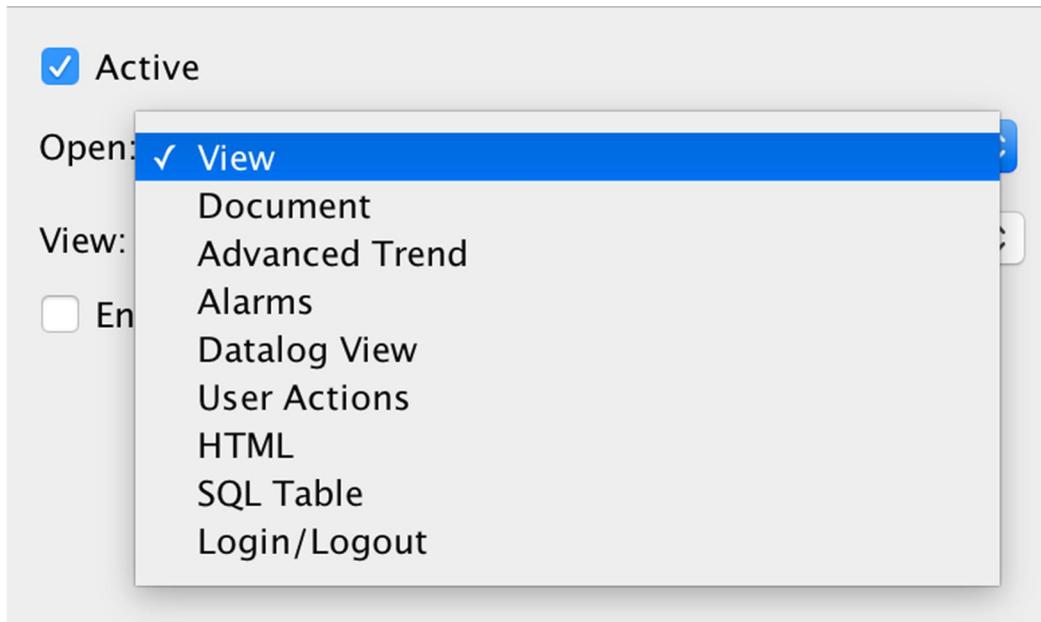


The Open dialog has three sections:

- Open: here you specify what you want to open
- Parameters: this section changes depending on your previous selection.
- Target: open in this view or in active area

## 16.1 Open Type

Select from the available options the entity type that should be opened by the command:



- *View* – can open a previous screen or any arbitrary view of your project
- *Parametric View* – parametric window will be opened
- *Document* – opens one of the project documents, *Index* property sets the page number the document will open on
- *Advanced Trend* – advanced trend will be opened
- *Alarms* – you can choose between *Online* alarms or *History* alarms
- *Data-log* – opens a data-log of your choice
- *User Actions* – opens a window with the user actions manager
- *HTML* – opens any web content you define in the dialog window
- *SQL Table* – shows SQL table
- *Login/Logout* – shows login dialog

## 16.2 Target

In Target you can specify if you wish to use the open command to replace the current view, open content in the Popup window, or show content in the active area. The Target option is only visible if you have Active Area in your view.



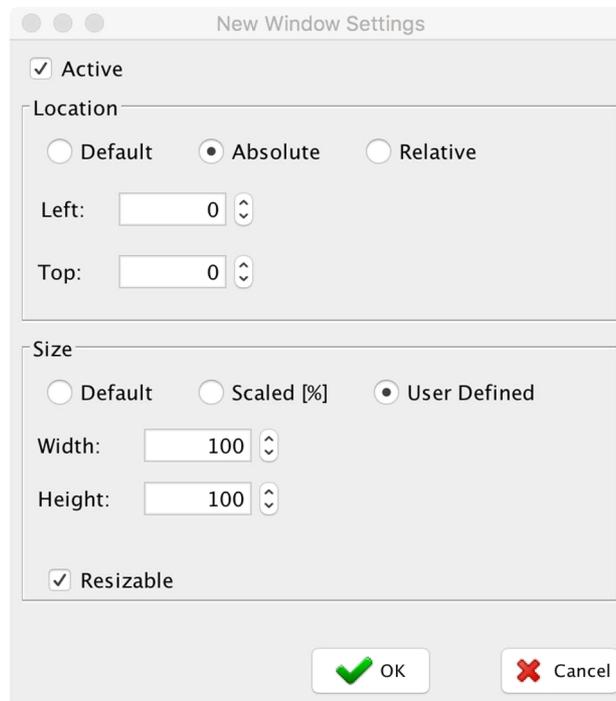
Set **“This view”** option to open the content in this window. Select **“Active area”** to set content in the Active area with a given number.

If you wish to open content in a new popup window, click on the **“...”** button. You will be presented with a new dialog to set up the parameters of the newly opened window.

## 16.3 Popup (Face Plate) Window

Popup Windows allow you to show a new window on top of your current window. The popup window can be useful to show multiple views at once, or you can use popup windows as a Face Plate.

- 1) Click on the “...” button in the Target area to set the popup window. You will be presented with the Popup Window Settings dialog.



### Location

Location sets the location on the screen where a new window will pop-up.

- Default: show window at the center
- Absolute: set absolute coordinates of top left corner
- Relative: set relative offset to current mouse position

All values are in pixels.

### Size

Size of a newly open window

- Default: original size of the window. A window will be shown with the same resolution as is defined in the designer.
- Scaled: scale window to given percentage
- User Defined: specify new dimensions of opened window

### Resizable

If resizable is selected, the user can resize the view. Otherwise, the size will be fixed.

## 16.4 View type

If you select this option, you can specify the view to be shown. You can display any view you have previously designed. For best results, use a window with the same resolution or at least the same aspect ratio as your active area.

Active  
 Open: View  
 View: TimeSequence

*TIP: If you specify the option "Previous View," open command will jump back to previously open view.*

### Parametric View type

If you select this option, you can specify the parametric view to be shown. You can display any view you have previously designed.

Active  
 Open: Parametric View  
 View: ParametricView[PAR]  
 Parameters: Not Set ...  
 Connections: Not Set ...

Aside from the Parametric View name, you can also specify the Indexes and Connection indexes.

For more details on how to deal with parametric views, please see chapter [Parametric Views](#).

### Document type

You can show a PDF document linked to the project. The PDF document will be automatically scaled to the size of your view.

Active

Open:

Document:

Page:

Select a PDF document. You can also specify on which page to open the document.

## Alarms type

This feature can show online alarms and alarm history. There are several parameters you can use to set up the specific visual appearance of the alarm table that will be shown. You can also limit shown data using filters.

Active

Open:

Alarms:

Toolbars:  Top  Bottom

Parameters:

Time Scale [min]:

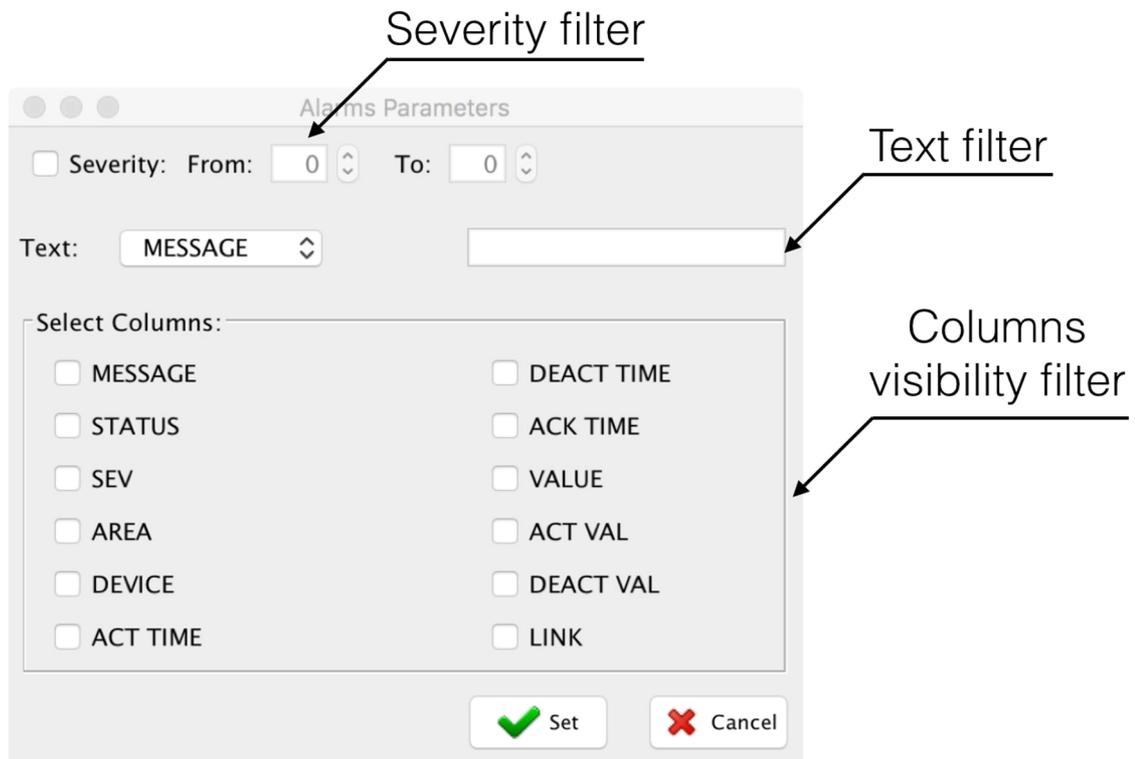
Background Color:  Transparent  Color

Row Click Callback:

Enable on Lock element or Lock Key

- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters and alarm settings.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g., the date selection controls).

- *Parameters* allow you to specify which columns will be visible in the alarm table. In addition, you can set a filter for severity and text filters for the message, area, and device.

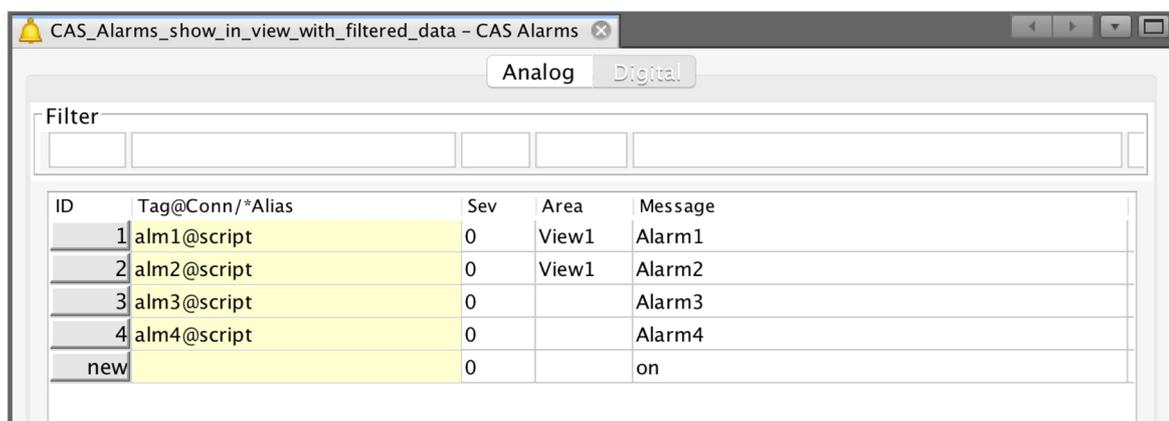


- *Time Scale [min]* parameter specifies the time interval that will be shown in the alarm history table. Units are in minutes.
- *Background Color* can be set to transparent or to a specific color. If you set this parameter to color, you can specify which color will be shown as the background.

### Example on showing filtered Alarm data

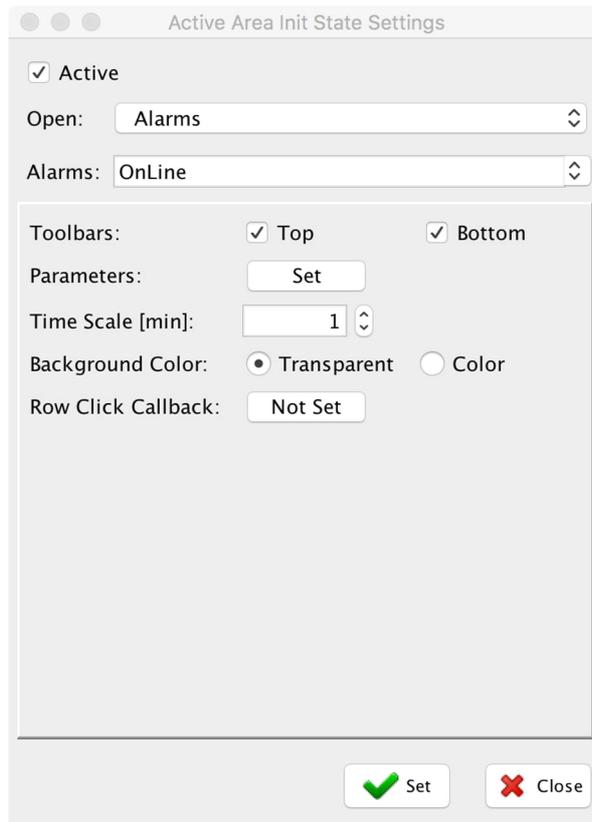
Suppose you want to show the alarm table on the screen, but only show data in respect to the given view.

1. Create alarm definitions in CAS Alarm Windows. To designate which alarms belong to the view you plan to show, specify the name of the view in the Area column.

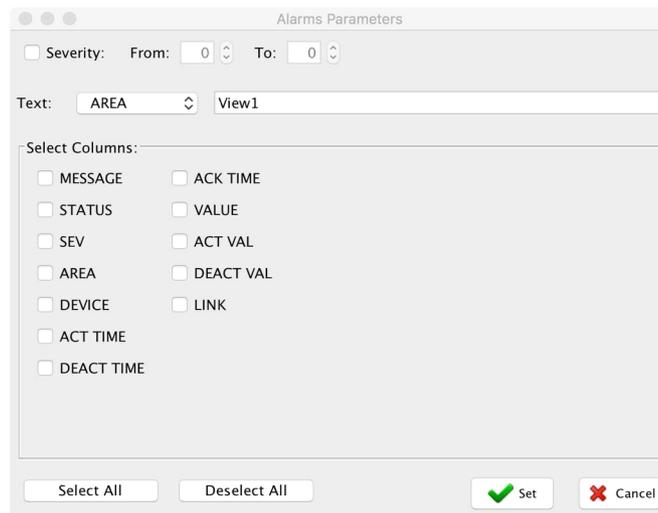


2. Create a new view and insert a button.

3. On the button set up an on click open command.



4. Set up data filter as follows:



5. Test your view.

## Advanced Trend type

You can show trends, as defined in the *Advanced Trends* section. The size of the trend is automatically adjusted to the size of the view.

The screenshot shows a configuration dialog for an Advanced Trend. It includes the following settings:

- Active
- Open:
- Trend:
- Toolbars:  Top  Bottom
- Filters:
- Time Scale [min]:
- Background Color:  Transparent  Color
- Row Click Callback:
- Enable on Lock element or Lock Key

- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters and advanced trend settings.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g., the date selection controls).
- Filters allow you to set a time range for the advanced trend shown. You can specify *Time From* value (in UNIX UTC time; e.g. seconds since the year 1970), *Time To* value, or *Time Scale* value. You can use constants, or you can use tags from PLCs or variables from View Scripts.  
If you use variables or tags in the bellow fields, you can change the time interval of the advanced trend dynamically.

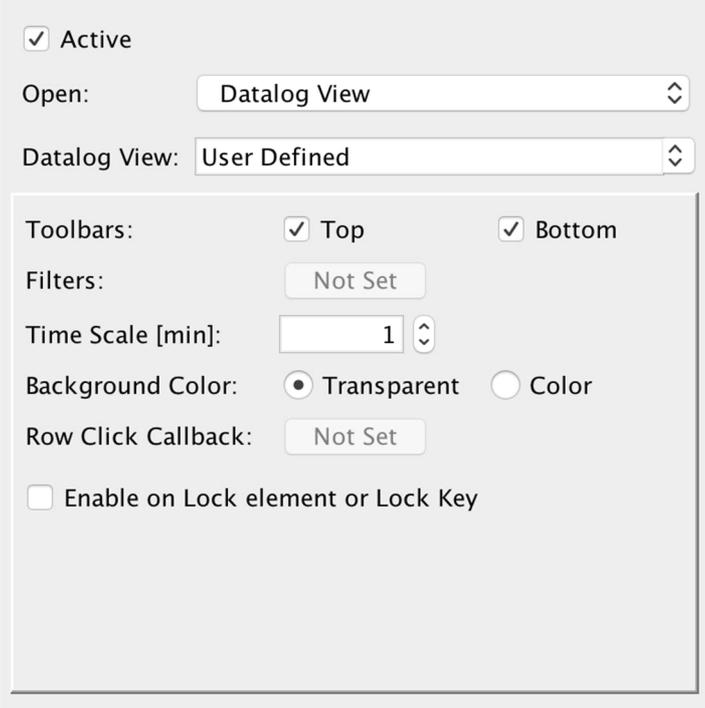
The screenshot shows a configuration dialog for Time [Tags (Address)]. It includes the following settings:

- Time [Tags (Address)]:
- Time From [sec]:  ...
- Time To [sec]:  ...
- Time Scale [sec]:  ...

- *Time Scale [min]* parameter specifies the time interval that will be shown by the Advanced Trend. Units are minutes.
- *Background Color* can be set to transparent or to a concrete color. If you set this parameter to color, you can specify the color that will be shown as the background.

## Data-log View type

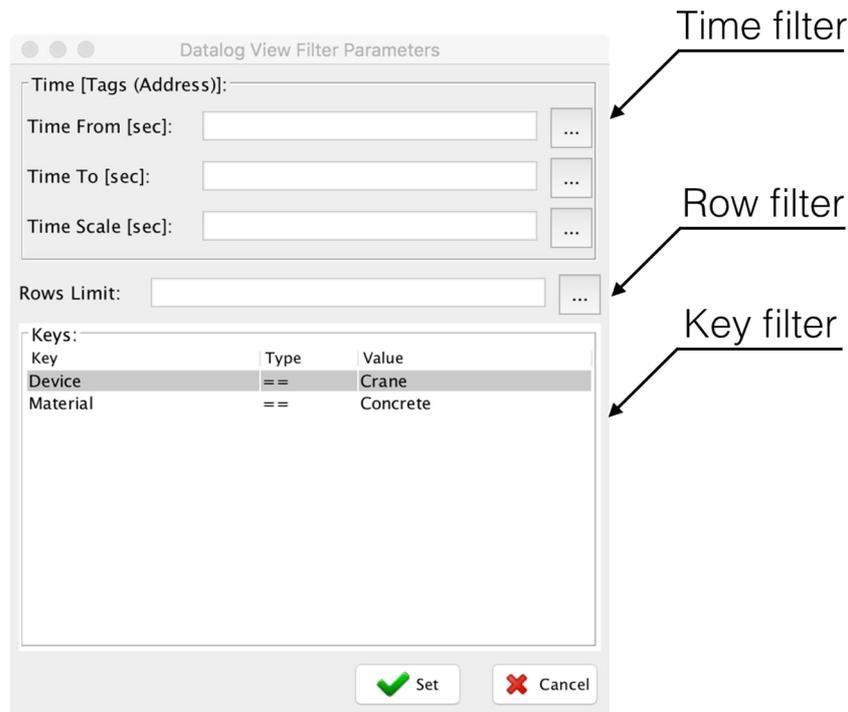
You can show data-log view, as defined in the *Data-log* section. The size of the data-log view will be automatically adjusted to the size of the view.



The screenshot shows a configuration dialog for the Data-log View. It includes the following settings:

- Active
- Open: Datalog View
- Datalog View: User Defined
- Toolbars:  Top,  Bottom
- Filters: Not Set
- Time Scale [min]: 1
- Background Color:  Transparent,  Color
- Row Click Callback: Not Set
- Enable on Lock element or Lock Key

- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters and data-log view settings.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g. the date selection controls).
- Filters allow you to limit the data shown by time, number of records, or by parameters.



**Time filter** allows you to set time ranges for records shown. You can specify a *Time From* value (in UNIX UTC time; e.g. seconds since the year 1970), *Time To* value, or *Time Scale* value. You can use constants, or you can use tags from PLCs or variables from View Scripts.

4. Using constants: just write the value in seconds in the text field
5. Using Tag: enter the tag value or use “...” button to specify a tag.
6. Using Variable: enter an = followed by the variable name.

**Row Filter** limits the total number of records shown. Again, you can use constants, or you can use tags from PLCs or variables from View Scripts.

**Key Filter** can limit the data shown by specifying filters for data items. Keys must be enabled in data-log definition to show up in the filter. For each defined key, you can specify the value. To specify a value, you can use constants, or you can use tags from PLCs or variables from View Scripts.

- *Time Scale [min]* parameter specifies the time interval that will be shown by the Advanced Trend. Units are minutes.
- *Background Color* can be set to transparent or to a specific color. If you set this parameter to color, you can specify which color will be shown as the background.

### Example on showing filtered datalog data

Suppose you want to show an on-screen data-log view table, but only show data filtered by user input.

1. Create data-log and data-log view. In the data-log tag definition, check the Key check box to enable filtering using this tag value.

Connection:

Log rate: Day: 0, Hour: 0, Min: 0, Sec: 0, Msec: 1

Read Refresh: 0, 0, 0, 2

Triggered logging:  Log single value,  Continuous logging

Number of samples to log: Before event: 0, After event: 0

Alarm IDs:

Id	Name	Description	Tag	Unit	Format	Hysteresis	Delta Abs	Type	Parameter	Alignment	Key
1	output		output@s...		#. #	<input type="checkbox"/>	0	Value	Decimal	left	<input checked="" type="checkbox"/>
new					###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>

Down ↓ Up ↑  Export CSV, PowerBI  Enable Data Filter

Data-log Views

Name	Description	Data Points (IDs)	Show in Menu	Hide ID	Hide Date	Accesses
default		1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Set
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Set

2. Create a new view and insert an button into it.
3. On the button, set up Open command.

Active Area Init State Settings

Active

Open:

Datalog View:

Toolbars:  Top  Bottom

Filters:

Time Scale [min]:

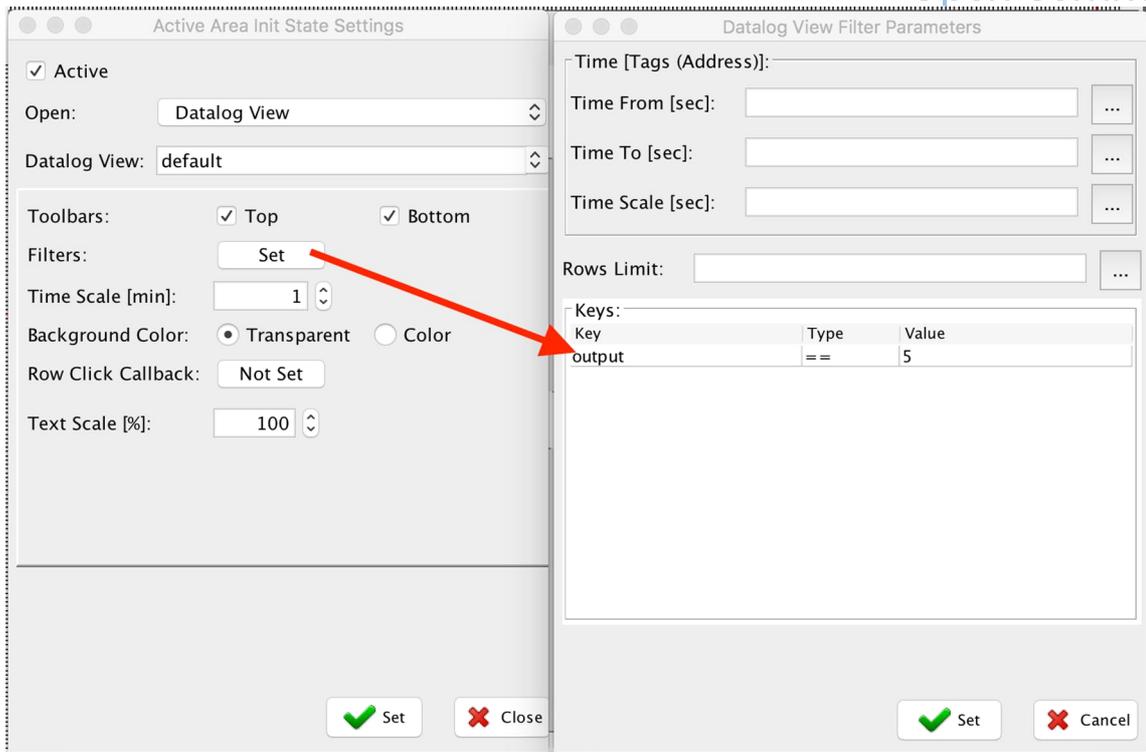
Background Color:  Transparent  Color

Row Click Callback:

Text Scale [%]:

Set  Close

4. Set up data filter as follows:



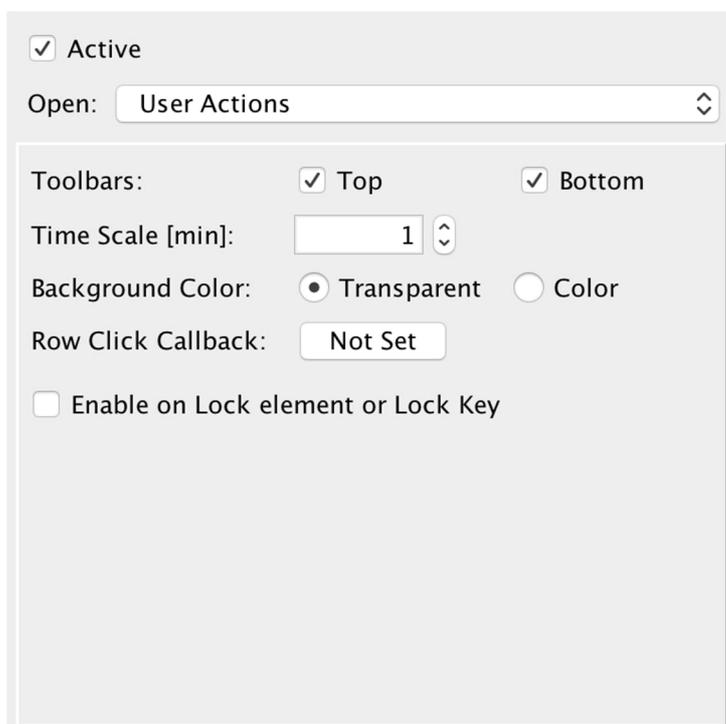
5. Test your view.

**DOWNLOAD DEMO PROJECT HERE:**

[ftp://nsa.myscada.org/history/projects/example/Showing\\_filtered\\_data\\_log\\_data.mep](ftp://nsa.myscada.org/history/projects/example/Showing_filtered_data_log_data.mep)

### User Actions type

You can show the history of all user actions. The size of the user actions view will be automatically adjusted to the size of the view.



- *Show Top Toolbar* check box allows you to show or hide the top toolbar with filters.
- *Show Bottom Toolbar* check box allows you to show or hide the bottom toolbar (e.g. the date selection controls).
- *Time Scale [min]* parameter specifies the time interval that will be shown in the alarm history table. Units are minutes.
- *Background Color* can be set to transparent or to color. If you set this parameter to color, you can specify which color will be shown as the background.

# 17 Write/Set Command

Watch video describing this functionality:

[https://www.youtube.com/watch?v=7z\\_0\\_DcmvSQ](https://www.youtube.com/watch?v=7z_0_DcmvSQ)

## Writing Values to PLC (or Database)

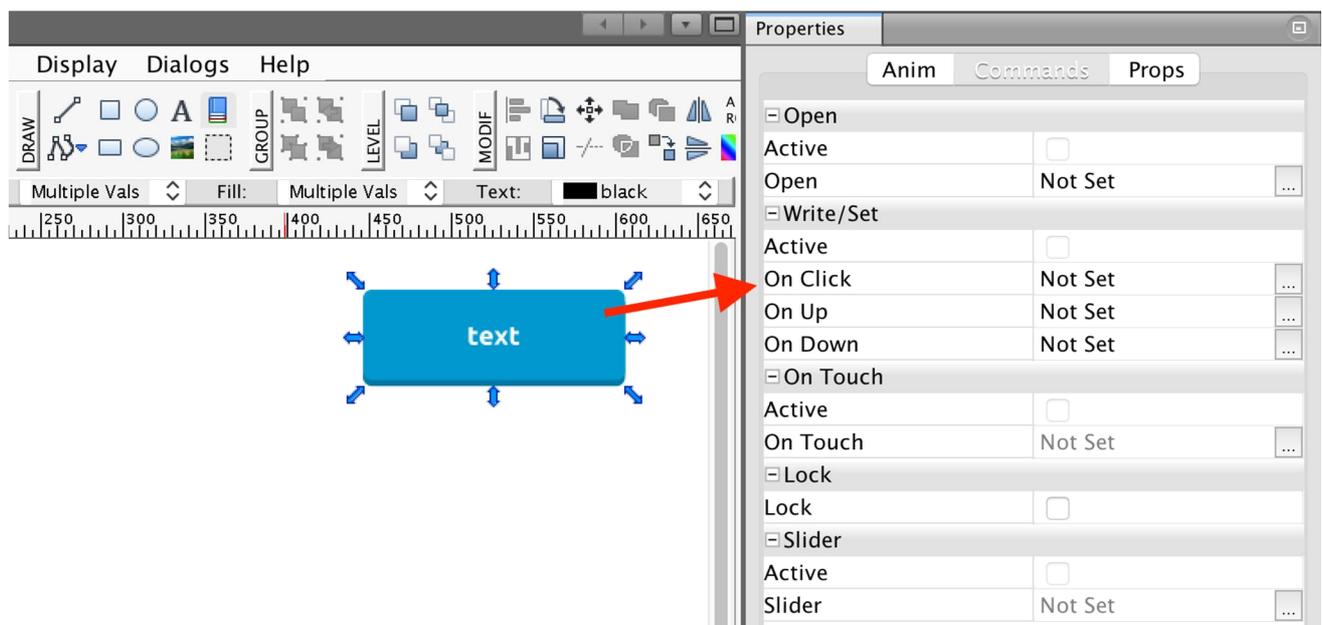
It is possible to configure any graphical object to write new values to the PLC (or database). The ability to toggle a specified bit, write a static predefined value, write any value given by the user, or write any value given from multiple choices from a user-created list are all possible.

There are three types of triggers available for each object, and each of them can write its own dataset. The data can be written using the *on Click*, *on Up*, and *on Down* actions. The *on Up* and *on Down* triggers are independent, so you can write two different sets of data with one button at the same time.

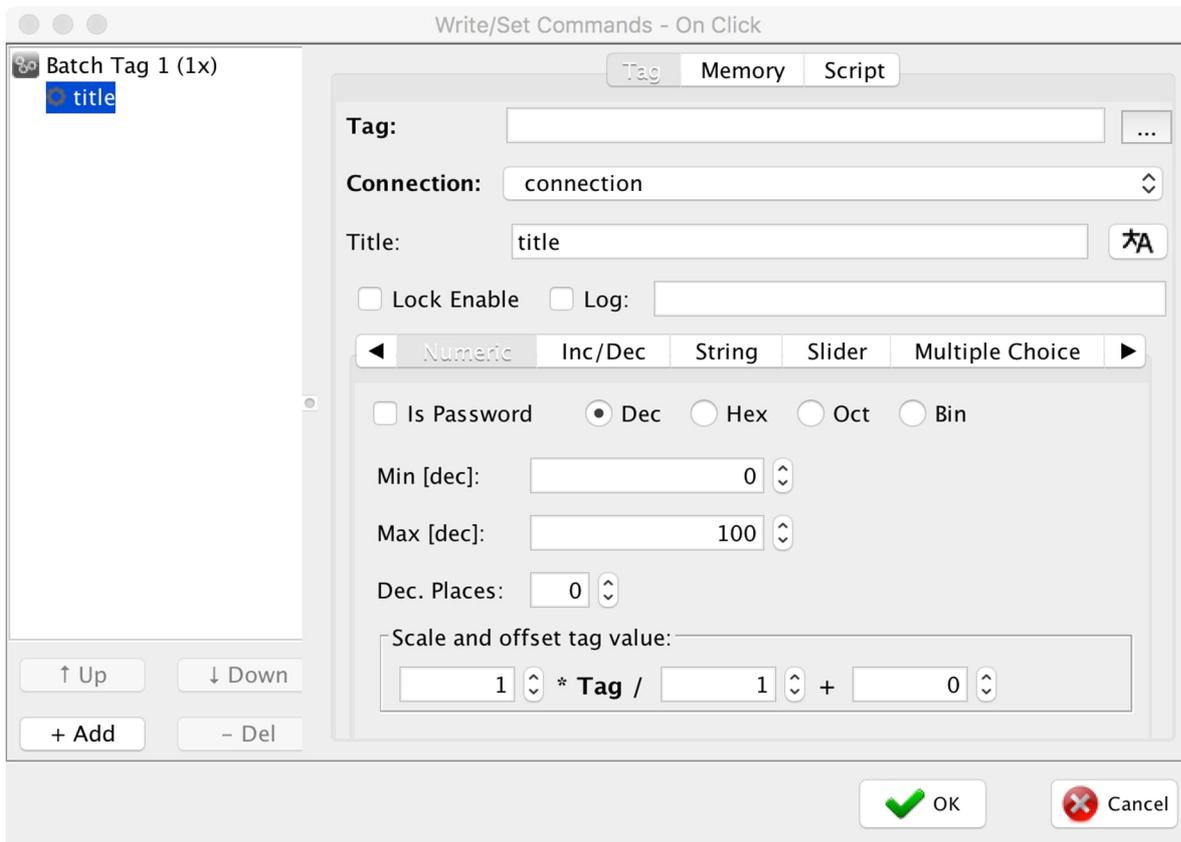
- *on Click* - Write/Set command will be activated when an object is clicked on
- *on Down* - Write/Set command will be activated when an object is pressed (click and hold)
- *on Up* - Write/Set command will be activated when an object is released

All three triggers have the same properties.

To use a write/set command, please select the graphical object you want to use for the set/write command and then navigate to Properties -> Write/Set command.



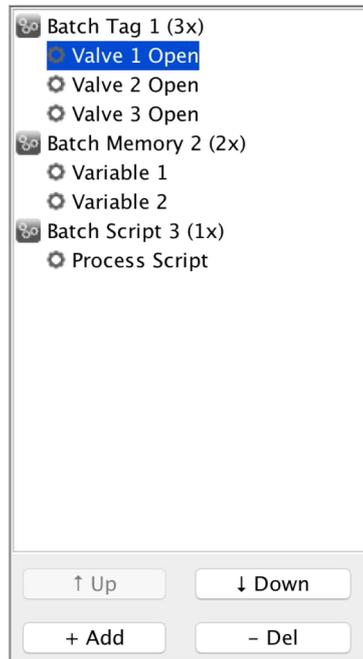
Select either the Click, on Down, or on Up command and click on the “...” button. You will be presented with the Write/Set Settings Dialog:



The dialog window is split into two sections. On the left, there is a Tree View showing all of your set commands in hierarchical order; the right section shows the parameters of the selected set command.

## 17.1 Using Batches

As you can see in the left pane of the dialog window, you can organize your set commands into batches. Each batch is a list of set commands that are processed at once. You can create an unlimited number of batches, and each batch can have as many set commands as you need.



In the batches pane, you can see all your batches listed.

A Batch is:

- A collection of set commands of the same type
- Processed from top to bottom
- Written to the PLC with one command

### Add Batch

To add a new Batch, please click on an existing batch and then click on the button “+ Add”.

### Remove Batch

To remove a Batch, select it and then click on the button “- Del”. All defined set commands in that batch will be deleted along with the batch.

### Add Batch Item

To add a batch item, please select a set command in the corresponding batch and then click on the button “+ Add”. The new batch item is added to the end of the batch list.

## Remove Batch Item

To remove a batch item, please select a set command in the corresponding batch and then click on the button “- Del”.

## Batch Type

Each batch can be of exactly one type. All set command in the batch must be of the same type. Available options are:

Type	Description
Tag	Write values to the PLCs
Memory	Set View Script variables
Script	Run custom function defined in View Script
SQL	Write values to the database

## 17.2 Specifying Set Command Parameters

To specify set command parameters, first, please select the batch in the corresponding batch list. You will see its properties in the right pane of the dialog.

### Tag Type

Select a Tag Type to write values to the PLC.

The screenshot shows a configuration dialog for the 'Tag' type. At the top, there are four tabs: 'Tag', 'Memory', 'Script', and 'SQL'. The 'Tag' tab is selected. Below the tabs, there are several input fields and controls:

- Tag:** A text box containing 'H:0' and a small button with three dots '...' to its right.
- Connection:** A dropdown menu showing 'connection' with a small arrow icon to its right.
- Title:** A text box containing 'Valve 1 Open' and a button with a star icon to its right.
- At the bottom, there are two checkboxes: 'Lock Enable' and 'Log:'. The 'Log:' checkbox is followed by an empty text box.

**Tag** – you can write the tag straight into the *Tag* field; you can also click on the ‘...’ button to prompt the tag database.

**Connection** – set connection to the PLC. Connection parameter is same for all set commands in the batch.

**Title** – Set title for this set command. If you require a response from the user in the form of dialog, the title of the dialog will be this title.



Use translation button to provide a translation for this set command.

**Lock Enable** – check this option if you want to use On Touch Lock for this element.

**Log** – check this check box to log this user action. The user action will be logged with the provided message.

## Memory Type

Select a Memory Type to set View Script Variables.

**Variable** – specify a View Script variable to set the value to.

**Title** – Set title for this set command. If you require a response from the user in the form of dialog, the title of the dialog will be this title.



Use translation button to provide a translation for this set command.

**Lock Enable** – check this option if you want to use On Touch Lock for this element.

**Log** – set this check box to log this user action. The user action will be logged with the provided message.

## SQL Type

Select SQL to write values to the database:

**Connection** – set connection to the database. Connection parameters are the same for all set commands in the batch.

**SQL** – specify SQL that will be used to write values to the PLC. The SQL parameter is the same for all set commands in the batch. When the batch is processed, all values from the batch are inserted into the specified SQL. The top set command in the batch will replace **#1#** keyword in the SQL; the second set command in the batch will replace **#2#** keyword, and so on.

**Title** – Set title for this set command. If you require a response from the user in the form of dialog, the title of the dialog will be this title.



Use translation button to provide a translation for this set command.

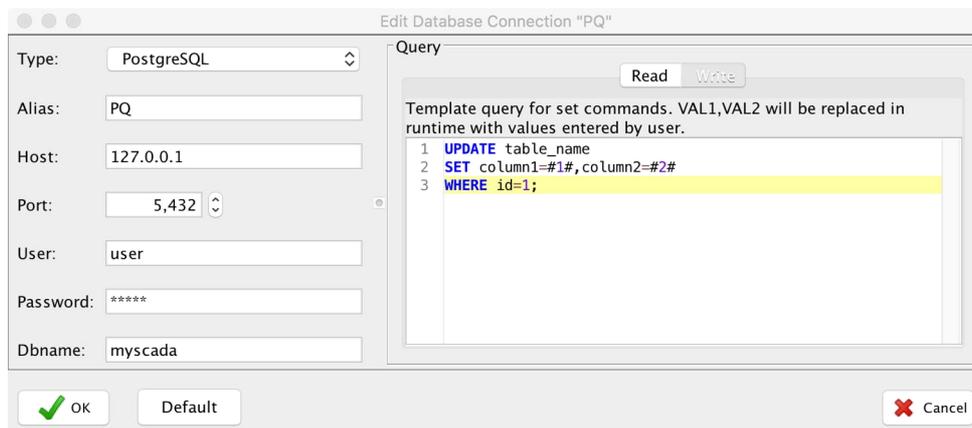
**Lock Enable** – check this option if you want to use On Touch Lock for this element.

**Log** – check this check box to log this user action. The user action will be logged with the provided message.

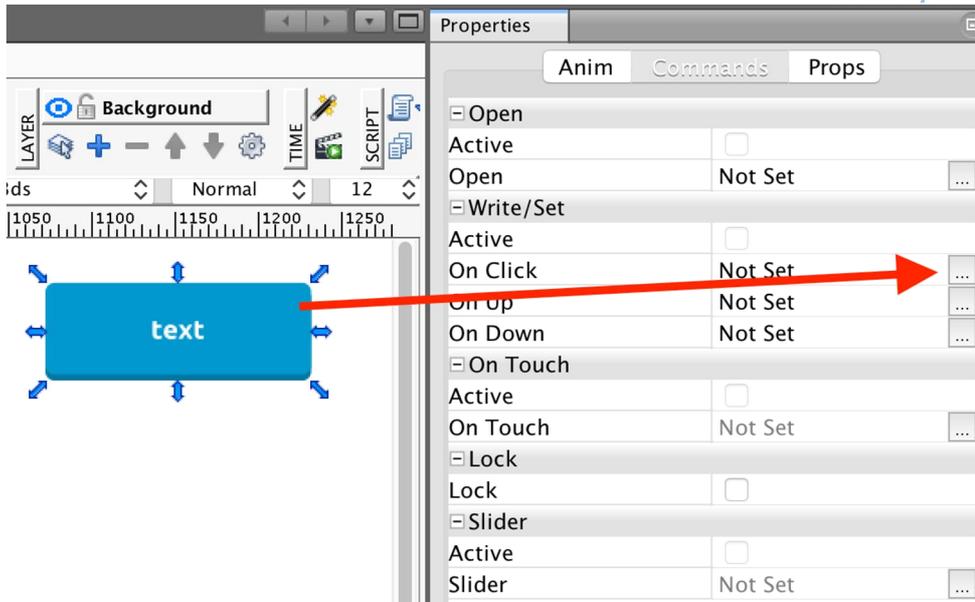
### Multiple Values Write Example

The following example will explain how to write multiple values into the SQL database with one batch set command. We will ask the user to enter two values which will be then written to the database using the SQL command.

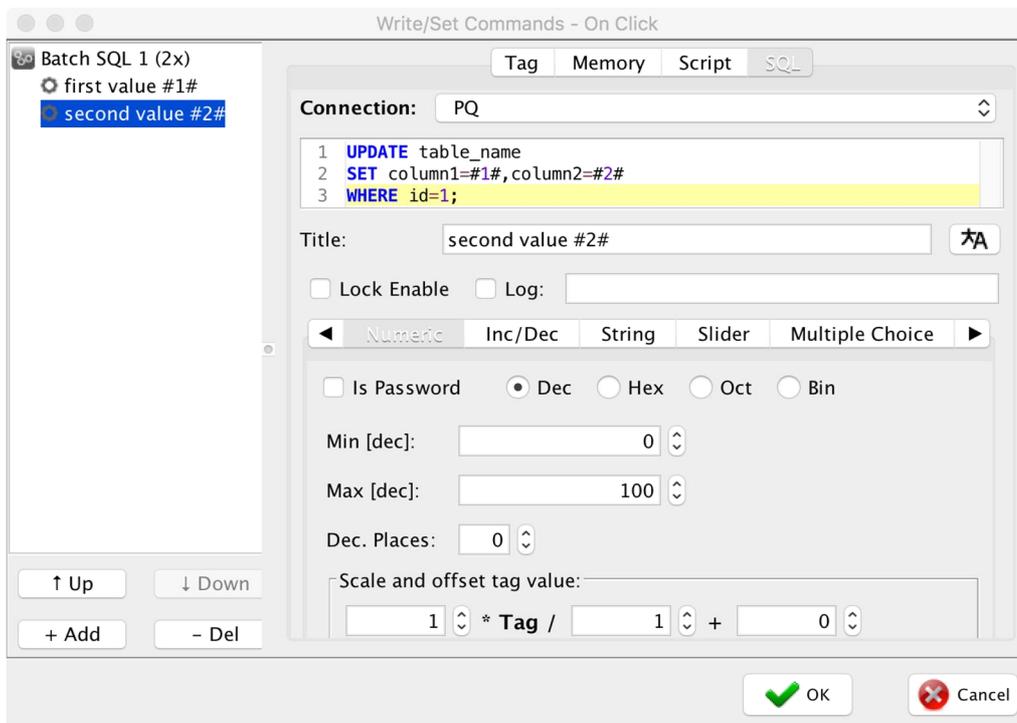
1. Create a connection to your database.



2. In your view, navigate to the button element and in Properties -> Commands, select Write/Set Command.



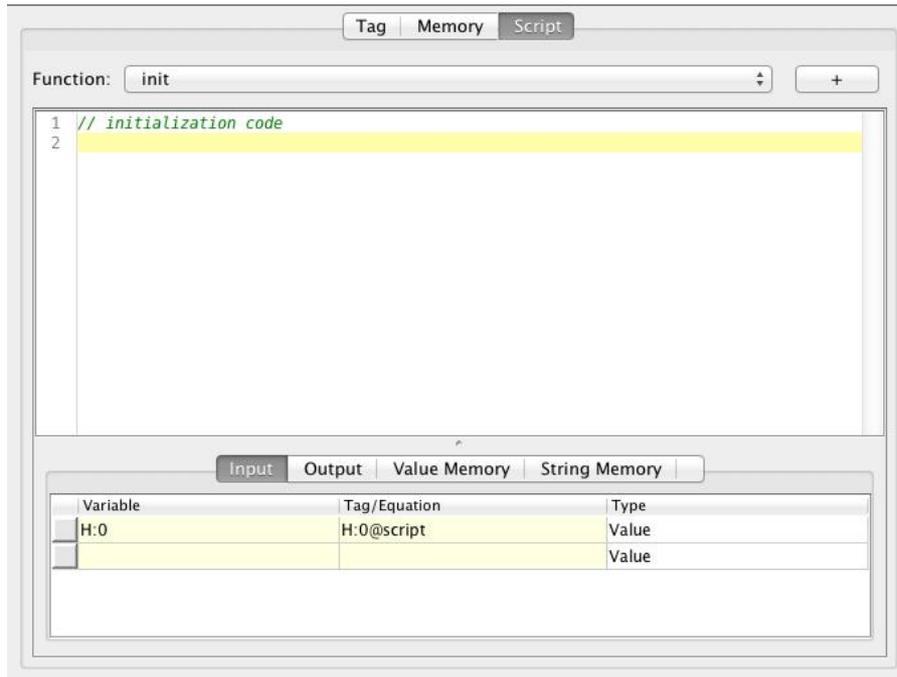
- Now in the set dialog, create two items in one batch like in the following picture:



- Now test your project. If you click on the button, the user will be asked twice to enter the value. Once the user has entered both values, mySCADA will execute the SQL. #1# will be replaced with the first value; #2# will be replaced with the second value.

## Script Type

This very useful feature allows you to oversee, set up, and enrich the whole *Write/Set* command functionality from the scripting level (i.e. you can freely create and add your own new arbitrary functionalities). When the set command is processed, the corresponding View Script is called.

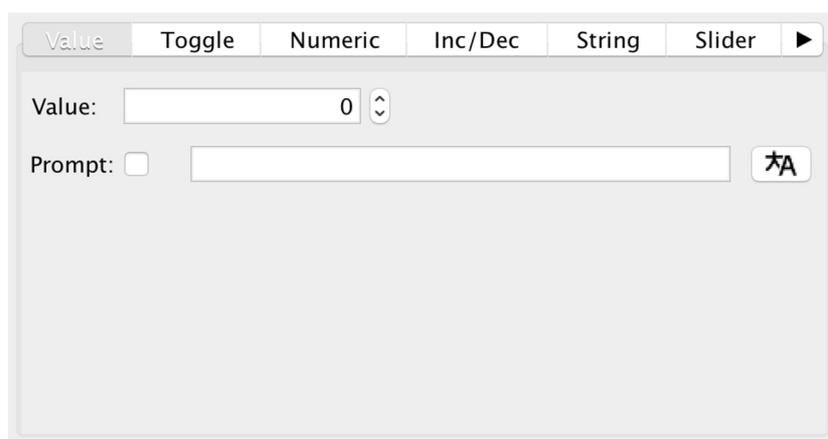


For a detailed description of how View Scripts work, please see section [View Scripts](#).

### 17.3 Value Options

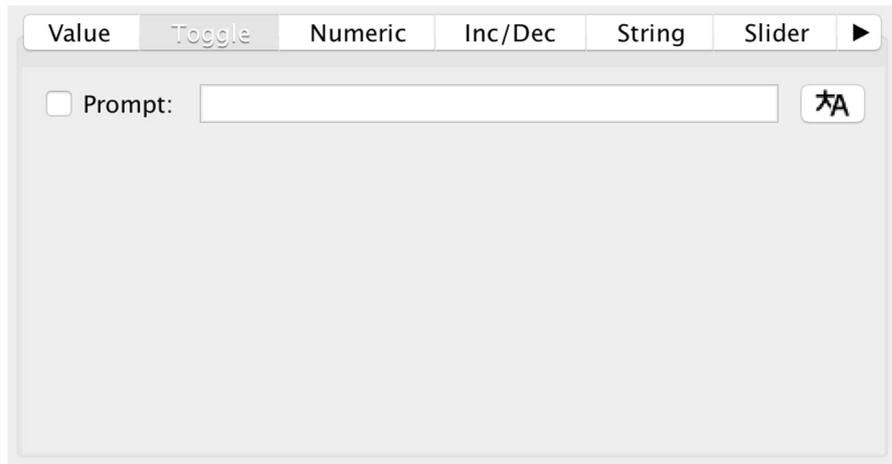
When writing values to the tags, memory variables, or the database, you must specify the value options. The following is a description of all possibilities.

**Value** - this option sends a static value to the PLC



Please specify a value to send to PLC. It can be any numeric value (integer or floating point). If you check the Prompt option, the user will be presented with a confirmation dialog. The value will be written only after the confirmation.

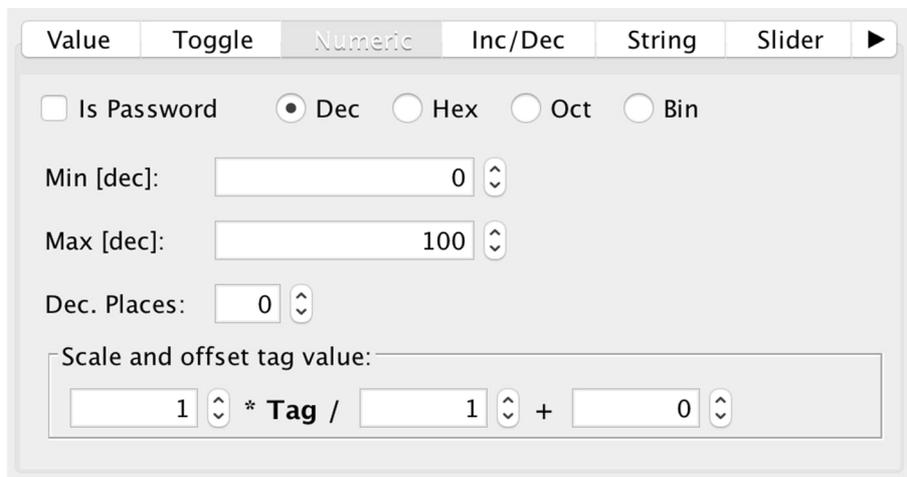
**Toggle** – binary switch; sets 0 to 1 and 1 to 0; should NOT be applied to non-binary



variables.

If you check the Prompt option, the user will be presented with a confirmation dialog. The value will be written only after the confirmation.

**Numeric** – prompts a dialog in the visualization where the user can enter any numeric value.



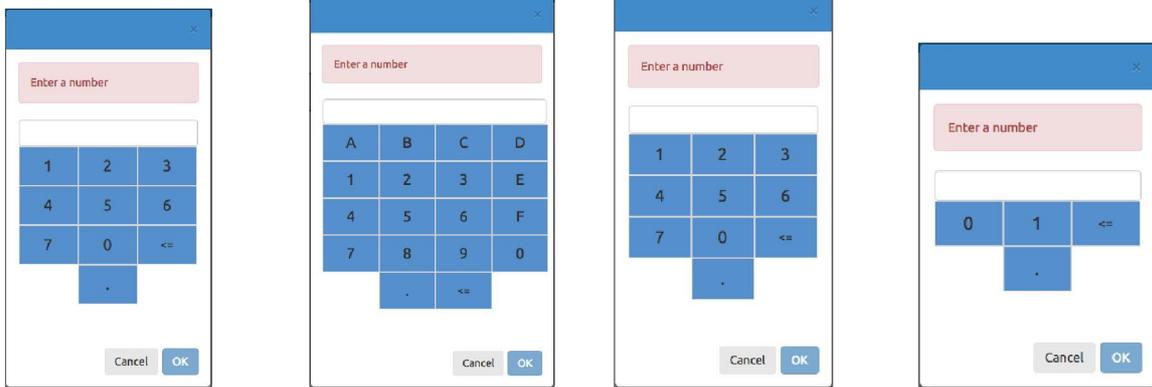
There are options to treat such dialogs as password input (hiding input values) and to set the allowed range of inputs with **Min** and **Max** limits and a number with decimal places. You can choose from four numeral systems:

*Decimal (Dec)*

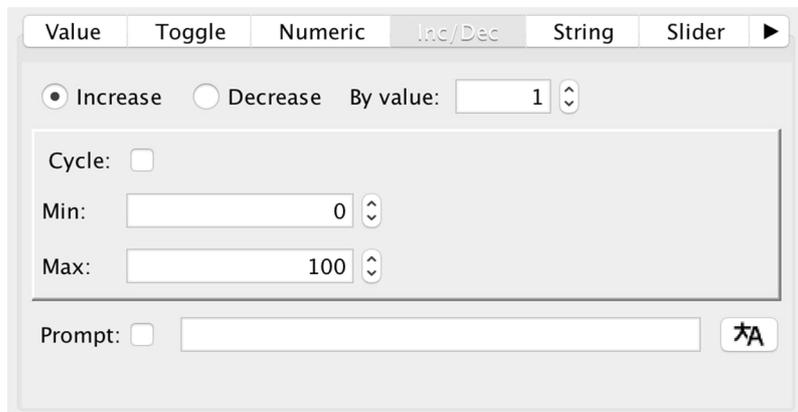
*Hexadecimal (Hex)*

*Octal (Oct)*

*Binary (Bin)*



**Inc/Dec** – increases or decreases PLC value by the set level:



If you select the *Cycle* option, the increasing / decreasing process will repeat. If the value goes over the **Maximum** value, it will automatically change to the minimum value. If the value goes below the **Minimum** value, it will automatically change to the maximum value.

If you check the *Prompt* option, the user will be presented with a confirmation dialog. The value will be written only after the confirmation.

**String** – prompts a dialog in the visualization where the user can enter any string value with other options such as password input (for hiding input values) and sets an allowed number of characters with **Min** and **Max** limits; predefined text can also be set with the **Predef. Text** option. You can also provide a translation of the predefined text by clicking on the translation button next to the text entry.

The dialog box has tabs for Value, Toggle, Numeric, Inc/Dec, String (selected), and Slider. The String tab is active, showing the following options:

- Type: Automatic (dropdown), Is Password (checkbox)
- Min No of Char: 1 (spin box), Length: 1 (spin box)
- Predef. Text: (text input field) with a star icon button

User Text Entry dialog:

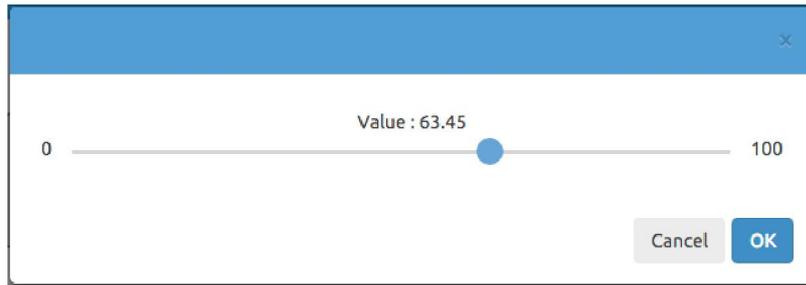
The dialog box has a blue title bar with a close button. A red error message box is displayed at the top, stating: "Text length must be between 1 and 31 characters". Below the message is a text input field. At the bottom right are "Cancel" and "OK" buttons.

**Slider** - prompts a dialog in the visualization where the user can select any numeric value with a slider; additional options allow setting **Min** and **Max** outer slider value limits and the number of decimal places with **Dec. Places**

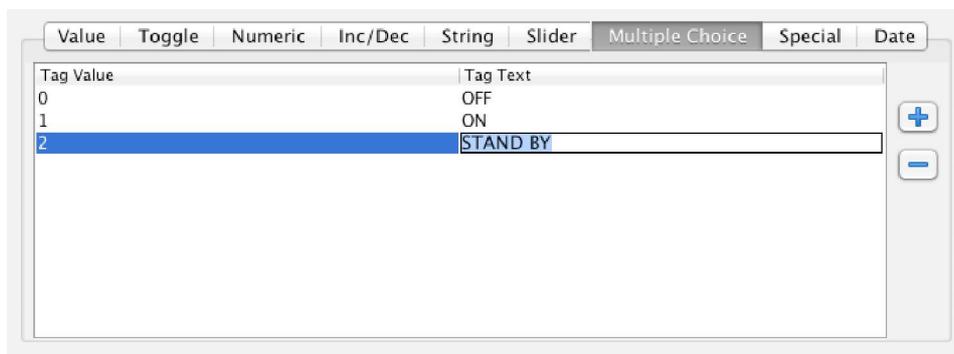
The dialog box has tabs for Value, Toggle, Numeric, Inc/Dec, String, and Slider (selected). The Slider tab is active, showing the following options:

- Min: 0 (spin box)
- Max: 100 (spin box)
- Dec. Places: 0 (spin box)
- Scale and offset tag value:  $1 * \text{Tag} / 1 + 0$  (formula input field)

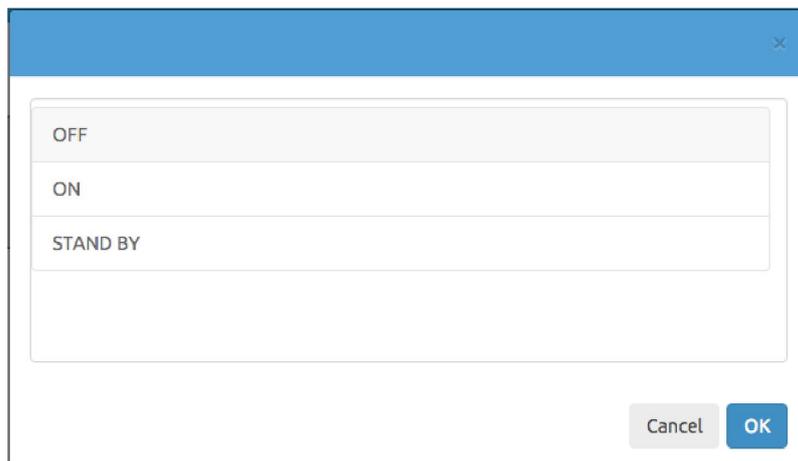
Slider set dialog:



**Multiple Choice** – prompts a dialog in the visualization where the user can select one of the predefined values, which should be entered as tag value/text pairs during the animation setting



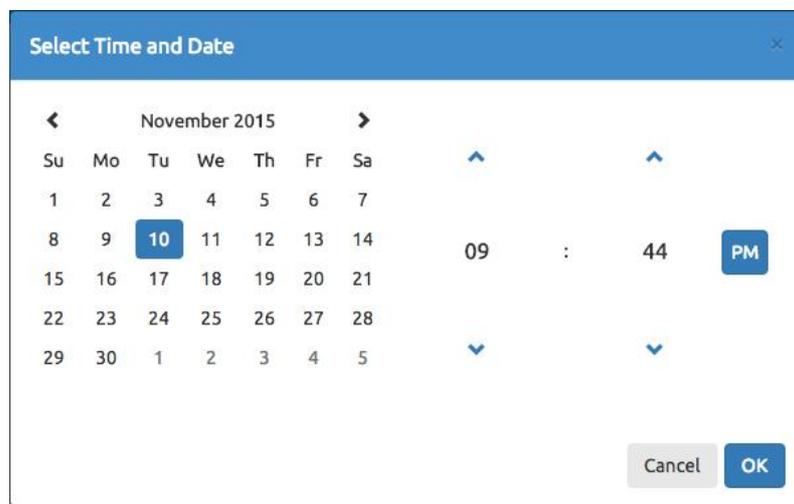
Multiple choice set dialog:



**Date** – with this feature you can write the current date, time value, or both (date + time) in the visualization



Date set dialog:



# 18 Scaling Set Values

The *Numeric* and *Slider* options have both settable scale and offset values being sent to the PLC.

Scale and offset tag value:  $1 * \text{Tag} / 1 + 0$

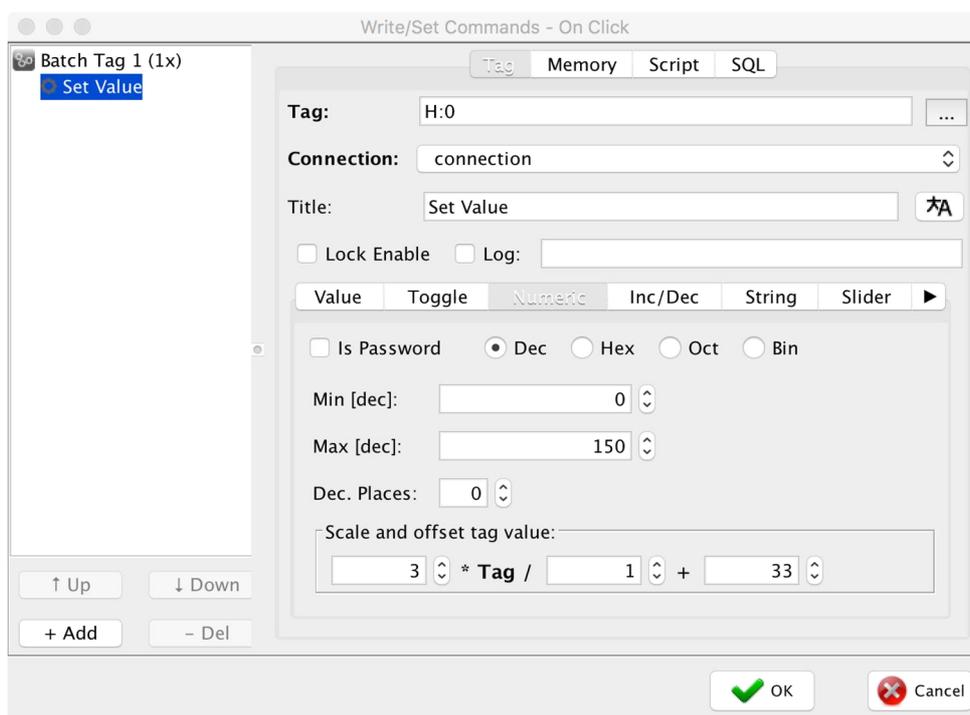
Usage is quite straightforward: imagine you want the user to enter a value by percent, e.g. between 0 and 100, but you need to have the value written to the PLC as a floating-point value in the range 0 to 1. In this case, you would set the scale as follows:

Scale and offset tag value:  $1 * \text{Tag} / 100 + 0$

## 18.1 Set Example:

The picture below shows the configuration setup so that the *on Click* command prompts the write dialog where the user can write a value in the range of 0 to 150.

The resulting value will be multiplied by 3 and increased by 33 before writing to the PLC as a result of the **Scale/Offset** addition.



Additionally, you can add a *Title* text, which will display during visualization in the dialog window when using the *Write/Set* command. If you want no confirmation message to be displayed upon the write action, unselect this field.

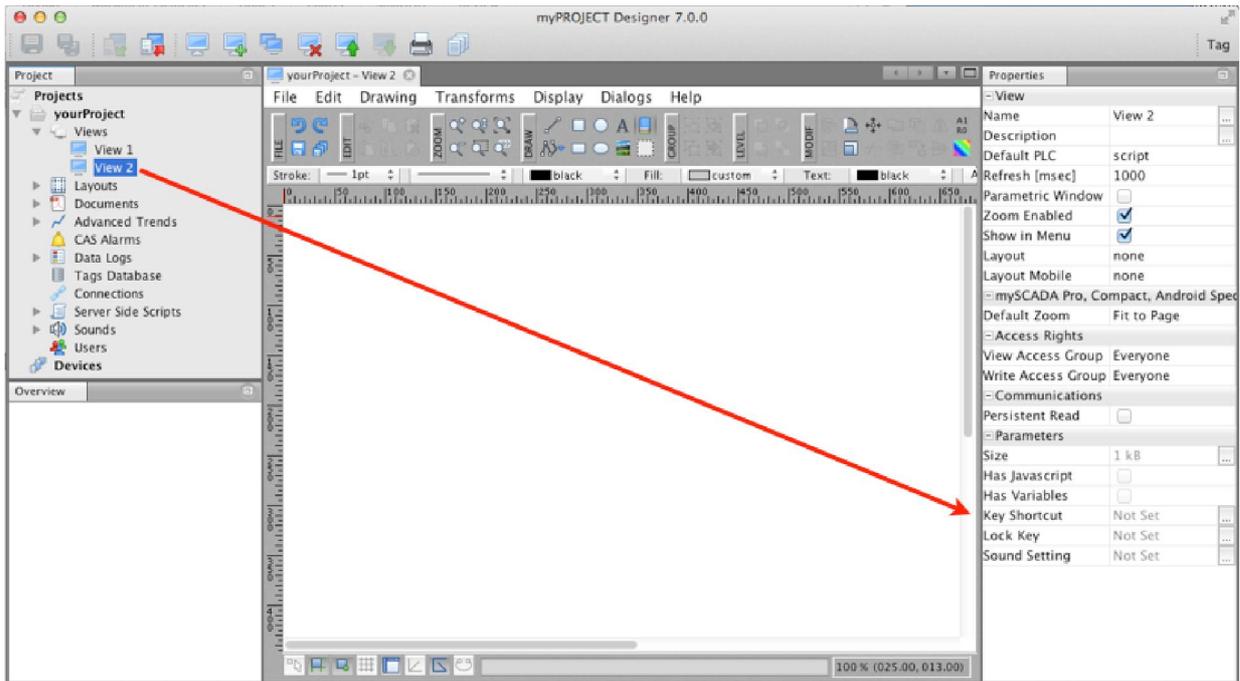
If you want to log the write command into the data-log, check the *Log* check box and fill in the message field.

# 19 Key Shortcuts

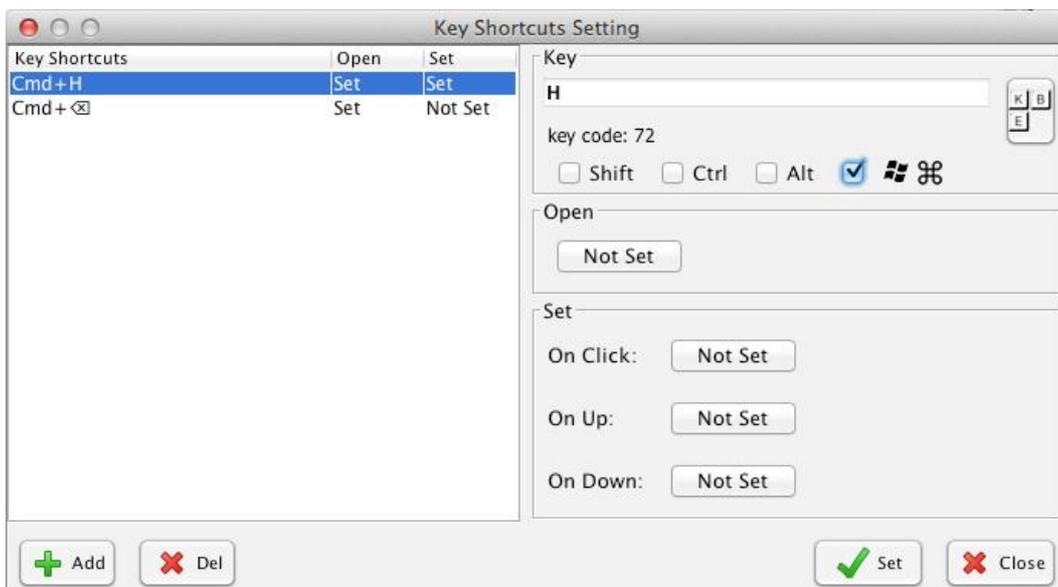
This feature is a very useful part of the *Commands* and allows you to prompt certain functions using set key shortcuts. On the key press, you can define write/set command or open command. This way, you can change values in the PLC or open another view, just using a simple press of a button.

To apply a Key Shortcut to your view, do the following:

- 1) Select the view you want to apply the shortcuts to and navigate to the section *Parameters* in the *Properties* window.

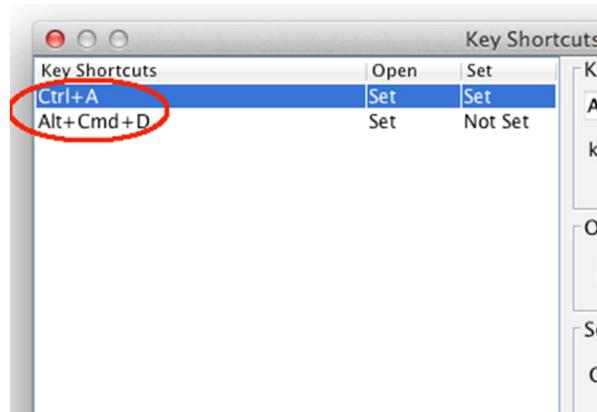


- 2) Click on the '...' icon to open the settings dialog window where you can set arbitrary key shortcut combinations and assign the commands to them.

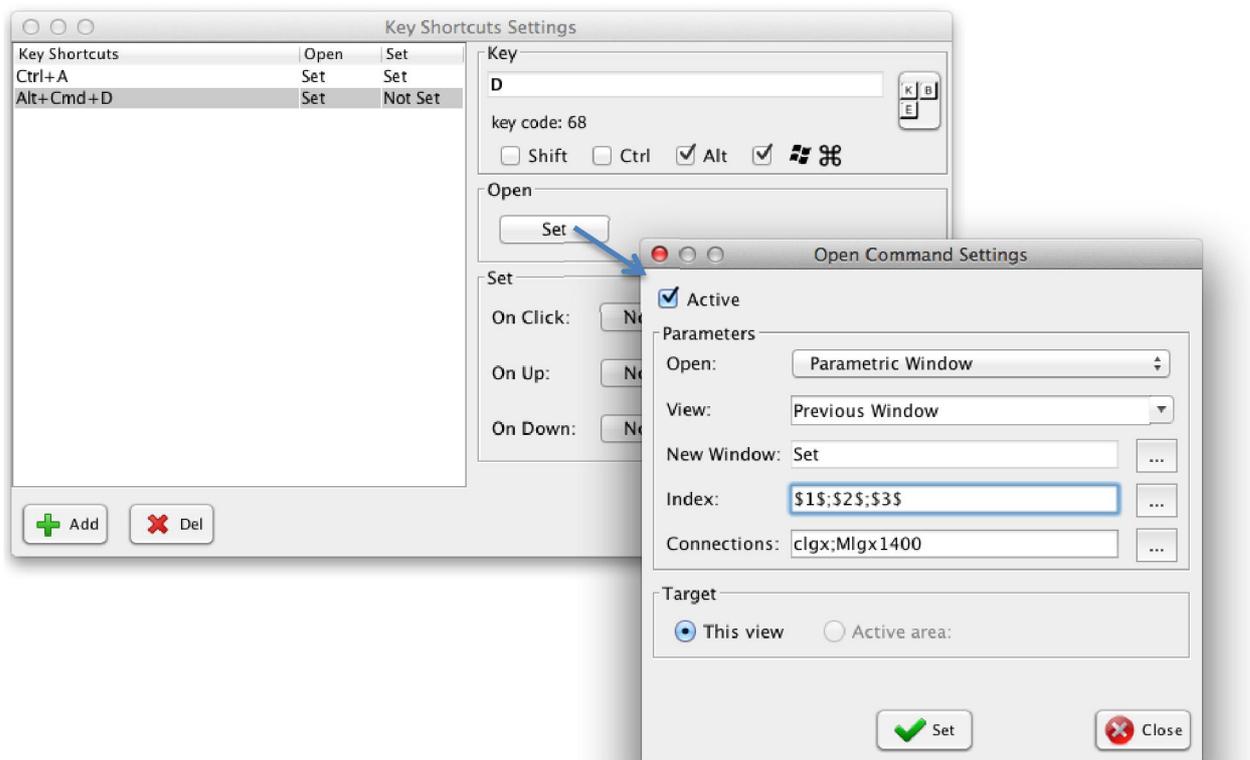


- 3) Click on the **Add** button and select the functional keys you want to use for the shortcut (*Shift, Ctrl, Alt* etc.), then click the *Key* box (or click on the 'KBE' button) and press any arbitrary key on the keyboard (letters or numbers).

You will see a review of the shortcut on the left side:



- 4) Now select the type of command the shortcut should prompt (*Open* or *Write/Set*) by clicking on **Set** and set the properties of the command in the dialog window as described earlier.



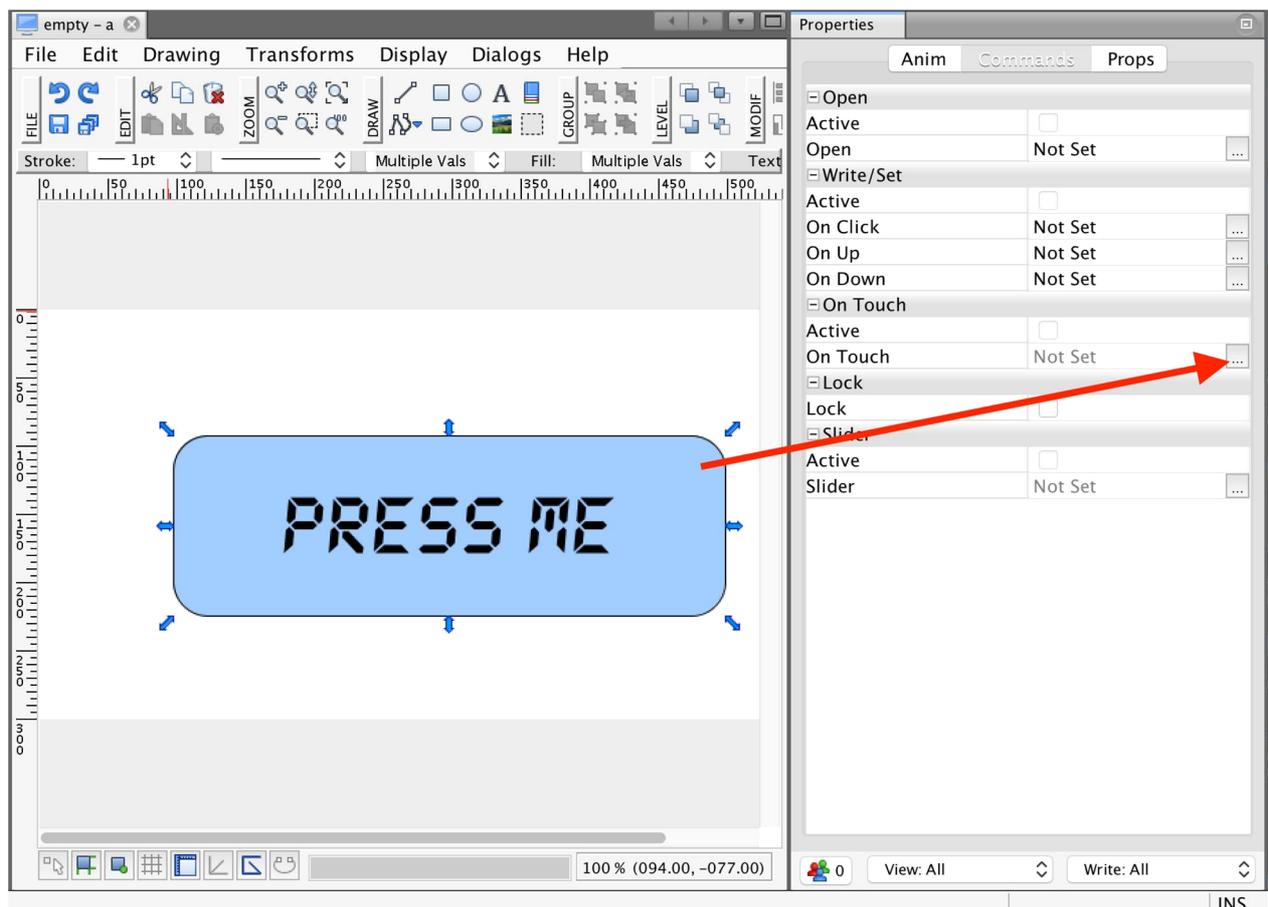
## 20 On Touch

This function brings user interactivity to your elements. It can change the object properties like fill and stroke color or visibility when you touch / release the element in the run-time.

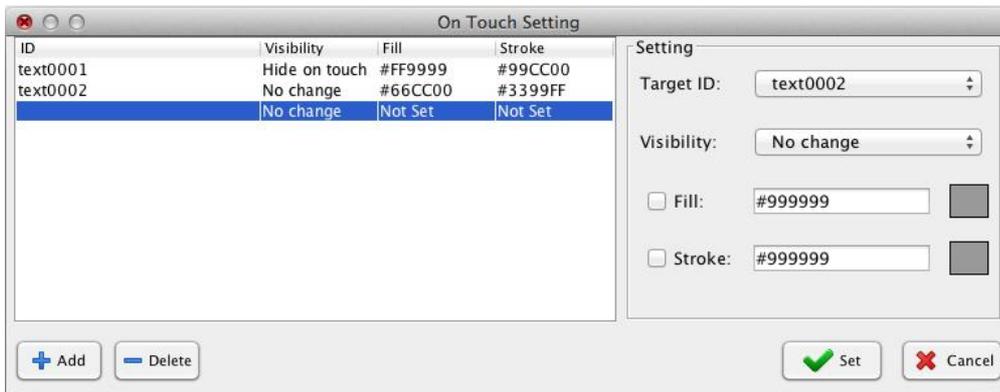
You can create interactive buttons where the user is notified of the press by a color change. In addition, you can create more complex scenarios like showing/hiding an object border and shadow when touched.

### 20.1 On Touch in Views

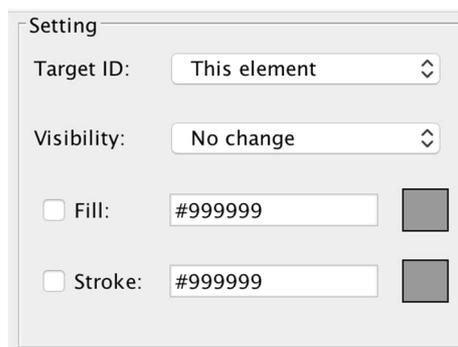
To use an On Touch event on elements in View, please select the element you want to use as your On Touch trigger. Then navigate to Properties -> Commands. Now click on the On Touch “...” button.



You will be presented with a new dialog window:



In this dialog, you can add multiple graphical objects and change their properties. To do so, click on the “+ Add” button. A new entry will be created. Please now look at the Settings section on the right side of the dialog:



The first option is the “Target ID”. The target ID specifies which graphical object you wish to apply settings to during the touch event. If you want to apply settings to the same element as your graphical object, please leave the option **This Element**.

Next, parameters control visibility, Fill, and Stroke color of the given object.

As you can see, you can create a simple scenario like changing the fill color of an object during a touch event, or you can create a complex scenario involving multiple different objects.

## 20.2 On Touch in Component

For the component, the on touch event is one for all items in the group. This is because each component is encapsulated in a group. If the user clicks on the group, all On Touch actions inside the group (all items in the component with set action On click) will activate.

To use On Touch animation in Component, please click on the component and navigate to the Properties -> Commands. Now click on the On Touch “...” button. You will be presented with the settings dialog:

Visibility:	No change	⌵
<input type="checkbox"/> Fill:	#999999	
<input type="checkbox"/> Stroke:	#999999	

Set visibility, fill, or stroke change for the selected element.

*TIP: If you want to modify multiple elements during the On Touch action, simply set the On Touch parameters for each of them.*

## 20.3 On Touch Example

The following Example will show you how to create a button component with an On Touch action.

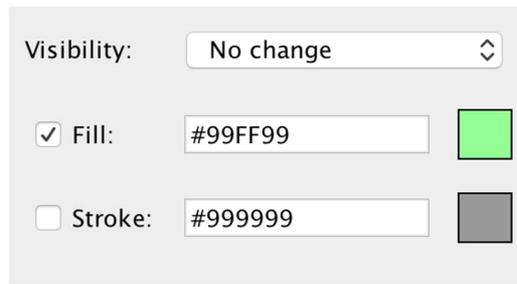
1. Create a new component and enter edit mode. If you don't know how to create components, please read the section [Components](#) first.
2. Now create a rectangle and put the text "Press me" over it.



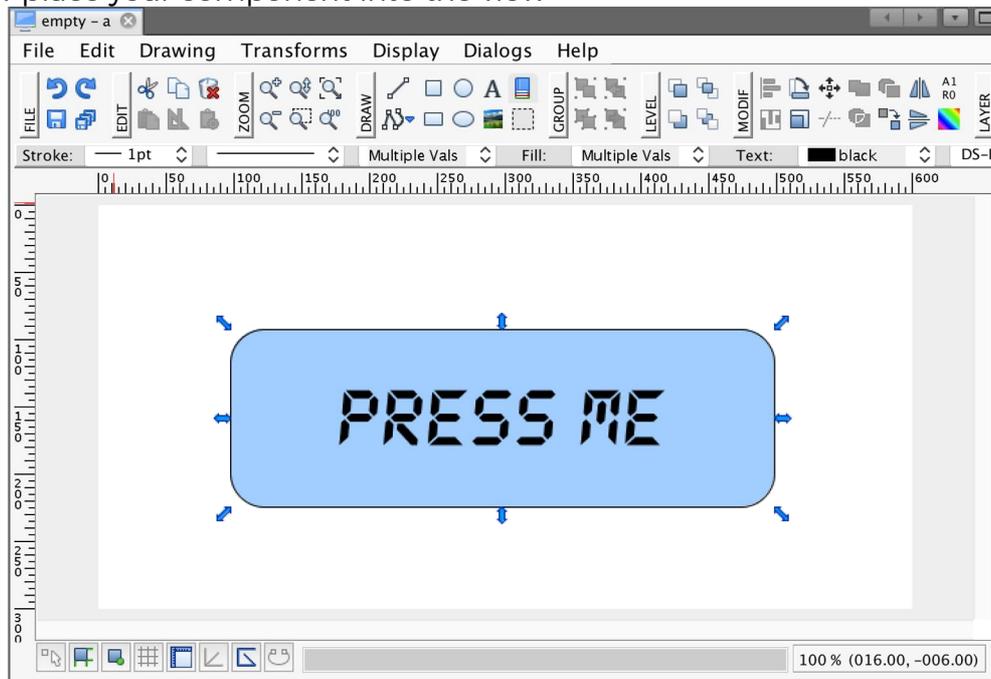
3. Select the rectangle and click on Properties -> Commands -> On Touch

Visibility:	No change	⌵
<input type="checkbox"/> Fill:	#999999	
<input type="checkbox"/> Stroke:	#999999	

4. Check the "Fill" check box and select a fill color



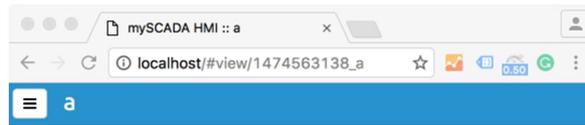
5. Now place your component into the view



When a user presses the button, its color will change to green



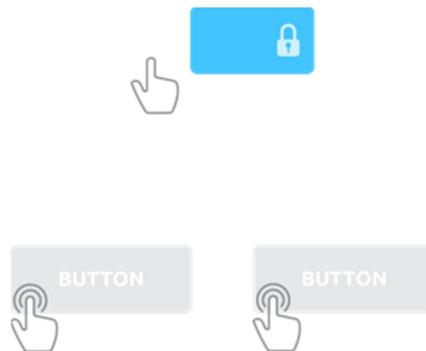
When the user releases the button, the color will return back to normal.



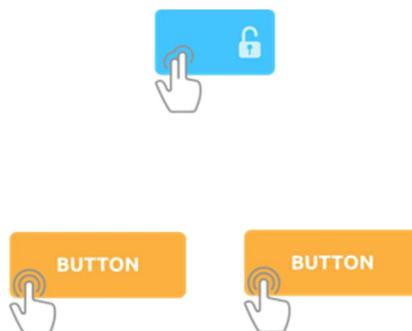
## 20.4 Lock Element

Lock, if applied, can turn any element of your graphics into a lock key. You can use Lock to disable accidental write/set commands from being executed. If enabled, the user must first press and hold the Lock element and then press the element with the set command.

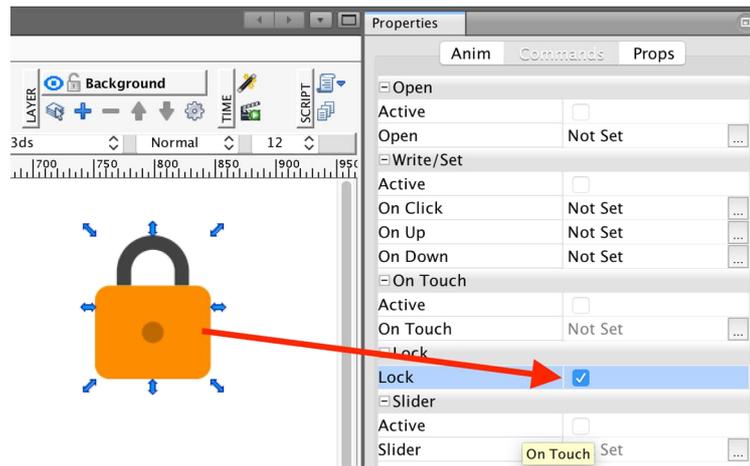
Lock element is not pressed = user cannot press the button.



Lock element is pressed = user can press the button

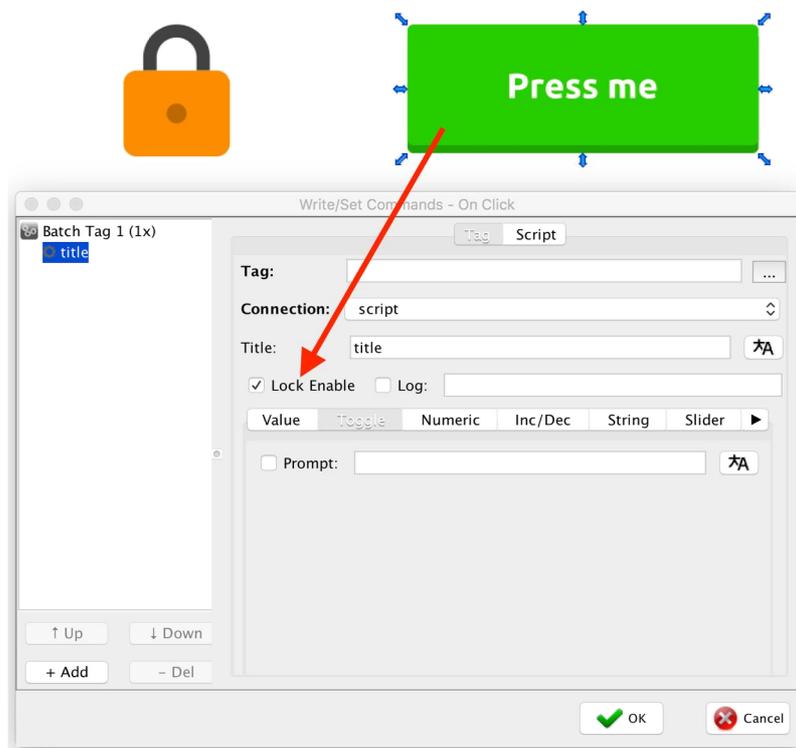


If you want to apply this function, select the object you want to set as a lock, navigate to the **Commands** tab of the *Properties* window, and check the box in the *Lock* section.



*Note: This function is only usable on multi-touch devices.*

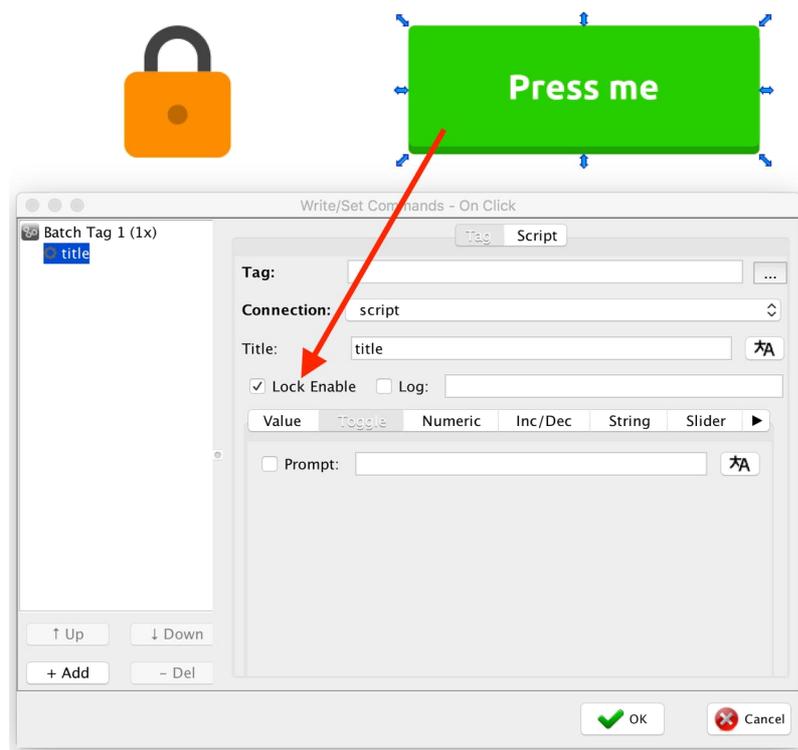
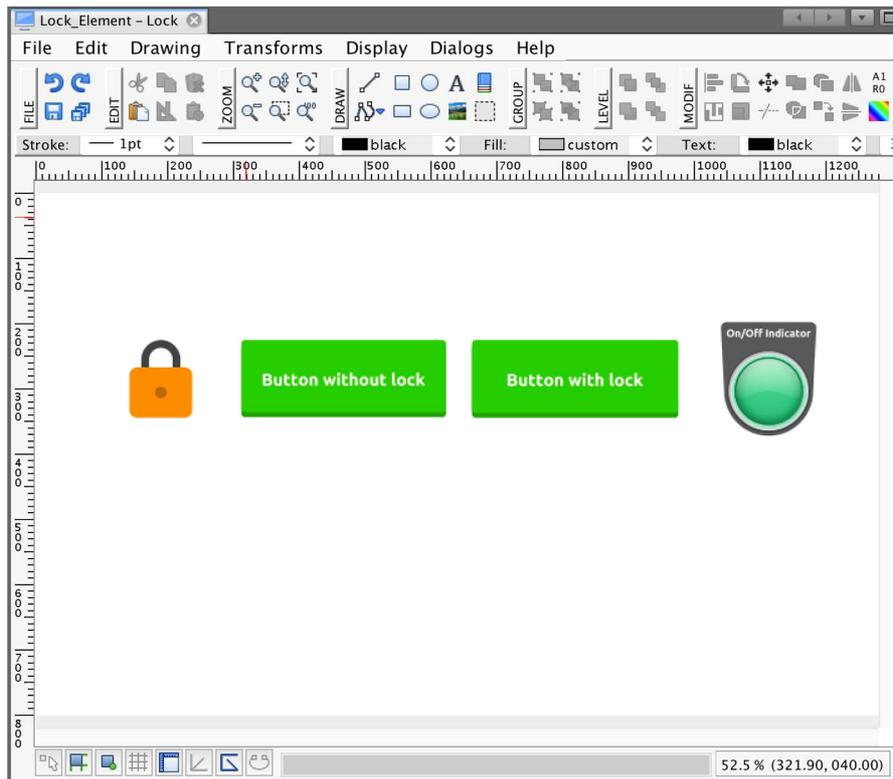
Once you have set up your Lock element, you can use it anywhere in the Write/Set Command. Simply apply the Write/Set Command to any element and check the "" check box.



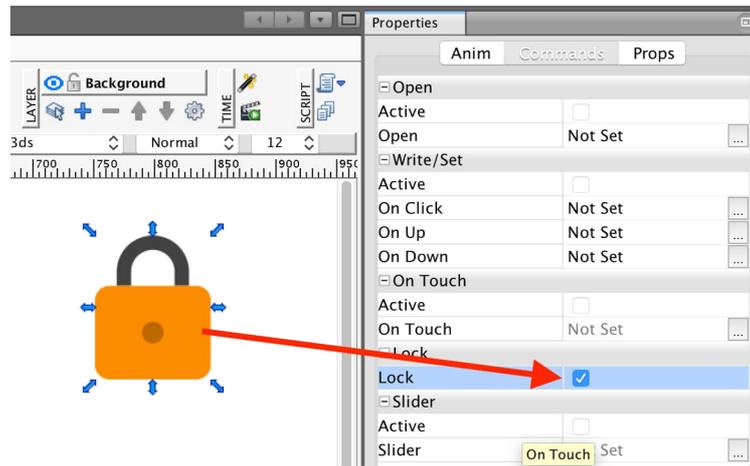
## Example

In the following example, we will create a simple view with a lock element and a button. The button will be enabled only if the user presses and holds the lock element simultaneously.

1. Create a new view and insert the Lock icon and a button. You can also insert an indicator to see if your button is working



2. Now enable the Lock command on the Lock icon



3. Now navigate to the Write/Set command and create a toggle action. Check the “**Lock Enable**” check box.
  
4. Now when you open your view, you will be able to press the button only if you simultaneously press the Lock icon.

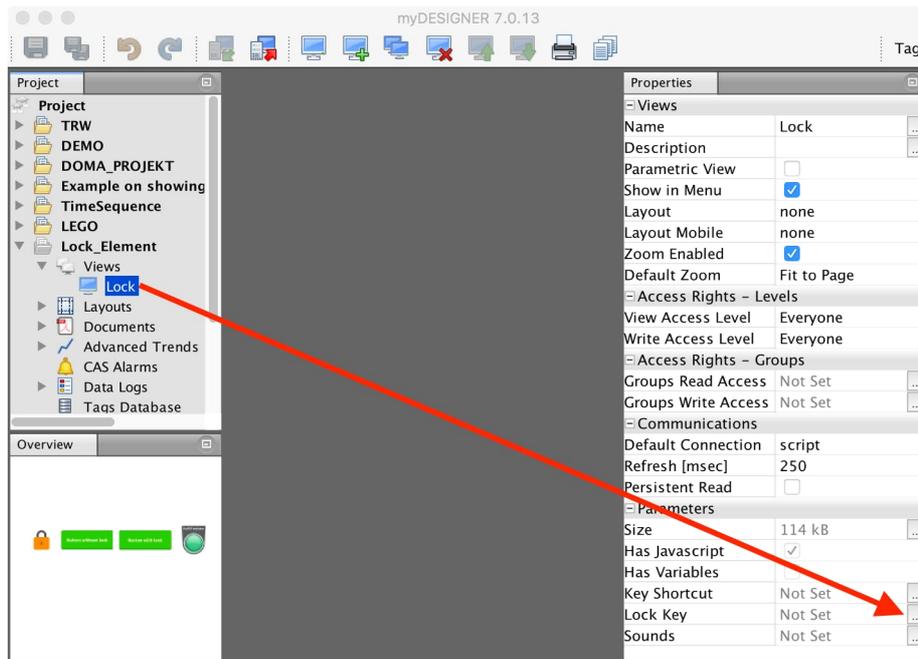
DOWNLOAD DEMO PROJECT HERE:

<ftp://nsa.myscada.org/history/projects/example/LockEnable.mep>

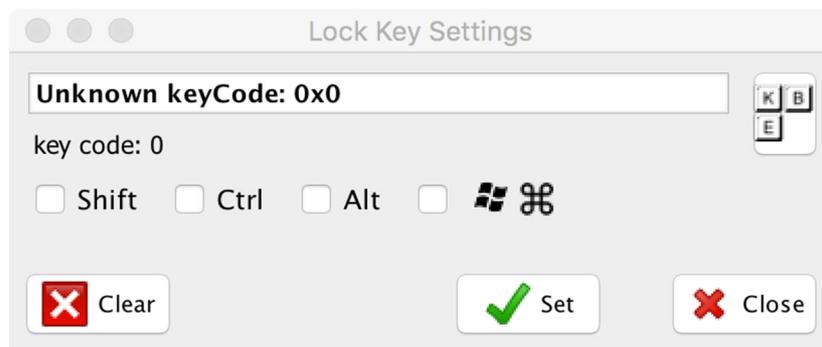
## 20.5 Lock Key

The Lock Element can be very useful to avoid accidentally pressing buttons. However, it is only suitable for multi-touch devices such as tablets or mobile phones. On a regular computer, you can use the analogous function called **Lock Key**. The principle is the same as for a Lock Element; however, instead of selecting a graphical object as your lock element, you select a key that must be pressed to enable the write/set commands.

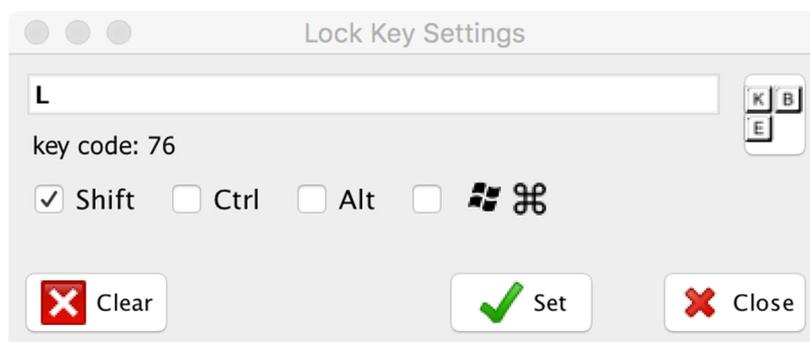
To enable this feature, please navigate to your view in the Project tree and press the **Lock Key** in the Properties window.



You will be presented with a new dialog window where you can register your Lock Key:



Now press the “KBE” button to register your new Lock Key. Now press the key you would like to use as your Lock Key. Your new Lock Key is registered.



Once your Lock Key is registered, you can work with your Write/Set commands. The usage is the same as described in the previous section.

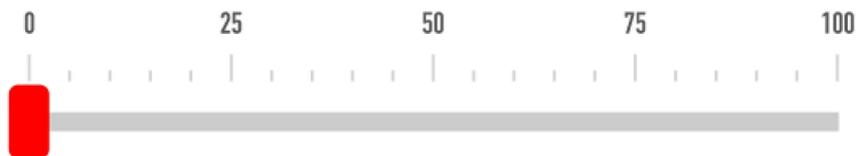
DOWNLOAD DEMO PROJECT HERE:

<ftp://nsa.myscada.org/history/projects/example/LockEnable.mep>

The right button will be unlocked by pressing and holding the space button.

## 20.6 Slider

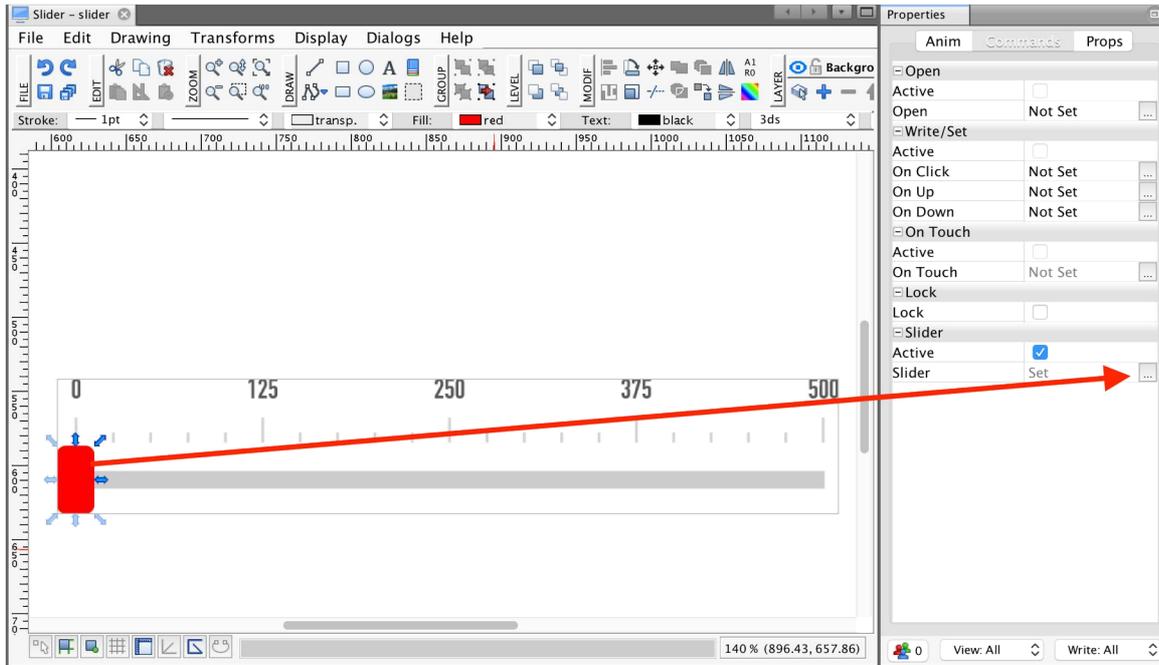
This is another possibility for writing arbitrary values to the PLC. You can create a slider control, enabling the user to move the slider and write values to the PLC. The slider can be a simple object as shown below, or it can be a complex scenario like moving items on a conveyor. Any graphical object on your screen can act as a slider.



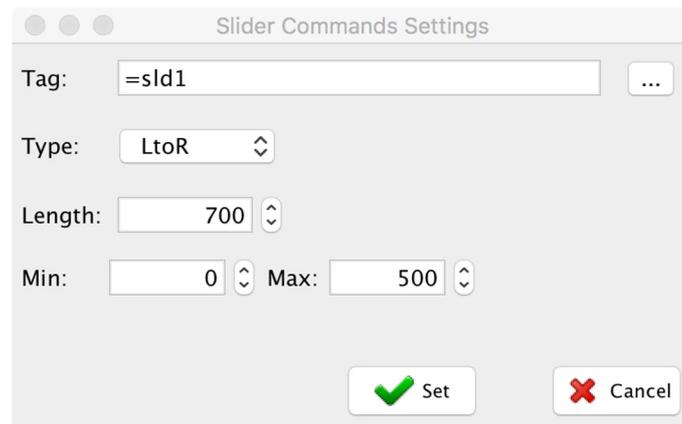
Some example slider controls:



To activate this control, select your graphical item, navigate to the *Write/Set* command options in the *Properties* window, and select *Slider*.



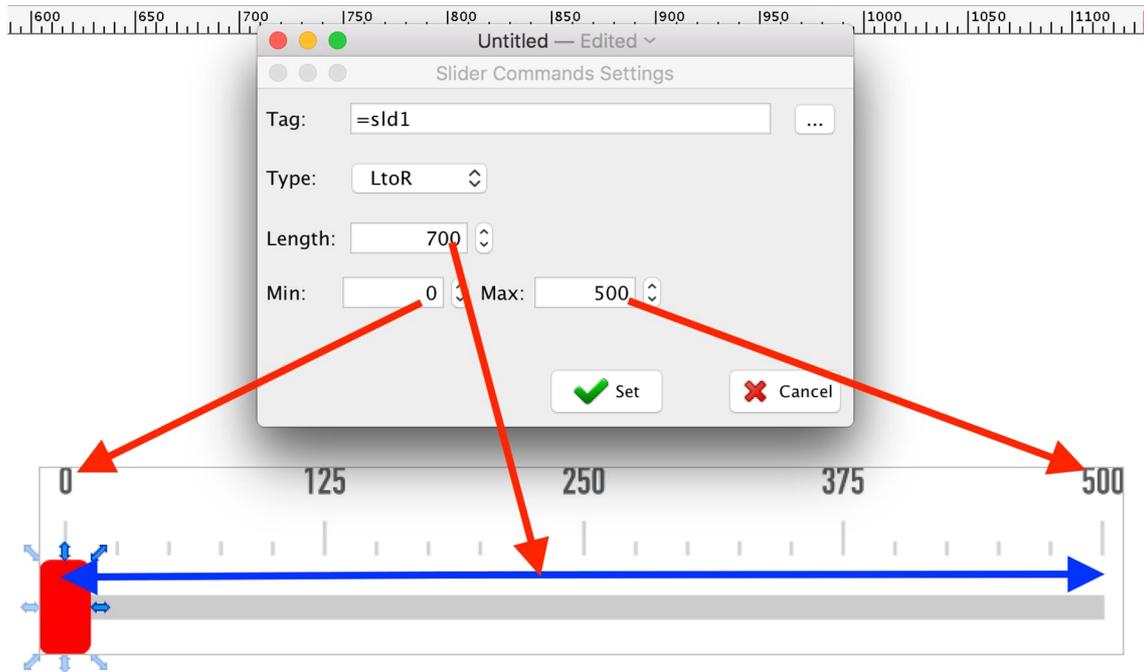
You will be presented with the settings window:



In the dialog window you have to set:

- **Tag** – either manually or from the tag database by clicking on the ‘...’ button.
- **Type** – you can choose the orientation of the slider from Left to Right, Top to Bottom, or vice versa
- **Min and Max** – range of Tag value
- **Length** – the length of the slider movement in pixels

The following picture demonstrates how parameters correlate to one another:

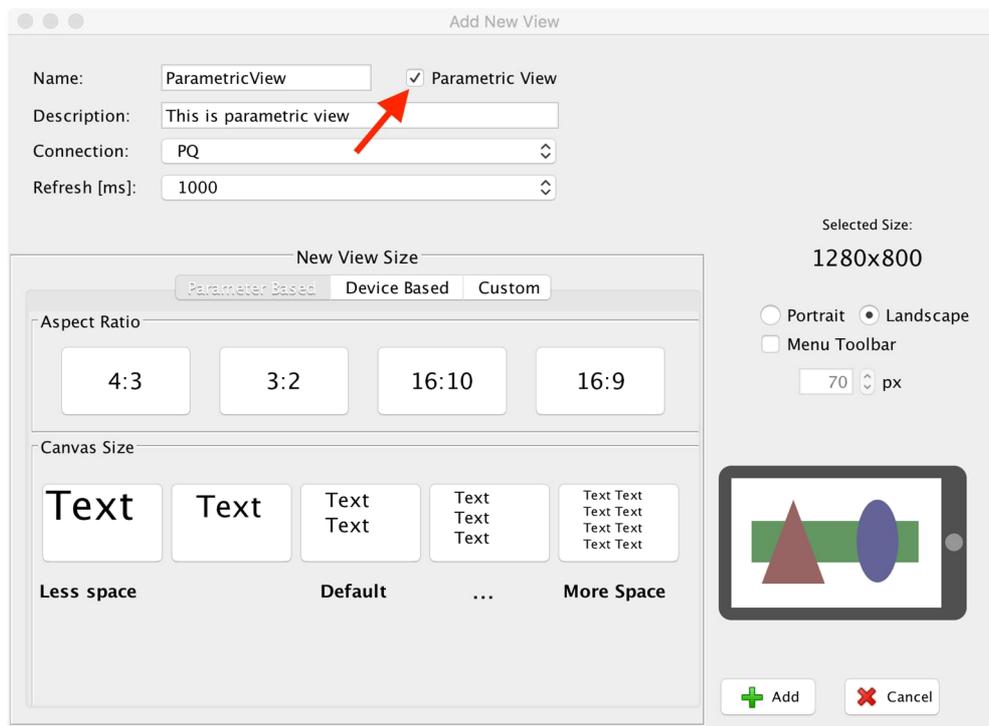


# 21 Parametric Views

Instead of designing multiple similar visualizations of multiple field technology that differentiates only in operational parameters and not in appearance, you can use one master view that is common for all. You can set and use operational parameters to link the views with specific pieces of technology.

Watch video describing this functionality:  
<https://www.youtube.com/watch?v=pAbXrOpUrd8>

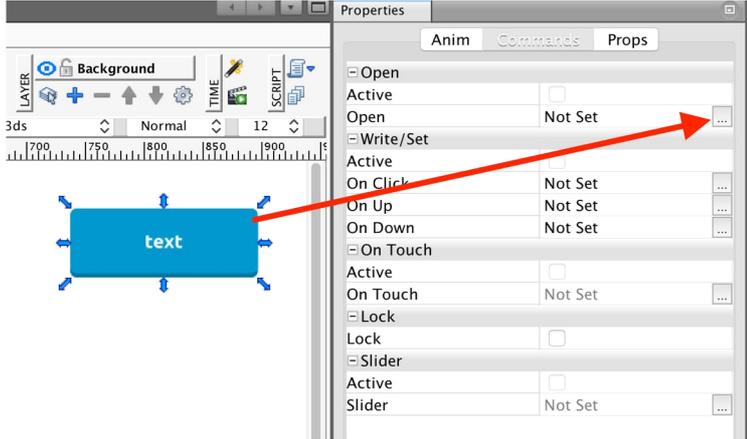
To create a Parametric View, click on the views in the Project tree and then click on the New Window icon in the main toolbar. Then, when naming the view, check the “Parametric” check box.



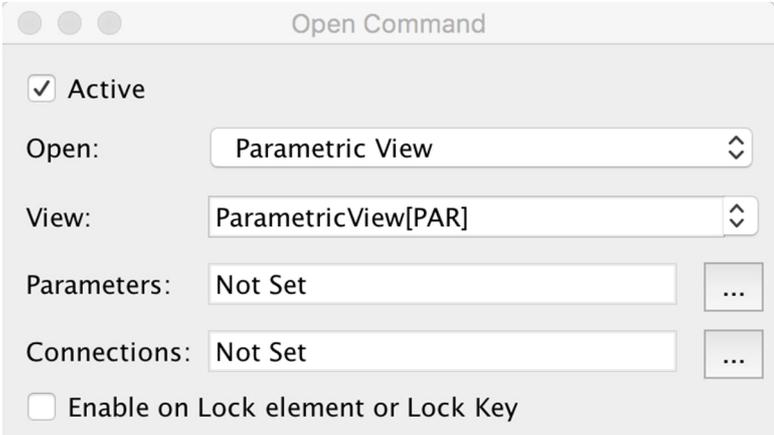
Click the “Add” button. Now your view is created. Once you have created your view, you can use symbolic access instead of absolute addresses.

## 21.1 Opening Parametric Views

The Parametric view is not visible in the main menu. There is a reason for this. If you use the parametric view in your project, you should use symbolic addresses instead of absolute ones. To show the parametric view, you must open it using the open command. Select an element you want to use as the open command, and in *Properties*, select *Commands* and click on the *Open* command.



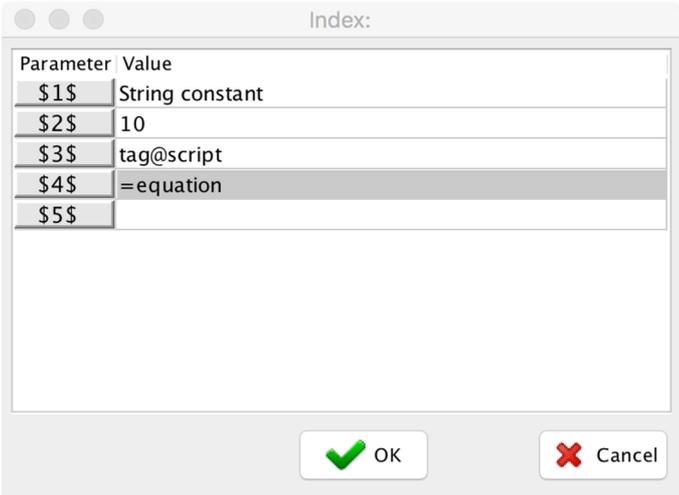
You will be presented with the Open Dialog.



For the Open command, use *Parametric View* and select your parametric view.

### Parameters

To use symbolic addresses in your parametric view, you should specify parameters in the **Parameters** options. To do so, click on the “...” button next to the Parameters field. You will be presented with a Parameters selection dialog:



As you can see in the dialog, you can specify multiple parameters. Each parameter can be of the following types:

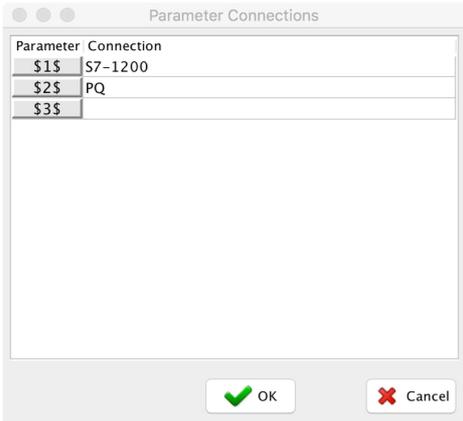
- String
- Value
- Tag (PLC address)
- Equation

If you set a Tag inside a parameter, mySCADA will pass the tag value as a parameter during the open command. In addition, when you specify an equation, mySCADA will evaluate the equation and pass the value as a parameter.

### Connections

You can also specify a connection during the opening command. Imagine you have multiple production lines, each controlled by one PLC. You can create one parametric window for one production line; then, when opening your parametric view, you can pass the connection to the given production line. This way you are able to have one view for all lines by simply passing the connection.

To set the connection during the opening command, please specify it by clicking on the “...” button next to Connections. You will be presented with a new dialog:



Now specify the connection or even multiple connections.

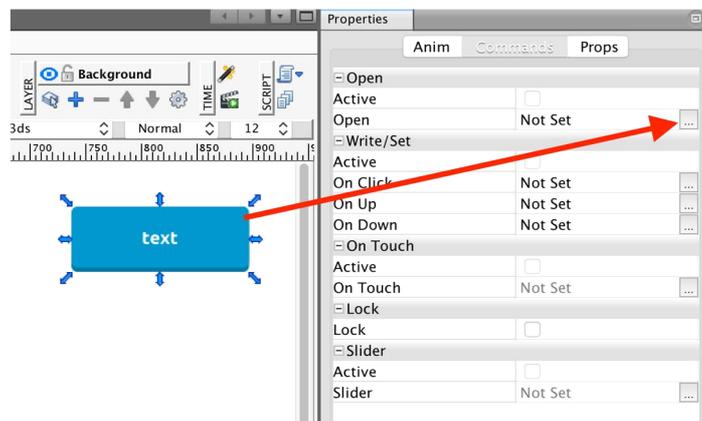
## 21.2 Symbolic Text Replacements

If you pass **Parameters** during the parametric view opening, you can then use any of the passed parameters in text elements. To do so, simply use the \$index\$ syntax.

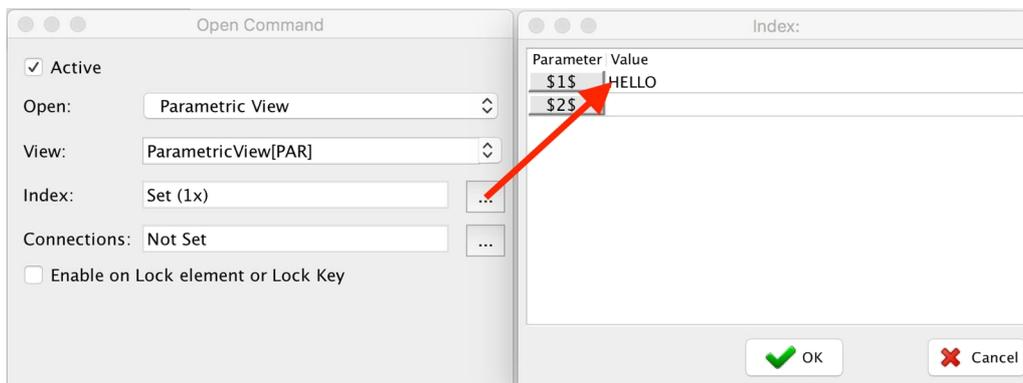
1. First, create a parametric view with text elements inside it:

**First Parameter is \$1\$**

2. Now in different view, select a button and use the open command



3. Specify the Parameter



4. When you open your parametric view by clicking on the button, you will see that **\$1\$** is replaced with the value specified in Parameters.

# First Parameter is HELLO

TIP: Download an example showing the Symbolic text replacement of the following parameters:

- string
- value
- tag value
- equation

DOWNLOAD DEMO PROJECT HERE:

[http://nsa.myscada.org/projects/example/Example\\_parametric\\_view.mep](http://nsa.myscada.org/projects/example/Example_parametric_view.mep)

[http://nsa.myscada.org/projects/example/Parametric\\_view.mep](http://nsa.myscada.org/projects/example/Parametric_view.mep)

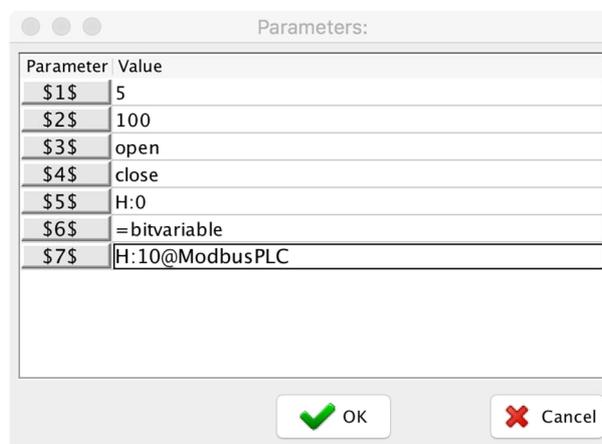
## 21.3 Symbolic Tag Creation

In a regular view, you would link your graphics to your PLC using hard (direct) address. In the Parametric View, you can construct your tag with the help of passing parameters, which you can use in the tag address.

When you construct your tag, use `$index$` as part of the tag. During the opening of the parametric view, `$index$` will be replaced by the parameter specified by the open command.

Some examples:

Open command parameters are specified like this:



## Parametric Views

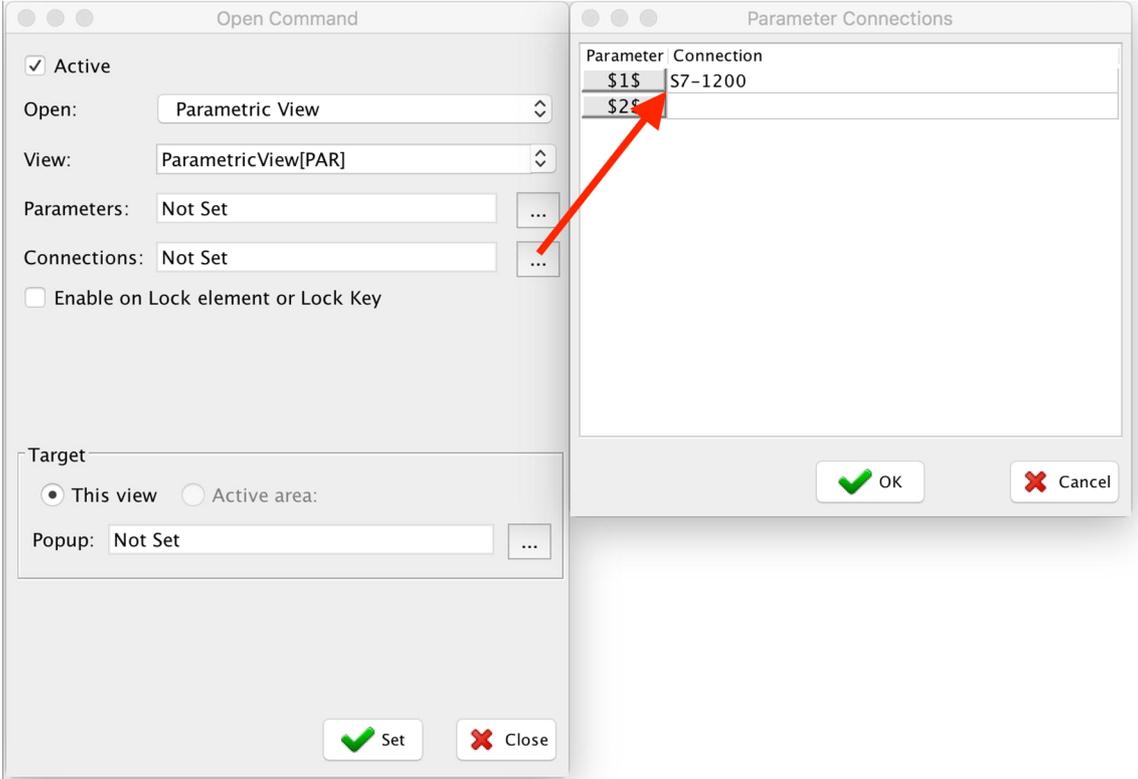
Let's look at the parameter \$6\$; here we use an equation to pass the value of variable **bitvariable** as defined in the View Script. In parameter \$7\$ we pass the value read from the PLC at address H:10.

Now some examples of how symbolic tag translates to hard link during the view opening:

Symbolic tag defined at view	Hard link translated during opening of view	Note
Valve[\$1\$]	Valve[5]	
DB\$2\$	DB100	
motor[\$1\$].\$3\$	motor[5].open	
motor[\$1\$].\$4\$	motor[5].close	
\$5\$	H:0	
DB100.\$6\$	DB100.1	bitvariable=1
DB100.\$7\$	DB100.2	Tag H:10 has value = 2

## 21.4 Replacing Connections

Replacing connections is extremely useful when you need to link your parametric view with multiple PLCs. Link your parametric view with specific PLC by passing the connection parameter at opening command.



As you can see, we are passing a connection to S7-1200 PLC by the first connection parameter \$1\$.

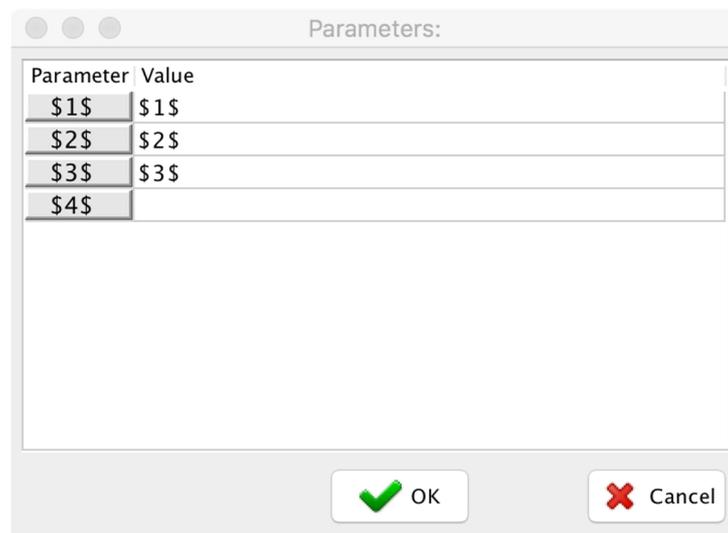
### Using Connection Parameter in Tag Definition

You can use connection parameters in the tag definition by simply using \$index\$ instead of a connection alias:

tag@\$1\$

## 21.5 Nesting Parametric Views

You can nest parametric views, e.g. you can open one parametric view from another. In this case, you can use parameters passed from the original view to the following parametric views. To do so, use the notion \$index\$ in the parameters dialog:



As you can see in the dialog, we are passing three parameters from the original view to the parametric view.

### 21.6 Usage Example:

Imagine you have to visualize a plant with 200 similar motors and, instead of designing 200 identical screens with different PLC links, you will only create one parametric (master) view. The operator then can select a specific motor by entering the desired parameters when opening the requested view.

# Parametric Views

Parameters

Open: Parameter Window

View: Motor[PAR] **View Name**

New Window: Not Set

Index: Crain 1A;Basement;DB100;1 **Opening parameters**

Connections:

Overview Window created by the user - every motor has different opening parameters

Parametric Views are denoted by [PAR] after its name. You can specify parametric views when designing a new view.

Name: Crain 1A  
 Location: Basement  
 Status: Running  
 RPM: 1270  
 Apms: 16.3  
 Hours: 12345

Concrete Motor Window

Now let's see what will happen when you use your parametric view in runtime:

**DESIGN** → **BEFORE VIEW OPENS** → **LIVE DATA**

Specify tags when designing the view with parameter values denoted by \$index\$. Use multiple parameters if necessary.

When opening the view, your tags will be automatically converted with parameter values

Online view will read correct real data from the PLC for a specific motor.

You can see that the text has been replaced as well:

replace \$1\$ with "Crain 1A"

text to replace

DOWNLOAD DEMO PROJECT HERE:

[ftp://nsa.myscada.org/history/projects/example/Example\\_parametric\\_view.mep](ftp://nsa.myscada.org/history/projects/example/Example_parametric_view.mep)

## 22 View Scripts

---

*myDESIGNER* offers many tools for performing the most common data acquisition, display, animation, and effects—and all of this without coding. For maximum flexibility *mySCADA* also includes a complete scripting language based on *JavaScript*, which allows you to interact programmatically with most of *mySCADA* functions with a high-level scripting language.

Watch video describing this functionality: <https://www.youtube.com/watch?v=t7BtSyaPVss>

### Easy to learn

Scripting can be used for all sorts of tasks. The possibilities are endless, as it is designed to be easy to use, extending the functions of *mySCADA*. You do not need to be an experienced programmer to be able to use scripting. Using *JavaScript* as a scripting language means you do not have to worry about memory allocations, leaks, or complex programming issues. Using it is very simple and straightforward.

### Features

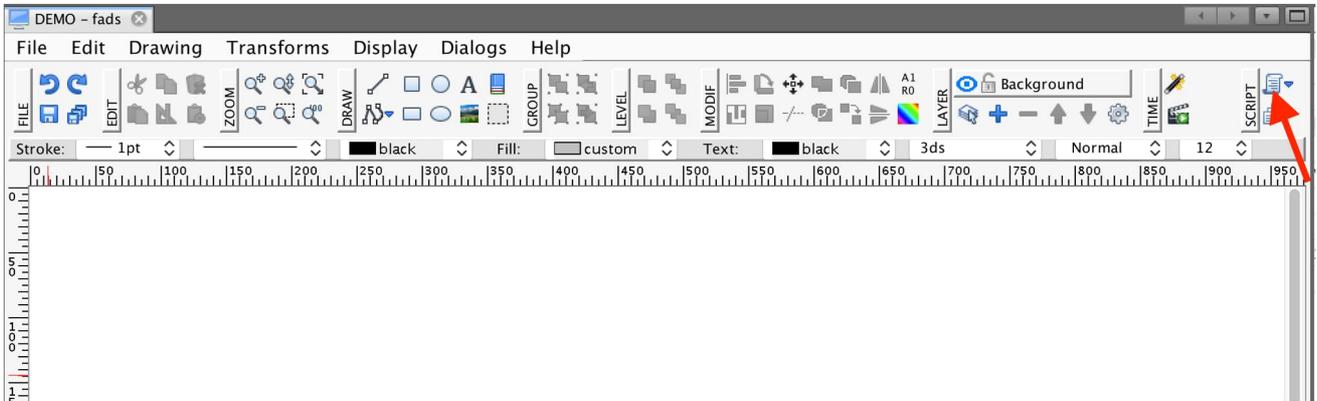
*mySCADA* has many built-in features that can be extended with scripting. If you are starting out, you can use all of the built-in features to acquire data, create graphics, make dynamic animations, and conduct other tasks without writing any code. You can then pick one area that needs a little extra flexibility and write some simple script while still using all of other functions. As you get more experienced, you can take further advantage of powerful scripting.

### 22.1 Using Script in Views

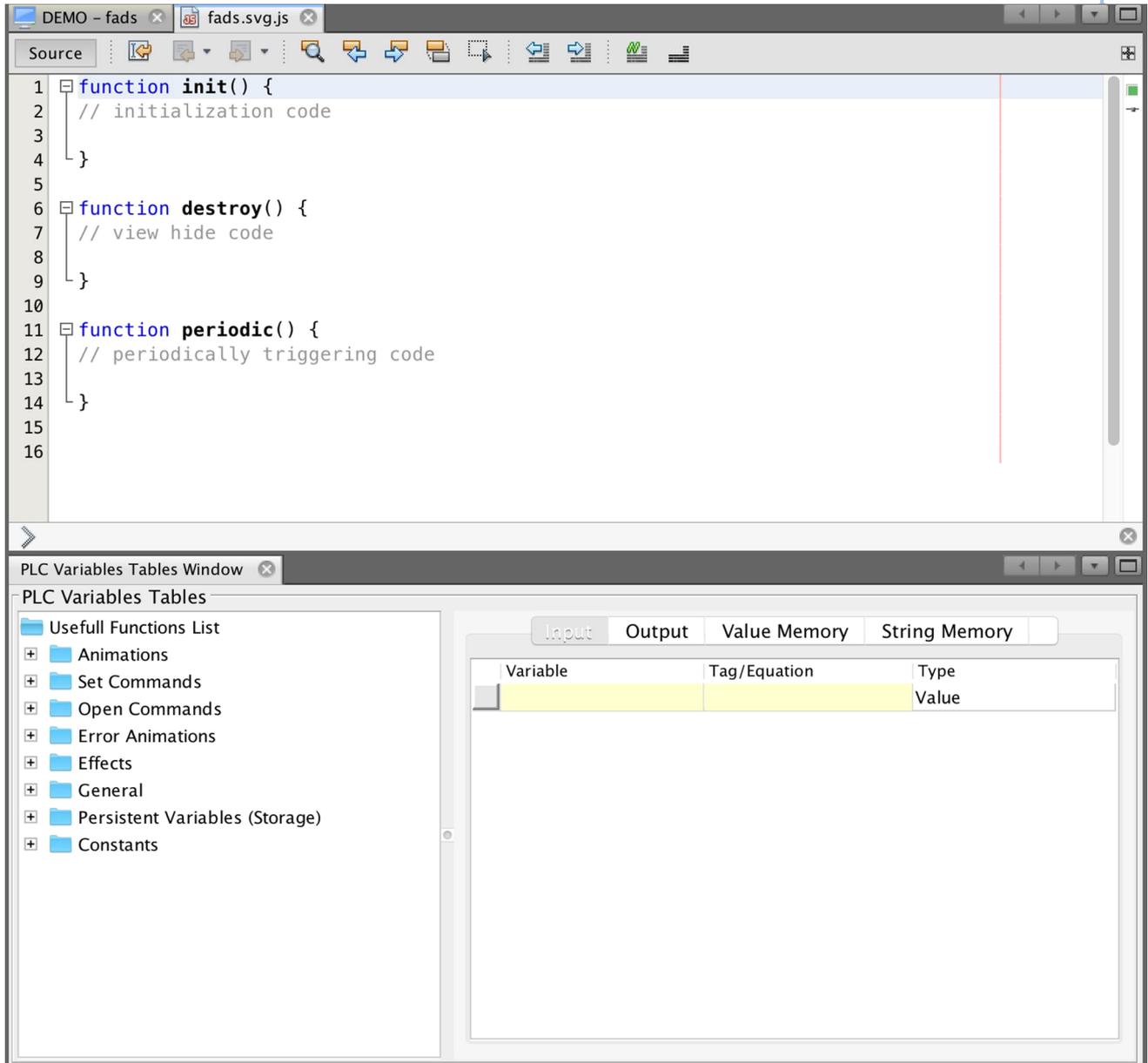
For each view, you can create your own script, which is evaluated each time an active screen is refreshed. The refresh logic works as follows:



First, if you want to add a user-defined script, open your view and click on the **Script** button in the main toolbar.

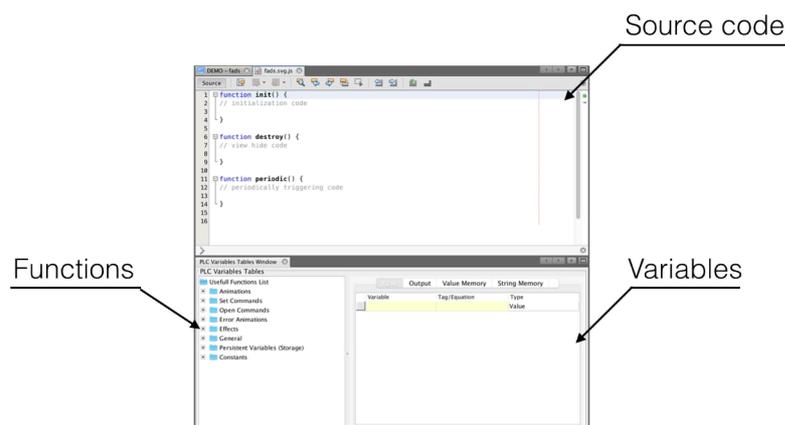


Then the script window will show up where you can insert your user-defined script:



The script editor is divided into 3 main windows:

- *Source Code*: this is a window where you put the user-defined script
- *Variables*: here you define your view variables
- *Useful Functions*: in this window, you can find specific functions that help you control your components in the corresponding window.



## 22.2 Declaring Variables

You can declare your variables in the *Variables Window*:

Variables Window			
Input	Output	Value Memory	String Memory
Variable	Tag/Equation	Type	
TankLevel1	=100*adr(F11:100)/1024	Value	
TankLevel2	F11:101	Value	
Flow1	F12:1	Value	
Flow2	F12:2	Value	
		Value	

There are four types of variables to use:

- *Input variables*: use these variables for reading data from the PLC; each time the script is evaluated it first reads the PLC tags and stores them in the input variables
- *Output variables*: after the script evaluation, the output variable values are written into the PLC; you can set whether it will write the value each time the script is evaluated or only upon change - this is controlled via the *Update* field (*Always, On Change*)
- *Value Memory*: here you can declare your persistent variables; when a view is loaded, memory variable has its default value; you can change this value in the script and it will prevail until you switch to a different view or close the application
- *String Memory*: String Memory variables are the same as Value Memory variables, but are of string type

## 22.3 Script Writing

You can write the script in the *Source code window*. You can use any function or expression from *JavaScript* - the editor automatically highlights your syntax and conducts an error check.

### Functions

You can declare your own functions using JavaScript function syntax. As you can see, when you open View Script, there are some already created functions:

```
function init() {
}
```

This function is called upon opening the view. You can put initialization code here.

```
function destroy() {
```

```
}

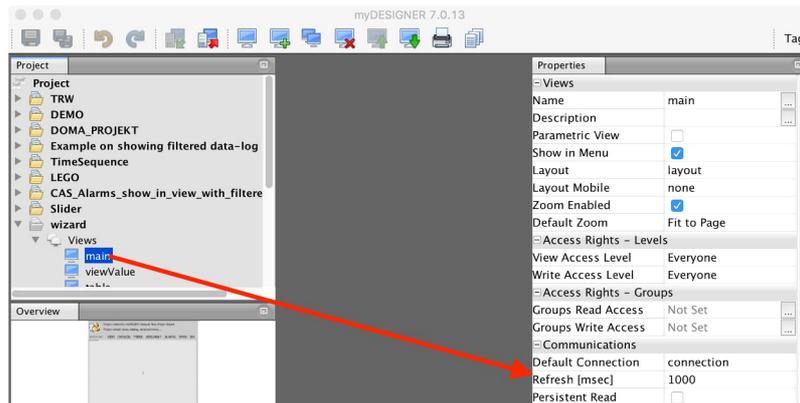
```

This function is called when a user closes the view or navigates to a different view. Usually, you don't have to fill this function in.

```
function periodic() {
}

```

This function is called every time the view is refreshed. The refresh period (e.g. the period this function is called) can be specified in view parameter – refresh



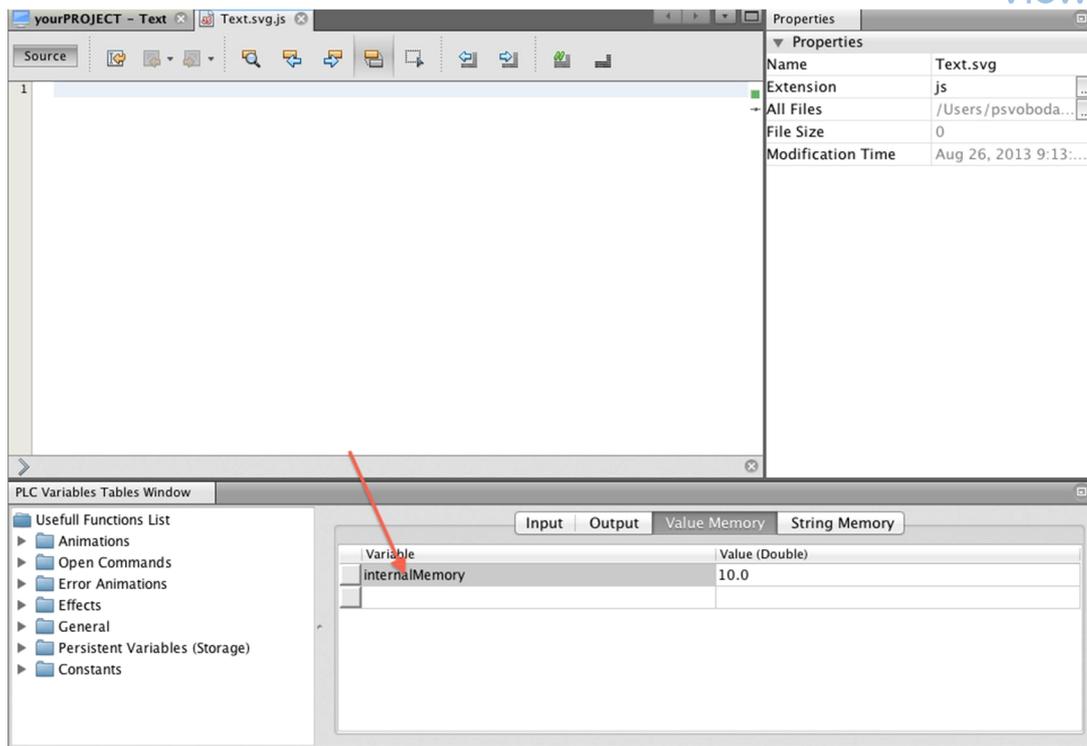
In periodic() function, you will probably write most (or all) of your code.

## 22.4 Using variables in animations and effects

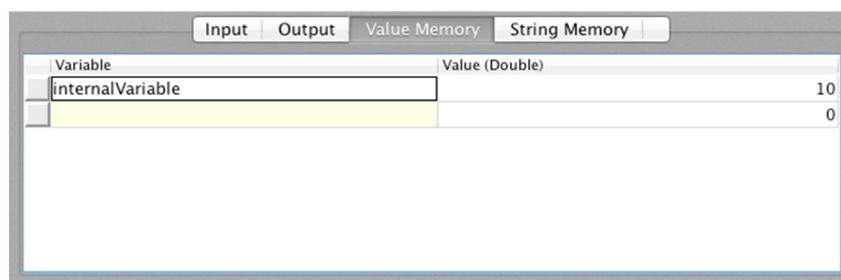
You can use declared view variables directly in animations and effects. To link the view variable with an animation, use the equation editor like with regular PLC tags.

### *Example:*

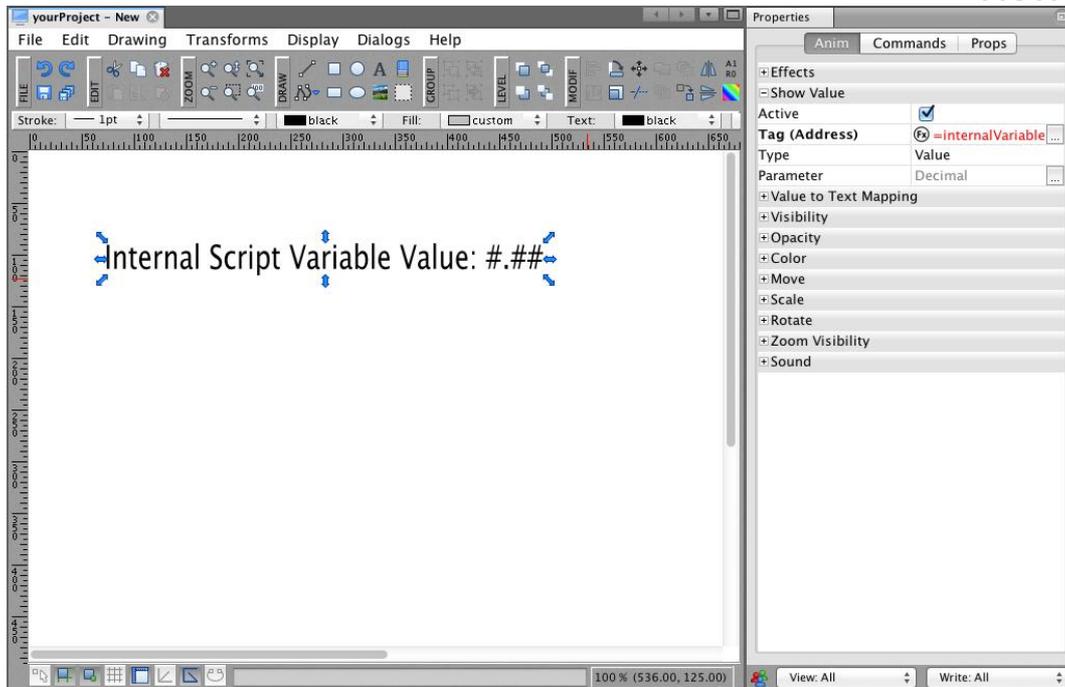
- 1) Let's show the value of the *Value Memory Variable* in a text element in your *view*.
- 2) Open the *Script window*.



- 3) In the **Value Memory** tab of the *Variables window*, create '*InternalVariable*' with a default value of 10.0.

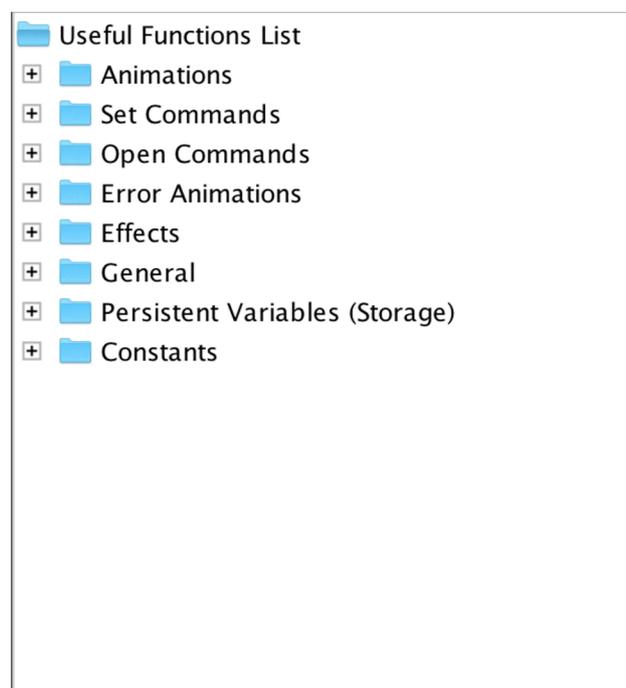


- 4) Switch back to the *view* you are editing.
- 5) Create a text element in your *view*, click on it, and create an animation with a memory variable by typing "*=InternalVariable*" or selecting memory variable with the equation editor.

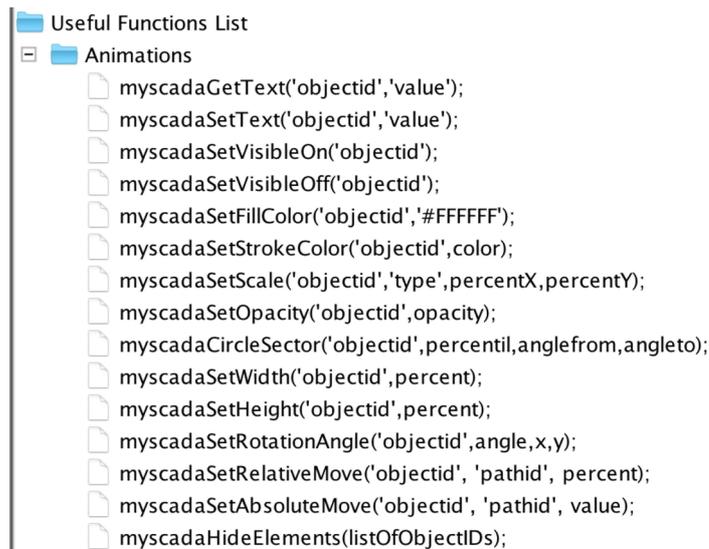


### Specific mySCADA functions

In the script, you can use any functions declared in JavaScript. To interfere with your graphics, you can use additional functions defined by mySCADA. A complete list of all available functions, including help, can be found in the Useful Functions List.



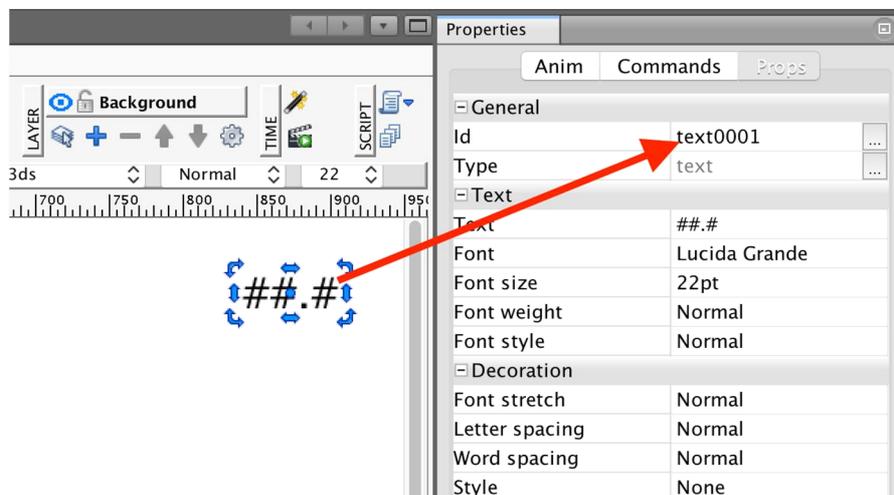
If you click on the category, you will see the list of all available functions:



Now select the function you are interested in, and you will see a description and help on the right side of the window.

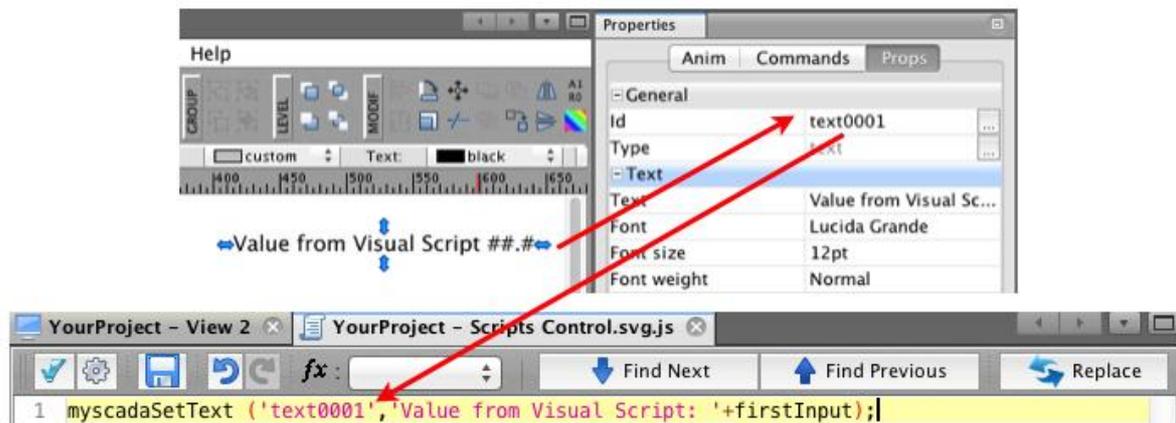
*TIP: You can drag the selected function and drop it into your source code.*

As you can see from the function list, most of the functions deal with your graphical objects. To be able to address your graphical object, you need to know its ID. Every graphical object in your view has its unique ID that can be found in properties. Click on the graphical object and then look in Properties – ID:



You can control animations of your graphic object directly in the script with specific *mySCADA* functions. To set the value of the element directly in the script, use the following function:

```
myscadaSetText('text0001',internalMemory);
```



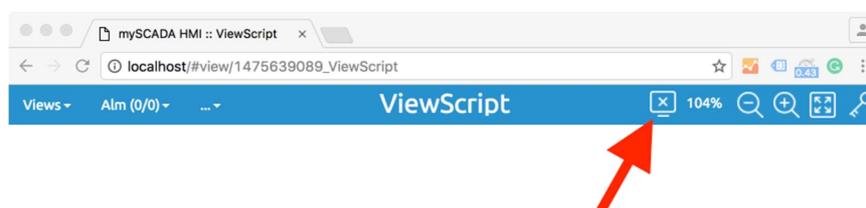
## 22.5 Debugging View Scripts

As you write your view scripts, there might come a moment when your scripts are not working for various reasons. In mySCADA, you have two easy options to troubleshoot and find where there might be a problem:

- Using debug screen on your view
- Using web browsers with an integrated debugger

## 22.6 Using debug screen on your view

To use a debug screen in your view, simply open the view in the browser. If you have an error in your view script code, you will see it right away - the icon in the main menu will show.



# View with view script

Now press **SHIFT + D**. You will be presented with the views' debug screen:

**Info about view ViewScript**

**View scripts info**

View script status	October 5, 2016 07:45:01 OK
function init() status	October 5, 2016 07:45:01 OK
function periodic() status	October 5, 2016 07:47:26 ReferenceError: nonexistentfunction is not defined

**Connections**

...no tag connections defined...

**View script variables**

Type	Name	Value

**System variables**

Name	Value
myscadaRunCount	146
myscadaLoggedUser	""
myscadaLoggedUserLevel	-1
myscadaLoggedUserGroups	[]
myscadaActiveAlarmsCount	0
myscadaNonAckAlarmsCount	0
myscadaIndexesString	[""]
myscadaBrowserIsMobile	false
myscadaViewname	"ViewScript"

**Active areas**

Id	Name
	...no active areas defined...

As you can see in the section "View scripts info," all functions are listed in the view script along with the status. If you have errors in the code, you can see the error status along with the description of the error:

function periodic() status	October 5, 2016 07:47:26 ReferenceError: nonexistentfunction is not defined
----------------------------	--

In addition, you can watch the live status of system variables in your view scripts in the section System variables:

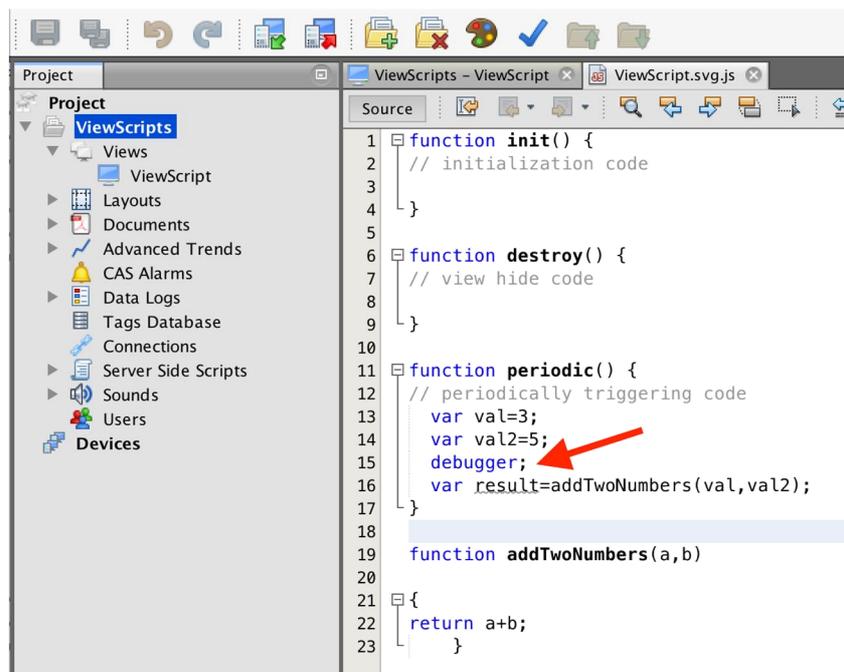
## System variables

Name	Value
myscadaRunCount	146
myscadaLoggedInUser	""
myscadaLoggedInUserLevel	-1
myscadaLoggedInUserGroups	[]
myscadaActiveAlarmsCount	0
myscadaNonAckAlarmsCount	0
myscadaIndexesString	[""]
myscadaBrowserIsMobile	false
myscadaViewname	"ViewScript"

## 22.7 Using web browsers integrated debugger

Another option is to use a debugger, which ships with your favorite web browser. Nowadays every modern browser is equipped with the option to debug JavaScript. To use the browser's debugger, do the following:

1. In your code, at the place where you want to start debugging, enter the command `debugger;`

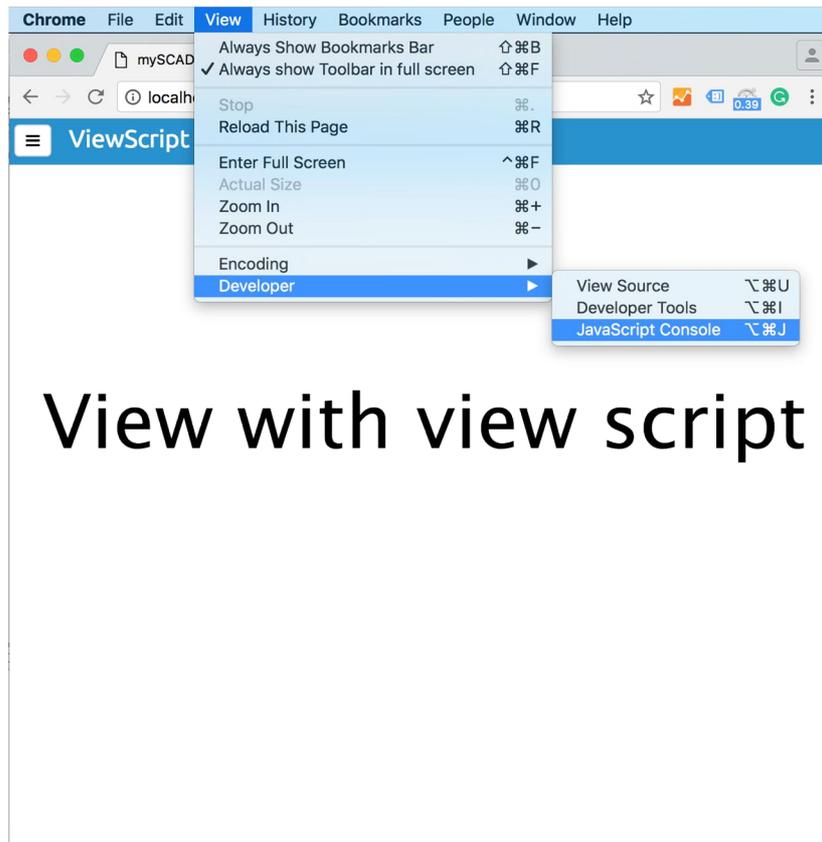


```

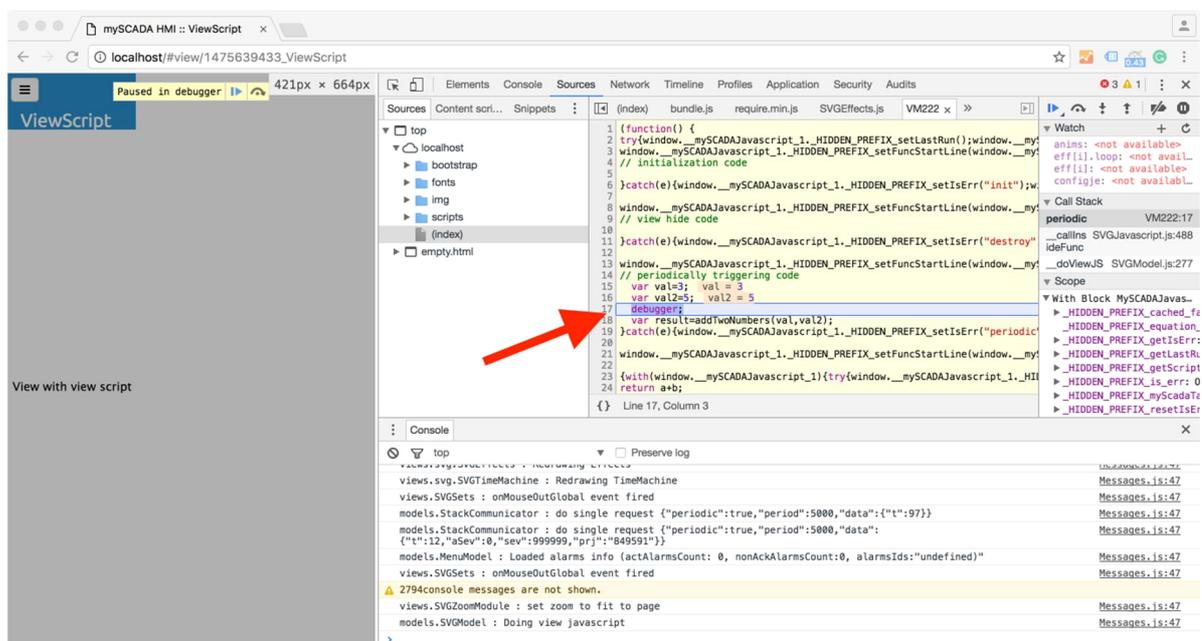
1 function init() {
2   // initialization code
3 }
4
5
6 function destroy() {
7   // view hide code
8 }
9
10
11 function periodic() {
12   // periodically triggering code
13   var val=3;
14   var val2=5;
15   debugger;
16   var result=addTwoNumbers(val,val2);
17 }
18
19 function addTwoNumbers(a,b)
20
21 {
22   return a+b;
23 }

```

2. Download your project to the device running mySCADA and navigate to your view



- Now enter the debug mode in your browser (Chrome shown in the picture)



- As you can see, your code has stopped where the keyword `debugger` was inserted. Now you can start debugging;

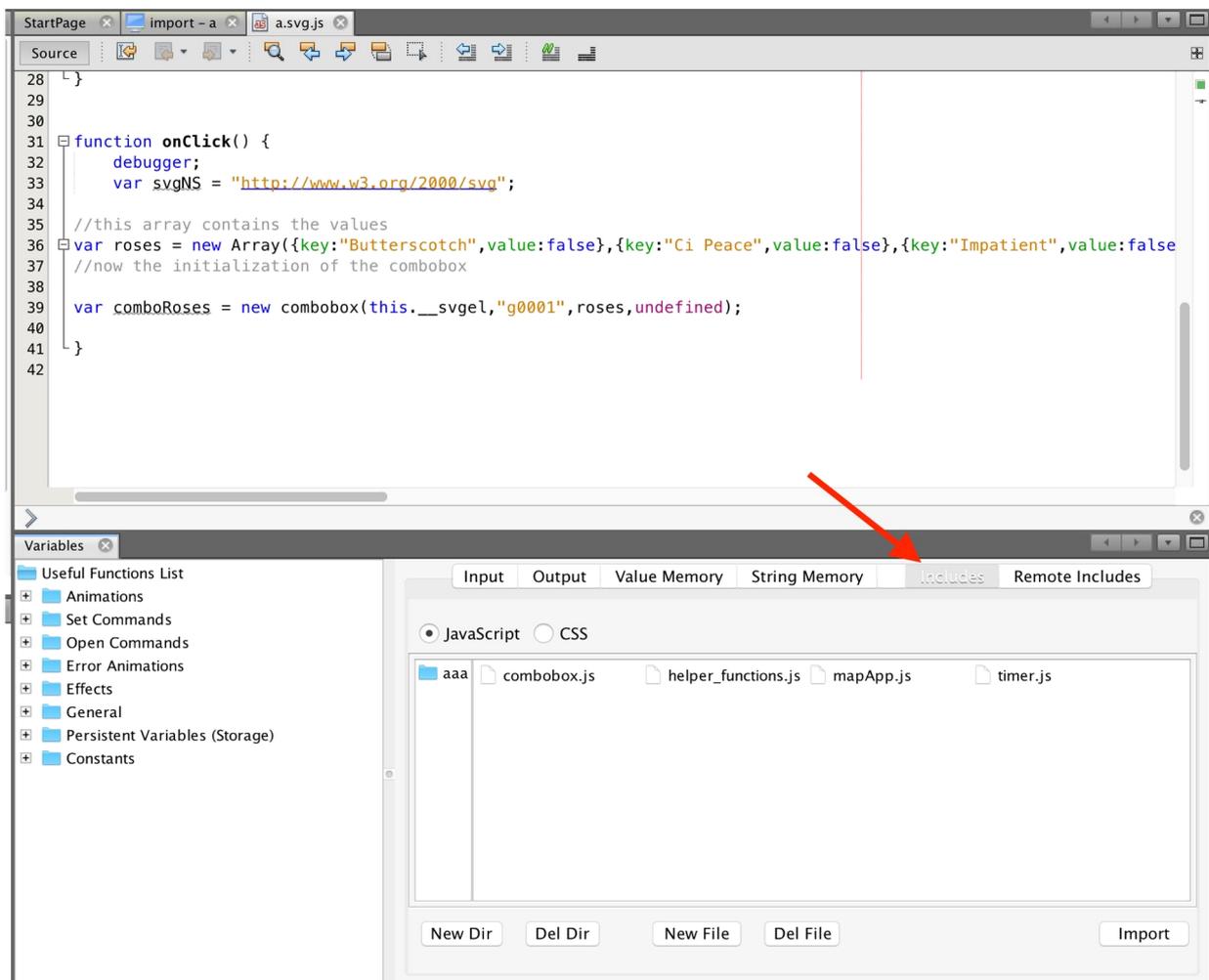
## 22.8 Using JavaScript Libraries - Includes

You can easily use your JavaScript libraries in your project. You simply can create your own library or include any JavaScript JavaScript library found on the internet. All the JavaScript resources are included in two folders in your project:

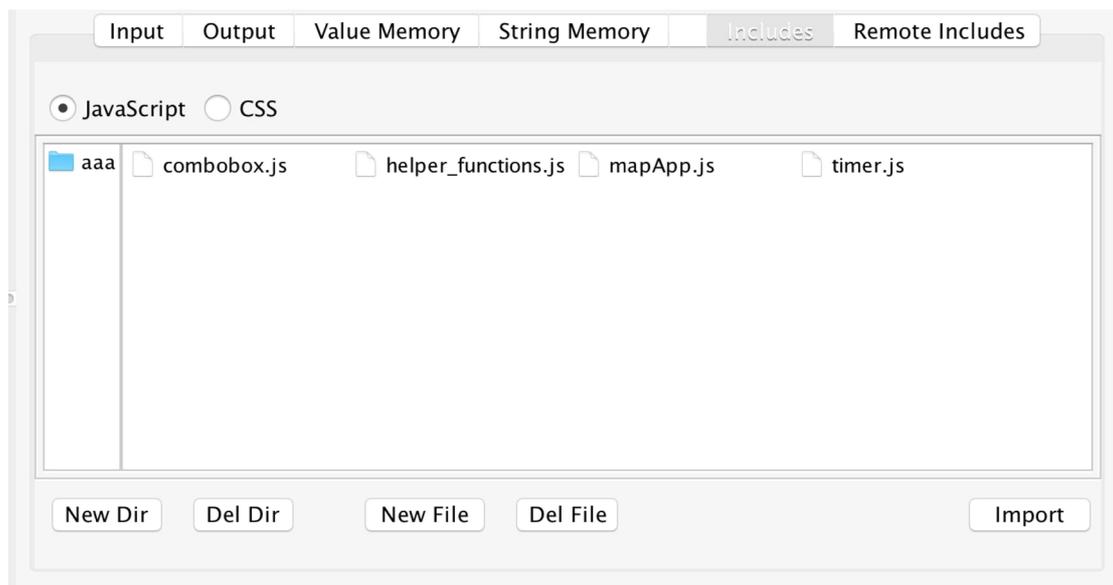
- js – all JavaScript sources
- CSS – all CSS sources

You can directly add all your library files into those two folders. myDESIGNER will find them automatically, and you can use them anywhere in your project.

To work with the JavaScript libraries directly in myDESIGNER, open a view script and navigate to the “Includes” tab.



Now, let's look at the details of the Includes window:



On the left side of the window, you can see all subdirectories. On the right side, you can find all your source files. If you double click on the file, you can edit it easily.

### Selecting JavaScript or CSS

First of all, you should choose if you want to work with JavaScript files or with CSS files. To select either of those:



### New Directory Creation

To create a new directory, press the **“New Dir”** button.

### Directory Deletion

To delete a directory, select it and press the **“Del Dir”** button.

### New File Creation

To create a new file, press the **“New File”** button and give it a name.

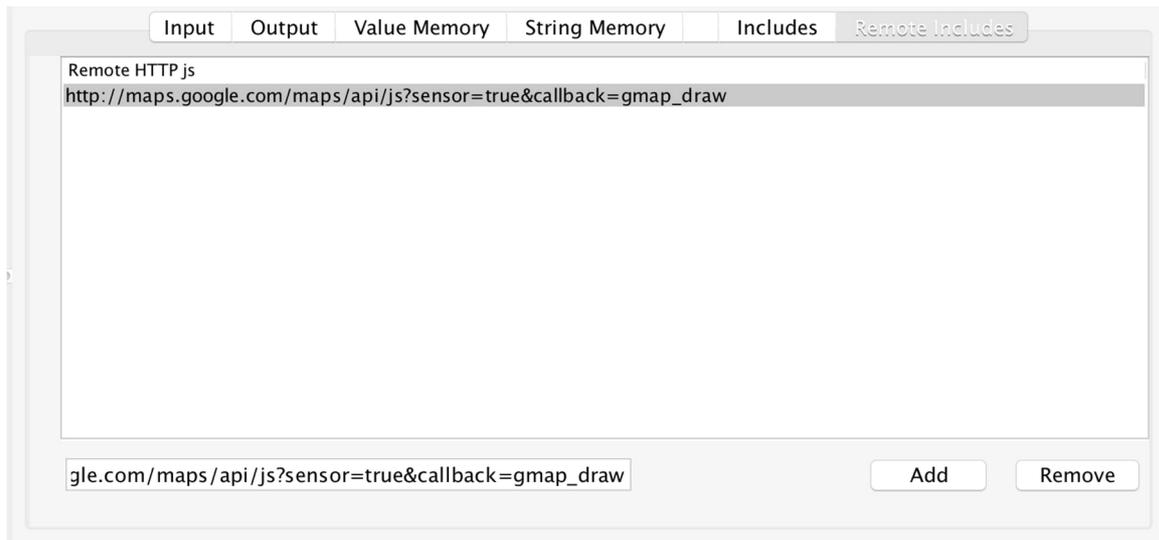
### Deleting File

To delete a file, select it and press the **“Del File”** button.

All JavaScript in the primary folder is automatically loaded in view. In addition, all CSS files in the primary folder are added to your view automatically.

## 22.9 Linking External JavaScript Libraries – Remote Includes

You can easily link remote JavaScript libraries by navigating to **Remote Includes**.



All the files specified in this window will be automatically loaded to your view.

*TIP: you can specify JavaScript as well as CSS files.*

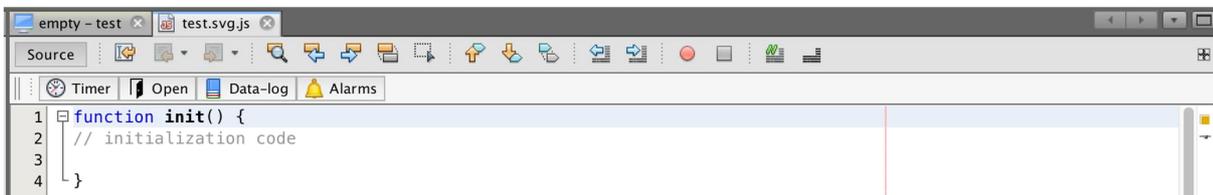
# 23 View and Server Side Scripts – Common Tasks

This chapter will show you how to achieve common tasks using the View or Server Side Scripts. You will learn how to simply create a timer, read historical data from data-logs and alarms, use open command, generate reports and more.

## 23.1 Graphical guides

For the all common tasks we have a prepared a simple to use graphical guides which will generate a code skeleton for you. Graphical guides are located at the top toolbar of the edited script:

### View Script Version



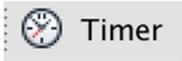
### Server Side Scripts Version



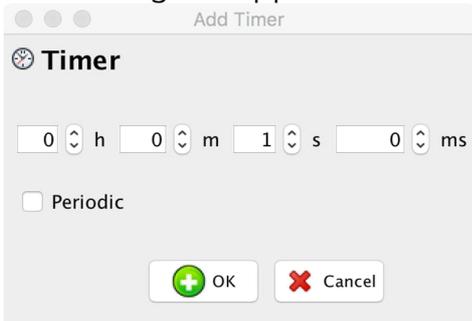
Now we will walk you to using each graphical guide.

### Creating Timer

To create a timer, click on a Timer Icon in the toolbar.



New dialog will appear:



## View and Server Side Scripts – Common Tasks

Select an time period and tick **repeat** if you want to trigger the timer periodically. Once you press OK, myDESIGNER will automatically create a code for you and insert it into the edited script.

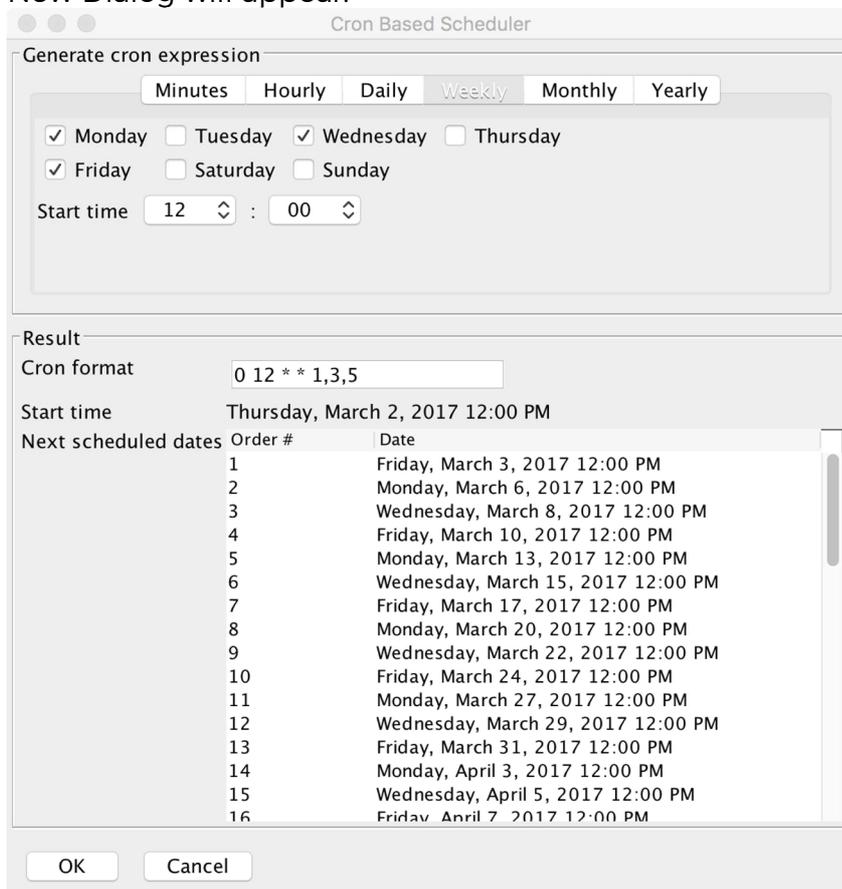
```
14 |   setTimeout(function() {  
15 |     //your code here!  
16 |   }, 1000); //Timeout value in milliseconds
```

### Creating Scheduled Event (Server Side Scripts Only)

To create a scheduled event, you can use the CRON tool. To activate it, please press the Schedule Icon:



New Dialog will appear:



The dialog box is titled "Cron Based Scheduler". It has a "Generate cron expression" section with tabs for "Minutes", "Hourly", "Daily", "Weekly", "Monthly", and "Yearly". The "Weekly" tab is selected. Below the tabs are checkboxes for days of the week: Monday (checked), Tuesday (unchecked), Wednesday (checked), Thursday (unchecked), Friday (checked), Saturday (unchecked), and Sunday (unchecked). There are also "Start time" dropdowns set to "12" and "00".

The "Result" section shows the "Cron format" as "0 12 \* \* 1,3,5". The "Start time" is "Thursday, March 2, 2017 12:00 PM". Below this is a table of "Next scheduled dates":

Order #	Date
1	Friday, March 3, 2017 12:00 PM
2	Monday, March 6, 2017 12:00 PM
3	Wednesday, March 8, 2017 12:00 PM
4	Friday, March 10, 2017 12:00 PM
5	Monday, March 13, 2017 12:00 PM
6	Wednesday, March 15, 2017 12:00 PM
7	Friday, March 17, 2017 12:00 PM
8	Monday, March 20, 2017 12:00 PM
9	Wednesday, March 22, 2017 12:00 PM
10	Friday, March 24, 2017 12:00 PM
11	Monday, March 27, 2017 12:00 PM
12	Wednesday, March 29, 2017 12:00 PM
13	Friday, March 31, 2017 12:00 PM
14	Monday, April 3, 2017 12:00 PM
15	Wednesday, April 5, 2017 12:00 PM
16	Friday, April 7, 2017 12:00 PM

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

Now select the dates and times, when you want your code to be run and press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
17 |   var CronJob = require('cron').CronJob;  
18 |   new CronJob('0 12 * * 1,3,5', function() {  
19 |     //your code here  
20 |   }, null, true);
```

### Export to CSV and Microsoft Power BI

You can make periodic export of data-log data into CSV file. This file is easily readable by Microsoft Power BI. To do so, please click on the button **Export**.



For detail description, please see section **Simple Periodic Export to CSV and Microsoft Power BI** in this manual.

### Read Historical data from data-logs

You can easily read and process historical data from data-logs. To do so, click on the Data-log Icon



New Dialog will be opened:

A dialog box titled "Read data-log data function". It contains a "Data-log" dropdown menu set to "default". Below it is a list of "Items" with checkboxes: "teplota1 (H:0@M)" and "teplota2 (H:1@M)", both checked. There are "Select All" and "Deselect All" buttons. A "Limit by time:" section has checkboxes for "Time From [sec]" and "Time To [sec]", both checked, with input fields containing "new Date()/1000-60\*60" and "new Date()/1000" respectively. A "Filter:" checkbox is checked. Below it is a table with columns "Key", "Type", and "Value". At the bottom, there is a "Limit records to:" field set to "1,000", radio buttons for "from start" (selected) and "from end", radio buttons for "Loop over results" (selected) and "Export to CSV", and "Cancel" and "OK" buttons.

Key	Type	Value
-----	------	-------

In the provided dialog, please start by selecting a data-log from which you want to read data. Then select an items you want to process, or leave all selected. Other options follow:

**Limit by time:** first you can limit the data by provided time. Time is specified in UTC format in seconds since 1.1.1970. You can enter a value or provide a variable where the value is stored.

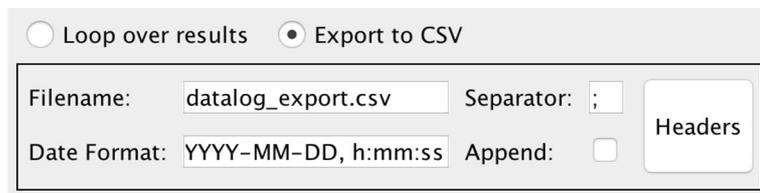
## View and Srever Side Scripts – Common Tasks

**Filter:** if the data-log contains a keyed value, you can limit results shown in your project by specifying a filter value. Again this can be a hard value or variable.

**Limit records to:** please provide limits for the number of records loaded. You can also specify if the limit is taken from the beginning eg. “**from start**” or **from end**.

**Loop over results:** if this option is checked, you will have an option to loop over the retrieved records and process them in your script.

**Export to CSV** (Server side scripts only): if you choose this option, the retrieved historical data will be exported into a CSV format and saved into a file. File is saved under a provided name into user data folder accessible over FTP or HTTP and HTTPS.



press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
18 //read historical data from data-log
19 var options=new Object;
20 options.dlgid=1;
21 options.tags=[1,2];
22 options.timeFrom=new Date()/1000-60*60;
23 options.timeTo=new Date()/1000;
24 options.limit=1000;
25 myscada.readDataLogData(options, function(err,result){
26     //process each row of returned data
27     for (i=0;i<result.length;i++)
28     {
29         var date=new Date(result[i].tm[0]*1000+Math.floor(result[i].tm[1]/1000));
30         var value0=result[i][1]; //teplota1 (H:0@M)
31         var value1=result[i][2]; //teplota2 (H:1@M)
32     }
33 });
```

## Reading and processing Alarms

You can read and process online and historical alarms. To do so, please click on the **Alarms** Icon:



New dialog will open:

In this dialog, you can choose if to process Online or Historical alarms. We will start with **Historical alarms**:

Historical alarms dialog has several sections. We will explain in details each section.

### Columns:

Columns allows you to select all the data retrievable from the history. Please select what columns you want to process.

### Filters:

Currently, you can filter retrieved data based on time interval. Future versions will allow to extend the filter for Message, Area and Device as well.

### Aggregates:

If you need historical alarms in the form they have been stored, leave aggregates to **none**. If you want to retrieve an aggregated data based on alarm occurrence count and overall activation time, please tick the **occurrence** option.

**Limit records to:** please provide limits for the number of records loaded. You can also specify if the limit is taken from the beginning eg. **“from start”** or **from end**.

**Loop over results:** if this option is checked, you will have an option to loop over the retrieved records and process them in your script.

**Export to CSV** (Server side scripts only): if you choose this option, the retrieved historical data will be exported into a CSV format and saved into a file. File is saved under a provided name into user data folder accessible over FTP or HTTP and HTTPS.

## View and Server Side Scripts – Common Tasks

Loop over results  Export to CSV

Filename:  Separator:

Date Format:  Append:

Headers

press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
16 //read historical alarm data
17 var options=new Object;
18 options.timeFrom=new Date()/1000-60*60;
19 options.timeTo=new Date()/1000;
20 options.limit=1000;
21 myscada.readAlarmsHistory(options, function(err,result)
22 {
23 //process each row of returned data
24 for (i=0;i<result.length;i++)
25 {
26 var msg=result[i].cell.msg; //Message
27 var status=result[i].cell.stat; //Alarms status
28 var severity=result[i].cell.sv; //Severity
29 var area=result[i].cell.area; //Area field
30 var acttime=new Date(result[i].cell.atm[0]*1000+Math.floor(result[i].cell.atm[1]/1000)); //Activation time
31 }
32 });
```

**Online Alarms** can be processed as well. To do so, click on the tab Online and dialog will change accordingly:

Alarms Function

Read alarms function

History Online

Select Columns:

- MESSAGE
- STATUS
- SEV
- AREA
- DEVICE
- ACT TIME
- DEACT TIME
- ACK TIME
- ACK TEXT
- ACT VAL
- DEACT VAL
- ACK VAL

Severity up to:

Loop over result  Export to CSV

OK Cancel

This function will retrieve active or non-acknowledge alarms at the time script is run. You can select which **columns** you want to process or **maximum severity** level. Other options are:

**Loop over results:** if this option is checked, you will have an option to loop over the retrieved records and process them in your script.

**Export to CSV** (Server side scripts only): if you choose this option, the retrieved historical data will be exported into a CSV format and saved into a file. File is saved

## View and Server Side Scripts – Common Tasks

under a provided name into user data folder accessible over FTP or HTTP and HTTPS.

<input type="radio"/> Loop over results	<input checked="" type="radio"/> Export to CSV	
Filename: <input type="text" value="datalog_export.csv"/>	Separator: <input type="text" value=";"/>	<input type="button" value="Headers"/>
Date Format: <input type="text" value="YYYY-MM-DD, h:mm:ss"/>	Append: <input type="checkbox"/>	

press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
16 //read online alarm data
17 var options=new Object;
18 options.severity=999999;
19 myscada.readAlarmsOnline(options, function(err,result)
20 {
21 //process each row of returned data
22 for (i=0;i<result.length;i++)
23 {
24 var msg=result[i].cell.msg; //Message
25 var status=result[i].cell.stat; //Alarms status
26 var severity=result[i].cell.sv; //Severity
27 var acktime=new Date(result[i].cell.acktm[0]*1000+Math.floor(result[i].cell.acktm[1]/1000));
28 var value=result[i].cell.v; //Current Value
29 var actval=result[i].cell.av; //Activation value
30 }
31 });
```

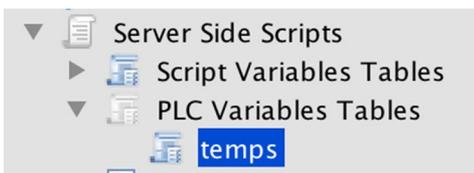
## 23.2 Read/Write data from/to PLC

To read or write data from PLC, you should use the **R/W tags** button.



R/W Tags

To read or write tags from PLC, you must create a PLC request table first. To do so, please navigate to the **PLC Variables Tables** section and create a new table. You can call it whatever you like.



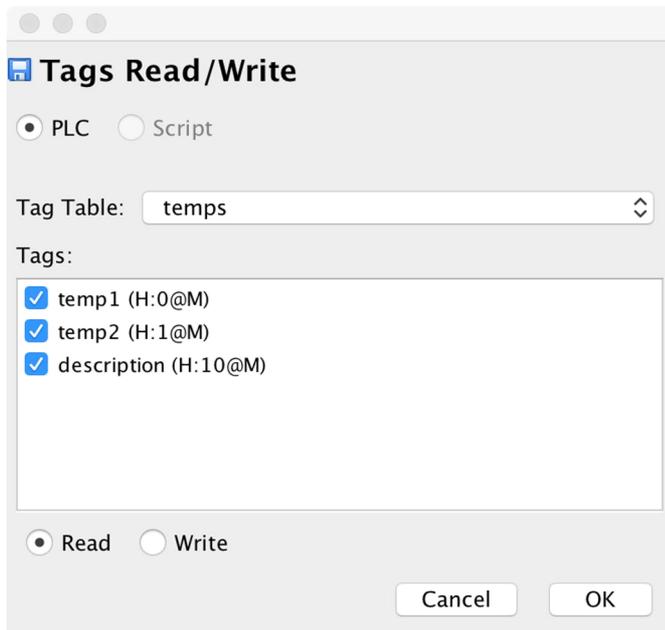
In **PLC Variables Table**, you define what tags will be read or written from the PLC.

Variable Name	Tag@Conn/*Alias	Number of Elem...	Type
temp1	H:0@M	1	Numeric
temp2	H:1@M	1	Numeric
description	H:10@M	10	String
		1	Numeric

## View and Server Side Scripts – Common Tasks

In this example, we will be reading two numerical values and one string from the PLC with alias M.

Once you create your request, you can proceed to the script. Click on the **R/W Tags** button and you will be presented with the following dialog:



In the dialog, you can see, we have specified our created PLC variable table called **temps**. You can see all the defined tags in the **Tags** section. If you wish to read/write only some tags, please, uncheck the others. Once you are done, press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

### For tags reading:

```
//Read tags for tag group temps
myscada.readTags("temps", function (err,data){
  if (!err){
    var temp1=data[0].value;
    var temp1_err=data[0].err;
    var temp2=data[1].value;
    var temp2_err=data[1].err;
    var description=data[2].value;
    var description_err=data[2].err;
  }
});
```

As you can see, myDESIGNER has prepared a code for you, which will read your tags. For each tag, you will get its value and also a status info if it was read or ended in error (for example if you missed typed the tag address)

### For tags write:

## View and Server Side Scripts – Common Tasks

```
//Write tags for tag group temps
var options={};
options['name']="temps";
options['values'] = {};
options['values']['temp1']=2.3;
options['values']['temp2']=7.7;
options['values']['description']="temperatures";
myscada.writeTags(options, function (err,data){
  if (err){
    //write error
  }
});
```

As you can see, first we create object options and we put all the values we want to write into the **options** -> **values** array. Then we call the function **myscada.writeTags**. On successful write you will get the callback to your function.

### 23.3 Generating Report

To generate a report, first create it in your project in section Reports. Then you can generate it in server side scripts. To do so, please click on the button **Report**.



For detailed explanation on how to generate report from server side scripts, please look at the **section Creating Report on Demand** in Reports chapter in this manual.

### 23.4 Other Guides

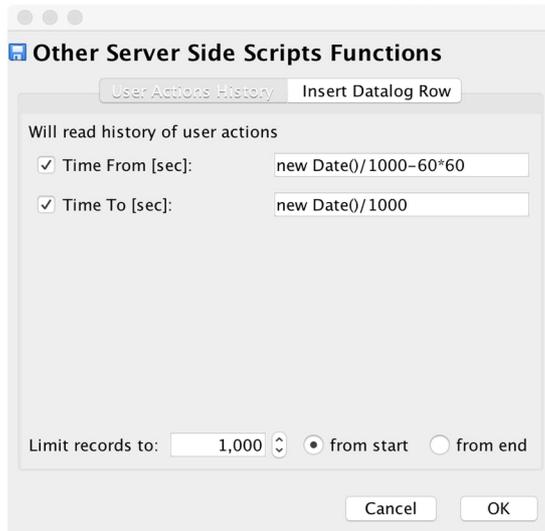
In other section you can find less common script guides. To invoke the Other guides dialog, please click on the **Other** button.



In Other guides dialog you are able to:

#### Read history of user actions

To read history of user actions, please select the tab **User Actions History**.



**Limit by time:** first you can limit the data by provided time. Time is specified in UTC format in seconds since 1.1.1970. You can enter a value or provide a variable where the value is stored.

**Limit records to:** please provide limits for the number of records loaded. You can also specify if the limit is taken from the beginning eg. **“from start”** or **from end**.

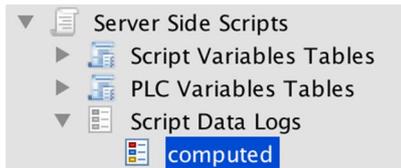
Once you are done, press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
//Read user actions from history
var options={};
options['timeFrom']=new Date()/1000-60*60;
options['timeTo']=new Date()/1000;
options['limit']=1000;
myscada.readUserActions(options, function (err,result){
  if (!err){
    //process each row of returned data
    for (i=0;i<result.length;i++)
    {
      var date=result[i].datetime;
      var text=result[i].text;
      var user=result[i].user;
      var sev=result[i].sv;
    }
  }
});
```

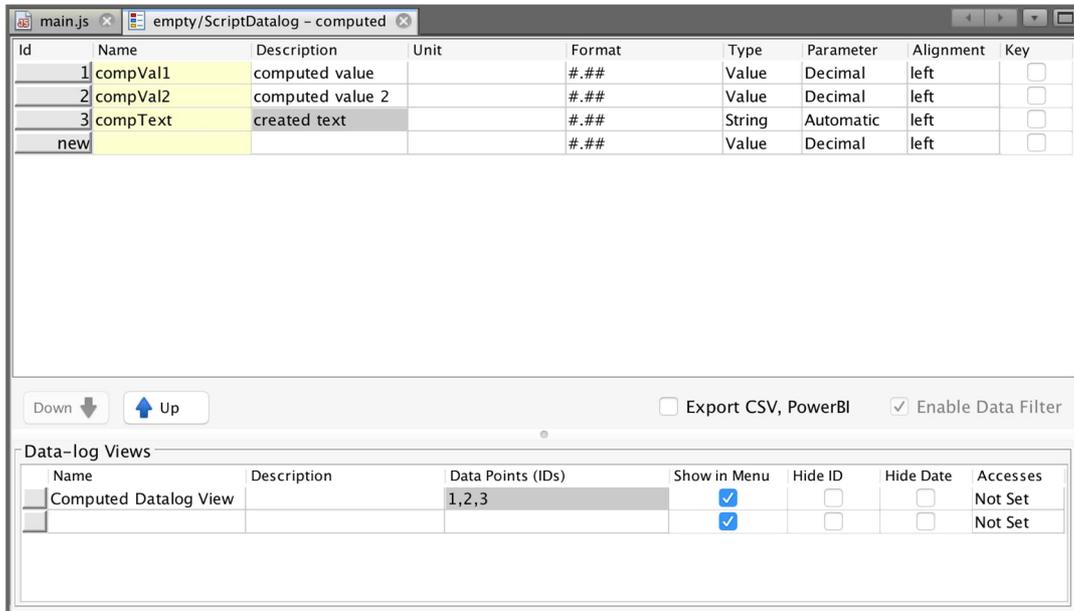
### Insert rows into the data-log

With this function, you can easily fill in data-log with data computed in server side scripts. This can be very convenient in cases when you want to log computed data instead of raw data from the PLC. We will show how to create a custom data-log and fill it with data in server side scripts.

1. Firstly, we need to create a custom data-log definition. This is an data-log which will hold our data. To do so, please navigate to Script Data Logs in server side scripts.

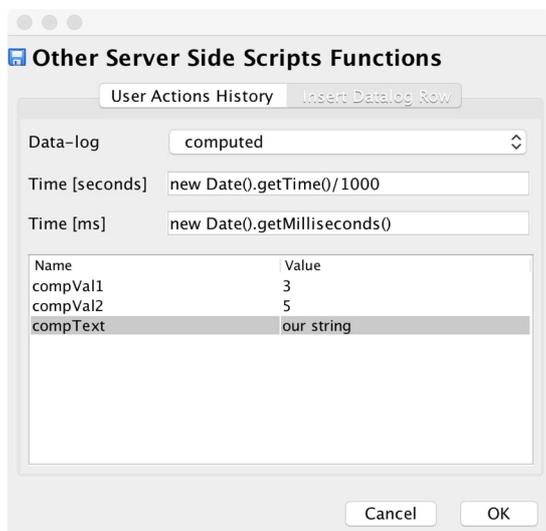


Here create a new data-log and give it a name.



As you can see, we have created data-log with 3 values, two numeric and one string. As you can see, Script data-log has the same definition as regular data-log. Only difference is, you don't link script data log to any data source, but you fill in its values from the server side scripts. Also, you can create a data-log views for this data-log as you would do with regular data-log. Once you are done with definition, we will proceed to data filling.

2. Filling the data-log with data. To fill our created script data-log with data, please click on the Other button and select an Insert Datalog Row tab.



## View and Server Side Scripts – Common Tasks

In the provided dialog, please select our created data-log by its name. Then fill in the time stamp of the record. Under this time, the record will be saved. Time has two parts, first is UTC time in seconds (since 1.1.1970) and second part is the number of milliseconds. If you leave the default values, new record will be logged at the time when your script is called.

Finally, fill in the values for each defined tag in data-log. Those can be hard coded values or variables.

Once you are done, press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

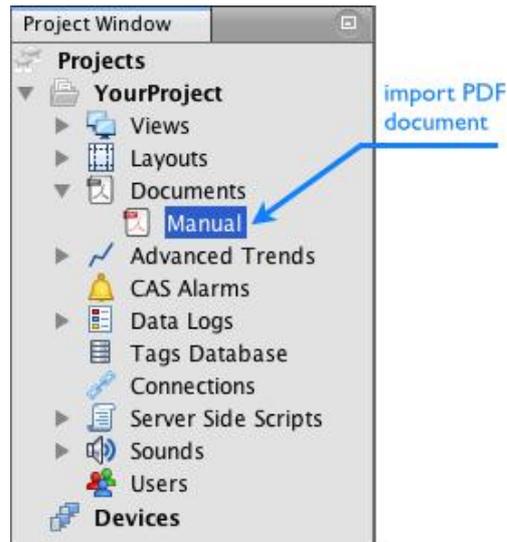
```
//Insert row into data-log computed
var options=new Object;
options['time']=new Date().getTime()/1000;
options['timeMs']=new Date().getMilliseconds();
options['dlgID']=2;
options['data']=new Array();
options['data'][0]=3;
options['data'][1]=5;
options['data'][2]="our string";
myscada.insertRowIntoDataLog(options, function (err,data){
  if (!err){
    //insert was successfull
  }
});
```

## 24 Documents

With this powerful function, you can link documents with your project. For example, you can link user manuals with components on the HMI screen, or you can attach a schema or picture to your project. As a result, you will have a complete project consisting of HMI screens and linked documentation files in one package.

Watch video describing this functionality:

<https://www.youtube.com/watch?v=4Q4HKTUFeE4>



You can attach any documents directly to your project by using the function *Import Document*.

Note that all attached documents must be in a **PDF format!**



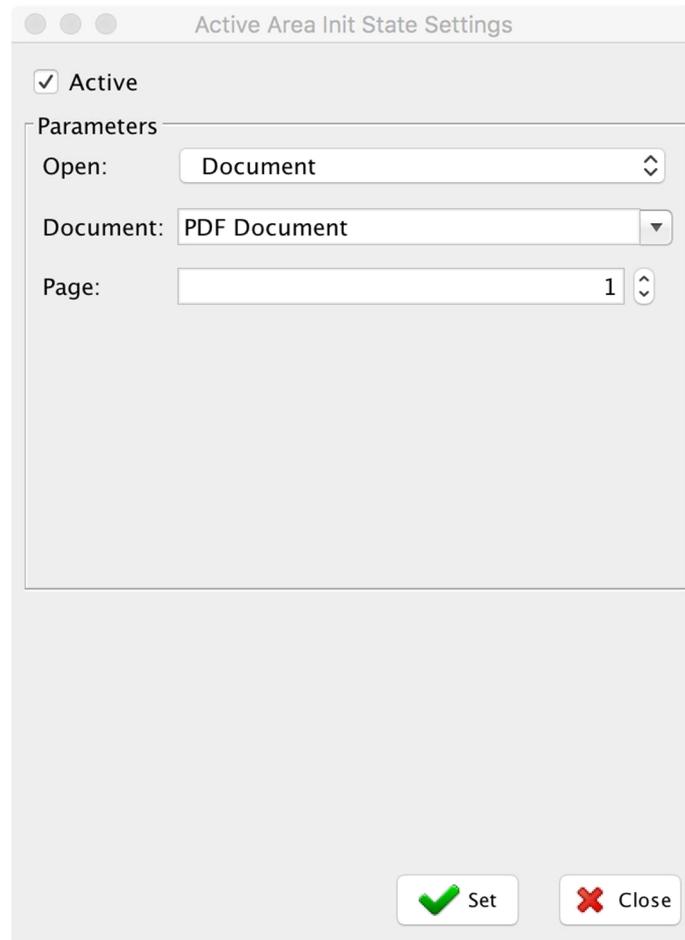
Click on the **Import document** icon in the main menu or right-click on the *Documents* in the *Project Window* menu and then select *Import*.

Once your documents are imported into your project, you can link them with any object on HMI screens. This can be done with the *Open* command function.

Showing the Document using Open Command

To show your document using open command:

1. Please select the graphical object you would like to use to open a document.
2. Go to Properties -> Commands -> Open Command and click on the "..." button. You will be presented with the Open Command Window
3. Now Select Open: Document and select your document



You can find more info about open command in the chapter [Open Command](#).

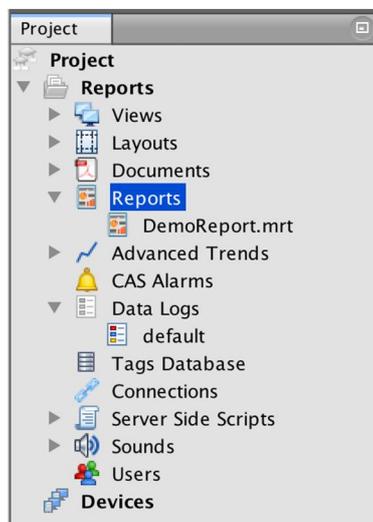
# 25 Reports

---

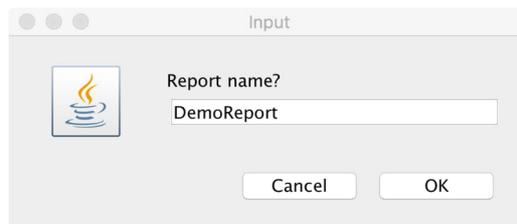
mySCADA provides a simple to use, convenient way for report creation and generation. With integrated reporting, you can create rich reports including tables, charts, and visual widgets. To get started with Reporting tools, please look at the explanatory video linked here: [LINK](#)

## 25.1 Creating Report Templates

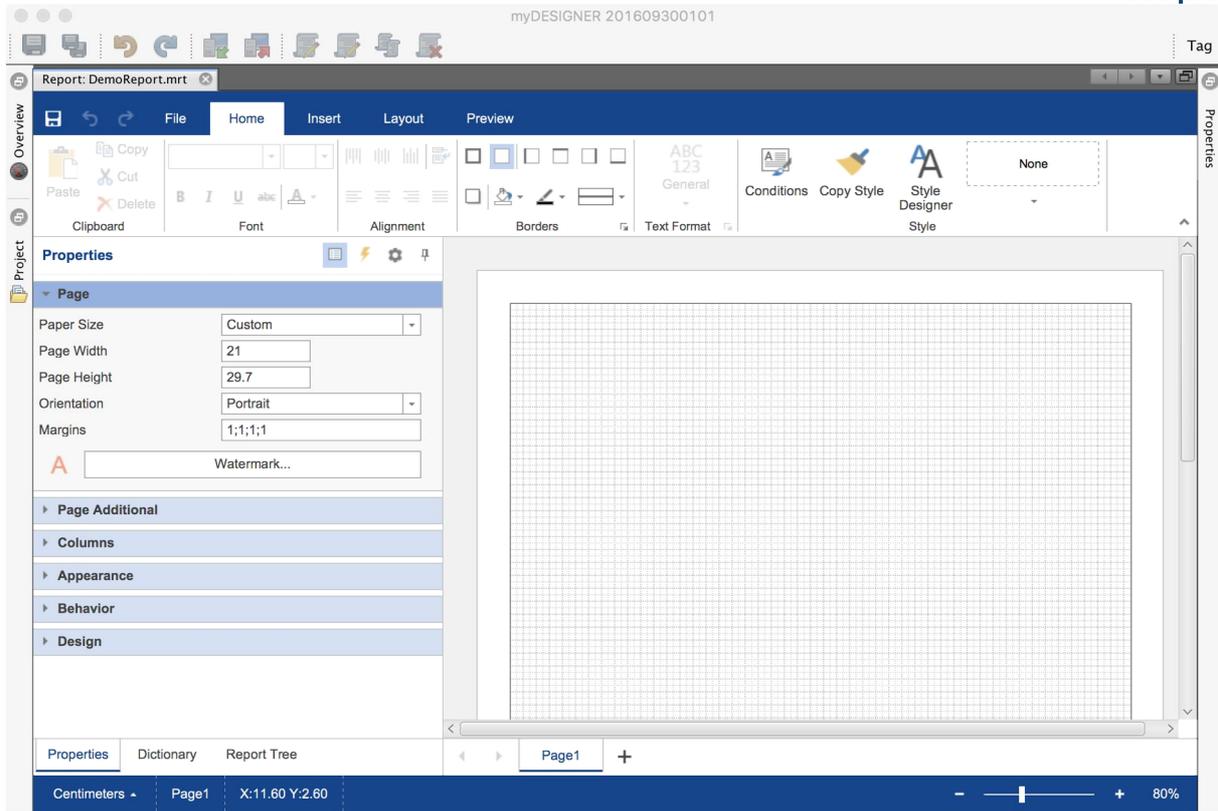
To create and integrate a report as part of your project is quite easy.



1. First of all, navigate to section Reports in your project:
2. Click on the Report section and select “Create new” on main Toolbar. You will be presented with new report dialog.



3. Enter the name of the new report and press OK.
4. New empty report template is created and report designer is automatically open.



Now, you can design your report. Once the report is designed, press save to save it.

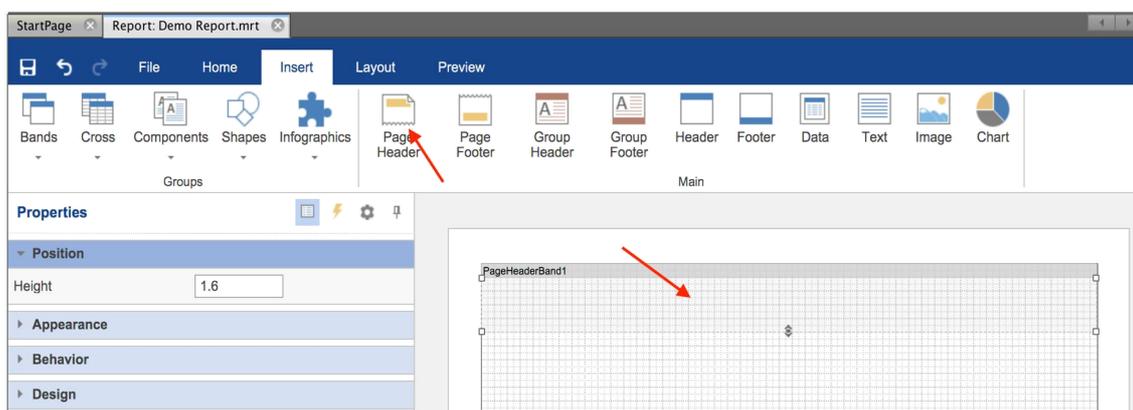
## 25.2 Designing Report

In this section, we will show, how to design a simple tabular report showing historical data-log and alarm data in a simple report. For complete set of functions and more complex examples on designing reports, please consult the Reporting Manual found here: XXXX

### Creating Header Section

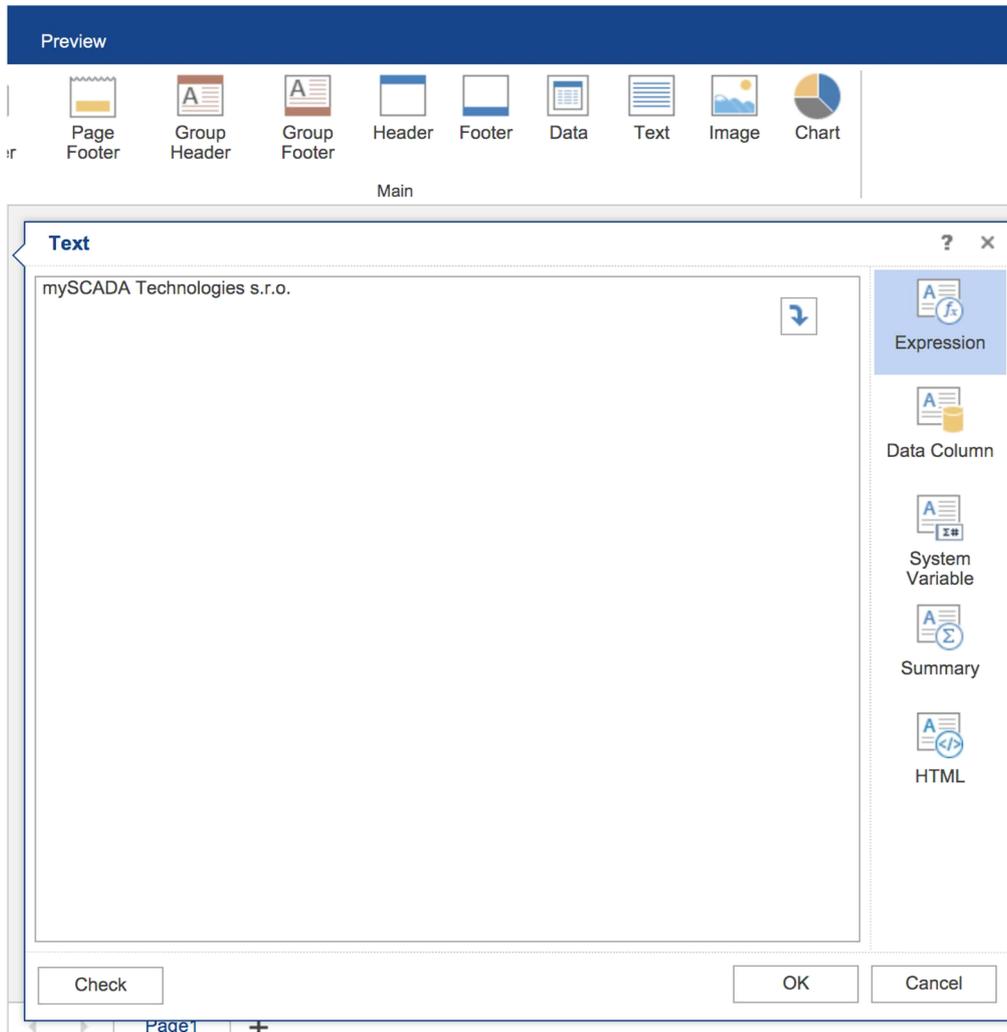
We will start by adding a company name and logo to the header section.

- a) Click on the **Insert** section in the ribbon and then click on **Page Header**. Then click into your report to insert it.

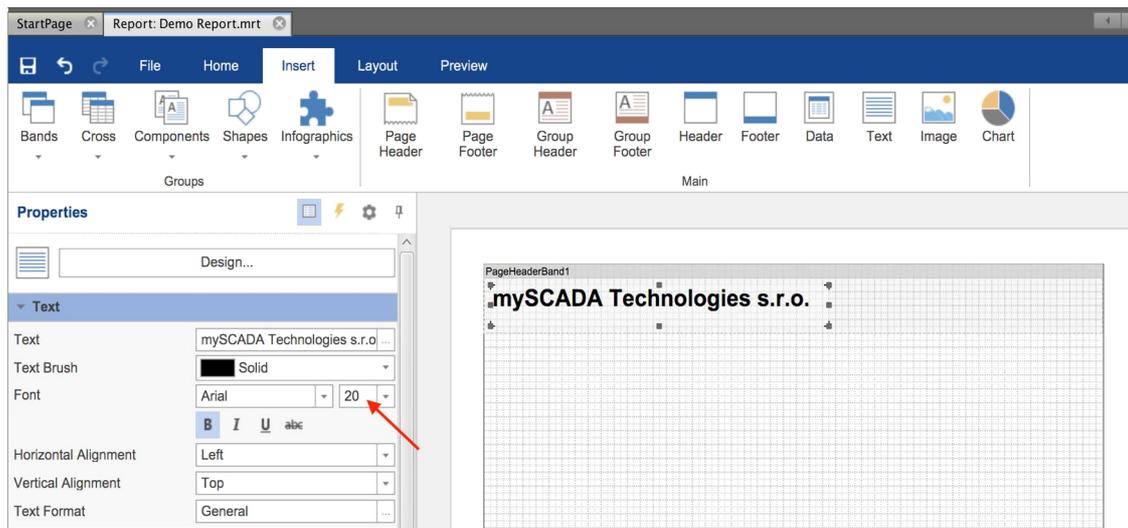


## 25.3 Inserting Text

- b) Now we will insert text into the header. Click on the **Text** icon and then click into the header in your report.



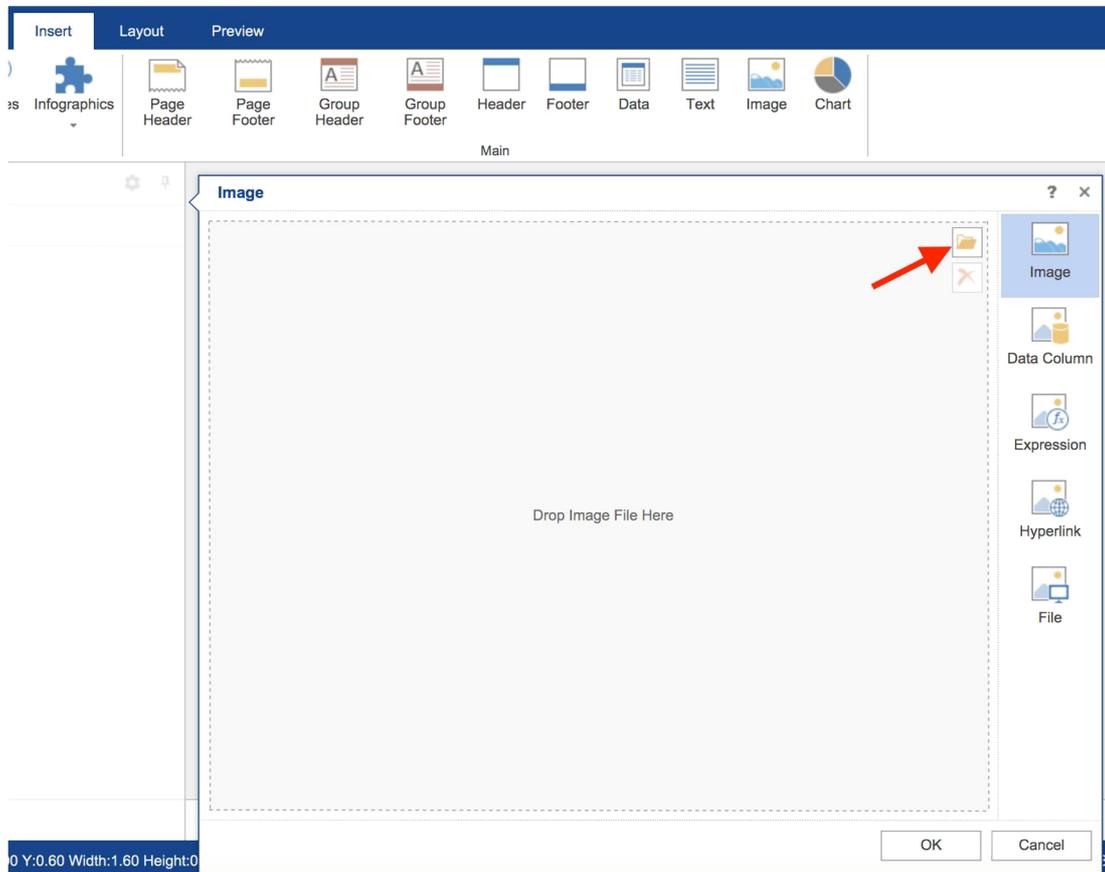
- c) Enter text into a dialog and press **OK** button.



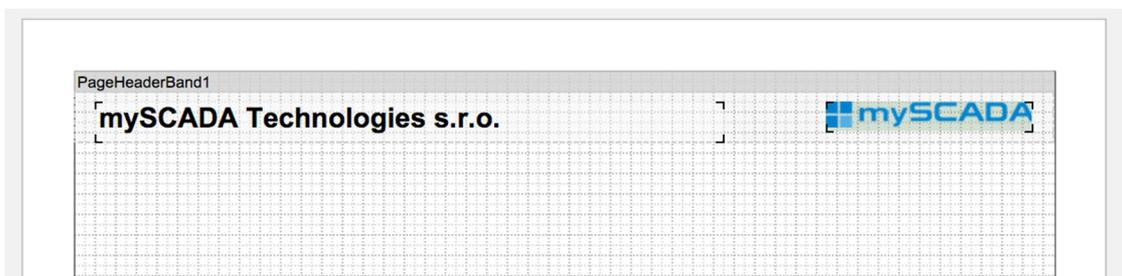
You can position text and change its properties like size, color etc. in the Properties section.

## 25.4 Adding Picture – logo

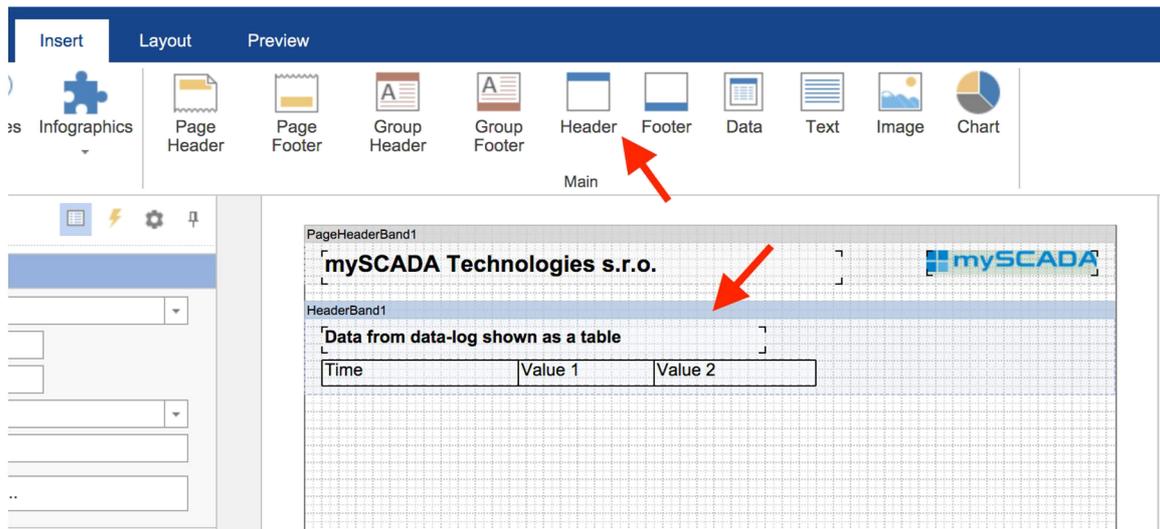
- d) Now we will enter image into the header section. Click on the image icon in the report and then click on the page header section in your report. You will be presented with image select dialog:



- e) In the image insert dialog, click on the folder icon and select your image. Once you select the image, you can preview it in the image select dialog and after you click on OK button it will be inserted into your report.

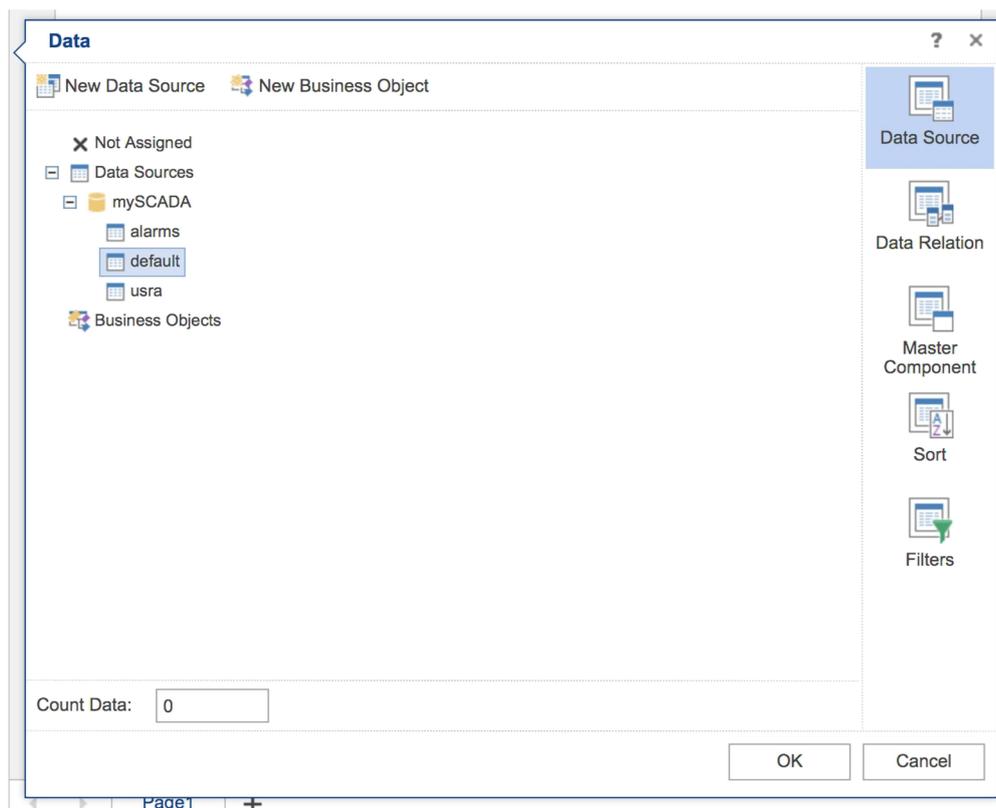


- f) Now your header section is done and we can continue to enter the historical data table into the report. Every table has 3 sections: Header, data and footer. First we will enter header into the report. To do so click on the **Header** button and then click on the report. Also add texts into the header section as shown in the image.

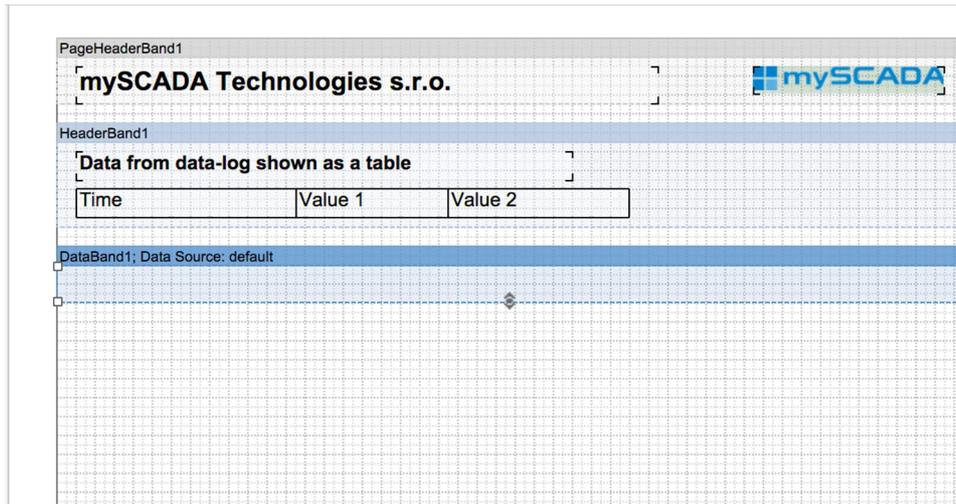


## 25.5 Table Data

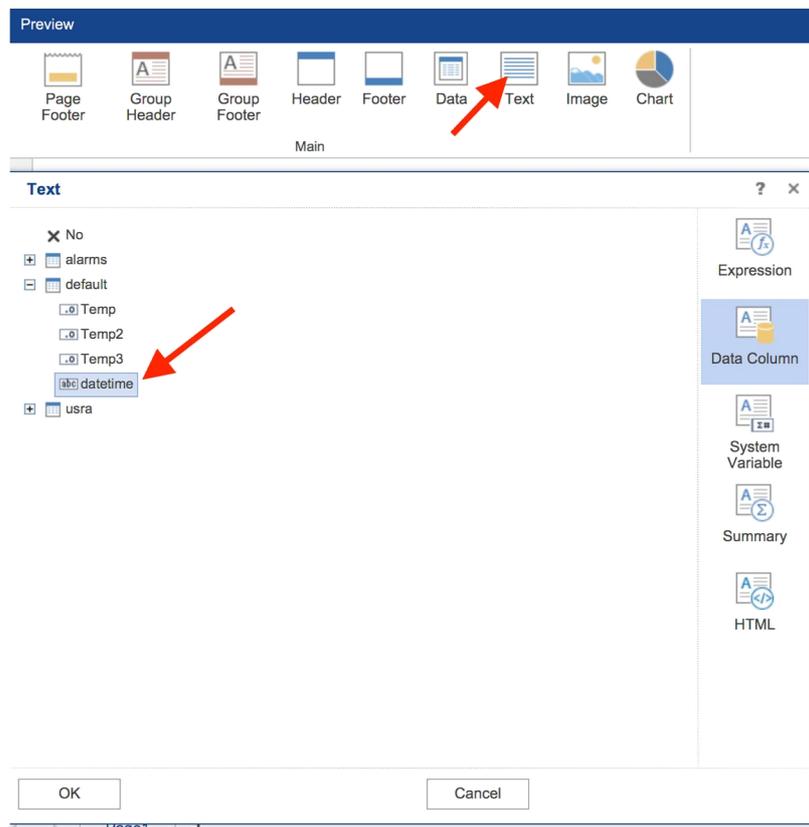
- g) Once we have header, we will insert data section. To do so, click on the **Data** button. In the provided dialog, select the default data-log as data source.



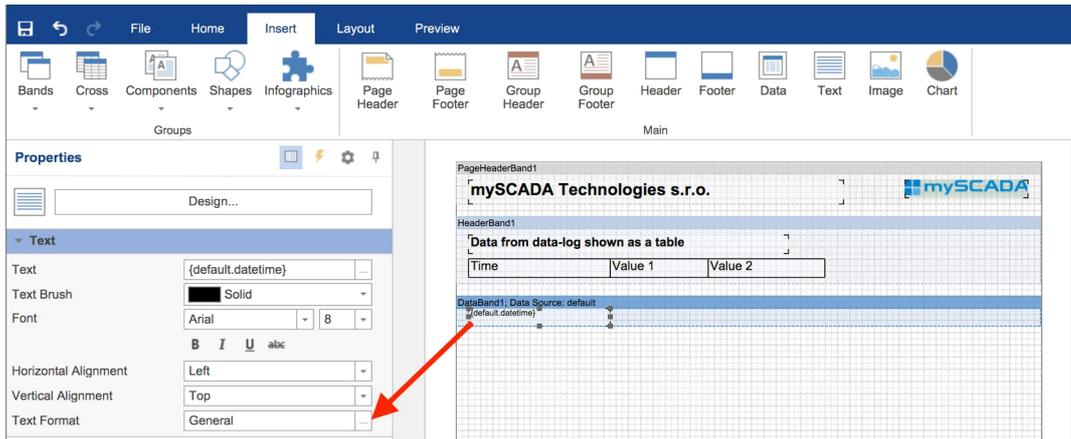
Once you click on OK button, data section is inserted into your report.



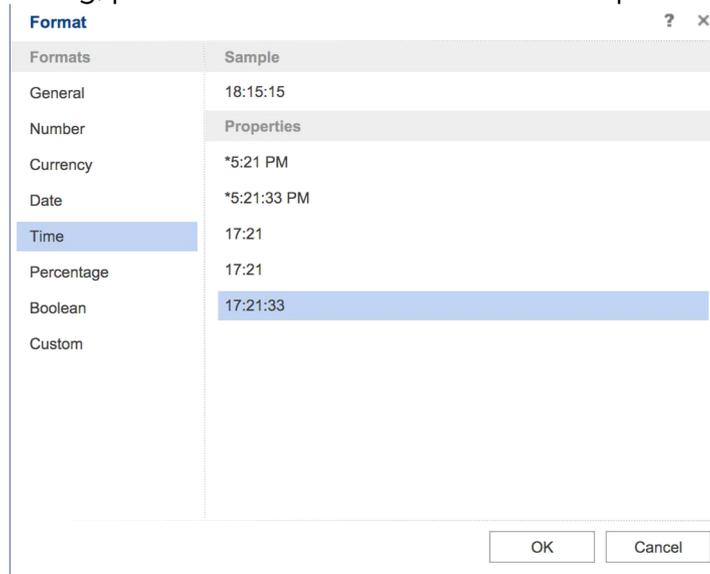
h) Once you enter the data section, you should fill it values to be shown inside. You just fill in one row template. Once the report is generated, there will be multiple rows created from this one row template. To do so, click on the Text button and then click inside the databand. Dialog to select value will open:



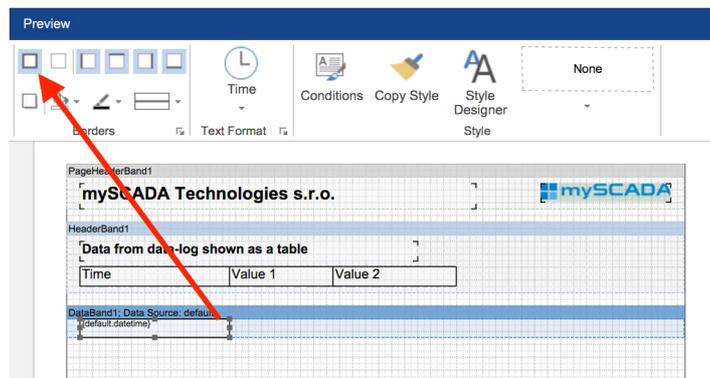
i) In the dialog, select the column from data-log you want to show. We will select and insert a datetime showing a date and time of the record. Once entered, you should format the text to show the time value. To do so, select **Text Format** in Properties.



In the popup dialog, please select time as shown in the picture:



- j) You can also add border around the table cell. To do so, click on the **border** button at the ribbon.



- k) Now add additional columns same way as you have added the date time column. You can leave the text format to general or change it to number and format as required.

PageHeaderBand1		
mySCADA Technologies s.r.o.		mySCADA
HeaderBand1		
Data from data-log shown as a table		
Time	Value 1	Value 2
DataBand1; Data Source: default		
{default.datetime}	{default.Temp}	{default.Temp2}

## 25.6 Table Summary

- I) Now we will add an footer to the table showing computed average values. To do so, click on the Footer button and then click into the report. The Footer Band will be added.

Preview



Page Footer



Group Header



Group Footer



Header



Footer



Data



Text



Image

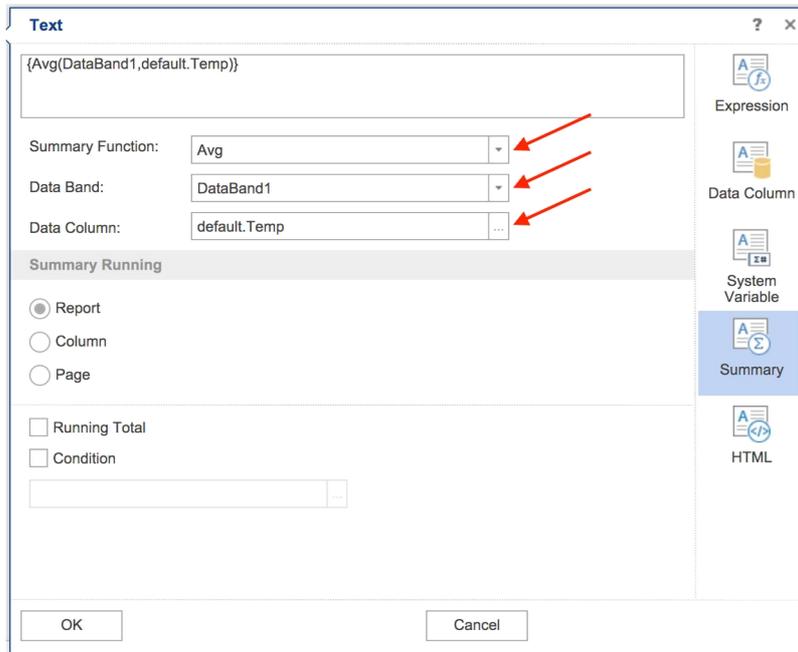


Chart

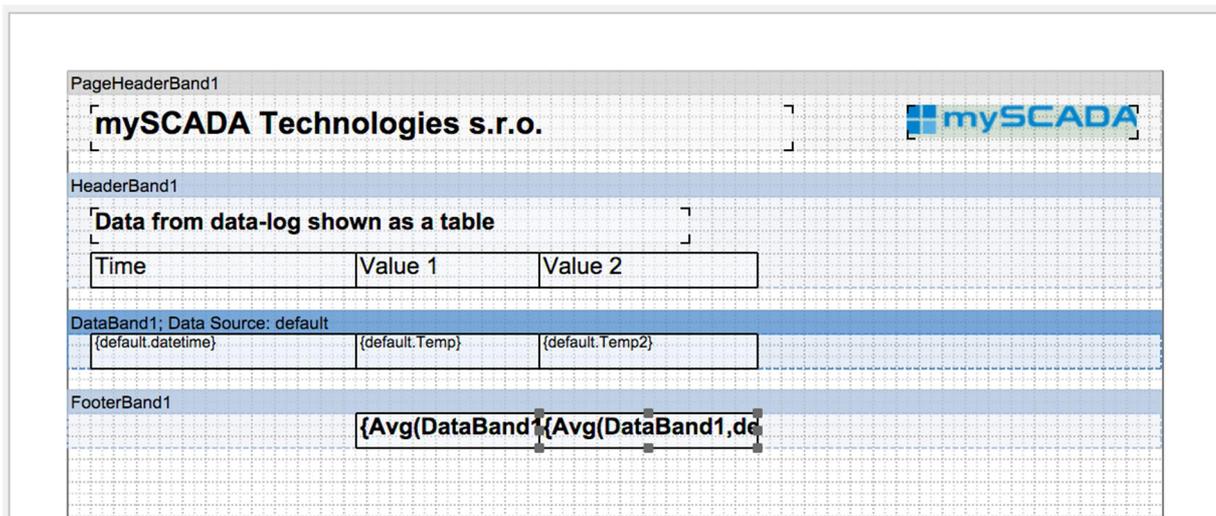
Main

PageHeaderBand1		
mySCADA Technologies s.r.o.		mySCADA
HeaderBand1		
Data from data-log shown as a table		
Time	Value 1	Value 2
DataBand1; Data Source: default		
{default.datetime}	{default.Temp}	{default.Temp2}
FooterBand1		

Now we will add two texts showing the average values. Click on the **Text** button, and then click inside the Footer Band. In the Text dialog, select **Summary**.



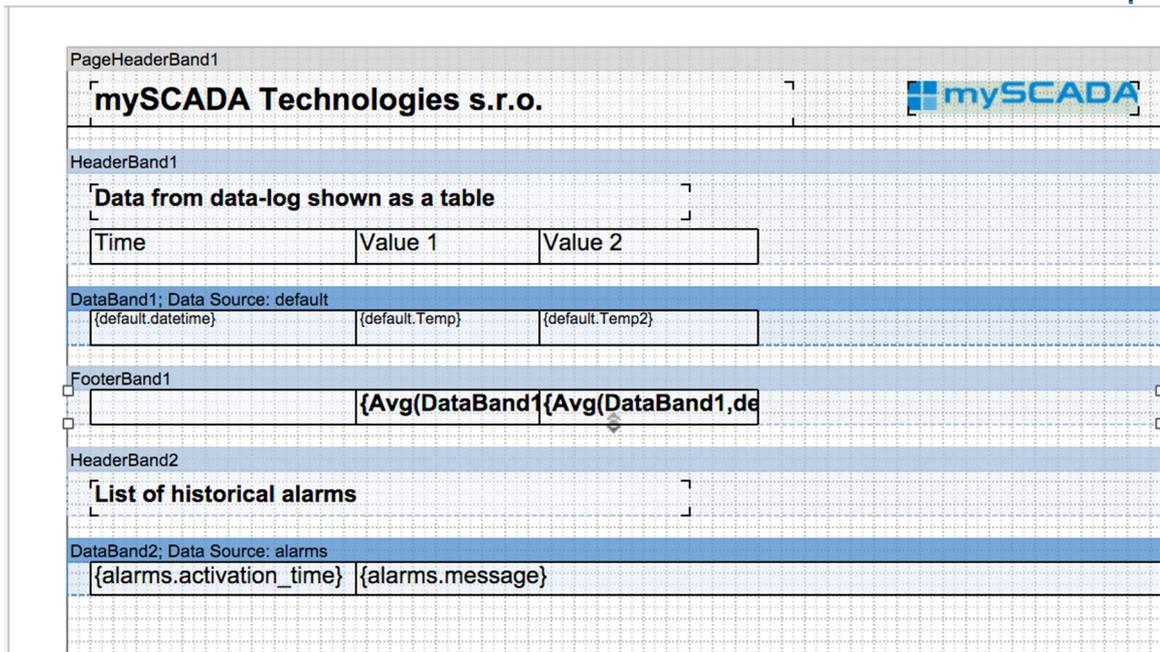
You need to fill in **Summary function**, **Data Band** and **Data Column** values. Then click on **OK** button.



Now, the footer section of the table is finished.

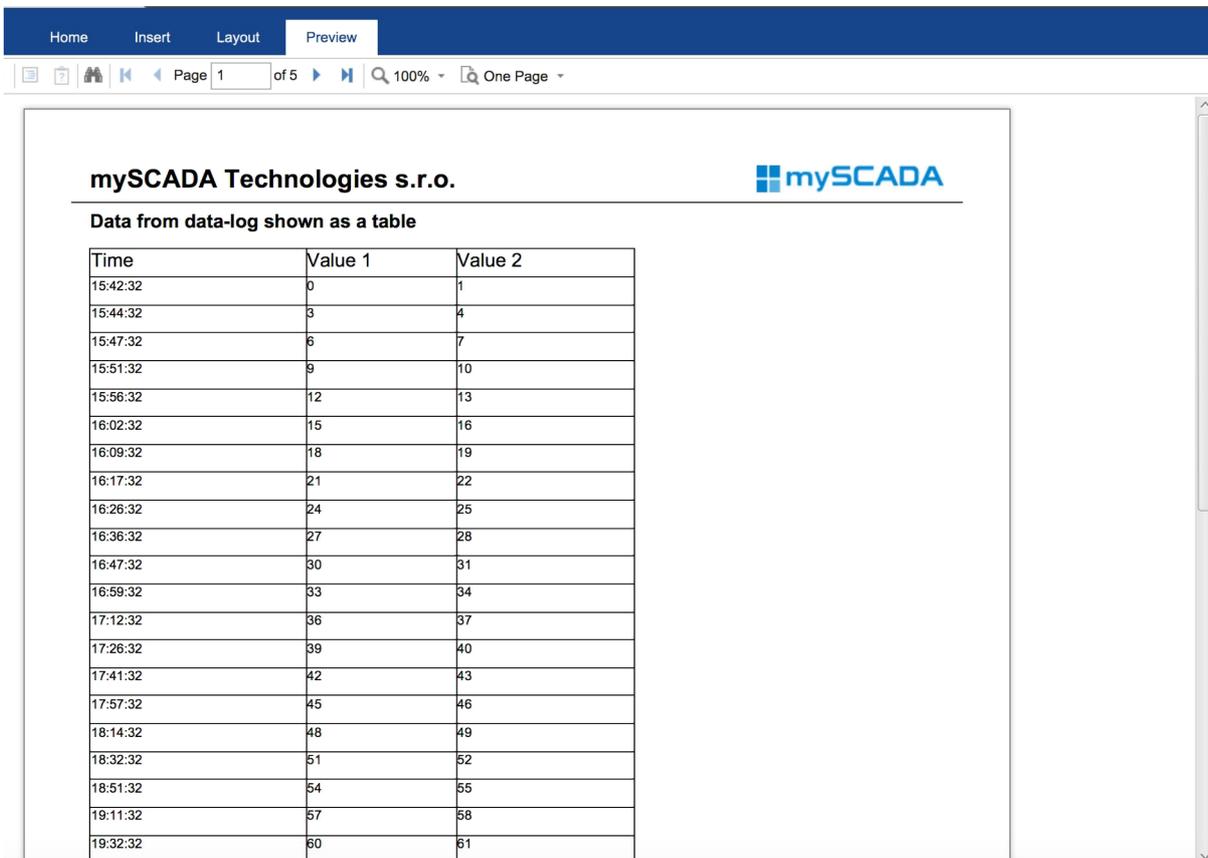
- m) Same way as creating table filled with historical data from data-logs, you can show list of historical alarms. To do so, follow the same steps as you have done when creating table from data-logs, but as a data source select **alarms**.

Our finished report looks like this:



## 25.7 Previewing Report during design

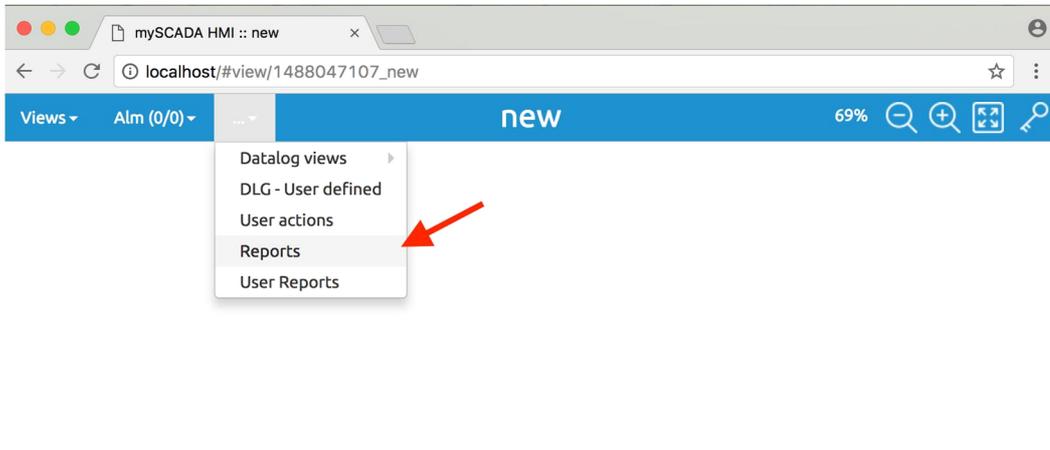
You can preview your report during design phase simply by clicking on Preview in the ribbon. Report will be rendered with demo data so you can see how your report will look like.



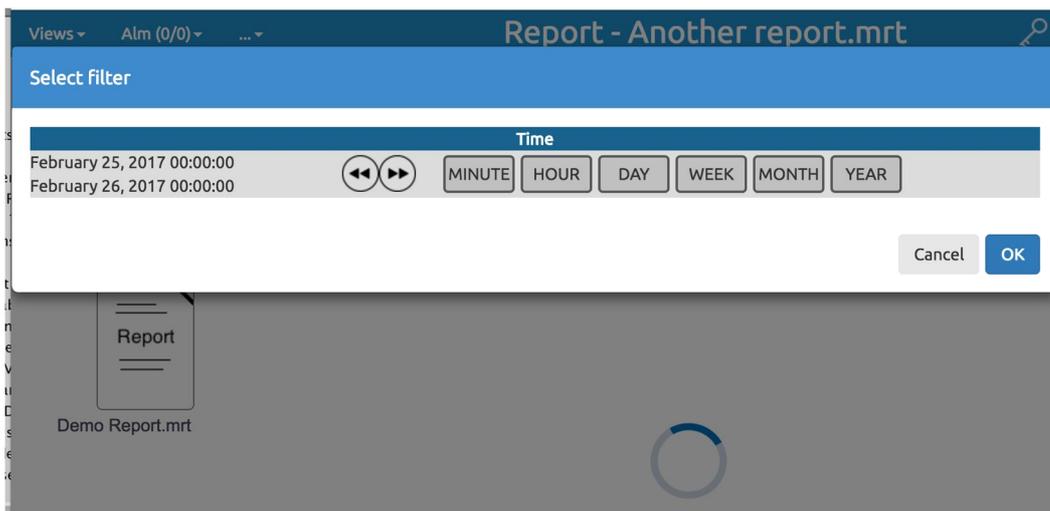
## 25.8 Showing Report in Runtime

Once you create and save your report template, it is available during runtime.

1. To show a report in runtime, simply go to the Reports section in main menu.



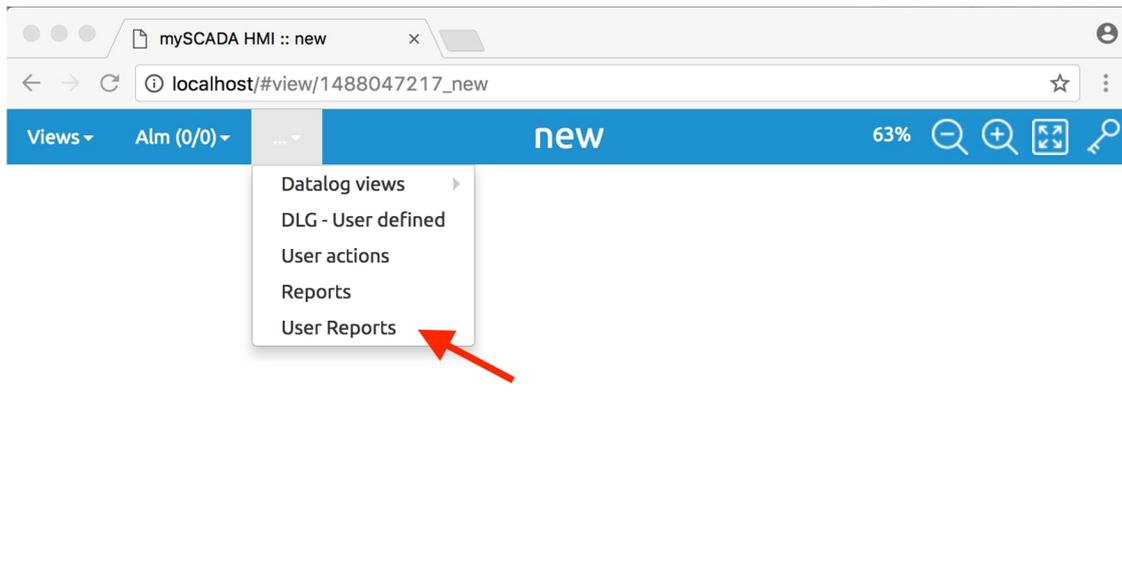
2. Now you will see all available reports on the left side. Click on the report you would like to render. You will be presented with time selection dialog:



3. Select desired start and end date and click OK. Your report will render
4. Now you can print your report or save it for later use.

## 25.9 Creating Custom Report in Runtime

User can also create custom reports on the fly. To create a custom report, user should navigate to section **User Reports**.



In this section, user has the ability to create, modify, rename and delete user reports. All the existing reports are located on the left side of the view. Each report has **rename** and **delete** icon right next to its icon.



### Deleting report

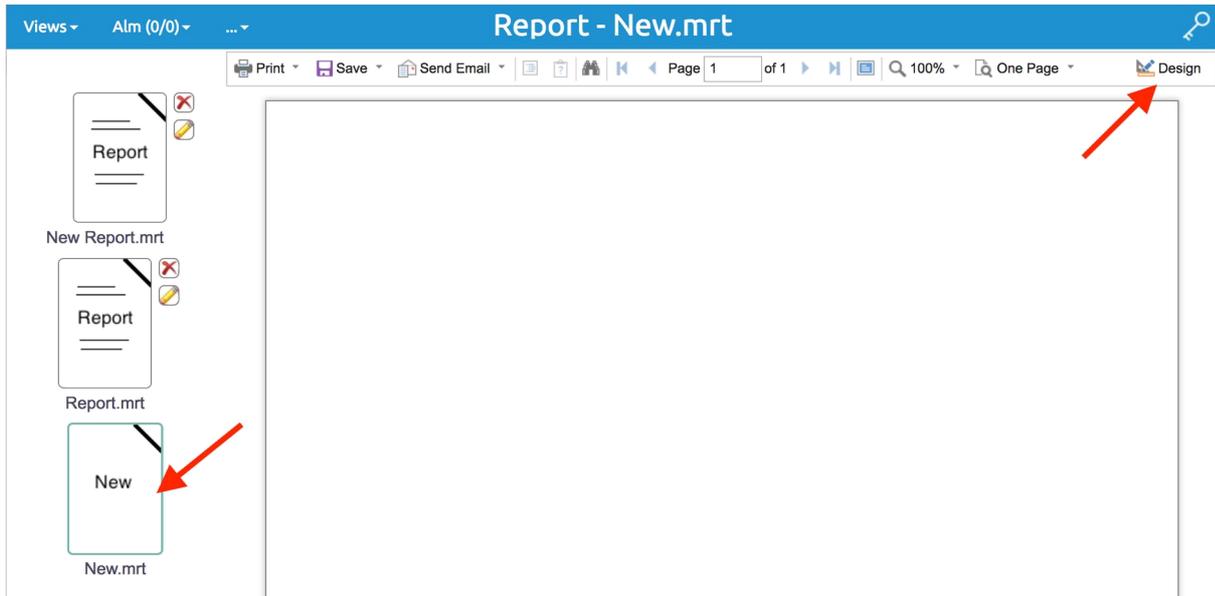
To delete existing report, click on the red cross icon next to the report.

### Renaming report

To delete existing report, click on the pen icon next to the report.

### Creating new report

To create a new report, select a **New.mrt** report from the left and then click on **Design** button.

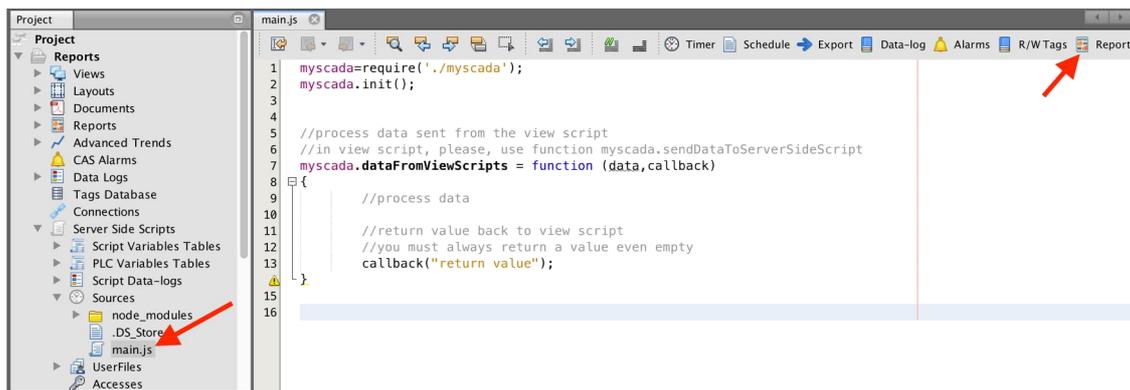


Report designer will open. Now end user can create custom reports exactly same way as done in myDESIGNER.

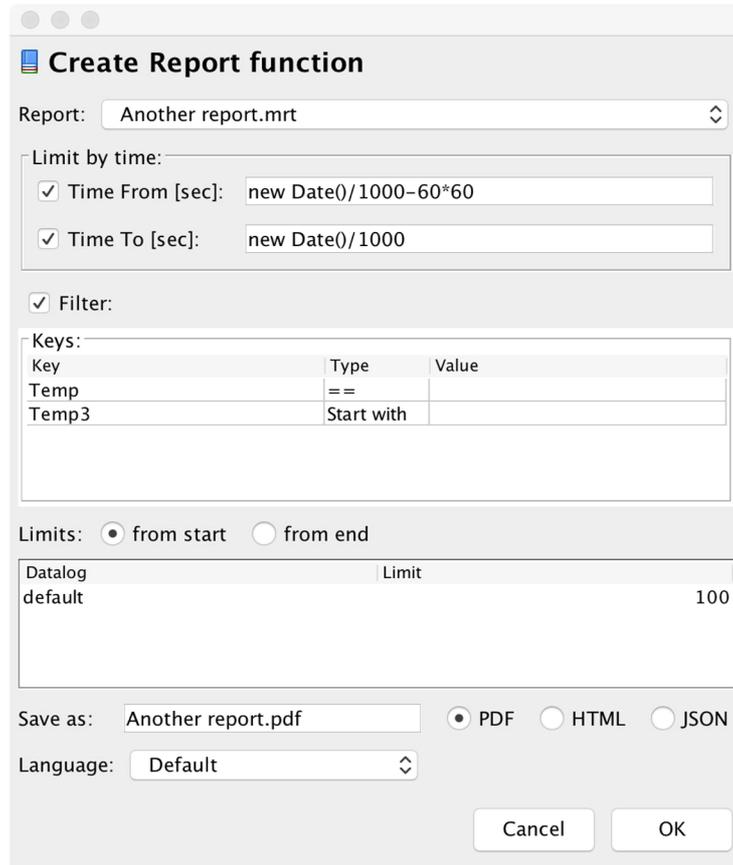
## 25.10 Creating Report on Demand

You can also create reports using server side scripts. This way, you can make periodic reports, or reports triggered by some event, options are limitless. To get started:

- 1 Create Report Template as described in previous chapter
- 2 Edit server side script – if you don't use your own module, simply edit main.js
- 3 Once you open the script, click on the Report icon



- 4 You will get the Report generation dialog. Select a report you want to generate, fill in the required values and press OK button.



Create report dialog has several sections:

**Limit by time:** first you can limit the report generation by time. Time is specified in UTC format in seconds since 1.1.1970. You can enter a value or provide a variable where the value is stored.

**Filter:** if the data-log contains a keyed value, you can limit results shown in your project by specifying a filter value. Again this can be a hard value or variable.

**Limits:** please provide limits for the number of records loaded. Each data-log or alarm values used in reports are provided in the Limits table. Please fill in the limit of rows to load. You can also specify if the limit is taken from the beginning eg. **“from start”** or **from end**.

**Save as:** provide a file name under which report will be saved to. You have an option to save a report in PDF format, in HTML format or pass it as JSON data object. Report is saved into user data folder accessible over FTP or HTTP and HTTPS.

**Language:** If your project is Multilanguage, please specify in what language report should be generated.

- 5 myDESIGNER will generate code template for you. When you run the code, the system will generate a report for you and save it into the user directory.

```
16 //generate report
17 var options=new Object;
18 options.report="Another report.mrt";
19 options.saveAs="Another report.pdf";
20 options.timeFrom=new Date()/1000-60*60;
21 options.timeTo=new Date()/1000;
22 options.exportAs="pdf";
23 options.limitEnd=false;
24 options.limits=[];
25 options.limits[1]=100; //limit no of records for datalog default
26 myscada.generateReport(options, function(err,report){
27 //after report generation
28
29 });
```

# 26 CAS Alarms

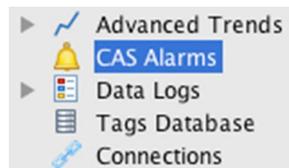
A very important feature of *mySCADA* is the ability to signal any dangerous or other important events with *Alarms*. They are an important part of most control applications as they alert operators if something goes wrong.

Watch video describing this functionality:

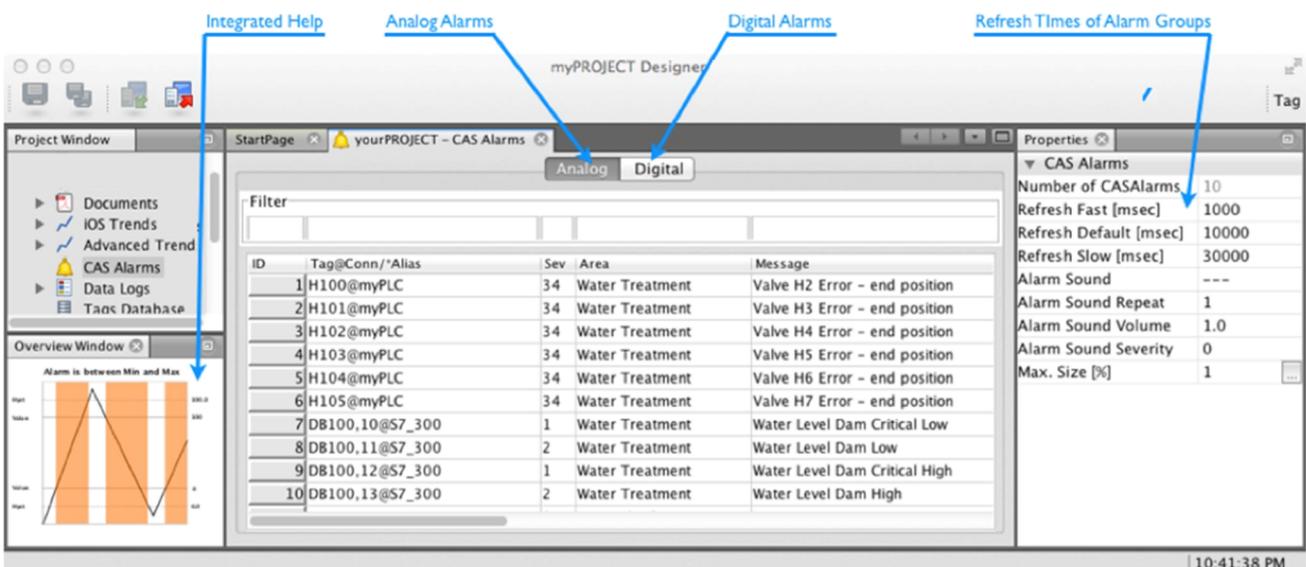
<https://www.youtube.com/watch?v=HwE-NMUmKxA>

Alarms can signal that a device or process has ceased operating within acceptable, predefined limits, or it can indicate breakdown, wear, or a process malfunction. Often it is important to keep a record of the alarms to know if they have been acknowledged.

To define alarms for your project, double-click on the *CAS Alarms* in the *Project window*.



For your convenience, the alarm definitions are split into two tables: one for **Analog** and second for **Digital** alarms:



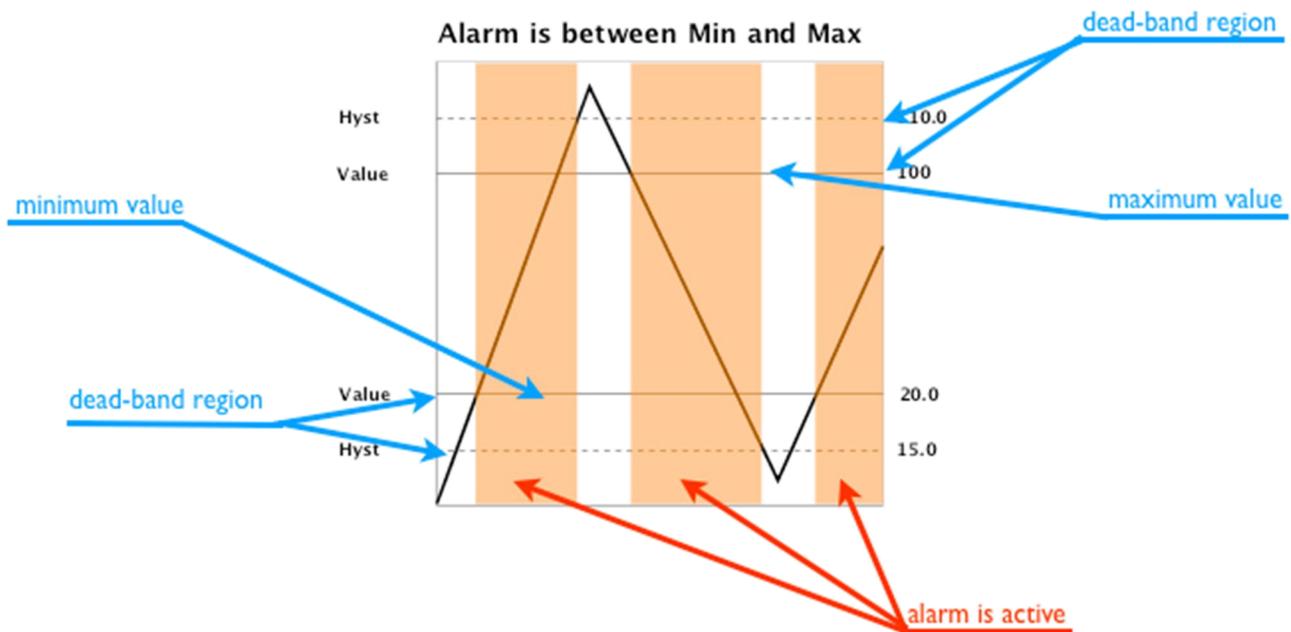
## 26.1 Digital Alarms

Digital alarms are tied to digital values read from the PLC. A digital alarm is either ON (1) or OFF (0). Instead of the thresholds, digital tags have alarm states 0 and 1.

## 26.2 Analog Alarms

Analog alarms are tied to analog values read from the PLC. Along with common parameters, which are the same for both *digital* and *analog* alarms, you specify the

minimum and maximum values for your tag or equation. You can also specify the *Dead-band* region for any value to eliminate false alarms. The number of alarms tied to one tag is not limited. In addition, you can define complex conditions, including multiple tags or mathematical conditions for a single alarm.



**Description of Alarm table fields:**

Unique ID	Automatically generated ID; needed if accessing alarms from the <i>Server-side Script</i>
Tag@Conn / *alias	Tag (Address) or <i>Equation</i> specifying data read from PLCs
Severity	Unsigned integer value specifying importance of given alarm; the lower the number, the higher the priority
Area	You can divide alarms by the geographical or virtual area they belong to; area is a string value that you can use to filter the alarms
Message	Message of your alarm
Device	Name or description of a device the alarm belongs to; one device can have multiple alarms defined
Inv (Inverse)	Inverts an alarm (if a digital alarm is inverted it will be active at 0; for analog alarm activation, area is reversed)
Hide	Enables hiding the alarm from the user; this is useful when you are using alarms as a condition in the triggered data-log; hidden alarms are not shown in the <i>Alarm window</i> and are not logged in the database.
Delay	Specifies the delay in milliseconds of how long the condition must be active to activate the alarm; this is the time hysteresis

	function
Refresh	Specifies how often your alarm will be refreshed; you can use <i>default, fast, or slow</i> . You can change the refresh values for each group in the <i>Properties window</i> .
Format	Defines how the value will be shown in the alarm table; use the '###.#' format for specification
e-mail	Sends an email upon each alarm activation or deactivation
SMS Act	Sends a message upon each alarm activation
SMS Deact	Sends a message upon each alarm deactivation
G0 - G9	Checks appropriate user group to receive alarms by email or SMS

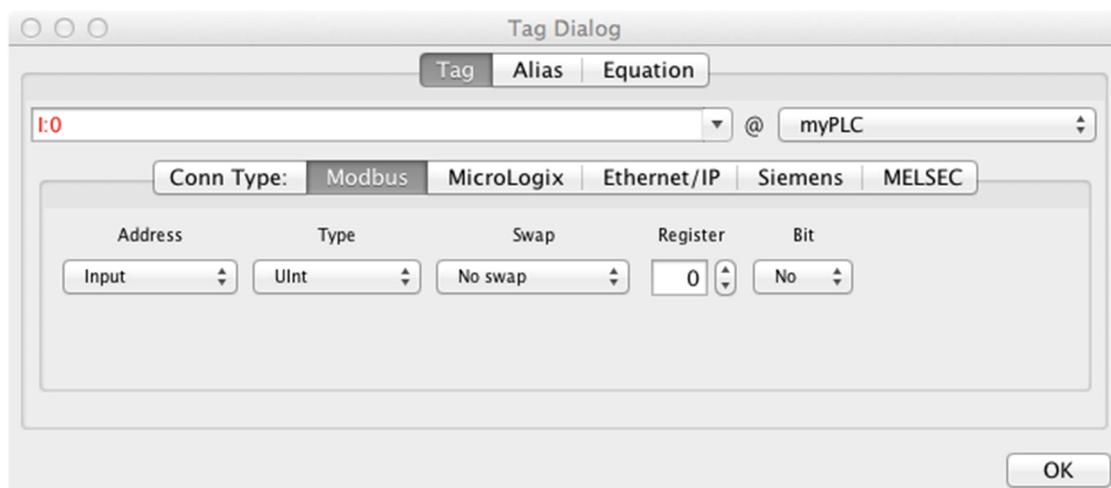
### Unique ID

Each alarm has a *Unique ID*, which is created when you define a new alarm; its value is fixed once the project is put into the runtime mode for the first time. This value is unique for all alarms, meaning that each alarm has its own *Unique ID* that is saved in the alarms' definition file. Once this ID has been created, it will remain until the alarm is removed from the definition table.

### Tag@Conn/\*alias

Each alarm must be connected to a tag or equation. The *Tag* is the value read from PLC (or your computed value from the script); with *Equation*, you can tie your alarm to multiple tags or evaluate complex formulas.

### Specifying a tag:



## Specifying a complex formula



You can create multiple alarms tied to one tag, as long as the alarm descriptions are different so that alarms operate correctly.

### *Severity*

Alarms can range in severity from 0 (most severe) up to an unsigned integer value (least severe), to indicate different levels of importance.

For example, an alarm with severity 10 might warn that a tank is half full of liquid, while another alarm with severity 5 indicates that the tank is about to overflow. Both alarms monitor the same tag but have different severity levels.

When you set up the alarm severity, specify what the severity levels mean and what actions they will trigger. Severity determines the order in which alarms are displayed in the alarm list.

### *Alarm areas*

The alarms can be grouped in different areas so that they can be displayed in the alarm window based on the area to which they belong. This may be helpful and enable you to divide the alarms according to the different plant zones they come from.

### *Message*

Alarm messages report information about alarms.

### *Device*

You can define multiple alarms for a single device. In the live alarm view or during a browsing of alarm history, you can filter your data based on device value.

### *Minimum and Maximum values*

Minimum and maximum values are available for analog alarms only. By default, you specify the region when the alarm will be active. So if you would like to activate an alarm when the level in a tank is equal to 90 and stop the alarm at level 100, your minimum value will be 90 and the maximum value will be 100.

### Inv (Inverse)

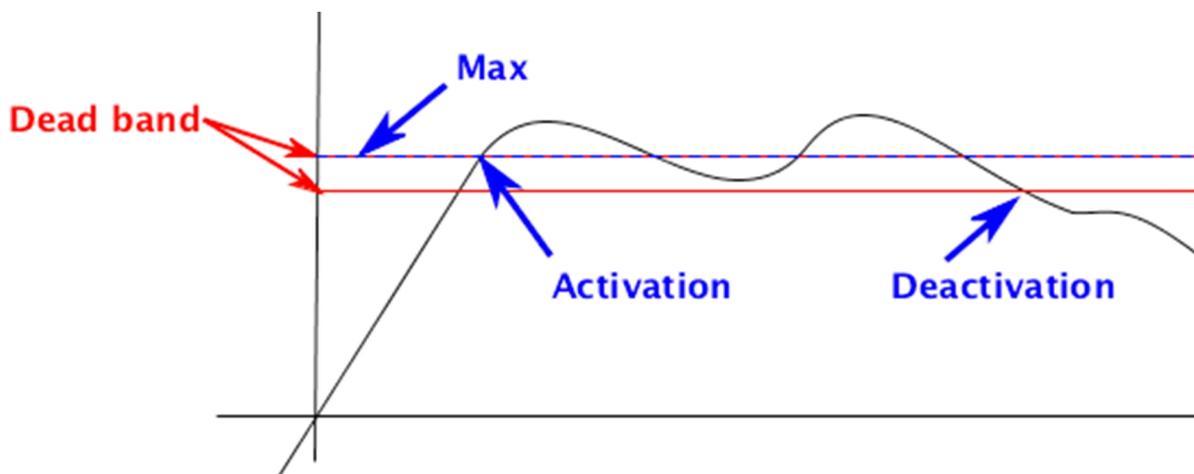
This parameter lets you invert your alarm definition. For a digital alarm, the alarm is activated when the value is equal to 0 and deactivated when equal to 1. For analog alarms, when 'inverse' is active, you specify the minimum and maximum values of the region when the alarm is **NOT** active.

### Recipients (GO up to G9)

Through these properties, you can select the recipient user group to which the message, SMS, E-mail, etc. is to be sent. The user profile, which is defined through the 'Users' settings, must contain a telephone number or E-Mail for sending messages.

### Dead-band ☒

With some measured values, such as line pressure, tag values can fluctuate rapidly above and below a critical threshold. Where such conditions exist, you can create a dead band as a buffer to prevent the fluctuations from triggering unnecessary alarms.



### Specifying frequency of alarm checks

The system does not check for alarms more frequently than the *Refresh* update rate specified in the alarm definition.

Match the maximum update rate to the rate at which you expect tag values to change. For example, if you are monitoring temperatures that fluctuate slowly, check alarms less frequently than when you have a manufacturing process that changes temperature rapidly.

You can specify one of three possible refresh rates: *default*, *slow*, or *fast*. Each refresh rate group can be changed in the *Properties window*.

Message	Device	Min	Value min	Dead band min	Max	Value max	Dead band max	Inv	Refresh
Default message for alarm	Device	<input checked="" type="checkbox"/>	40.0	0.0	<input checked="" type="checkbox"/>	60.0	0.0	<input type="checkbox"/>	slow
on		<input checked="" type="checkbox"/>	0.0	0.0	<input checked="" type="checkbox"/>	100.0	0.0	<input type="checkbox"/>	slow

## 26.3 Alarm Window

The alarm window allows the operator to perform complete management of the technology alarms. This window allows you to visualize the alarms present in the technology or in a restricted area of the technology.

The alarm window can display all the technological alarms or a set of alarms arranged by the user-defined areas. If necessary, the user can click on the filter button and fill in the area name.

#	MESSAGE	STATUS	SEV	AREA	DEVICE	ACT TIME	DEACT TIME	ACK
5	Maximal Power	TEXT N...	0	Hydro p...	Electrics	September 26, 2016 22:3...	September 26, 2016 22:3...	
4	High Upper Level	TEXT N...	50	Hydro p...	Levels	September 26, 2016 22:3...	September 26, 2016 22:3...	
3	Generator disconnec...	TEXT N...	0	Hydro p...	Electrics	September 26, 2016 22:2...	September 26, 2016 22:3...	
2	Low Head	TEXT N...	0	Hydro p...	Levels	September 26, 2016 22:2...	September 26, 2016 22:3...	
1	Low Lower Level	TEXT N...	50	Hydro p...	Levels	September 26, 2016 22:2...	September 26, 2016 22:2...	

### *Alarm acknowledgement*

The operator can acknowledge the alarms displayed in the alarms summary. This does not correct the alarm triggering condition but indicates that the operator is aware of it.

### *Alarm suppression*

You can suppress alarm monitoring for one or multiple alarms. This is useful for testing, repairing, or maintaining a piece of equipment. Click on the **Suppress** button to suppress alarm monitoring. To view the list of the tags not being monitored, use the *Suppressed list*. You can also turn monitoring back on from this list.

### *Sorting and filtering in run-time*

By default, alarm information in the alarm summary is sorted first by date and time, then by severity, and finally by area name.

This means that alarms are presented in a chronological order: if two or more alarms have the same time and date, they are presented in order of severity; if any alarms have the same time and date and the same severity, they are presented by the area name.

## 26.4 Alarm History

The *mySCADA* engine automatically (if not disabled) logs your alarms into history. Every alarm action is logged with all relevant data such as current time (precise to 1 millisecond). You can browse through the alarm history in the *Alarm History* window. Along with direct data browsing, you can filter your data based on the criteria.

You can also export the alarm history into an XLS file.

History alarms										
#	MESSAGE	STATUS	SEV	AREA	DEVICE	ACT TIME	DEACT TIME	ACT VAL	DEACT VAL	USER
535	Generator disconnec...	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:34:27		0.0	1.0	system
1	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 16:35:46		260.0	259.2	system
533	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 22:31:57	September 26, 2016 22:32:53	260.0	259.2	system
4	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 16:35:46	September 26, 2016 16:37:01	260.0	258.3	system
531	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:31:57		260.0	259.6	system
7	Generator disconnec...	ACT	0	Hydro power plant	Electrics	September 26, 2016 16:38:39		0.0	1.0	system
529	Generator disconnec...	DEACT	0	Hydro power plant	Electrics	September 26, 2016 22:29:14	September 26, 2016 22:30:23	0.0	1.0	system
9	Generator disconnec...	DEACT	0	Hydro power plant	Electrics	September 26, 2016 16:38:39	September 26, 2016 16:39:18	0.0	1.0	system
527	Generator disconnec...	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:29:14		0.0	1.0	system
11	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 16:40:52		260.0	258.3	system
525	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 22:26:38	September 26, 2016 22:27:42	260.0	259.6	system
14	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 16:40:52	September 26, 2016 16:42:23	260.0	258.4	system
523	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:26:38		260.0	259.8	system
17	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 16:46:00		260.0	258.4	system
521	Generator disconnec...	DEACT	0	Hydro power plant	Electrics	September 26, 2016 22:24:08	September 26, 2016 22:25:03	0.0	1.0	system
20	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 16:46:00	September 26, 2016 16:47:43	260.0	258.0	system
519	Generator disconnec...	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:24:08		0.0	1.0	system
23	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 16:51:11		260.0	258.0	system
26	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 16:51:11	September 26, 2016 16:52:59	260.0	258.2	system
516	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 22:21:16	September 26, 2016 22:22:35	260.0	259.8	system
29	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 16:56:26		260.0	258.2	system
32	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 16:56:26	September 26, 2016 16:58:11	260.0	258.2	system
513	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:21:16		260.0	258.5	system
35	Maximal Power	ACT	0	Hydro power plant	Electrics	September 26, 2016 17:01:45		260.0	258.2	system
511	Generator disconnec...	DEACT	0	Hydro power plant	Electrics	September 26, 2016 22:19:07	September 26, 2016 22:19:35	0.0	1.0	system
38	Maximal Power	DEACT	0	Hydro power plant	Electrics	September 26, 2016 17:01:45	September 26, 2016 17:03:19	260.0	259.5	system
509	Generator disconnec...	ACT	0	Hydro power plant	Electrics	September 26, 2016 22:19:07		0.0	1.0	system

## 26.5 CAS Alarms in View - Example

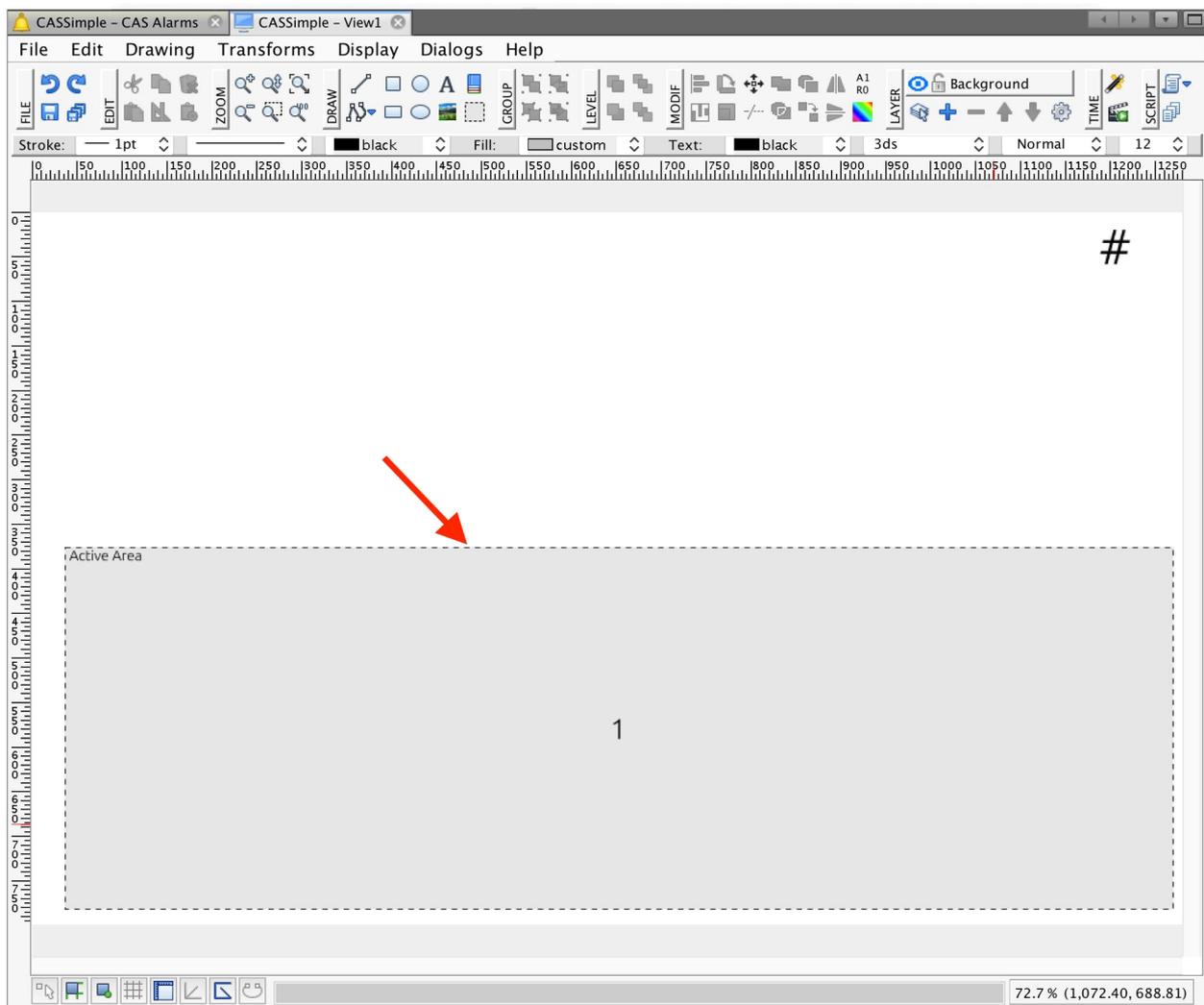
The following example will show you how to embed the CAS Alarms window in the view.

1. Fill in the CAS Alarms

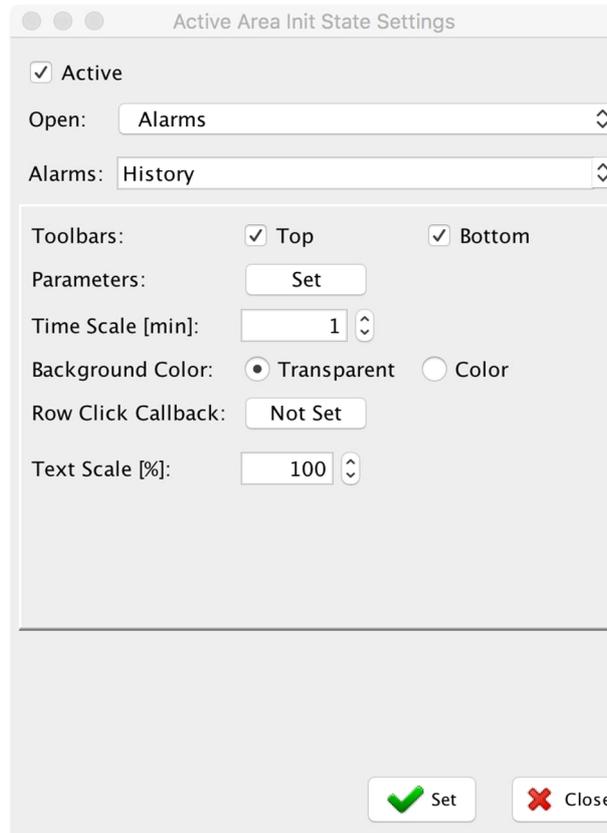


ID	Tag@Conn/*Alias	Sev	Area	Message	Device	Min	Value n
1	value1@script	0		Alarm ON		<input checked="" type="checkbox"/>	
new		0		on		<input checked="" type="checkbox"/>	

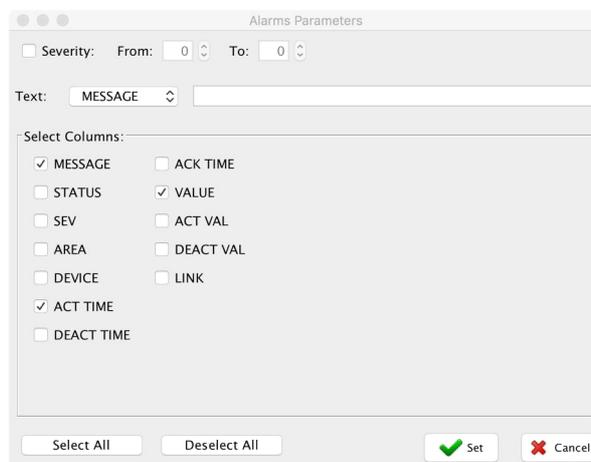
2. Create your view and place inside Active Area



3. In Active Areas' **init** state, set it to CAS Alarms -> History



4. Specify which columns should be visible in the CAS Alarms table by clicking on the Parameters button. We only want to show the alarm message, activation time, and activation value.



5. Save your project and test.

The screenshot shows the CAS Alarms interface. At the top, there is a blue header bar with 'Views' and 'Alm (1/1)' on the left, 'View1' in the center, and '80%' with zoom and search icons on the right. Below the header, there is a toolbar with icons for 'LIMIT: 10000', 'ACT', 'DEA', 'ACK', 'SUP', 'UNS', 'SEVERITY', 'TEXT', and 'Export'. The main area displays a table of alarm messages with the following columns: '#', 'MESSAGE', 'ACT TIME', and 'ACT VAL'. The table contains 12 rows of data, all with 'Alarm ON' messages and a value of '5.0'. The messages are sorted by time, with the most recent at the top. At the bottom of the table, there is a scroll bar and a '10 minutes' indicator.

#	MESSAGE	ACT TIME	ACT VAL
55	Alarm ON	October 11, 2016 00:22:49	5.0
54	Alarm ON	October 11, 2016 00:22:39	5.0
53	Alarm ON	October 11, 2016 00:22:39	5.0
52	Alarm ON	October 11, 2016 00:22:29	5.0
51	Alarm ON	October 11, 2016 00:22:29	5.0
50	Alarm ON	October 11, 2016 00:22:19	5.0
49	Alarm ON	October 11, 2016 00:22:19	5.0
48	Alarm ON	October 11, 2016 00:22:09	5.0
47	Alarm ON	October 11, 2016 00:22:09	5.0
46	Alarm ON	October 11, 2016 00:21:59	5.0
45	Alarm ON	October 11, 2016 00:21:59	5.0
44	Alarm ON	October 11, 2016 00:21:49	5.0

DOWNLOAD DEMO PROJECT HERE:

[ftp://nsa.myscada.org/history/projects/example/Cas\\_alarms\\_in\\_view.mep](ftp://nsa.myscada.org/history/projects/example/Cas_alarms_in_view.mep)

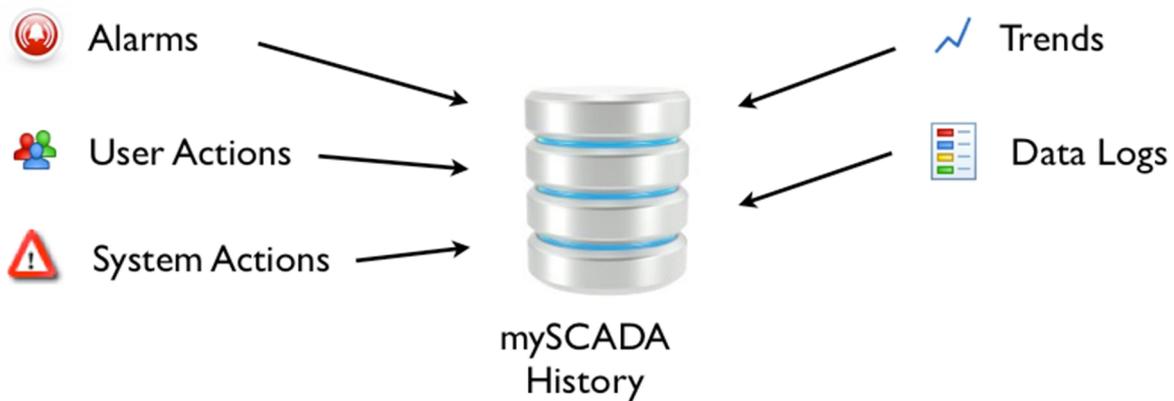
# 27 Data Logging

You can log and access a complete alarm history, all user actions, and any technological data you wish to log. The historical data are grouped into logical sections called *Data-logs*.

*Data log* has many options you can set up to fine tune your logging options. You can easily optimize your data logs for speed and storage.

Data logs can be periodic or event driven.

Using *pre* and *post* event buffers, you can save a collection of data before a specified event has happened.

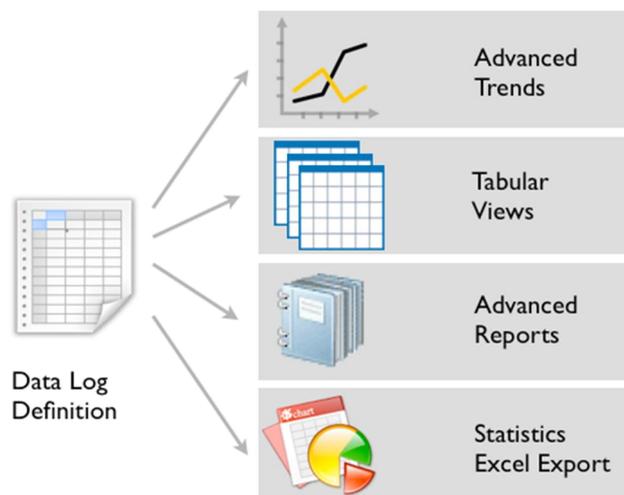


## 27.1 Data-logs

Watch video describing this functionality:

[https://www.youtube.com/watch?v=wzxKQ\\_AUCn8](https://www.youtube.com/watch?v=wzxKQ_AUCn8)

You can log any data or information available in *mySCADA*. For user convenience and easy access, data are grouped in the *data-logs*. You can think of a data log as a collection of similar data. It might be, for example, a set of temperatures read each second from the PLC, motor start-up voltage and current logged every 100 milliseconds, the running hours of machinery, operator actions, or computed production statistics. You can also log any user-defined variables from *Server-side Scripts* via a virtual PLC.



**Data-logs:** data sets grouped together. Each data log has a set of parameters, such as log period, *pre* and *post* event buffers, and so on. Data logs are defined by the data you wish to read and log. Data is the collection of data points. Data points can be variables read from the PLCs, user-defined variables, and computed statistics from *Server-side Scripts*.

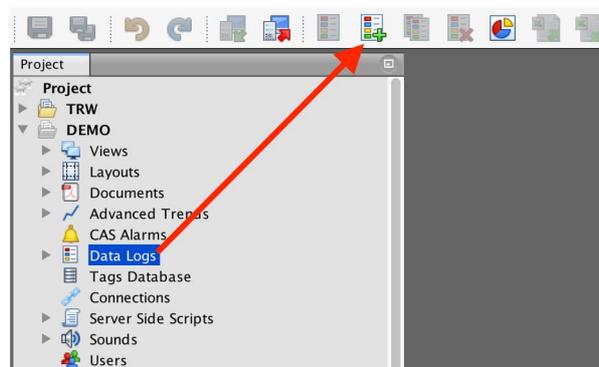
**Data point:** can be either a numerical value, string, or date. Numerical values can be of any numerical type such as *Boolean*, *Integer*, *Float*, *Double*, *Signed*, or *Unsigned*. Numerical values are always automatically converted and logged as double values. This way you do not have to worry about the data type and its conversions.

There are two types of *Data-logs*:

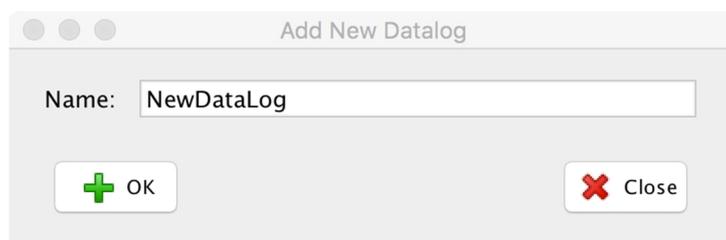
- Continuous Data-logs log periodically without interruption; this type of data log is useful mainly for persistent processes.
- Triggered Data-logs is dependent on some event/condition. The condition is specified by the alarm ID (or multiple alarm IDs). This type of data-log is useful for repetitive or random processes where you can specify the triggering condition.

### To create a data-log:

- 1) Go to the **Data-Logs** section in the *Project Window*, right-click on the *Data logs*, and select *Add New*. You can also use the button “New Data-log” in the main toolbar.



- 2) A new dialog window will show up; fill in the data log name and click on **OK**.



- 3) Double-click on your newly created data-log in the **Data-Logs** folder. A new definition window will show up. It is split into two horizontal sections: data log definition and data log views.

DEMO/Datalog - NewDataLog

Datalog Settings

Connection: PQ

Log rate: Day Hour:Min:Sec.Msec  
0 : 0 : 0 : 5 . 0

Read Refresh: 0 : 0 : 0 : 5 . 0

Triggered logging

Log single value

Continuous logging during event

Number of samples to log  
Before event: 0  
After event: 0

Alarm IDs:

Id	Name	Description	Tag	Unit	Format	Hysteresis	Delta Abs	Type	Parameter	Alignment	Key
new					###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>

Down ↓ Up ↑

Export CSV, PowerBI  Enable Data Filter

Predefined Datalog Views

Name	Description	Used ID	Totals	Hide ID	Hide Date
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## 27.2 Defining Connection

Each data-log can log data from only one connection. To specify a connection, please select it from the combo box:

Connection: script

Log rate: Day Hour:Min:Sec.Msec  
0 : 0 : 0 : 5 . 0

Read Refresh: 0 : 0 : 0 : 5 . 0



## 27.3 Defining Data Points

Each data-log should have at least one data point. You can comfortably define data points in the provided table:

Id	Name	Description	Tag	Unit	Format	Hysteresis	Delta Abs	Type	Parameter	Alignment	Key
1	Temp1	Temperature 1	Temp1@sc...		###	<input checked="" type="checkbox"/>	0.5	Value	Decimal	left	<input type="checkbox"/>
2	Temp2	Temperature 2	Temp2@sc...		###	<input checked="" type="checkbox"/>	0.5	Value	Decimal	left	<input type="checkbox"/>
3	Temp3	Temperature 3	Temp3@sc...		###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
4	Temp4	Temperature 4	Temp4@sc...		###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
5	Temp5	Temperature 5	Temp5@sc...		###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
new					###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>

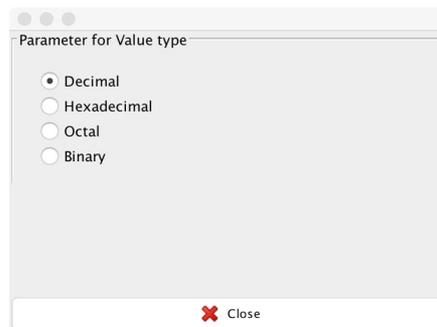
Each data point can have the following properties:

Parameter	Description
Id	Unique ID. Each data point has an auto-generated unique ID. This ID is used as a reference to find your record in the database.
Name	Name of the data point - will be shown in Data-log view table as column name.
Description	Short description - will be shown as Tooltip when hovering over column name.
Tag	Specify a valid tag (address). All specified tags must be from the same connection.
Unit	Engineering unit.
Format	This parameter is used to format numerical value based on a given pattern. More info on formatting is in chapter <a href="#">Formatting Numerical values</a> .
Hysteresis	Check this option if you want to log data only if the value of the data point changes. If you check this option, you must also set up the following parameter "Delta abs"
Delta Abs	Used when hysteresis is selected. Specify the rate of change in absolute value to log it.
Type	Type of data. Options are: <ul style="list-style-type: none"> <li>• Value - any number such as integer, float, or Boolean</li> <li>• String - String value</li> <li>• Date - value representing a date</li> </ul>
Parameter	This option is for specific settings based on selected type.
Alignment	Align the data shown in Data-log view table to the <b>left</b> , <b>right</b> , or to the <b>center</b>
Key	If you check this option, mySCADA will automatically create a search index for this data item. You can then search based on this key in Data-log view tables.

## Data Point Types

A data point can be a value (numeric), string, or date value. For each data point type, you can specify additional options.

For Value data point type, you can specify if you wish to present the value as a decimal, hexadecimal, or binary. To do so, click and select the corresponding option in the Parameter cell.



For String type value, you can specify how the string is represented in PLC. For a detailed description, please see the section Animations - Get Animation (link [Animations](#)). The usage is exactly the same.



For date type, you can specify if you wish to show only the date, only the time, or both:



*Information: mySCADA expects a date to be saved in PLC as an integer value. The date should be in UNIX UTC format.*

## 27.4 Continuous Data Logging

The purpose of the continuous data log is to log data periodically without interruption. This type of data log is useful mainly for persistent processes.

### Log Period

With continuous data logging, you define the log rate in milliseconds. In this period, all data defined in a data log are read from PLCs and logged into the internal database.

### Hysteresis

For any data point, you can define hysteresis. When hysteresis is enabled, mySCADA will read values each **Read** cycle, defined by the Read Refresh parameter; however, it will log data only if the value has changed by more than the specified **Delta Abs** value.

Id	Name	Description	Tag	Unit	Format	Hysteresis	Delta Abs	Type	Parameter	Alignment	Key
1	Temp1	Temperature 1	Temp1@sc...		###	<input checked="" type="checkbox"/>	0.5	Value	Decimal	left	<input type="checkbox"/>
2	Temp2	Temperature 2	Temp2@sc...		###	<input checked="" type="checkbox"/>	0.5	Value	Decimal	left	<input type="checkbox"/>
3	Temp3	Temperature 3	Temp3@sc...		###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
4	Temp4	Temperature 4	Temp4@sc...		###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
5	Temp5	Temperature 5	Temp5@sc...		###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
new					###	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>

When hysteresis is enabled for at least one data point, you should specify read refresh and log refresh.

**Read refresh:** how often data will be read from the PLC

**Log refresh:** how often data will be logged when there is no change in value (e.g. no log triggered by value change > delta abs).

Hysteresis function can be especially useful for logging analog data that does not change very often. When you set hysteresis of some data points in the data log, you can specify the log rate – the period in which all data points of the data log will be logged, regardless whether they have changed or not.

By enabling hysteresis, you can dramatically decrease the amount of historical data logged while maintaining data precision and time continuity.

## 27.5 Triggered Data Logs

The purpose of the triggered (event driven) data log is to log data depending on some event/condition. This is useful for repetitive or random processes where you can specify the triggering condition. You can, for example, log production data only at the time when the production line is running or log data upon system failure and use the logged data for diagnostics. With an event driven data log, you specify the start of the event by a condition. If the trigger condition is met, the system starts recording the data.

To enable a triggered data log, check the “**Triggered logging**” checkbox:

The screenshot shows a configuration panel for triggered logging. At the top, the "Triggered logging" checkbox is checked. Below it, there are two radio button options: "Log single value" (which is selected) and "Continuous logging during event". To the right of these options is a section titled "Number of samples to log" containing two spinners: "Before event:" and "After event:", both set to 0. At the bottom, there is a text input field labeled "Alarm IDs:" followed by a button with three dots.

Now you have two options:

- Continuous logging during event
- Log single value

### Continuous Logging During Event

If you select continuous logging during the event, mySCADA will log data as long as the trigger conditions are valid.

### Log Single Value

Log single value option will log just one record at the time when the trigger condition occurs.

### Triggering Conditions

mySCADA uses CAS Alarms as a triggering condition. You can select multiple CAS Alarms to use as triggers. Triggering conditions are specified by the Alarm ID field.

Click on the “...” button to specify which alarms should be used as triggers.

Select	Id	Tag	Area	Message	Device	Min	Value min	Dead ban...	Max	Value max	Dead ban...	Inv	Refresh
<input checked="" type="checkbox"/>	1	alm1@script	View1	Alarm1		<input type="checkbox"/>	0.0	0	<input type="checkbox"/>	1.0	0	<input type="checkbox"/>	default
<input checked="" type="checkbox"/>	2	alm2@script	View1	Alarm2		<input type="checkbox"/>	0.0	0	<input type="checkbox"/>	1.0	0	<input type="checkbox"/>	default
<input type="checkbox"/>	3	alm3@script		Alarm3		<input type="checkbox"/>	0.0	0	<input type="checkbox"/>	1.0	0	<input type="checkbox"/>	default
<input type="checkbox"/>	4	alm4@script		Alarm4		<input type="checkbox"/>	0.0	0	<input type="checkbox"/>	1.0	0	<input type="checkbox"/>	default

### Pre-trigger buffering

You specify the number of time samples to log before the trigger condition is met. The system automatically keeps the number of defined time samples in the memory. If the event occurs, the system will flush all the buffered data from the database and continue logging.

### Post-trigger buffering

The system will continue to log your data even after the trigger condition has stopped. You should specify the number of the time samples to be logged after the trigger condition has finished. The time sample duration is equal to the read period.

## 27.6 Triggered Logging Example

The following example will show you how to set up triggered logging. Our objective is to log a set of values when a triggering condition occurs.

1. Create a CAS Alarm that will be used as our trigger.

2. Open default data-log and select Triggered logging.
3. Select our created CAS Alarm as our trigger condition
4. Fill in the data points (e.g. values you want to log)
5. Test your project

## 27.7 Simple Periodic Export to CSV and Microsoft Power BI

There is a simple way to export data-log historical data into .csv files periodically. You define the export period (or times), and mySCADA will periodically refresh the file with fresh data. The exported file can be easily opened in MS Excel, processed by another program, or even imported into Microsoft Power BI.

Watch video describing this functionality:

<https://www.youtube.com/watch?v=vZyWoRu59c>

To set up the periodic Export, open your data-log definition file and check the **Export to CSV, PowerBI** check box.

The screenshot shows the 'empty/Datalog - default' configuration window. It includes the following elements:

- Connection:** A dropdown menu set to 'M'.
- Log rate:** A field with a 'Day' dropdown and a time input field (Hour:Min:Sec.Msec) set to 0:0:0.100.
- Read Refresh:** A field with a time input field set to 0:0:0.100.
- Logging Options:**
  - Triggered logging
  - Log single value
  - Continuous logging
- Number of samples to log:** A sub-panel with 'Before event' and 'After event' spinners, both set to 0.
- Alarm IDs:** A text input field with an ellipsis button.
- Table:** A table with columns: Id, Name, Descripti..., Tag, Unit, Format, Hysteresis, Delta Abs, Type, Parameter, Alignment, Key. It lists 10 temperature points (Temp1 to Temp10) and a 'new' entry.
- Bottom Panel:** Contains 'Down' and 'Up' buttons, and two checkboxes:  Export CSV, PowerBI (highlighted with a red arrow) and  Enable Data Filter.

You will be presented with an export settings dialog:

In this dialog, you can set up all required parameters.

First of all, you should select a file name for this export. By default, the file will be saved in the user file folder.

Then you should set the refresh period or times for when to run the export. This is done by pressing the “...” button right next to the **Refresh Time** text box.

Order #	Date
1	Friday, September 30, 2016 12:00 PM
2	Monday, October 3, 2016 12:00 PM
3	Wednesday, October 5, 2016 12:00 PM
4	Friday, October 7, 2016 12:00 PM
5	Monday, October 10, 2016 12:00 PM
6	Wednesday, October 12, 2016 12:00 PM
7	Friday, October 14, 2016 12:00 PM
8	Monday, October 17, 2016 12:00 PM
9	Wednesday, October 19, 2016 12:00 PM
10	Friday, October 21, 2016 12:00 PM
11	Monday, October 24, 2016 12:00 PM
12	Wednesday, October 26, 2016 12:00 PM
13	Friday, October 28, 2016 12:00 PM
14	Monday, October 31, 2016 12:00 PM
15	Wednesday, November 2, 2016 12:00 PM
16	Friday, November 4, 2016 12:00 PM

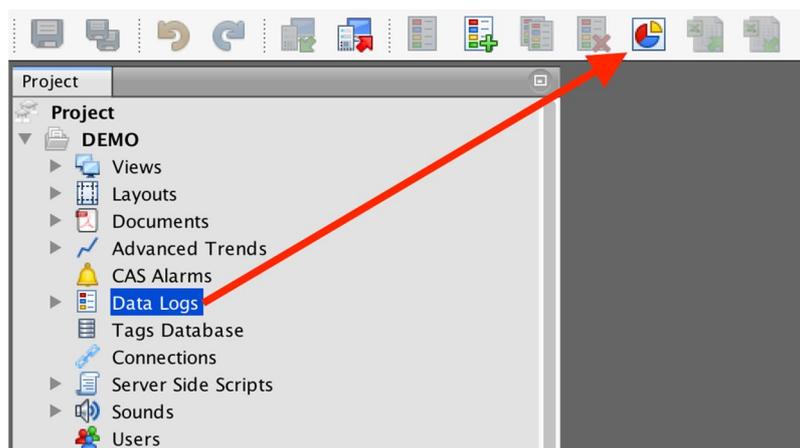
In addition, you can directly write the expression in the **Refresh time** text box. The expression is based on CRON syntax. The following figure explains how to create expressions. More info can be found here: <https://en.wikipedia.org/wiki/Cron>

```
# _____ min (0 - 59)
# | _____ hour (0 - 23)
# | | _____ day of month (1 - 31)
# | | | _____ month (1 - 12)
# | | | | _____ day of week (0 - 6) (0 to 6 are Sunday to
# | | | | Saturday, or use names; 7 is also Sunday)
# | | | | |
# | | | | |
# | | | | |
# * * * * * command to execute
```

Set up the maximum number of records to limit the size of the export file.

## 27.8 Data-logs Properties

You can specify several parameters in data-logs properties, including among others, size distribution, maximum log time, and position of data-log views in the menu. To do so, please select a data-log folder in the Project tree and click on the data-log properties icon.



You will be presented with the settings dialog:

Datalogs Settings

Datalog	Max. Size [%]	Remove After
100vals	16	365 00:00
10vals	16	365 00:00
500vals	16	365 00:00
default	16	365 00:00
Materials	16	365 00:00
Alarms	10	365 00:00
User Action	10	365 00:00

Datalog: 100vals

Max. Size [%]: 16

Delete records older than:

unlimited

Day: 365

Hour: 0 Min: 0

Total size: 100.00 %

Max. Size Uniformly

ID

Pie  Bar

Data-log Views Order

Material Batches

First Floor

Second Floor

10vals

100vals

500vals

Down Up

OK Close

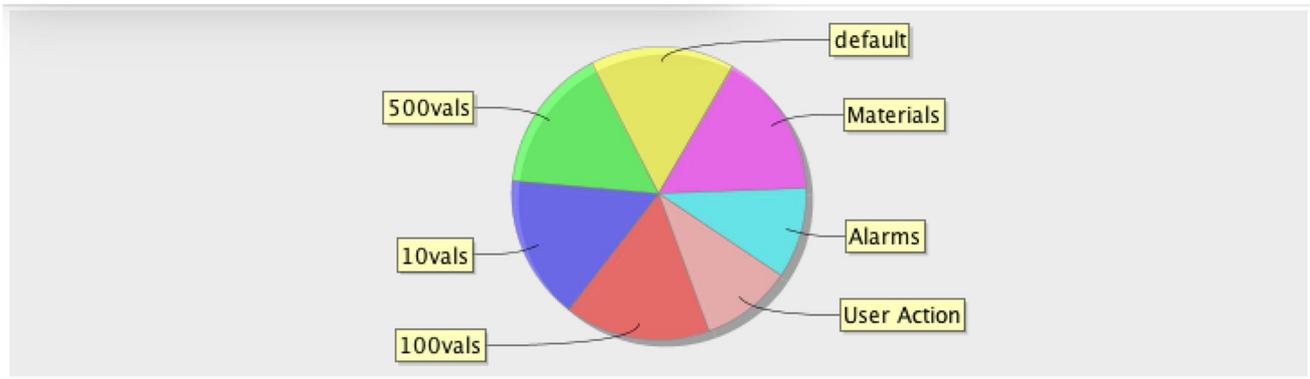
### Setting the Size Distribution of Data-logs

The most important part of the dialog is the ability to set the distribution size for all data-logs defined in your project. Running out of space will start deleting your oldest records. In this section, you can specify which data-log historical data will be removed first. Once the space reserved for data-logs is depleted, mySCADA will delete data according to these specified properties.

In the following table, you can see the percentage of how much space your data-log will have.

Datalog	Max. Size [%]	Remove After
100vals	16	365 00:00
10vals	16	365 00:00
500vals	16	365 00:00
default	16	365 00:00
Materials	16	365 00:00
Alarms	10	365 00:00
User Action	10	365 00:00

You can change the size in percent. The overall percentage should be equal to 100 percent. You can check the distribution of your data-logs in the graph below the table:



TIP: You can use decimal numbers to specify maximum size; this can be especially useful if you have a large number of data-logs and need to split disk space precisely.

### Setting the Maximum Log Time

For each data-log, you can set the maximum log time. If the oldest records in the data-log are beyond this threshold, they will be automatically deleted. To do so, please select the corresponding data-log in the table and then change the “Delete records older than” parameter.

### Distributing data-logs uniformly

You can distribute data-logs uniformly by pressing the button “Max. size uniformly”.

Size of Data-logs		
Datalog	Max. Size [%]	Remove After
100vals	16	365 00:00
10vals	16	365 00:00
500vals	16	365 00:00
default	16	365 00:00
Materials	16	365 00:00
Alarms	10	365 00:00
User Action	10	365 00:00

Datalog: 100vals

Max. Size [%]:

Delete records older than:

unlimited

Day:  Min:

Hour:  Min:

### Changing the position of Data-log Views in Menu

You can change the position of data-log views on the right side of the dialog:



# 28 Data-Log Views

Each data log can have defined multiple tabular views.

Watch video describing this functionality:  
[https://www.youtube.com/watch?v=wzxKQ\\_AUCn8](https://www.youtube.com/watch?v=wzxKQ_AUCn8)

Id	Name	Description	Tag	Unit	Format	Hysteresis	Delta Abs	Type	Parameter	Alignment	Key
1	Temp1		H:0@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
2	Temp2		H:1@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
3	Temp3		H:2@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
4	Temp4		H:3@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
5	Temp5		H:4@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
6	Temp6		H:5@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
7	Temp7		H:6@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
8	Temp8		H:7@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
9	Temp9		H:8@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
10	Temp10		H:9@M		#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>
	new				#.##	<input type="checkbox"/>	0	Value	Decimal	left	<input type="checkbox"/>

Name	Description	Data Points (IDs)	Totals	Hide ID	Hide Date
First Floor	Temperatures on first floor	1,2,3,4,5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Second Floor	Temperatures on second floor	6,7,8,9,10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

You can define the data-log views in the same window as you define the data-log.

Tabular views enable displaying all captured data from the data-log in the form of a table.

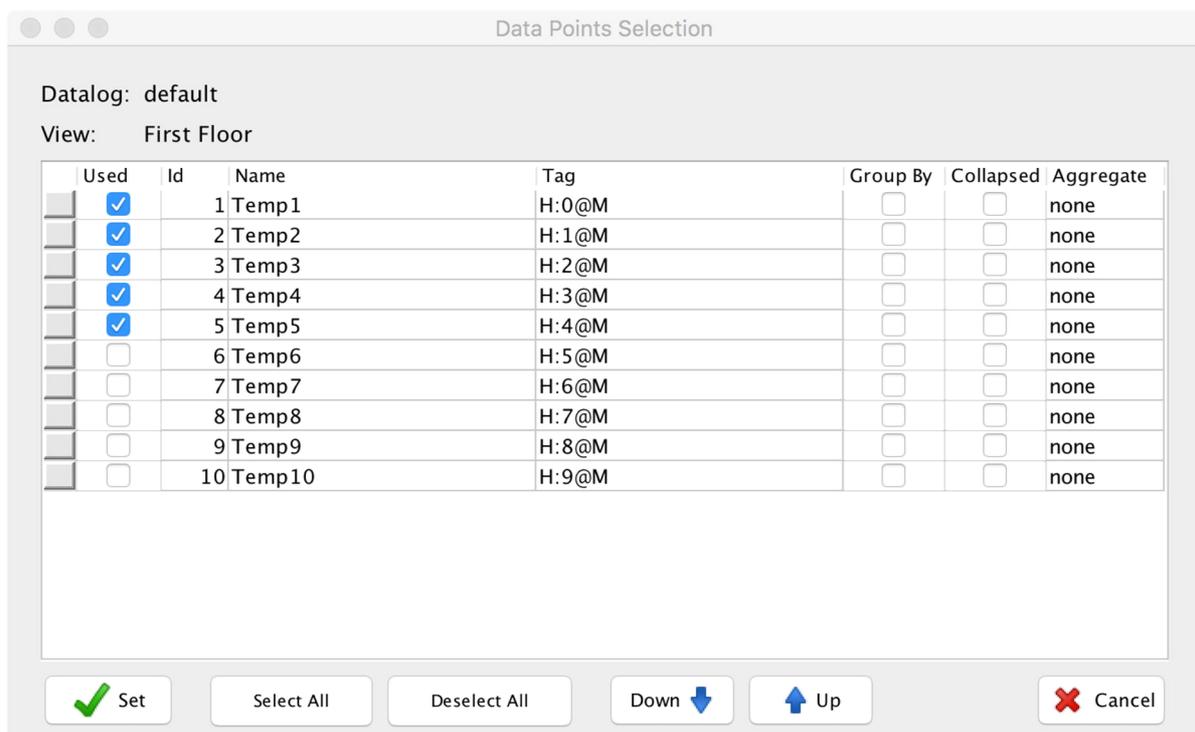
The Data-log view has following parameters

Parameter	Description
Name	Enter a name for data-log view
Description	Short description
Data Points (IDs)	List of data points used in this data-log view. For each data point, you can set up configuration parameters in the editor.
Totals	Shows the totals at the end of the table.
Hide ID	Hides the ID (#) column.
Hide Date	Hides the Date column.

First, fill in the name and description. Then click the **Data Points** column and click on the “...” button to include data points in your data-log view. You will be presented with the data points selection dialog.

## 28.1 Data Points Selection

The data points selection dialog enables you to select data points to be included in your data-log view. Select the data points you want to include on the left side of the table.



Once you specify all required parameters, you can save your data-log view and show it in runtime.

A list of all data-log views can be found in Main Application Menu under the “...” -> data-logs section.

Once you select your data-log view by name, you will see your data in tabular form:

The screenshot shows a data-log view interface for 'First Floor'. At the top, there is a blue header bar with 'Views', 'Alm (0/0)', and a dropdown menu. Below the header, there is a settings gear icon, 'LIMIT: 10000', and an 'Export' button. The main data is presented in a table with the following columns: #, TIME, Temp1, Temp2, Temp3, Temp4, and Temp5. The table contains 15 rows of data, all showing a temperature of 23.00 for Temp1, 22.00 for Temp2, 25.00 for Temp3, 24.00 for Temp4, and 27.00 for Temp5. The TIME column shows dates from September 29, 2016, with times ranging from 23:26:19 to 23:26:46. At the bottom of the interface, there is a playback slider with a play button icon and a '1 minute' label.

#	TIME	Temp1	Temp2	Temp3	Temp4	Temp5
185	September 29, 2016 23:26:19	23.00	22.00	25.00	24.00	27.00
369	September 29, 2016 23:26:48	23.00	22.00	25.00	24.00	27.00
367	September 29, 2016 23:26:48	23.00	22.00	25.00	24.00	27.00
366	September 29, 2016 23:26:48	23.00	22.00	25.00	24.00	27.00
365	September 29, 2016 23:26:48	23.00	22.00	25.00	24.00	27.00
364	September 29, 2016 23:26:47	23.00	22.00	25.00	24.00	27.00
363	September 29, 2016 23:26:47	23.00	22.00	25.00	24.00	27.00
362	September 29, 2016 23:26:47	23.00	22.00	25.00	24.00	27.00
361	September 29, 2016 23:26:47	23.00	22.00	25.00	24.00	27.00
360	September 29, 2016 23:26:47	23.00	22.00	25.00	24.00	27.00
359	September 29, 2016 23:26:47	23.00	22.00	25.00	24.00	27.00
358	September 29, 2016 23:26:46	23.00	22.00	25.00	24.00	27.00
357	September 29, 2016 23:26:46	23.00	22.00	25.00	24.00	27.00
356	September 29, 2016 23:26:46	23.00	22.00	25.00	24.00	27.00
355	September 29, 2016 23:26:46	23.00	22.00	25.00	24.00	27.00
354	September 29, 2016 23:26:46	23.00	22.00	25.00	24.00	27.00

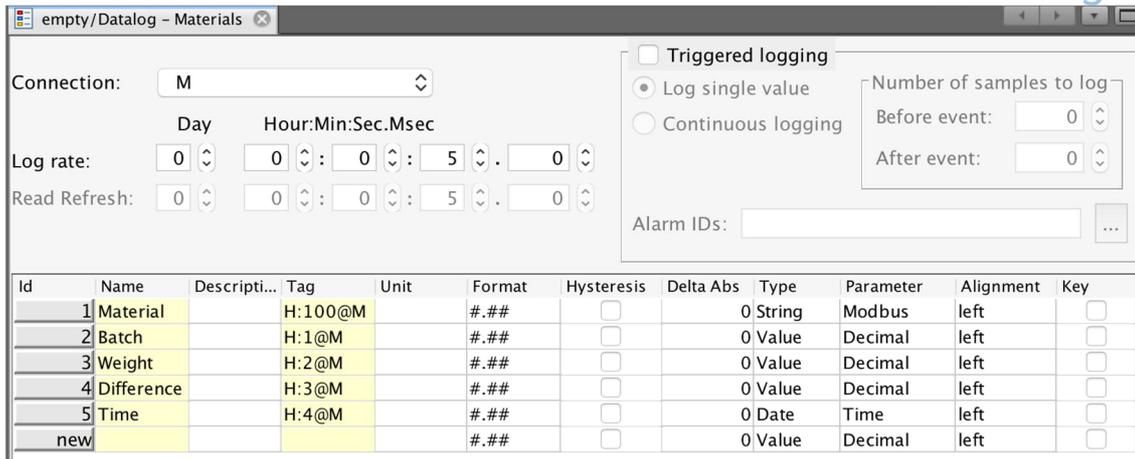
## 28.2 Data Grouping

You can group data in the data-log view table. You can even have multiple groups nested. To use data grouping, check the **Group By** check box for the data points you want to group them by. Then, for the rest of the data points, specify the Aggregate function; you can use **none**, **sum**, **average**, **minimum**, and **maximum**. If you don't provide an aggregate function, the corresponding field will be left blank.

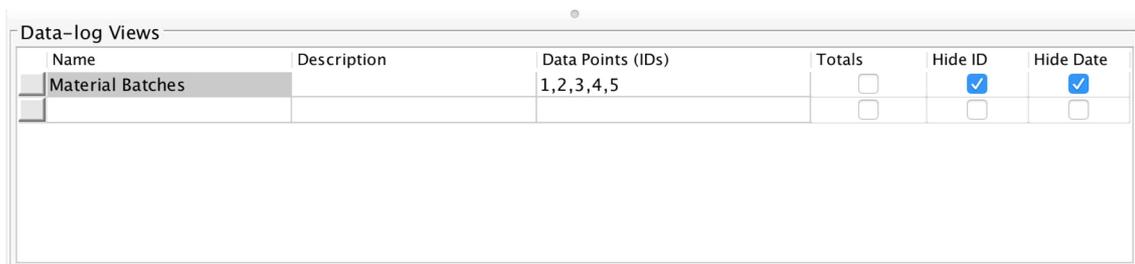
### Data Grouping Example

Suppose you have a batch process where you would like to show production data grouped by material and then by batch numbers. To do so, we will create a data-log and corresponding data-log view.

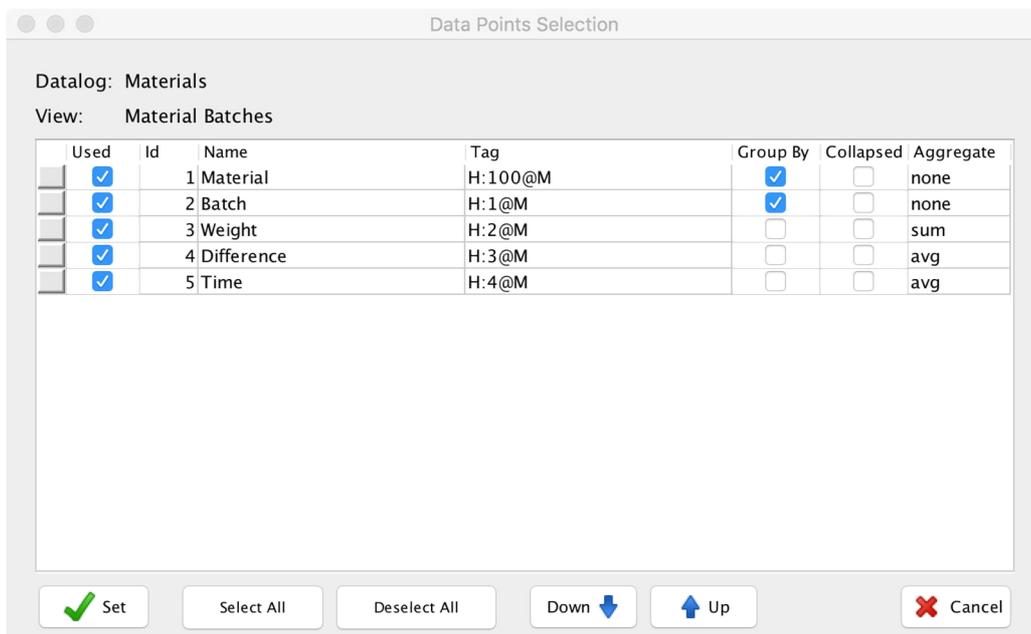
1. Create your data-log



2. Create data-log view



3. Set up data points. Check **Group By** for Materials and Batch data points. For ungrouped data points, select appropriate aggregate function.



## 4. Test your view

Views ▾ Alm (0/0) ▾ ... ▾ Double_Group				
 LIMIT: 10000  FILTER  TIME ON  Export				
Material	Batch Code	Weight	Difference	Time
[-] Material: Stone 4-6 (11 items)				
[-] Batch Code: 12440.00 (11 items)				
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
Stone 4-6	12440.00	9367.00	27.00	0:00:27
		Σ 103037.00	× 27.00	× 0:00:27

As you can see, your data are grouped by Material and then by Batches. You can collapse each group by clicking on the “+” button.

## 28.3 Filtering Data

You can easily filter your data in the data-log view based on time intervals. However, in some situations, it might be necessary to filter your data based on the data itself. In the example below, you can look at a situation where filtering data is an essential functionality.

To enable data filtering, you must specify which data points should be used as data filters. You do this in the data point definition in your data-log definition. Usage is quite simple, just check the **Key** check box for all the data points you want to use in data filtering.



Now write down the material name and you will see that your table will show this material.

## 29 Aggregated Data logs

---

Aggregated data logs allow you to automatically compute aggregated data from regular data logs and alarms.

Usually, you have rough data recorded from your PLC in regular data logs. For example, you can be reading temperature values every 10 seconds and have them logged into the data log. Now imagine you would like to know the minimum, maximum and average temperature values per hour, per day and per month. This is where aggregated data logs come in play.

With aggregated data logs you can aggregate by

- time period (minutes, hours, days, ...)
- some logged value change
- Alarm activation

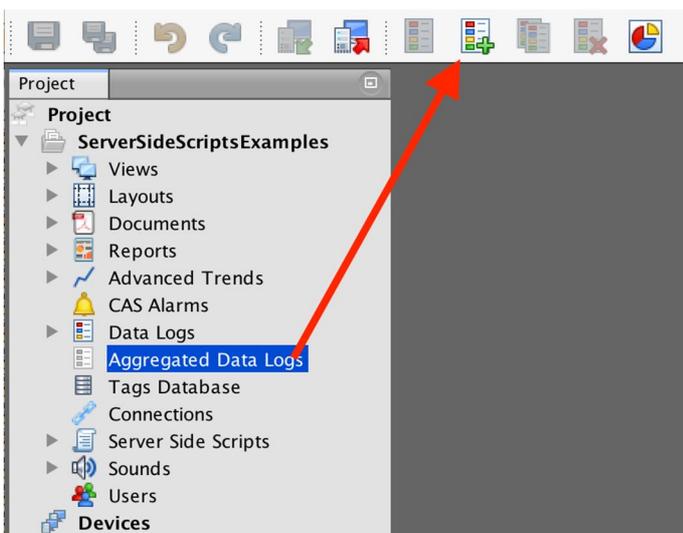
You can compute statistics based on functions

- Count
- Sum
- Average
- Minimum
- Maximum
- Alarm time activation
- Alarm count

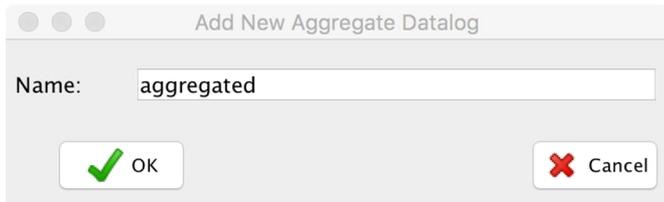
To create an aggregated data log, first create all sources - eg. Data logs you want to aggregate data from. Then you can start creating your data log.

### 29.1 Creating aggregated data log

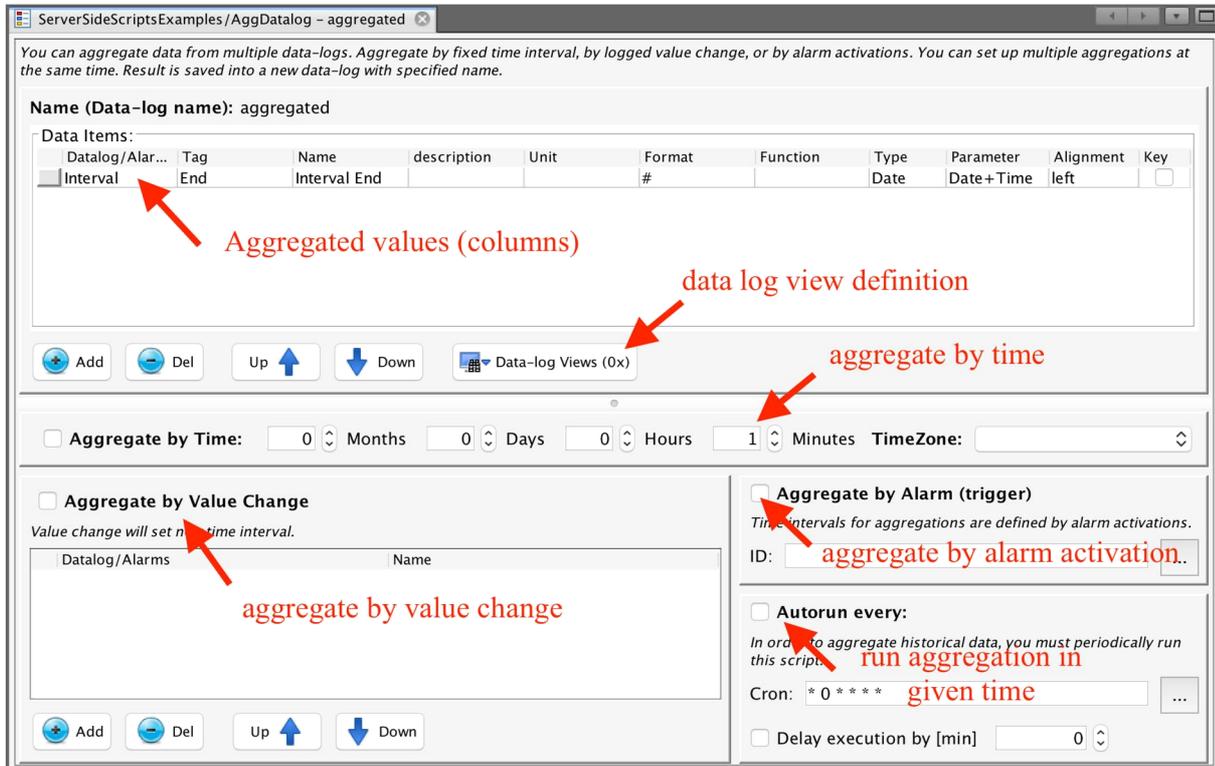
To create an aggregated data log, navigate to **Aggregated Data Logs** and click on **New**.



You will be presented with the new dialog, please select a name and confirm.



Once confirmed, a window with the definition of your aggregated data log will be opened.



As you can see, the window is split into several sections. We will explain each section in detail below.

## 29.2 Aggregated Values:

Aggregated values are the values from one or multiple data logs aggregated using the “aggregate” function. You can add an arbitrary number of values (columns) from multiple data logs.

First item in the table is always Interval End. Aggregated data log has always two dates: start and end of interval.

Then you can add as many aggregated items as you wish. Table has following columns:

Column	Description
Datalog/Alarm	Source datalog (or Alarm)
Tag	Source tag (or Alarm Severity)
Name	Name - eg. name of aggregated item
Description	Description of aggregated item

Unit	Unit (useful in trends)
Format	Format for numeric values use ### notation
Function	Aggregate function (sum, avg, min, max, ..)
Type	Type of item eg. Numeric or String
Parameter	Additional format parameter for Type item
Alignment	Align item in data log view table to the left, right, or middle
Key	If you want to perform a search by this item, tick it please.

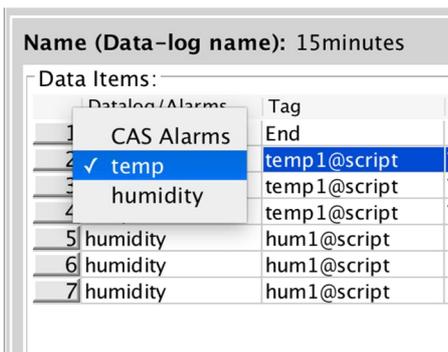
To add aggregated value for data log:

1. Select data log name in combo box
2. Select tag in combo box
3. Fill rest of the table

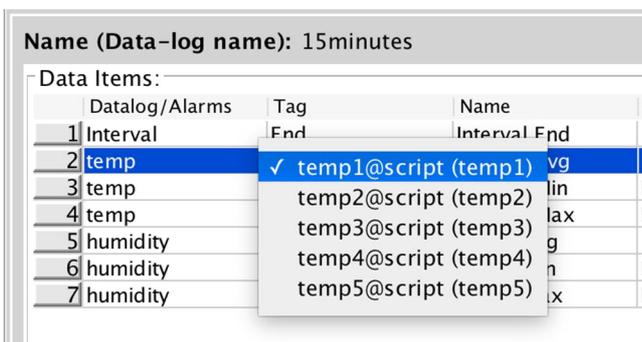
To add aggregated alarm count or alarm activation time

1. Select CAS Alarm
2. Enter maximum severity
3. Fill rest of the table

First, select a data log



Secondly, select a tag from datalog



Then fill in the Name, description and also the aggregate function:

Name (Data-log name): 15minutes

Data Items:

	Datalog/Alarms	Tag	Name	description	Unit	Format	Function
1	Interval	End	Interval End			#	sum
2	temp	temp1@script	Temp1 Avg			#.##	✓ avg
3	temp	temp1@script	Temp1 Min	Minimum		#.##	min
4	temp	temp1@script	Temp1 Max	Maximum		#.##	max
5	humidity	hum1@script	Hum1 Avg	15 mins hum			cnt
6	humidity	hum1@script	Hum1 Min	Minimum		#.##	val
7	humidity	hum1@script	Hum1 Max	Maximum			

Once you are done, don't forget to also set up the data-log view, you can do it by clicking on the "Data log views" button:

*You can aggregate data from multiple data-logs. Aggregate by fixed time interval, by logged va saved into a new data-log with specified name.*

Name (Data-log name): 15minutes

Data Items:

	Datalog/Alarms	Tag	Name	description	Unit
1	Interval	End	Interval End		
2	temp	temp1@script	Temp1 Avg		
3	temp	temp1@script	Temp1 Min	Minimum	
4	temp	temp1@script	Temp1 Max	Maximum	
5	humidity	hum1@script	Hum1 Avg	15 mins hum	
6	humidity	hum1@script	Hum1 Min	Minimum	
7	humidity	hum1@script	Hum1 Max	Maximum	

You will be presented with data-log view definition, it is the same as when you define a data log view in data log definitions (please see chapter Data logs)

Aggregate Data-log Views Edit

Name	description	Data Points (IDs)	Show in Menu	Hide ID	Hide Date	Accesses
15 mins Temps		1,2,3,4,5,6,7	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Set

### Simple Example:

In this example, we are aggregating data from 2 different datalogs "temp" and "humidity"

You can aggregate data from multiple data-logs. Aggregate by fixed time interval, by logged value change, or by alarm activations. You can set up multiple aggregations at the same time. Result is saved into a new data-log with specified name.

Name (Data-log name): 15minutes

Data/Alarms	Tag	Name	description	Unit	Format	Function	Type	Parameter	Alignment	Key
1 Interval	End	Interval End			#		Date	Date+Time	left	<input type="checkbox"/>
2 temp	temp1@script	Temp1 Avg	15 mins temps		###	avg	Value	Decimal	left	<input type="checkbox"/>
3 temp	temp1@script	Temp1 Min	Minimum		###	min	Value	Decimal	left	<input type="checkbox"/>
4 temp	temp1@script	Temp1 Max	Maximum		###	max	Value	Decimal	left	<input type="checkbox"/>
5 humidity	hum1@script	Hum1 Avg	15 mins hum		###	avg	Value	Decimal	left	<input type="checkbox"/>
6 humidity	hum1@script	Hum1 Min	Minimum		###	min	Value	Decimal	left	<input type="checkbox"/>
7 humidity	hum1@script	Hum1 Max	Maximum		###	max	Value	Decimal	left	<input type="checkbox"/>

## 29.3 Time Aggregates

The most common way to aggregate data is by the time interval.

**Aggregate by Time:**
 Months
  Days
  Hours
  Minutes
 **TimeZone:**

Please tick the check box “**Aggregate by Time:**” and then fill in the time period for your aggregate. mySCADA is using the ISO8601 norm when you specify the interval.

You should also specify the **time zone** of your areas (if you aggregate by days, this will define start time of your day)

### Simple Example:

Aggregating data from multiple data logs by 15 minute intervals.

Definition of aggregated data log:

You can aggregate data from multiple data-logs. Aggregate by fixed time interval, by logged value change, or by alarm activations. You can set up multiple aggregations at the same time. Result is saved into a new data-log with specified name.

Name (Data-log name): 15minutes

Data/Alarms	Tag	Name	description	Unit	Format	Function	Type	Parameter	Alignment	Key
1 Interval	End	Interval End			#		Date	Date+Time	left	<input type="checkbox"/>
2 temp	temp1@script	Temp1 Avg	15 mins temps		###	avg	Value	Decimal	left	<input type="checkbox"/>
3 temp	temp1@script	Temp1 Min	Minimum		###	min	Value	Decimal	left	<input type="checkbox"/>
4 temp	temp1@script	Temp1 Max	Maximum		###	max	Value	Decimal	left	<input type="checkbox"/>
5 humidity	hum1@script	Hum1 Avg	15 mins hum		###	avg	Value	Decimal	left	<input type="checkbox"/>
6 humidity	hum1@script	Hum1 Min	Minimum		###	min	Value	Decimal	left	<input type="checkbox"/>
7 humidity	hum1@script	Hum1 Max	Maximum		###	max	Value	Decimal	left	<input type="checkbox"/>

Definition of time interval:

**Aggregate by Time:**
 Months
  Days
  Hours
  Minutes
 **TimeZone:**

### Result:

Raw Data (shown only small part):

Temperatures						Humidity							
#	TIME	temp1	temp2	temp3	temp4	temp5	#	TIME	hum1	hum2	hum3	hum4	hum5
1	August 13, 2017 20:47:45	25.19	22.46	24.04	28.92	23.39	1	August 13, 2017 20:47:45	46.67	45.64	53.03	48.35	49.23
2	August 13, 2017 20:47:43	25.47	22.50	24.40	28.96	23.38	2	August 13, 2017 20:47:45	48.70	45.64	53.03	48.35	49.23
3	August 13, 2017 20:47:42	25.22	22.69	24.58	28.67	23.79	3	August 13, 2017 20:47:43	48.70	44.22	52.24	49.12	50.33
4	August 13, 2017 20:47:41	25.75	22.80	24.49	28.03	23.44	4	August 13, 2017 20:47:42	47.95	44.27	52.59	49.44	51.56
5	August 13, 2017 20:47:41	25.75	22.80	24.49	28.03	23.44	5	August 13, 2017 20:47:42	47.95	44.27	52.59	49.44	51.56
6	August 13, 2017 20:47:40	25.73	22.11	24.96	28.32	23.74	6	August 13, 2017 20:47:41	47.62	44.09	53.49	49.33	51.73
7	August 13, 2017 20:47:39	25.50	22.24	24.43	28.87	23.70	7	August 13, 2017 20:47:41	47.62	44.09	53.49	49.33	51.73
8	August 13, 2017 20:47:38	25.89	22.37	24.58	28.76	23.92	8	August 13, 2017 20:47:40	48.17	46.94	52.75	48.62	50.78
9	August 13, 2017 20:47:37	25.99	22.31	24.78	28.87	23.60	9	August 13, 2017 20:47:40	48.17	46.94	52.75	48.62	50.78
10	August 13, 2017 20:47:36	25.99	22.23	24.84	28.21	23.09	10	August 13, 2017 20:47:39	48.73	44.80	53.71	49.66	50.55
11	August 13, 2017 20:47:36	25.99	22.23	24.84	28.21	23.09	11	August 13, 2017 20:47:39	46.93	46.95	53.71	50.17	50.45
12	August 13, 2017 20:47:35	25.57	22.24	24.05	28.90	23.96	12	August 13, 2017 20:47:38	46.93	46.95	52.14	50.17	50.45
13	August 13, 2017 20:47:34	25.92	22.18	24.16	28.76	23.13	13	August 13, 2017 20:47:37	48.42	45.18	52.13	48.42	51.20
14	August 13, 2017 20:47:32	25.08	22.52	24.81	28.55	23.58	14	August 13, 2017 20:47:37	46.47	45.18	52.13	48.42	51.20
15	August 13, 2017 20:47:31	25.14	22.22	24.79	28.83	23.79	15	August 13, 2017 20:47:36	46.47	44.28	53.51	49.75	50.09
16	August 13, 2017 20:47:30	25.54	22.56	24.06	28.13	23.94	16	August 13, 2017 20:47:36	46.47	44.28	53.51	49.75	50.09
17	August 13, 2017 20:47:29	25.79	22.27	24.05	28.87	23.24	17	August 13, 2017 20:47:35	47.18	44.36	52.38	50.39	49.58
18	August 13, 2017 20:47:29	25.27	22.81	24.05	28.87	23.24	18	August 13, 2017 20:47:35	47.91	44.36	52.38	50.39	49.58
19	August 13, 2017 20:47:28	25.27	22.81	24.43	28.04	23.97	19	August 13, 2017 20:47:34	47.91	45.90	54.91	50.51	49.24
20	August 13, 2017 20:47:27	25.63	22.16	24.05	28.84	23.41	20	August 13, 2017 20:47:34	47.28	45.90	54.91	50.51	49.24
21	August 13, 2017 20:47:26	25.53	22.68	24.44	28.73	23.37	21	August 13, 2017 20:47:32	47.28	44.29	53.11	48.10	51.20
22	August 13, 2017 20:47:26	25.53	22.68	24.44	28.73	23.37	22	August 13, 2017 20:47:32	48.67	44.29	53.11	48.10	49.27
23	August 13, 2017 20:47:25	25.69	22.69	24.25	28.80	23.69	23	August 13, 2017 20:47:31	48.67	46.39	54.53	49.36	49.27
24	August 13, 2017 20:47:24	25.52	22.83	24.88	28.95	23.58	24	August 13, 2017 20:47:31	48.67	46.39	54.53	49.36	49.27
25	August 13, 2017 20:47:22	25.52	22.25	24.43	28.49	23.61	25	August 13, 2017 20:47:30	48.44	46.30	53.34	48.63	49.11

And final aggregated result:

15 mins Temps								
#	TIME	Interval End	Temp1 Avg	Temp1 Min	Temp1 Max	Hum1 Avg	Hum1 Min	Hum1 Max
1	August 13, 2017 20:30:00	August 13, 2017 20:45:00	25.49	25.00	26.00	47.5	46.00	49.0
2	August 13, 2017 20:15:00	August 13, 2017 20:30:00	25.49	25.00	26.00	47.5	46.00	49.0
3	August 13, 2017 20:00:00	August 13, 2017 20:15:00	25.49	25.00	26.00	47.5	46.01	49.0
4	August 13, 2017 19:45:00	August 13, 2017 20:00:00	25.51	25.00	26.00	47.4	46.00	49.0
5	August 13, 2017 19:30:00	August 13, 2017 19:45:00	25.49	25.00	26.00	47.5	46.00	49.0
6	August 13, 2017 19:15:00	August 13, 2017 19:30:00	25.50	25.00	26.00	47.5	46.00	49.0
7	August 13, 2017 19:00:00	August 13, 2017 19:15:00	25.48	0.00	26.00	47.5	0.00	49.0

## 29.4 Value Change Aggregates

You can also aggregate data by value change. The usage is quite simple, please define from which datalog and what tags should be tracked for value change. When value in data log changes it is used as aggregate interval.

To use Value Change Aggregates, select the “Aggregate by Value Change” check box and add at least one item by which you want to track value change.

**Aggregate by Value Change**

Value change will set new time interval.

Datalog/Alarms	Name
production	product@script

Add
 Del
 Up
 Down

In this simple example, we will show you how to track production change. We have 2 data logs:

1. Raw data, where we log each produced item on the production line
2. Product change data log, here we log when we start producing new product

We will create an aggregated data log, which will show how many items we have produced for each product.

Aggregated data log definition:

You can aggregate data from multiple data-logs. Aggregate by fixed time interval, by logged value change, or by alarm activations. You can set up multiple aggregations at the same time. Result is saved into a new data-log with specified name.

**Name (Data-log name):** Product\_Change

Data Items:										
Datalog/Alarms	Tag	Name	description	Unit	Format	Function	Type	Parameter	Alignment	Key
1	Interval	End	Interval End		#		Date	Date+Time	left	<input type="checkbox"/>
2	temp	temp1@script	Items produced			cnt	Value	Decimal	left	<input type="checkbox"/>
3	production	product@script	Product			val	String	Automatic	left	<input type="checkbox"/>

Data - log with product label change:

☰
Product\_Change

⚙️    **LIMIT: 10000**    📄 **Export**

#	TIME	product
85	August 13, 2017 21:55:21	Product No. 85
86	August 13, 2017 21:55:40	Product No. 86
87	August 13, 2017 21:55:43	Product No. 87
88	August 13, 2017 21:56:07	Product No. 88
89	August 13, 2017 21:56:17	Product No. 89
90	August 13, 2017 21:57:33	Product No. 90
91	August 13, 2017 21:59:17	Product No. 91
92	August 13, 2017 21:59:36	Product No. 92
93	August 13, 2017 22:00:08	Product No. 93
94	August 13, 2017 22:00:23	Product No. 94

And Resulting aggregated data log:

#	TIME	Interval End	Product	Items
1	August 13, 2017 22:00:24	August 13, 2017 22:00:27	Product No. 94	3.00
2	August 13, 2017 22:00:08	August 13, 2017 22:00:24	Product No. 93	15.00
3	August 13, 2017 21:59:37	August 13, 2017 22:00:08	Product No. 92	26.00
4	August 13, 2017 21:59:18	August 13, 2017 21:59:37	Product No. 91	18.00
5	August 13, 2017 21:57:34	August 13, 2017 21:59:18	Product No. 90	104.00
6	August 13, 2017 21:56:17	August 13, 2017 21:57:34	Product No. 89	79.00
7	August 13, 2017 21:56:08	August 13, 2017 21:56:17	Product No. 88	8.00
8	August 13, 2017 21:55:43	August 13, 2017 21:56:08	Product No. 87	22.00
9	August 13, 2017 21:55:40	August 13, 2017 21:55:43	Product No. 86	2.00
10	August 13, 2017 21:55:22	August 13, 2017 21:55:40	Product No. 85	17.00
11	August 13, 2017 21:54:48	August 13, 2017 21:55:22	Product No. 84	29.00
12	August 13, 2017 21:54:36	August 13, 2017 21:54:48	Product No. 83	11.00
13	August 13, 2017 21:54:27	August 13, 2017 21:54:36	Product No. 82	8.00
14	August 13, 2017 21:53:07	August 13, 2017 21:54:27	Product No. 81	77.00
15	August 13, 2017 21:52:55	August 13, 2017 21:53:07	Product No. 80	14.00
16	August 13, 2017 21:52:09	August 13, 2017 21:52:55	Product No. 79	43.00

## 29.5 Aggregates based on Alarm Activation

Alarm activation aggregates can be useful if you use alarm as your trigger for time interval changes. Usage is quite simple and straight forward, the alarm activation sets the aggregate interval.

To define, aggregates base on Alarm activations, please check the “**Aggregate by Alarm (trigger)**” check box and fill in the alarm ID or IDs.

**Aggregate by Alarm (trigger)**

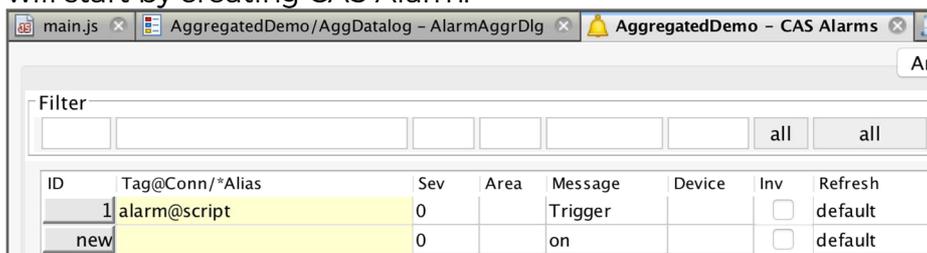
*Time intervals for aggregations are defined by alarm activations.*

ID:

### Simple example:

We will log production based on the production line activation signal.

1. We will define our activation signal. For this purpose, we will use CAS Alarms. Eg. we will start by creating CAS Alarm:



2. Now we will use the aggregated data log to aggregate the data based on the alarm activation:

You can aggregate data from multiple data-logs. Aggregate by fixed time interval, by logged value change, or by alarm activations. You can set up multiple aggregations at the same time. Result is saved into a new data-log with specified name.

Name (Data-log name): AlarmAggrDlg

Data Items:

	Datalog/Alar...	Tag	Name	description	Unit	Format	Function	Type	Parameter	Alignment	Key
1	Interval	End	Interval End			#		Date	Date+Time	left	<input type="checkbox"/>
2	humidity	hum1@script	Total				sum	Value	Decimal	left	<input type="checkbox"/>
3	humidity	hum1@script	Average			#.##	avg	Value	Decimal	left	<input type="checkbox"/>

Aggregate by Time: 0 Months 0 Days 0 Hours 1 Minutes TimeZone:

Aggregate by Value Change  
 Value change will set new time interval.

Aggregate by Alarm (trigger)  
 Time intervals for aggregations are defined by alarm activations.

ID: 1

Autorun every:  
 In order to aggregate historical data, you must periodically run this script.

Cron: \* 0 \* \* \* \*

Delay execution by [min] 0

And the result:

Alarm activations:

#	MESSAGE	STATUS	SEV	AREA	DEVICE	ACT TIME	DEACT TIME
1	Trigger	DEACT	0			August 14, 2017 07:05:22	August 14, 2017 07:06:22
2	Trigger	ACT	0			August 14, 2017 07:07:22	
3	Trigger	DEACT	0			August 14, 2017 07:07:22	August 14, 2017 07:08:22
4	Trigger	ACT	0			August 14, 2017 07:09:22	
5	Trigger	DEACT	0			August 14, 2017 07:09:22	August 14, 2017 07:10:22
6	Trigger	ACT	0			August 14, 2017 07:11:23	

And corresponding aggregated data

#	TIME	Interval End	Total	Average
1	August 14, 2017 07:05:23	August 14, 2017 07:06:23	2533.4	47.80
2	August 14, 2017 07:07:23	August 14, 2017 07:08:23	2858.0	47.63
3	August 14, 2017 07:09:23	August 14, 2017 07:10:23	3705.6	47.51

As you can see, for each alarm activation period, you get one row in the aggregated data log.

## 29.6 Running the Aggregation Periodically

You need to run the aggregated function periodically to get the calculated data log updated. Ideally, you should run the aggregation just after your data is ready, eg. shortly after a defined period. To do so, please tick **“Autorun every:”** check box.



The screenshot shows a configuration dialog box with the following elements:

- A checked checkbox labeled **Autorun every:**.
- A note: *In order to aggregate historical data, you must periodically run this script.*
- A text field labeled **Cron:** containing the value `0 */1 * * *`, followed by a button with three dots (`...`).
- An unchecked checkbox labeled **Delay execution by [min]**, followed by a text field containing `0` and a spinner control.

Then define the period by clicking on the “...” button. You will be presented with dialog, where you can define at what time intervals to run the aggregation function.

If you need to further delay the execution of aggregation function (for example, you need to run every hour and 5 minutes), you can select **“Delay execution by”** and add number of minutes to delay an execution.

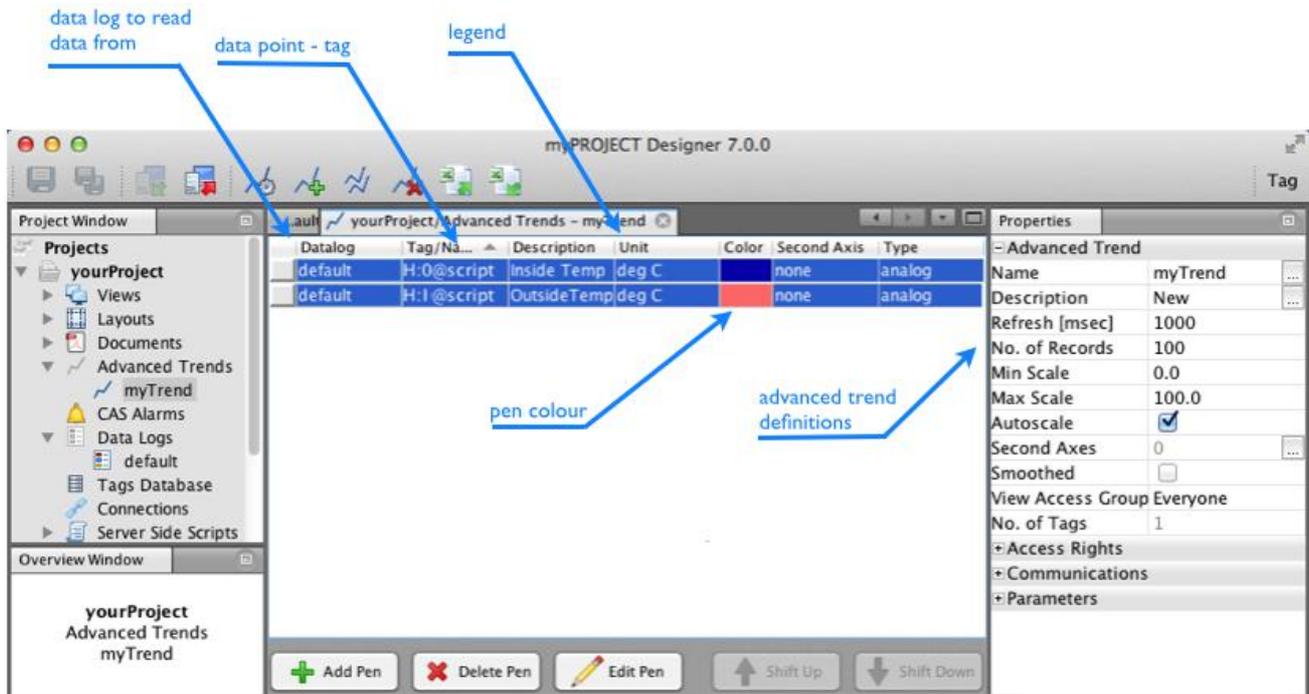
**IMPORTANT: IF YOU DON'T RUN THE AGGREGATION FUCTION, YOU WILL HAVE NO DATA IN AGGREGATED DATA LOG**

## 30 Advanced Trends

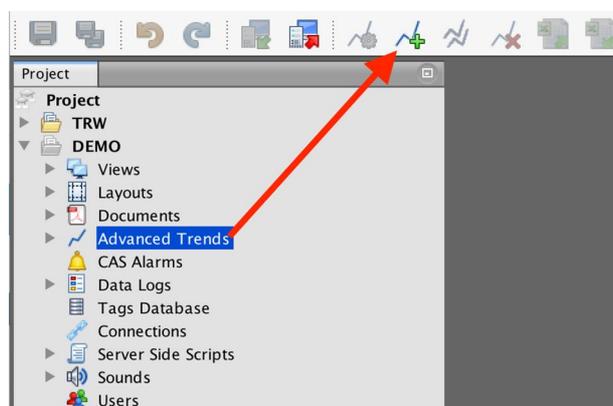
You can easily present your historical data in the form of a historical trend.

Watch video describing this functionality: <https://www.youtube.com/watch?v=NhvLI-QZL34>

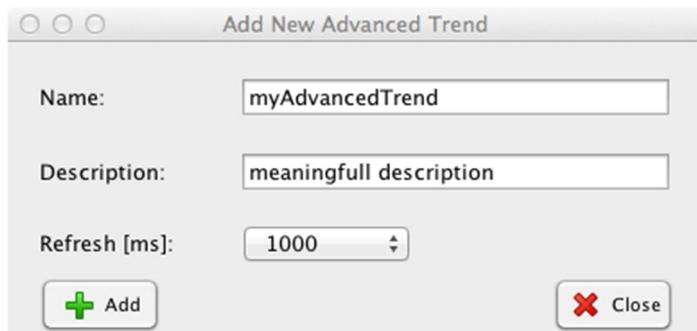
To do so, first look at the chapter [Data Logging](#) to set your data-log from which to retrieve data.



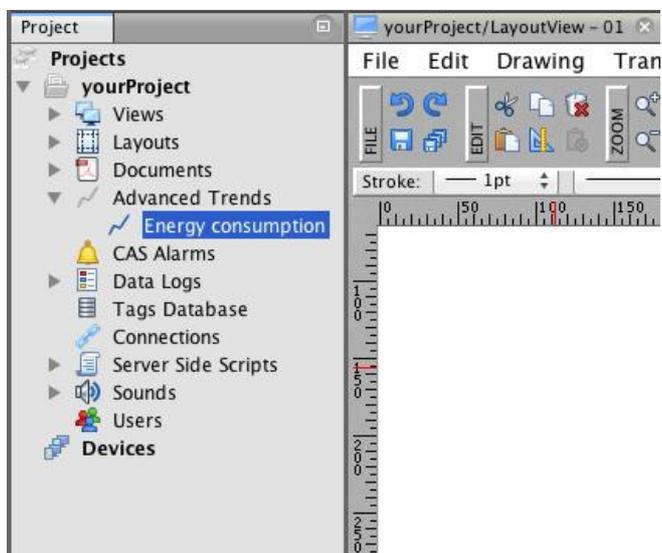
- 1) If you have your *Data log* ready, create a new historical trend by going to the **Advanced Trends** folder and selecting a new trend, as shown in the following picture:



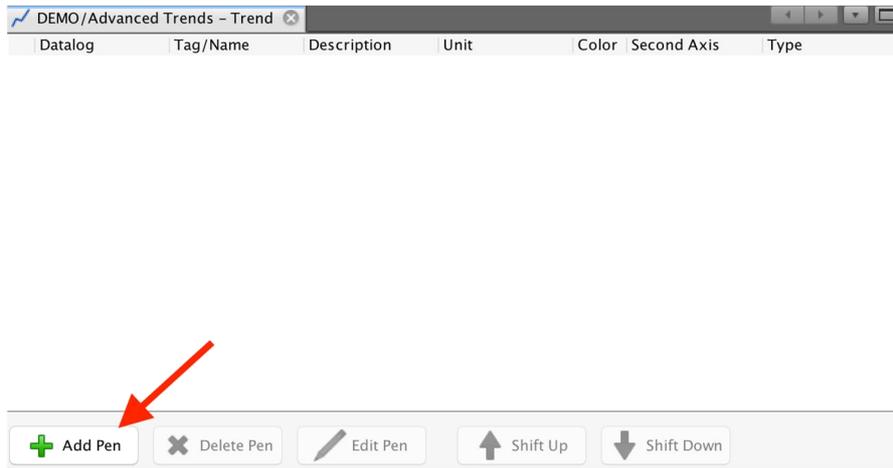
- 2) A new dialog window will show up where you can fill in the trend name and description. Then click on **OK**.



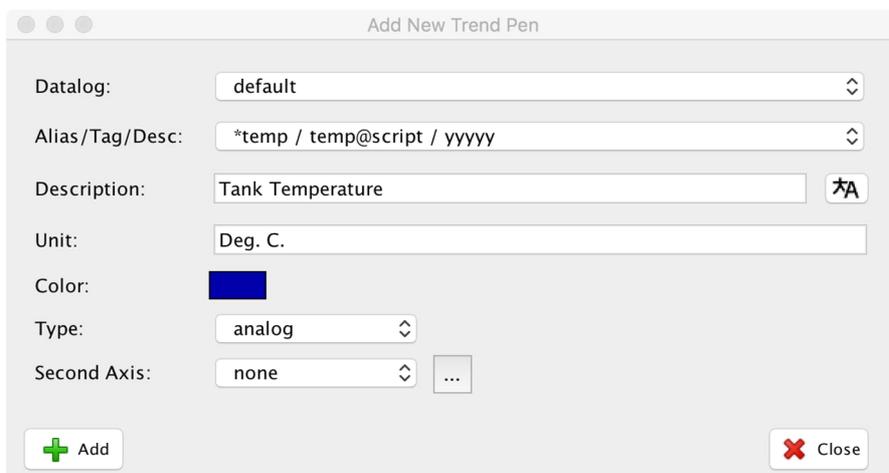
- 3) Now the new trend '*myAdvancedTrend*' is created and you can find it in the *Advanced Trends* folder of the *Project window*.



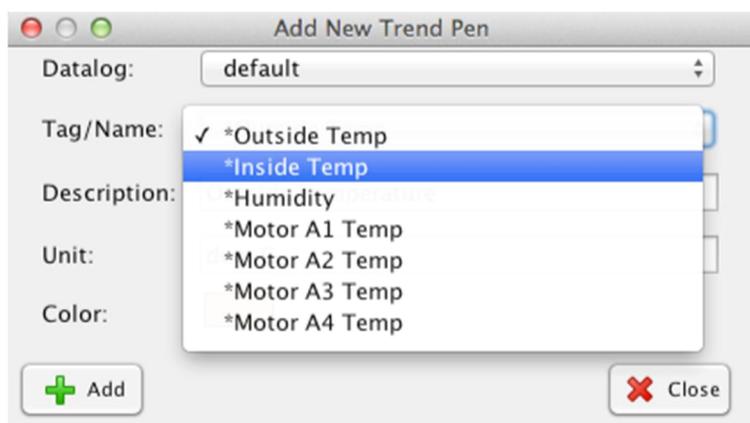
- 4) Double click on the newly created trend to open it.
- 5) Click on the **Add Pen** button in the lower part of the *Advanced Trend Definition window*.



You will be presented with the Pen settings dialog:



- 6) In the pop-up dialog box, you can define your new pen to show. First, you should select a data log from which to show data, then select your data point to show (only the tags from the selected data log are available):



- 7) Finally, fill in the *Description* and *Unit*. You can also change the pen color and confirm by pressing the **Add** button. The pen is now listed in the table below:

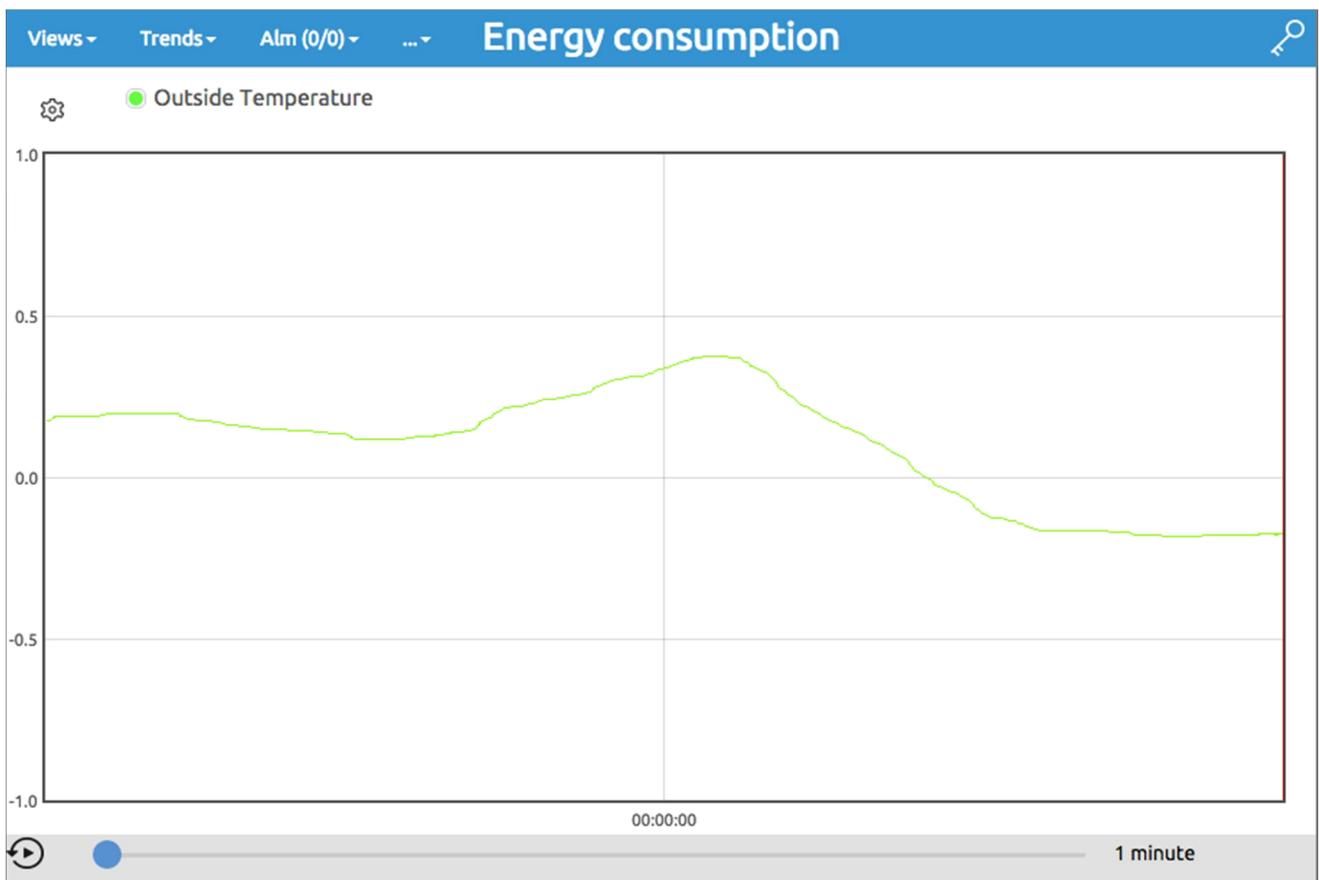
Datalog	Tag/Name	Description	Unit	Color	Second Axis	Type
default	temp@script	Tank Temper...	Deg. C.	<span style="color: blue;">█</span>	none	analog

+ Add Pen
× Delete Pen
✎ Edit Pen
↑ Shift Up
↓ Shift Down

*Note: You can freely add as many pens as you wish; they can read data from multiple data-logs.*

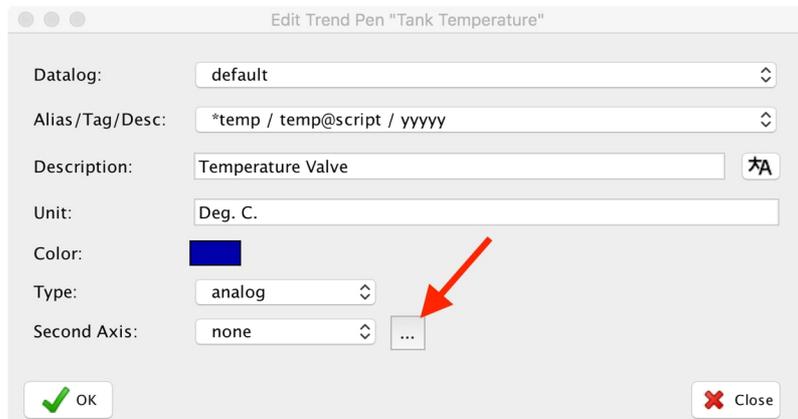
You can see how your trend will look in real-time:



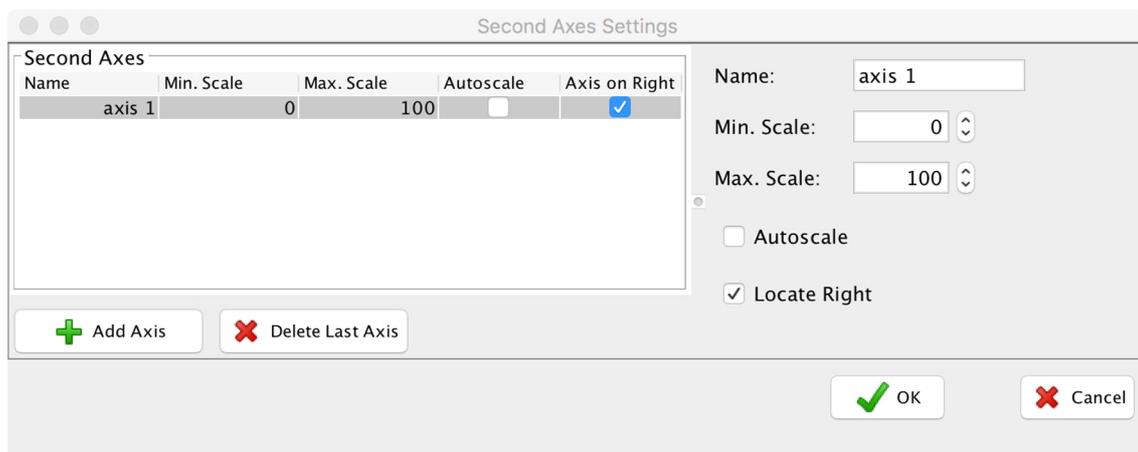
## 30.1 Using Multiple Axes

By default, all pens use the same vertical axis. You can define multiple axes. Each axis can be used just for one pen, but it can be also shared by multiple pens. To add a new axis:

1. Click on the "+ Add Pen" dialog. Now click on the "... " button next to Second Axis.



2. You will be presented with the Axes Settings Dialog:



3. In the dialog, you can add a new Axis and set its parameters like **auto-scale**, **location**, and **name**.
4. Once you have created the new axis, you can select it for a given pen:

Edit Trend Pen "Tank Temperature"

Datalog: default

Alias/Tag/Desc: \*temp / temp@script / yyyy

Description: Temperature Valve

Unit: Deg. C.

Color:  

Type: analog

Second Axis: axis 1

OK Close

5. Now you can look at how the trend will look with multiple axes.



## 30.2 Advanced Trends – Visual Appearance

You can easily modify visual appearance of the trend to fit it to your requirements. Eg, you can change the background color, color of the grid, text labels, etc. To change the visual appearance of the trend, please click on the trend in the project window and navigate to the properties window, to the section Visuals.

Visuals	
Background	<input type="checkbox"/> [255,255,255] ...
Hide X Grid	<input type="checkbox"/>
Hide Y Grid	<input type="checkbox"/>
Hide X Label	<input type="checkbox"/>
Hide Y Label	<input type="checkbox"/>
X Axis Font Size	11
X Axis Font Color	■ [0,0,0] ...
Y Axis Font Size	11
Y Axis Font Color	■ [0,0,0] ...
X Axis Color	■ [0,0,0] ...
Y Axis Color	■ [0,0,0] ...
Grid Border Color	■ [0,0,0] ...
Markings/Limits	Not Set ...

Here you can change the parameters to your requirements. Changing these parameters will allow you to customize the appearance to fit your brand standards or to make it more visually appealing to you.

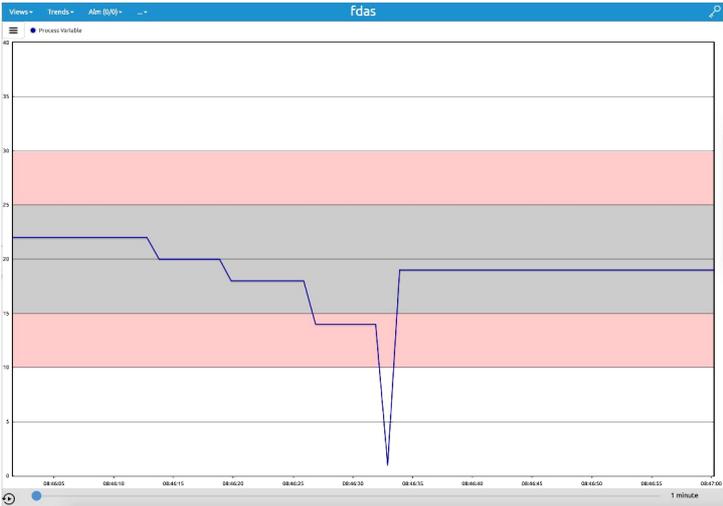
### Markings/Limits:

Markings allow you to draw regions in your chart. This way, you can easily visually highlight the limits for variable. You can create multiple markings for one trend. To add/delete marking please click on the “...” next to the markings. You will be presented with the dialog to set markings.

Advanced Trend Markings/Limits Edit

Markings/Limits		
Y Axis From	Y Axis To	Color
<input type="checkbox"/>	10	30 #FF9999
<input type="checkbox"/>	15	25 #CCCCCC

The following example shows a chart with 2 markings regions enabled. One gray region in the range of 15 to 25 and one light red region in the interval 10 to 30.



- Line thickness
- Fill region under

Edit Trend Pen "Process Variable"

Datalog: default

Alias/Tag/Desc: \*H:0 / H:0@M /

Description: Process Variable

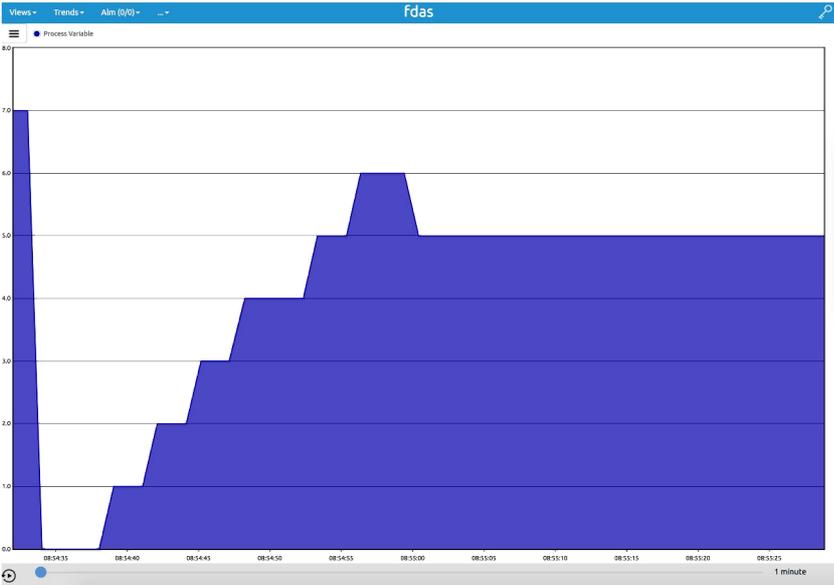
Unit:

Color:  Thickness [px]: 2 Fill under:  Alpha: 0.7

Type: analog

Second Axis: none

OK From Tag Close



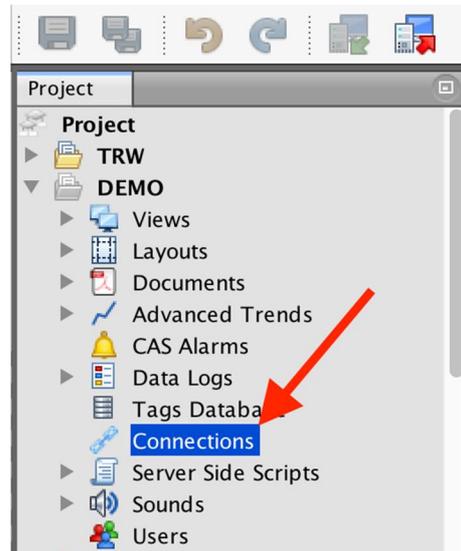
Example of chart with option fill under (opacity set to 70%)



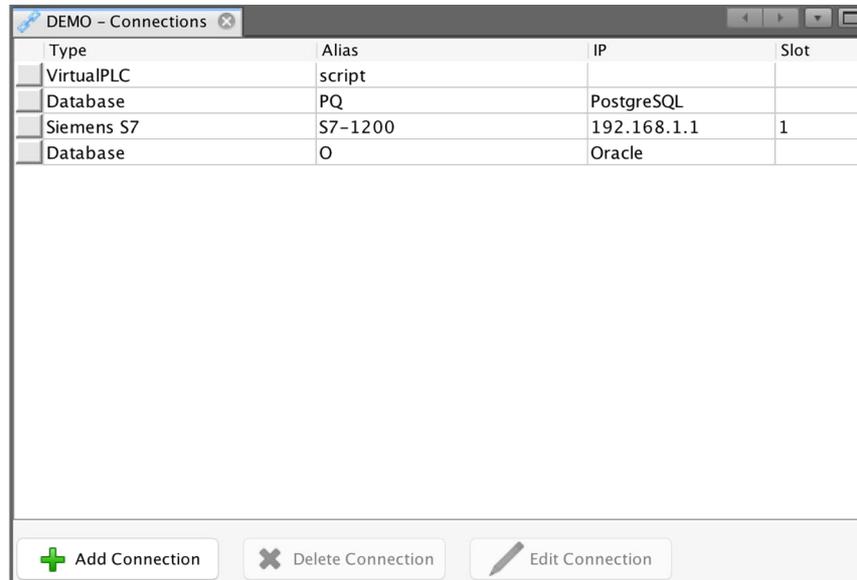
# 31 Connections

Watch video describing this functionality: [https://www.youtube.com/watch?v=9\\_iUEjRtFAc](https://www.youtube.com/watch?v=9_iUEjRtFAc)

To view existing connections, select the *Connections* folder in the *Project Window*.



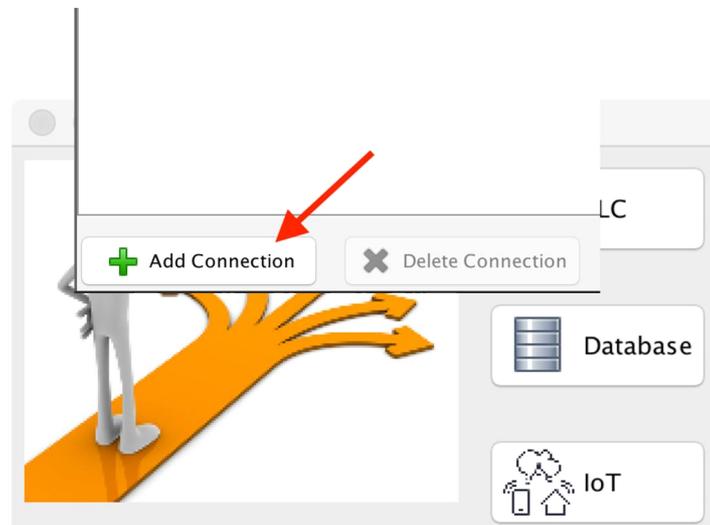
Now double click on the Connections folder, and you will be presented with the Connections Window:



In this window, you can see all of your defined connections.

## 31.1 Creating New Connection

- 1) Create a new connection by clicking on the **Add Connection** button at the bottom of the main window.



To add a new connection, you must first select the type of connections you want to create:

There are three possible types:

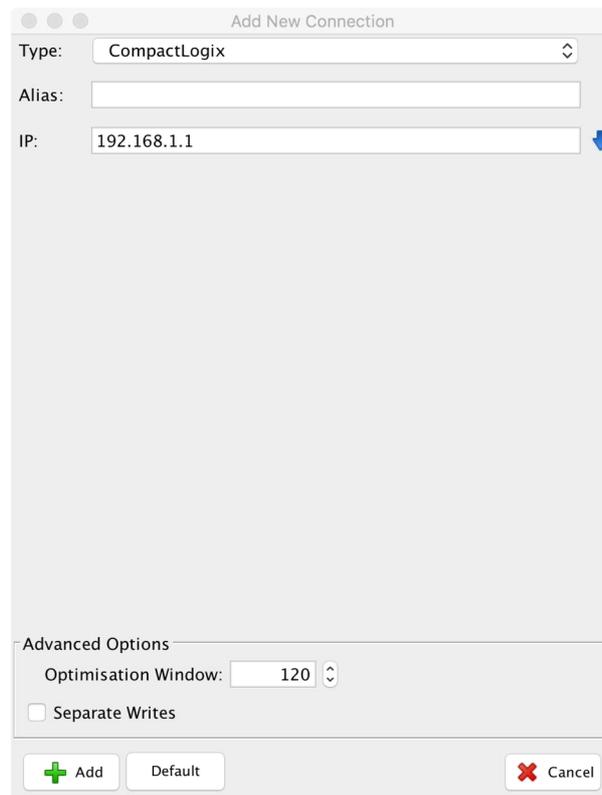
**PLC:** PLC, PAC, and DCS connections

**Database:** SQL Database connections

**IoT (IIoT):** Internet Of Things Protocols

## 31.2 Creating PLC Type Connection

First, click on the PLC type connection. A dialog window will open where you can enter specific connection parameters in order to create a viable communication channel with the PLC.



The screenshot shows a dialog window titled "Add New Connection". It contains the following fields and options:

- Type:** A drop-down menu with "CompactLogix" selected.
- Alias:** An empty text input field.
- IP:** A text input field containing "192.168.1.1" and a blue downward arrow button.
- Advanced Options:** A section containing:
  - Optimisation Window:** A text input field with "120" and a drop-down arrow.
  - Separate Writes:** An unchecked checkbox.
- Buttons:** At the bottom, there are three buttons: "+ Add" (green plus icon), "Default", and "Cancel" (red X icon).

- **Type** – select the type of PLC from the drop-down box at the top of the window
- **Alias** – enter the name of the connection
- **IP** – enter the IP address of the PLC

*NOTE: The dialog content depends on the selected PLC type.*

2) Once all the valid information is entered, click on “+ **Add**” at the bottom of the window.

## 31.3 Creating Database Type Connection

mySCADA is very strong in connecting to external SQL databases. It is surprisingly easy to connect and use SQL databases in your project. To get started, please select the Database type. You will be presented with the configuration dialog:

The screenshot shows a window titled "Add New Connection". On the left, there are several input fields: "Type" (a dropdown menu with "PostgreSQL" selected), "Alias" (text input "PSQL"), "Host" (text input "192.168.1.1"), "Port" (text input "5,432" with a small spinner), "User" (text input "myscada"), "Password" (text input "myscada"), and "Dbname" (text input "main"). On the right, there is a "Query" editor with a "Read" button and a "Write" button. The query text is "1 SELECT \* FROM table\_name". At the bottom of the window, there are three buttons: a green "+" button labeled "Add", a "Default" button, and a red "X" button labeled "Cancel".

## Database Type

As you can see, the first option is the Type of the database. You have the following options:

- PostgreSQL
- Microsoft SQL
- Oracle
- mySQL
- sqlite
- ODBC

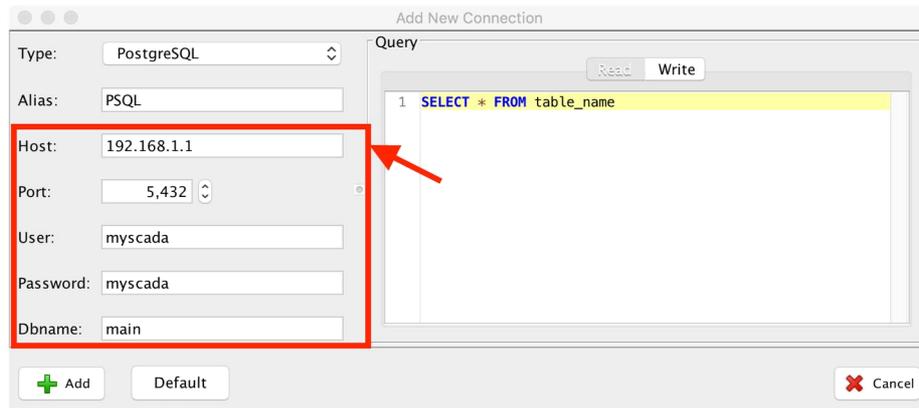
As you can see, all major databases are supported. On the Windows platform, you can also connect the other database types using the ODBC driver.

## Alias

Again, the next option is **Alias**, and you must fill it to use your connection in your project.

## Database Specific Options

Then you should fill in the database specific options:



## SQL Query Specification

On the right side of the dialog, you can specify an SQL Query to the database. This query will be used to retrieve data from the database. Then you can use this data anywhere in your project. Query dialog type is divided into two sections:

### Read

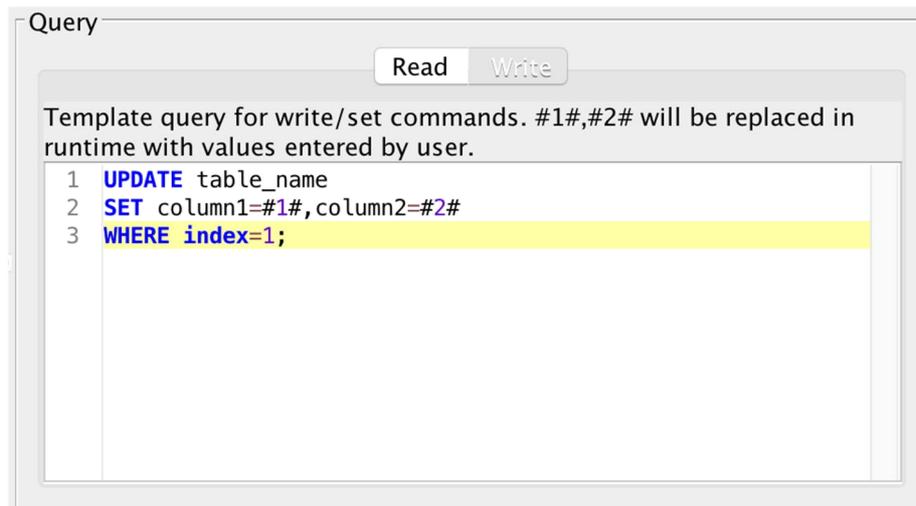
SQL Read Query. This query is used to read values from the database. You can specify any valid query to the database, including conditional statements.

### Write

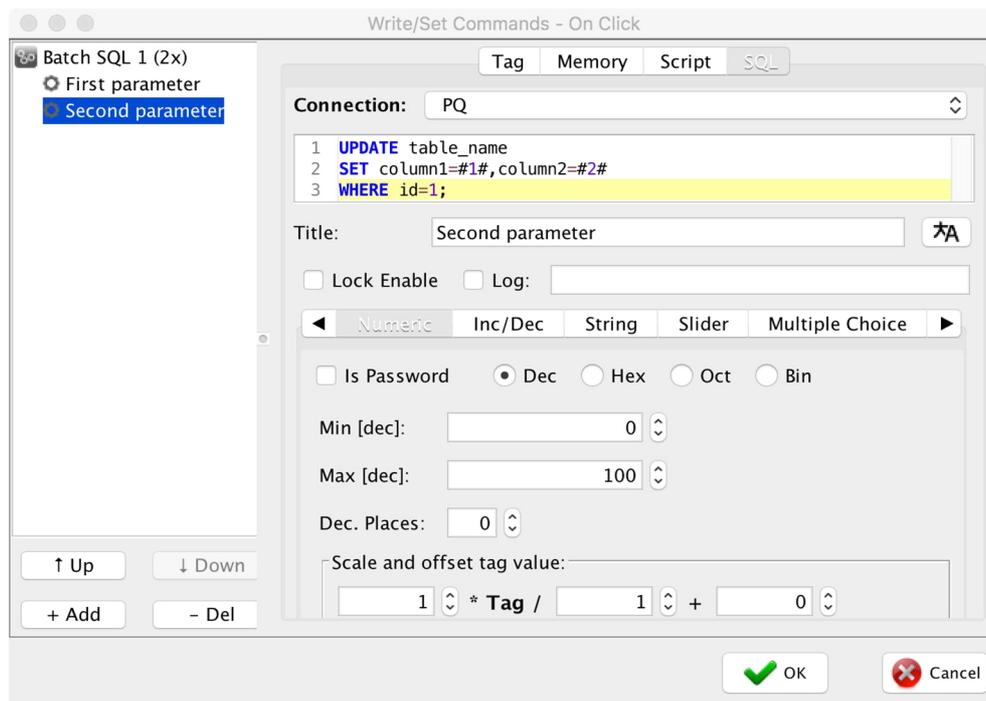
In the Write Query section, you should specify a TEMPLATE for the SQL Query, which will be used later for the write commands. In the query, you should use a placeholders #1#, #2#, and so on in place of values. The placeholders will be replaced with real values provided from the write/set command during runtime.

Follow these steps to write values to the database:

1. Specify a Write SQL Query in the connection dialog. In our case, we will write 2 values to the database



2. Use the Write/Set command to write values to the database using the template query



3. When the write/set command is executed, your query to the database will change to:

```

UPDATE table_name
SET column1=10,column2=12
WHERE index=1;

```

TIP: You can create multiple connections to one database to use different SQL queries and retrieve multiple data.

## 31.4 Creating IoT Type Connection

mySCADA is a modern evolving system built on open standards. As the industry evolves, we are evolving with it. Currently, mySCADA supports the international widespread IoT network – SigFox. You can connect to the SigFox servers directly from mySCADA and visualize any data you can get from the network. To do so, please select the IoT type in the

The screenshot shows a dialog box titled "Add IoT Connection". At the top, there is a "Type:" label followed by a dropdown menu currently showing "SigFox". Below this is the SigFox logo with the tagline "One network A billion dreams". Underneath the logo are several input fields: "Alias:" with the text "SFX", "Login:" with "LoginHash", and "Password:" with "PasswordHash". There is also a "Timeout:" section with a checked checkbox, an input field containing "600", and "[sec]" to its right. At the bottom left is a green "+ Add" button, and at the bottom right is a red "X Cancel" button.

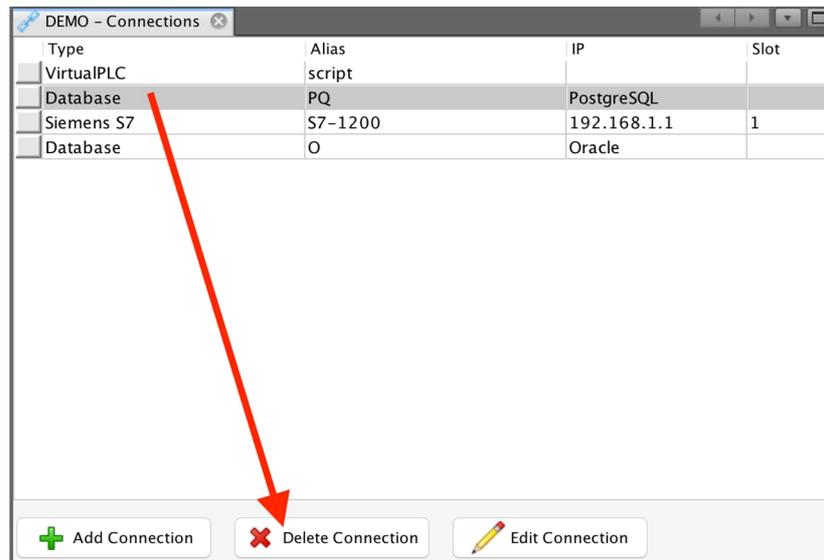
connection. Then you will be presented with the settings dialog:

Fill in the Alias to be able to use this connection in your project. Then fill in the Login and Password generated from your account in the SigFox backend server.

If you specify a timeout interval, mySCADA will process the time of every message, and if the time is over the specified limit, you will be notified by an error communication about it.

## 31.5 Deleting Connections

If you want to delete some connections, select them and click on the **Delete Connection** button right next to the **Add Connection** button at the bottom of the main window.



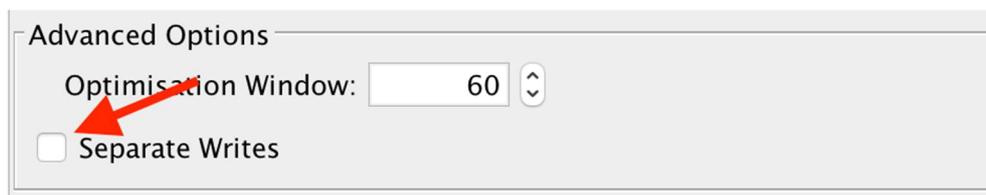
*TIP: To edit any of the existing connections, double-click on the connection row.*

## 31.6 Advanced Options - Optimizations

mySCADA in default settings is highly optimized for optimal speed and memory requirements of typical hardware. However, further optimizations can be made for speeding up sensitive applications.

### Separate Writes Optimization

When a connection to PLC is defined, mySCADA will use one socket for communication with this PLC. If there are write requests in the queue, mySCADA will process them before it starts reading. If there is a large number of write requests, the read period will increase accordingly. To eliminate this behavior, mySCADA can set write requests to PLC on an independent channel (e.g. New socket connection). To do so, enable "Separate Writes" in the Connection definition.



### Optimization Window – Block Read Optimization

mySCADA is using optimized data read and writing from PLCs; this is achieved by combining all read/write requests for the given PLC into blocks of data to be read. Each connection has the maximum size of the data block set to default values. You can override this value by changing the size of the data block to be read at once. There might be two reasons to do so:

- Your PLC is not able to provide data in such a large block; in that case, you must decrease the **Optimization Window** size

- Your PLC can provide data in a larger block than selected, if you increase the **Optimization Windows** size, you can achieve higher read speeds from your PLC.



Advanced Options

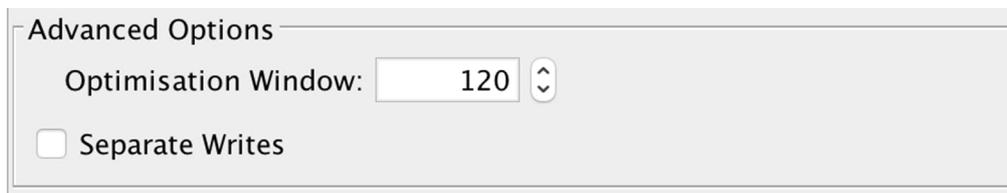
Optimisation Window:

Separate Writes

### Multiple Channels Optimization

As already stated previously, mySCADA uses one communication link to the PLC. If there are multiple, fast-changing requests (for example, a high number of users changing views), mySCADA will optimize communication data quite often. To eliminate this, you can create multiple connections to the same PLC. For example, you can create one connection for CAS Alarms, one for data-logs, and one for views. This way mySCADA will use three simultaneous connections to the PLC. As CAS Alarms and data-log requests don't change, mySCADA does not have to optimize the data blocks for these two connections. Only the third connection will have to be optimized. This way high-speed data exchanged with PLC can be achieved.

To use multiple channel optimization, your PLC must allow for enough simultaneous connections.



Advanced Options

Optimisation Window:

Separate Writes

## 32 User Accesses

---

Security is a major concern for every modern SCADA system. In *mySCADA*, you can limit access from the whole project to a single element control. This way you can easily control user access to your project and keep security at the highest level. In addition, if there is more than one person using the technology, you can utilize the *user access* function to set certain limitations.

### 32.1 Access Levels

Watch video describing this functionality:

<https://www.youtube.com/watch?v=A6lloVDPH4>

Access Levels are security groups with a chosen level of access. You can set different access levels for different users. For example, maintenance personnel would have lower system access level than administrators, but higher than operators. This way you can control who can access your system and who can operate given technology. *mySCADA* has ten *Access Levels* numbered from “0” to “9”. “0” (zero) is the lowest access available while “9” is the highest access level possible.

For better user orientation, you can give specific names to these user access levels.

### 32.2 Access Groups

Watch video describing this functionality:

<https://www.youtube.com/watch?v=9maAmwbKPDc>

Every user in the *mySCADA* system has a defined unique name and security password. In *mySCADA*, every user can be a part of a user group or multiple groups. There can be an unlimited number of groups in the system. Groups can be assigned to tie users together for common security, privilege, and access purposes. This is the foundation of *mySCADA* security and access. In *mySCADA*, you can grant access based on group names.

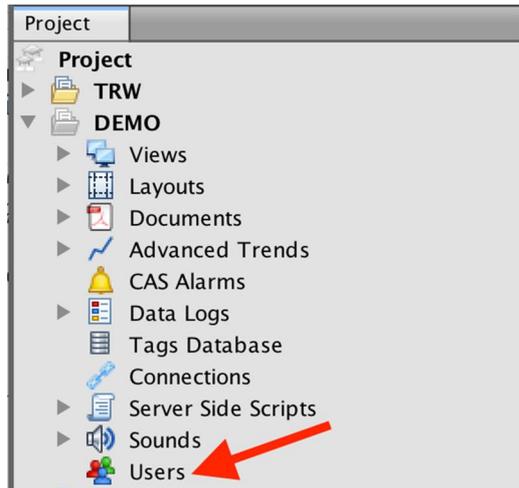
For every user, you can specify a group or groups to which a user belongs. Then, anywhere in the project, you can limit access to this user (or users) by specifying the group to allow access.

### 32.3 User Accounts

In order to use User Access Levels, you have to specify users for your project. You can create as many users as you need. Each user must have a specified Access Group ranging from 0 to 9 - the higher the number, the higher access rights for the user.

*TIP: If you wish to use authentication based only on Access Groups, you can leave the Access Level equal to 0.*

To specify user accounts, select Users from the project tree and double click it.



You will be presented with the Users Window

The screenshot shows the 'Users' window with the following components:

- Projects Panel:** Shows a tree view with 'yourProject' expanded to show 'Users'.
- Users Table:**

ID	Name	Access Group	E-mail	Tel.	Set system	Set network	SMS Control
2	admin	9			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	worker1	4			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	worker2	3			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
- Buttons:** '+ Add User', 'X Delete User', and 'Edit User'.
- Groups Table:**

Group	Name	Reporting	Severity Up To
0	No Access	<input type="checkbox"/>	0
1	Visitor	<input type="checkbox"/>	0
2	Junior Operator	<input type="checkbox"/>	0
3	Senior Operator	<input type="checkbox"/>	0
4	Supervisor	<input type="checkbox"/>	0
5	Manager	<input type="checkbox"/>	0
6	IT Specialist	<input type="checkbox"/>	0
7	Instrument Technician	<input type="checkbox"/>	0
8	Engineer	<input type="checkbox"/>	0
9	Administrator	<input type="checkbox"/>	0
- Properties Panel:** Shows 'Number of Users' as 3.

To add or modify existing users, simply write the values into the provided table:

The screenshot shows the 'DEMO - Users' window with the following table:

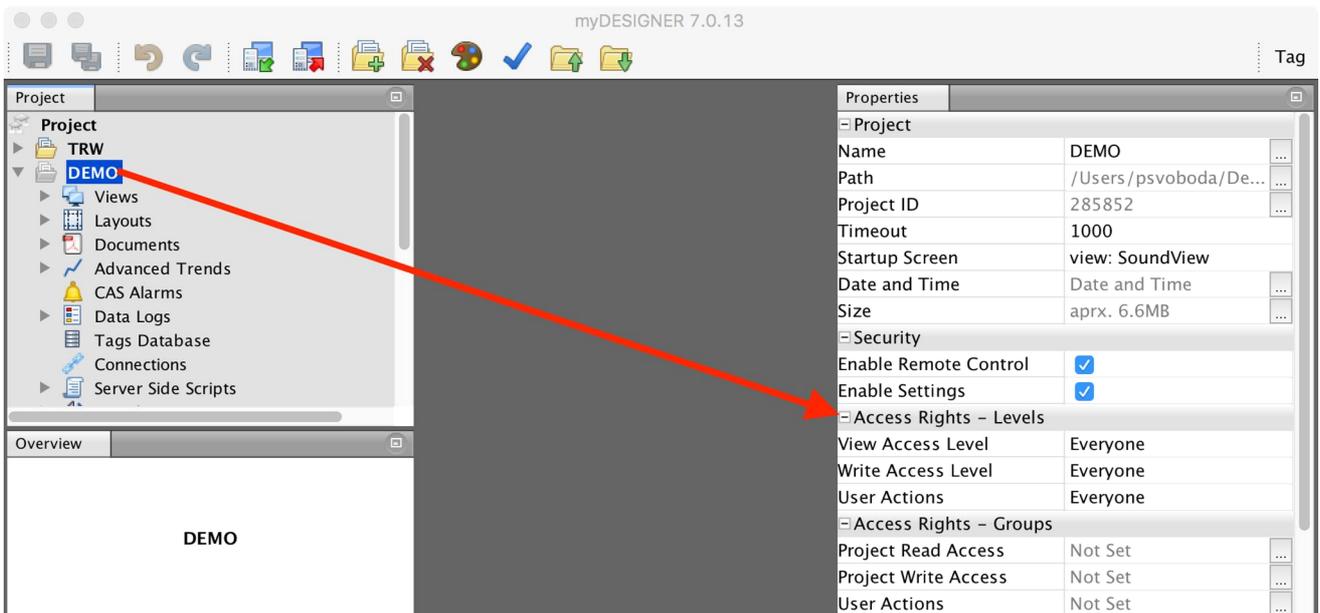
ID	Name	Password	Access Level	Part of Groups	E-mail	Tel.	Set system	Set network	SMS Control
1	Operator	***	1	operators	operator@myscada....	+420602418889	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	Administrator	***	9	admins, operators, e...			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
new			0				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Available options are:

Option	Description
Name	Username
Password	Password. Saved password is encrypted for higher security.
Access Level	Specify an access level for given user
Part of Groups	Specify an access group or groups of which a given user is a part.
E-mail	Provide user's email to receive email notifications.
Tel.	Provide user's phone number in the international format to receive SMS notifications.
Set system	myBOX specific settings: enable to change system setting on myBOX
Set network	myBOX specific settings: enable to change networking setting on myBOX
SMS Control	myBOX specific settings: enable to control myBOX with SMS commands remotely.

## 32.4 Limiting Access for Whole Project

You can specify which users will have access to the project. To specify which users can access the project, select your project from the project tree and look at the properties.



As you can see, you can limit access to your project either based on Access Levels or based on Access Groups.

### Setting Access Levels:

Navigate to the Access Rights - Levels and fill in the appropriate values:

Access Rights - Levels	
View Access Level	Everyone
Write Access Level	Everyone
User Actions	Everyone

**View Access Level:** Sets the minimal level of access for viewing your project.

**Write Access Level:** Sets the minimal level of access for writing values to the PLCs

**User Actions:** Enables access for viewing user actions log

### Setting Access Groups

Navigate to the Access Rights - Groups and fill in the appropriate values:

Access Rights - Groups	
Project Read Access	Not Set
Project Write Access	Not Set
User Actions	Not Set

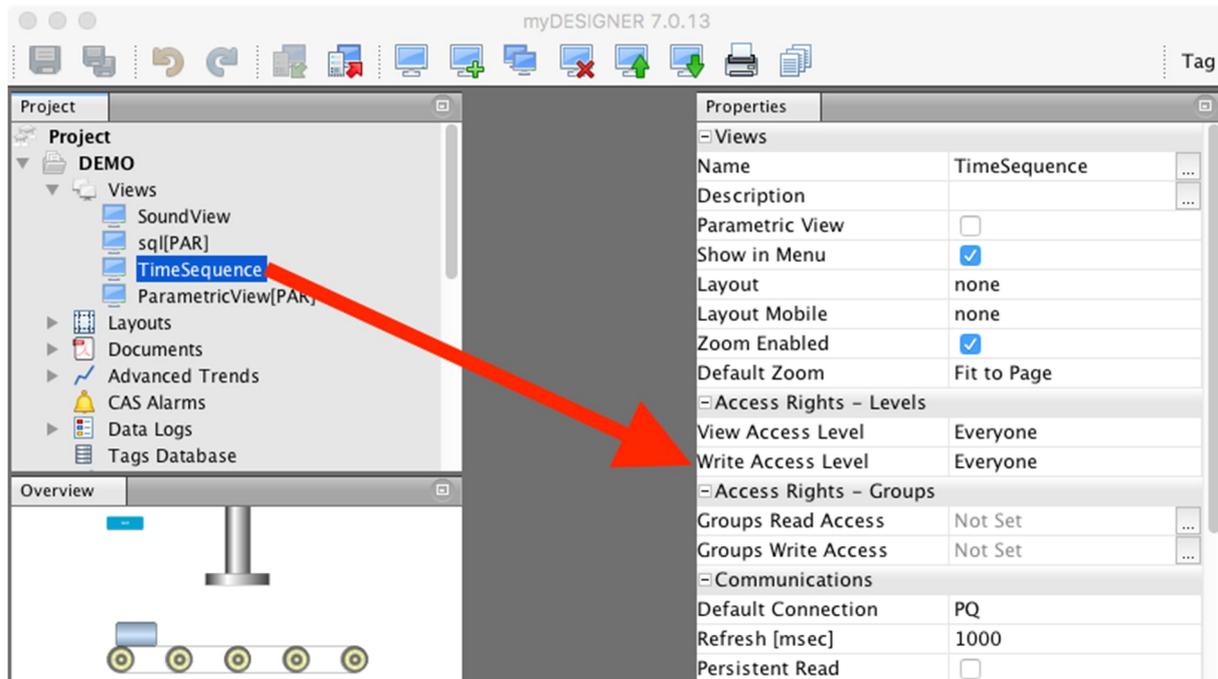
**Project Read Access:** Specifies which groups of users can view your project

**Project Write Access:** Specifies which groups of users can write values to the PLCs

**User Actions:** Enables access for viewing user actions log

## 32.5 Limited Access for Views and Trends

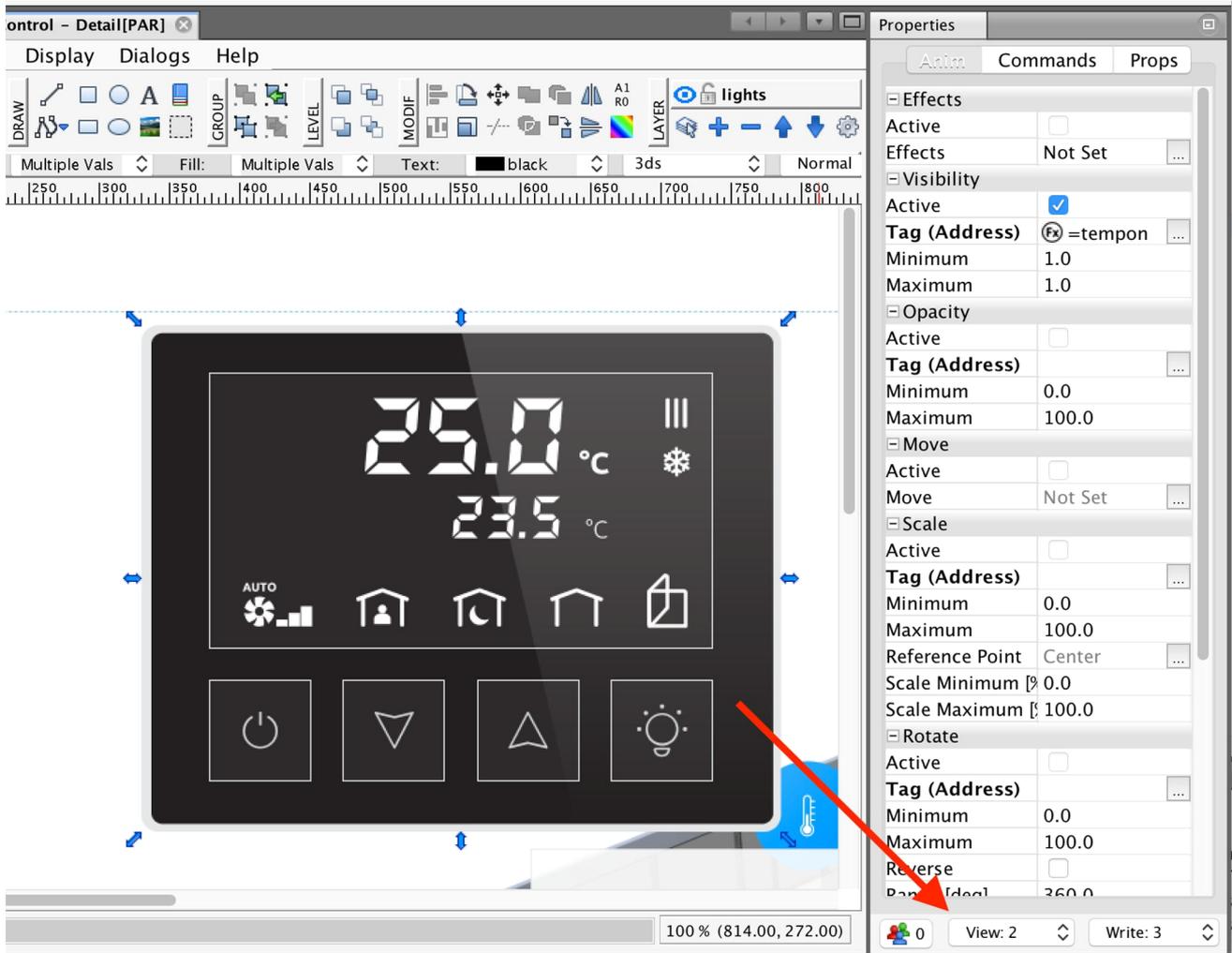
You can limit the user access for each view and trend individually. Select a desired view or trend in the *Project tree* to display its properties.



## 32.6 Limited Access of Arbitrary Object in Views

You can set the user access rights to even more precise detail. The access rights can be set for each object in a view, allowing creation of complex and customizable visualizations.

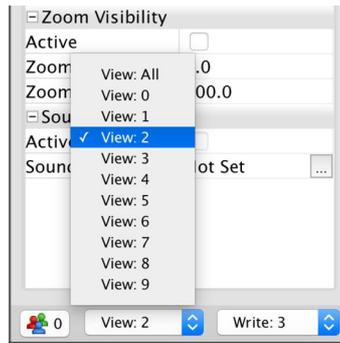
Select the target object and set the *View* and *Write Access* values:



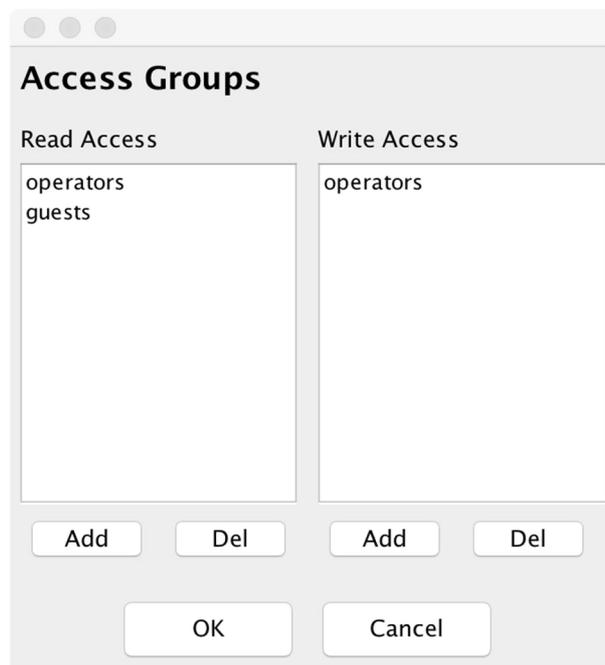
As you can see, there is an Access Settings section right at the bottom of the properties window:



As you can see, you can limit access to the access levels by simply selecting the desired level from the combo box:



Alternatively, you can limit access based on the user groups simply by clicking on the Access Groups Button. Then you will be presented with the Access Groups Settings dialog:



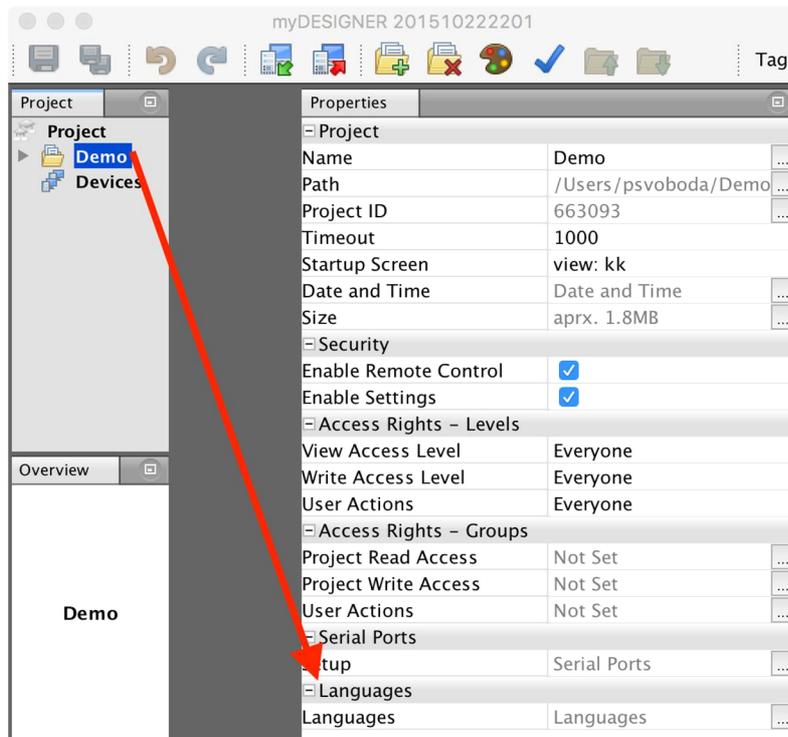
As you can see, there are two sections in the dialog, allowing you to set up read (visibility) access and write access independently.

**DOWNLOAD DEMO PROJECT HERE:**

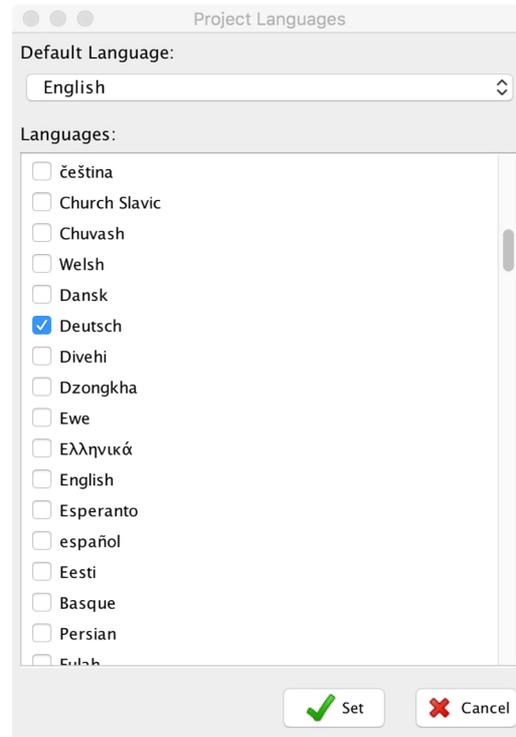
[http://nsa.myscada.org/projects/example/Limited\\_Access.mep](http://nsa.myscada.org/projects/example/Limited_Access.mep)

## 33 Multi-language support

In mySCADA, you have the ability to create projects with multiple languages. This way, based on user preference, users can view your project in multiple languages. To get started, click on your project; in properties, go to the section Languages and click on “...” button.



A new Dialog, is shown:



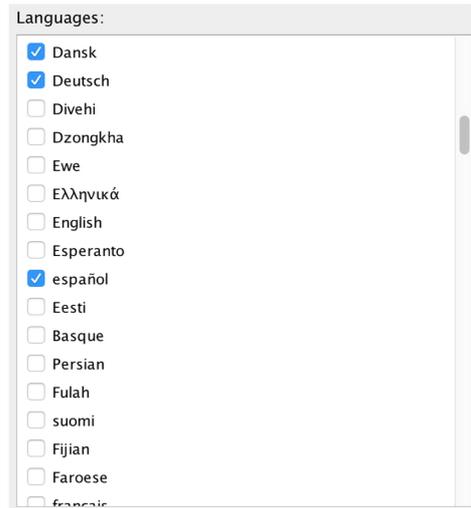
Here, you can select all languages that you want to have in your project.

First of all, look at the section “**Default Language.**” This is the default language for your project. All defined texts in views, names of views, trends, data-logs, etc. are in your default language. The default language is automatically set upon project creation to the default language of your computer.



*Important: If you wish to create a project in a different language than the language of your computer, please change it upon project creation!*

The next section in the Languages Dialog is available languages. Please select all the languages that you want to support in your project. For the parts of the project where you don't provide translation to a given language, those parts will be shown in the project's default language.



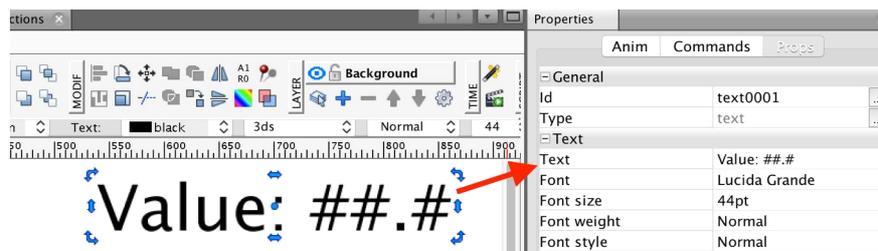
*Tip: You can add languages to your project at any time.*

### 33.1 Providing translations inside a Project

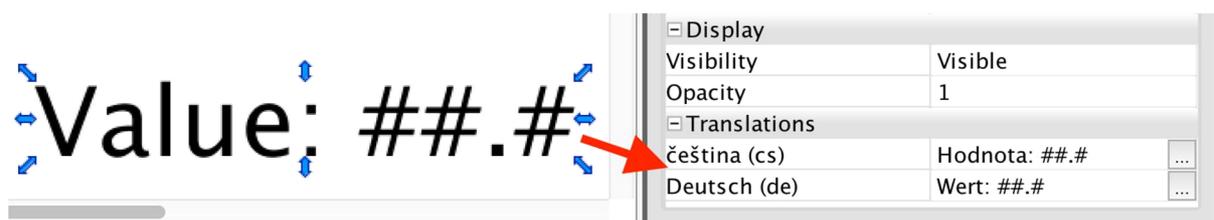
Every textual part of the project, such as text elements in views, view names, trend names, data-log names, etc. can have its own translation. The following screens show how to prepare translations for every part of your project.

#### Textual elements in Views:

Each textual element in view has its own properties. In properties, you can find the default text under the Section "Text" in item "Text".

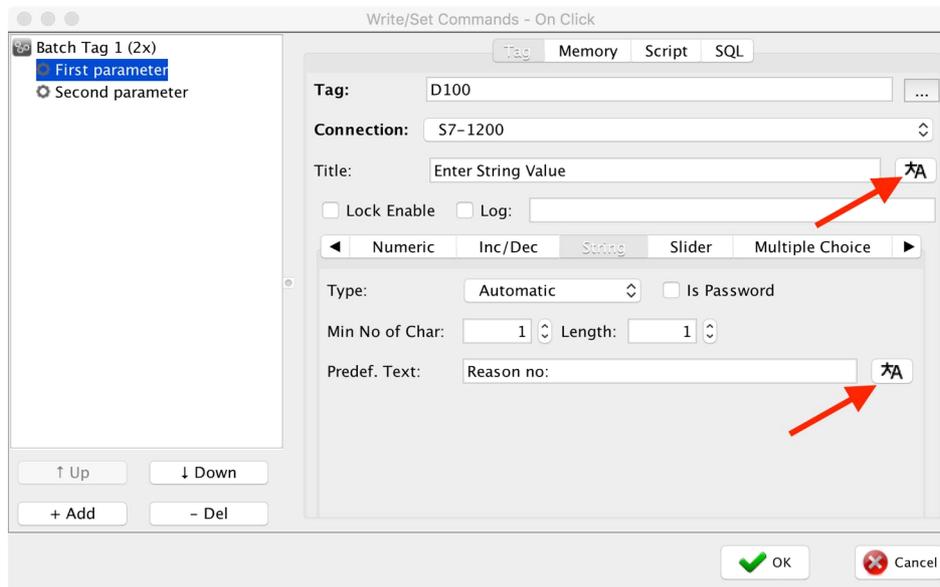


To provide a Translation for textual elements, fill in the section "Translations" in properties. Each language defined in the project has its own item. Here you can fill translation for every language.



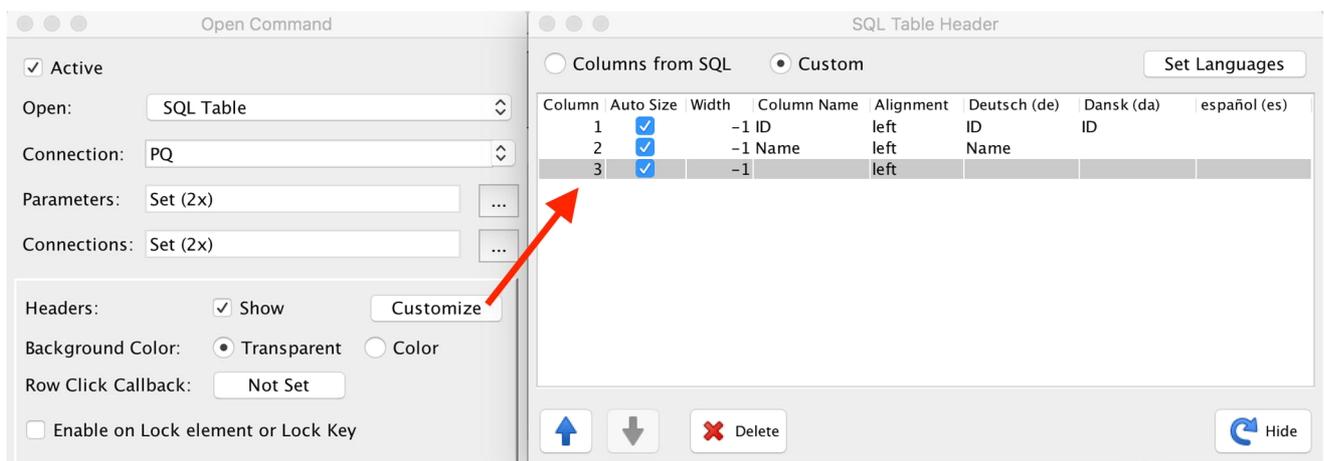
## Write/Set Commands:

Write/Set command can contain text as well. You can comfortably set multiple languages for write/set command text items by clicking on the translation button located next to the text item:



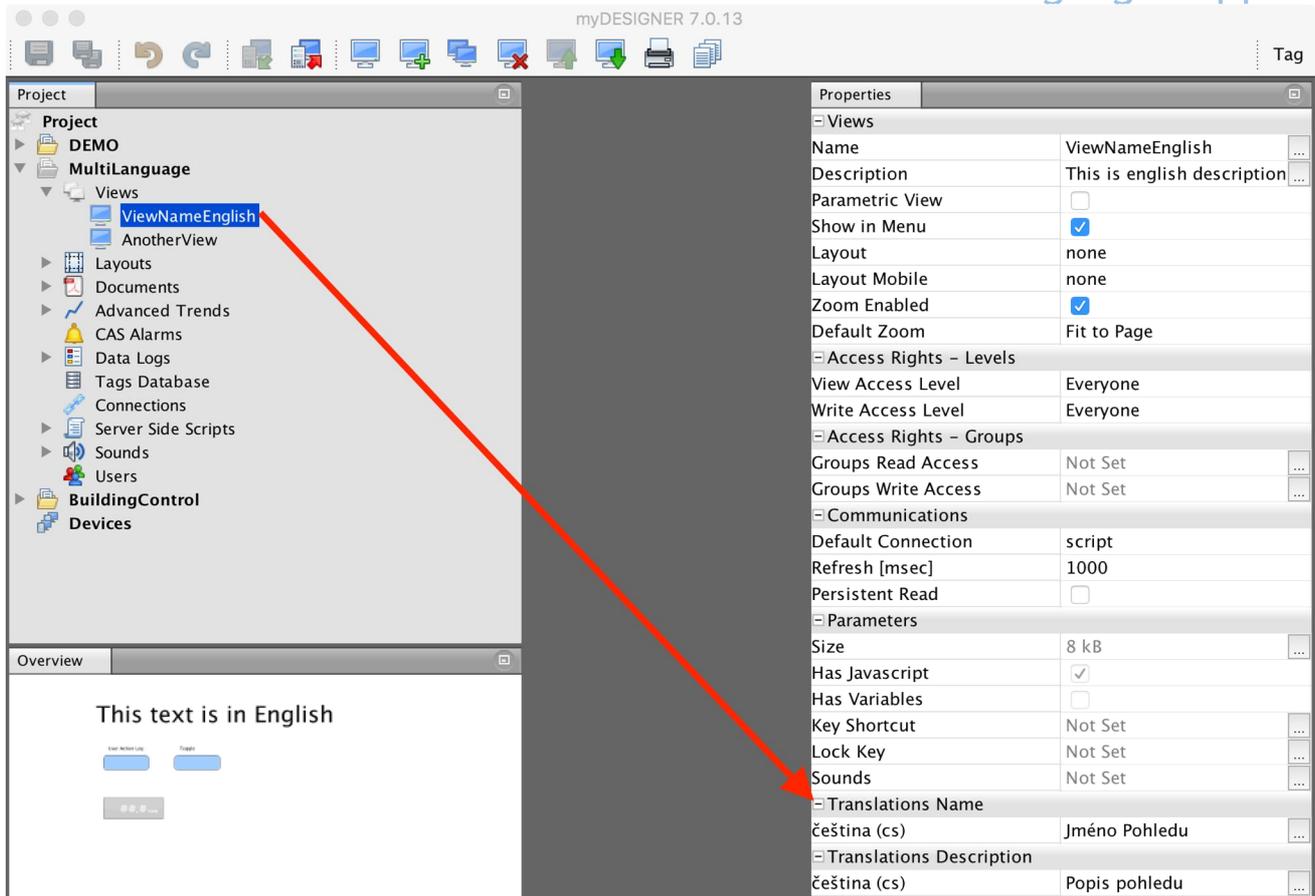
## Translating Open Command

Open Command provides language-specific settings for the option SQL Table. You are able to provide the header column names in the default language as well as in all predefined languages.



## 33.2 Translating Names of Views

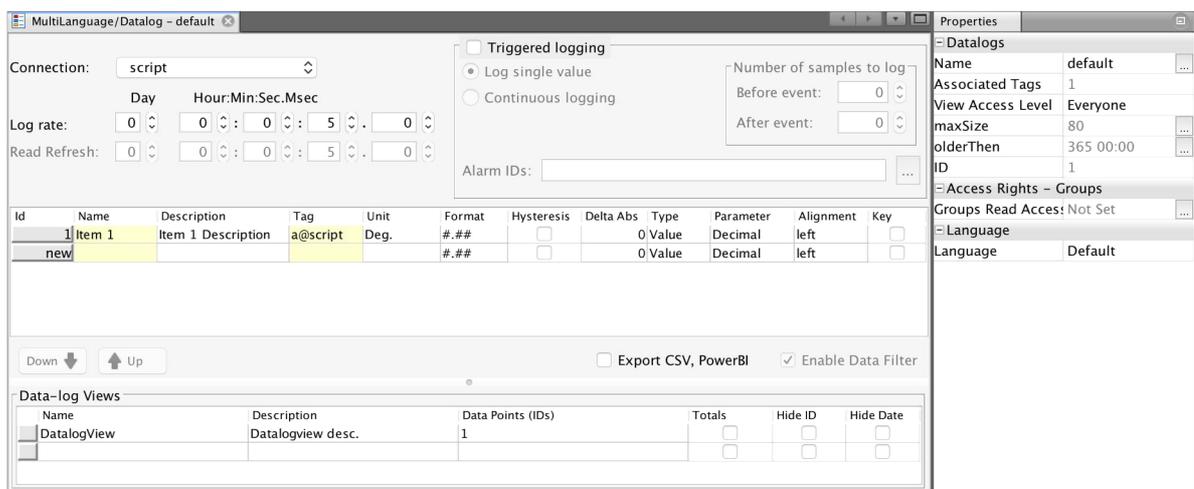
Select your view from the Project tree and then look at the Properties window in the section Translations.



### 33.3 Translating Data-logs

Translation of data-logs is easy. Please follow these steps to provide translation for data-logs.

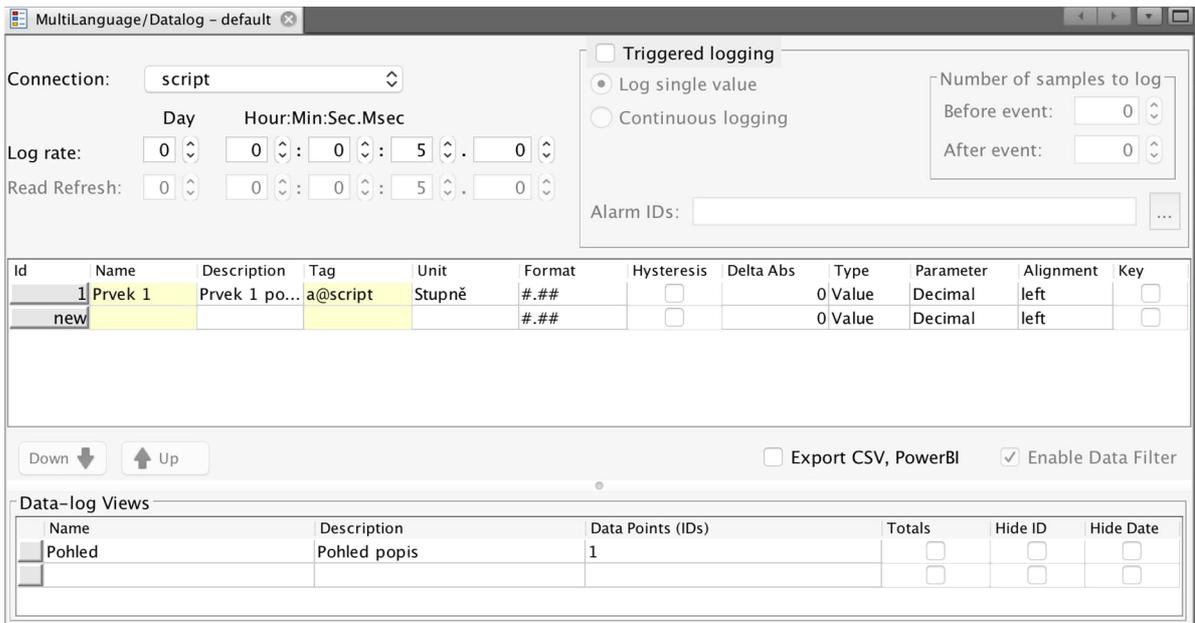
1. You first create data-log in your primary language.



2. Now switch the language by selecting it in the properties



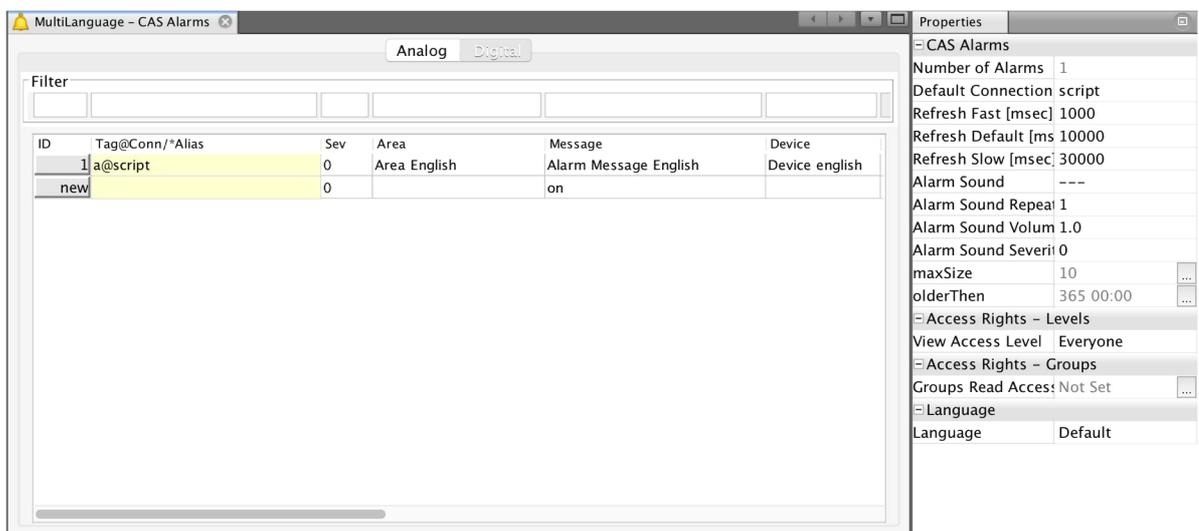
- Your data-log will be now presented in a selected language. When you make changes now, they will be saved in the selected language.



### 33.4 Translating CAS Alarms

Translation of CAS Alarms is easy. Please follow these steps to provide translation for CAS Alarms.

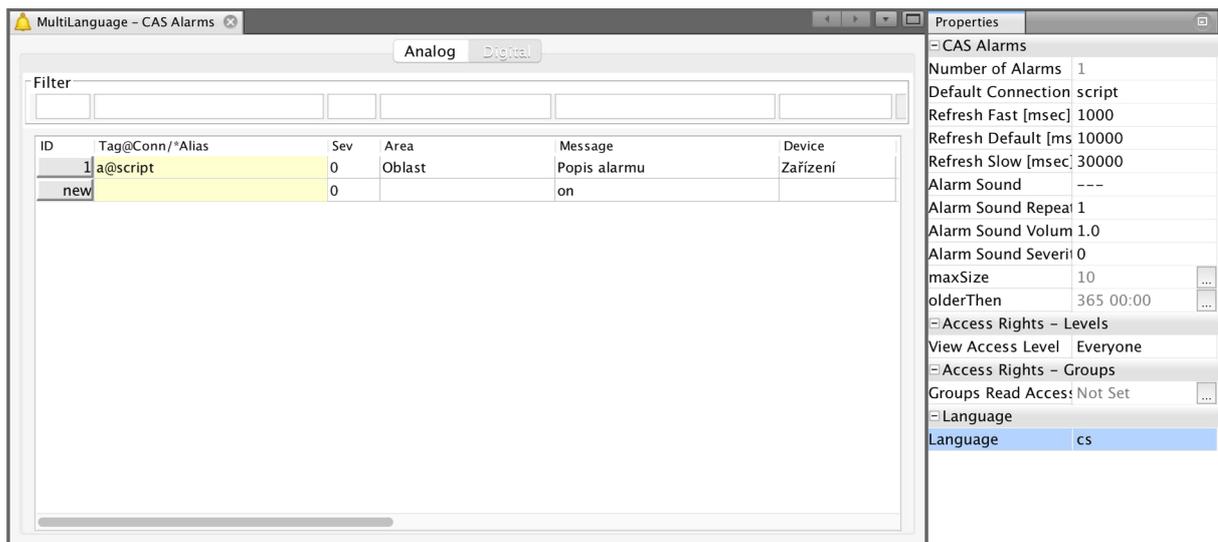
- You first create CAS Alarms in your primary language.



- 2) Now switch the language by selecting it in the properties

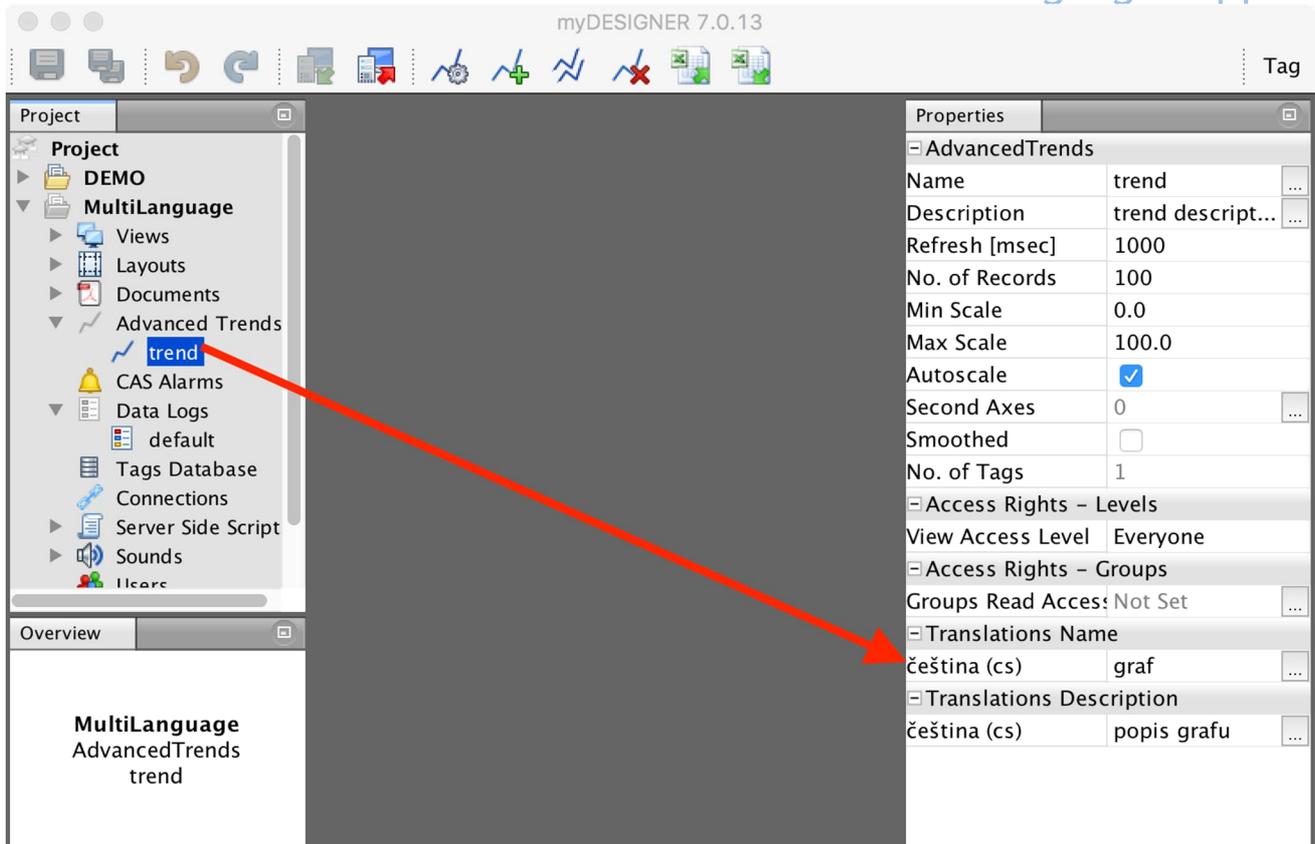


- 3) Your CAS Alarms will be now presented in the selected language. When you make changes now, they will be saved in the selected language.

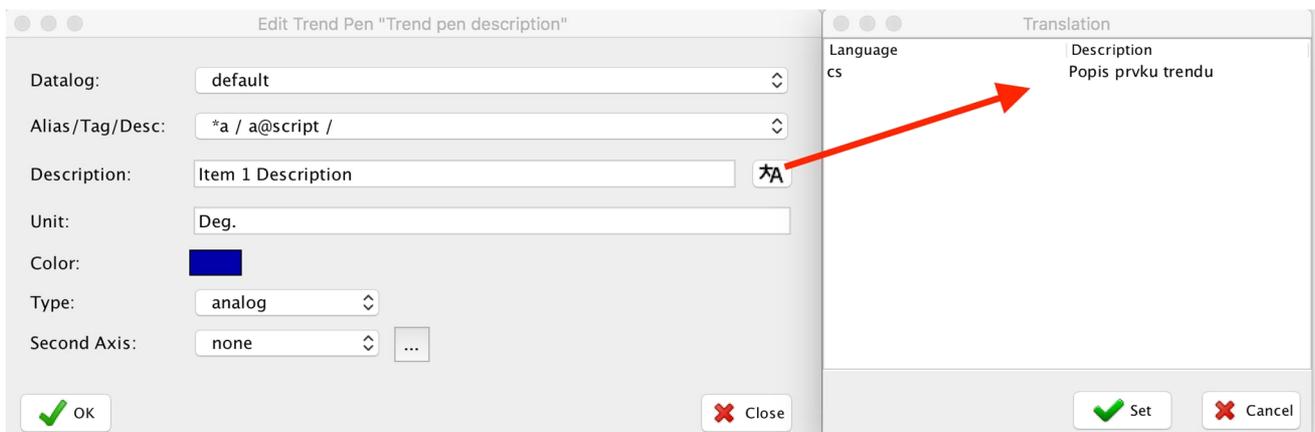


### 33.5 Translating Advanced Trends

Select your view from the Project tree and then look at the Properties window into the section Translations.



Provide a translation for the trend name and description. Then open your trend by double clicking on its name and edit a pen.



When you are editing a pen (or creating a new one) you can provide a translation by clicking on the translation button.

# 34 Server-side Scripts

---

## 34.1 Introduction

Server-side scripting is an easy-to-use, extremely powerful option to extend the functionality of your *mySCADA* system. Server-side scripting uses *JavaScript* as a primary language for writing scripts, which is one of the simplest, straightforward, versatile, and effective scripting languages. It is relatively easy to learn since it uses syntaxes close to English. In addition, you can find a lot of resources and *JavaScript* libraries on the web - one of the main reasons why *JavaScript* has been chosen for server-side scripting on the *mySCADA* platform.

The execution of *JavaScript* is based on the excellent *V8* library from *Google*, which is now the fastest *JavaScript* engine available. Rather than interpreting *JavaScript*, as the old engines used to do, *V8* uses the *Just-In-Time compiler* to produce and execute native instructions tailored to a processor on which the application is running. The generated instructions are cached, avoiding the overhead of repeated code generation and deleted if no longer needed.

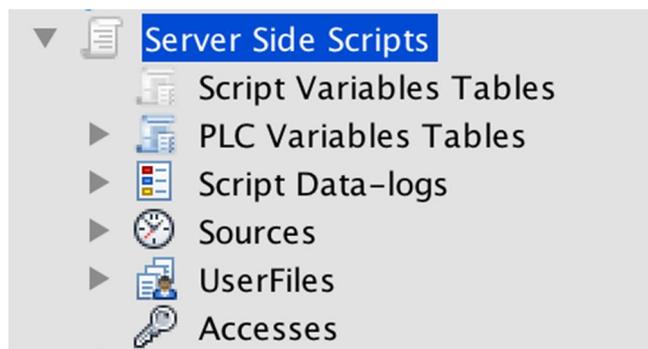
For networking and advanced functionalities, *mySCADA* server-side scripts use the *NODE.JS* toolbox for building fast and scalable network applications. *NODE.JS* uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications.

What can you achieve with *mySCADA* server-side scripting?

- process and analyze the PLC data
- compute statistical data
- create reports and serve them over a web server to a client
- implement complex alarming
- communicate with devices over Ethernet or Serial line
- implement own protocols for specialty devices

## 34.2 Server-side Scripts folder

The *Server-side scripts* folder is located in the *Project Window* menu, as displayed in the picture below.



### 34.3 Server-side Scripts Folder Structure

**Script Variables Tables** – In this section, you can define multiple tables containing the Global script variables. You can use later use this table to read/write into variables inside these tables from scripts.

**PLC Variables Tables** – In this section, you can define multiple tables containing the Global variables linked to the PLC tag. You can use later use this table to read/write tag values defined in the table.

**Script Data-logs** – data-logs and data-log views defined for the server side scripts

**Sources** – source files for the server side script.

**User Files** – used for the user-defined files. This is where you can set user access to generated files, reports, etc.

**Accesses** – limit user access to the generated files from server side scripts.

### 34.4 Variables Tables

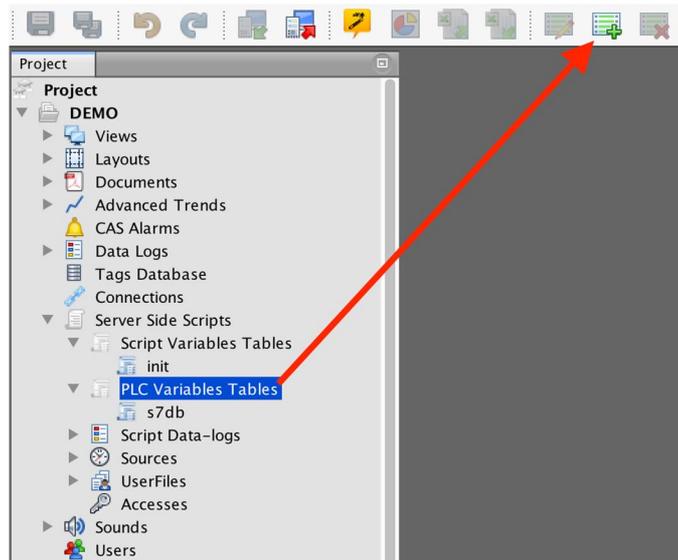
Variables Tables are used to read/write data from different data sources. This is a place where you can define all your global variables connected to the PLCs, databases, or to the memory variables. Once you define your variables in the corresponding tables, you can use them anywhere in the project as well as in server-side scripts.

There are two types of variables tables:

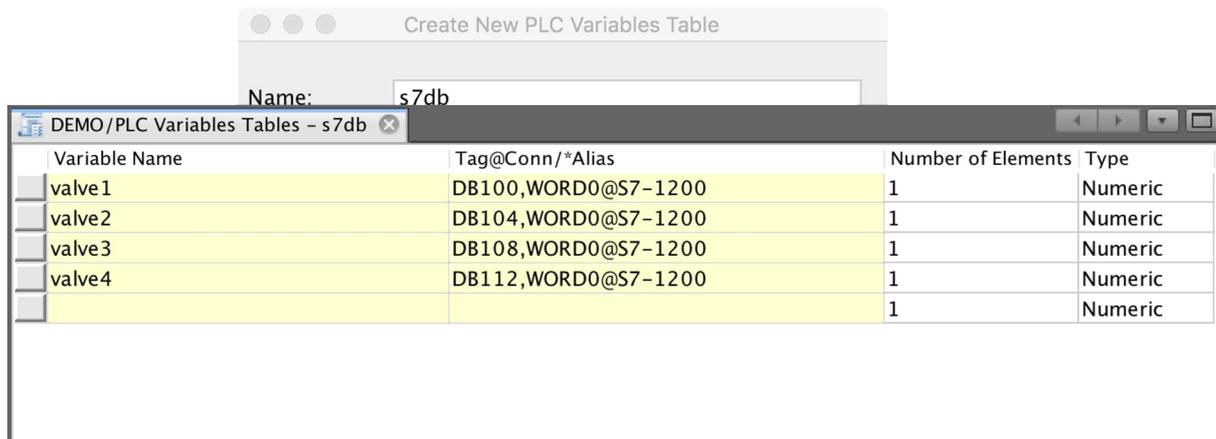
- Script Variables Tables – work with internal memory variables (e.g. virtual PLC)
- PLC Variables Tables – use these tables to read/write values from PLCs, databases, ...

To create a new variables table:

1. Select a Script or PLC Variables Table from the Project / Server side scripts tree and click on **New** in the main toolbar.



2. Enter a new table name. As you can see, each table is associated just with one connection.



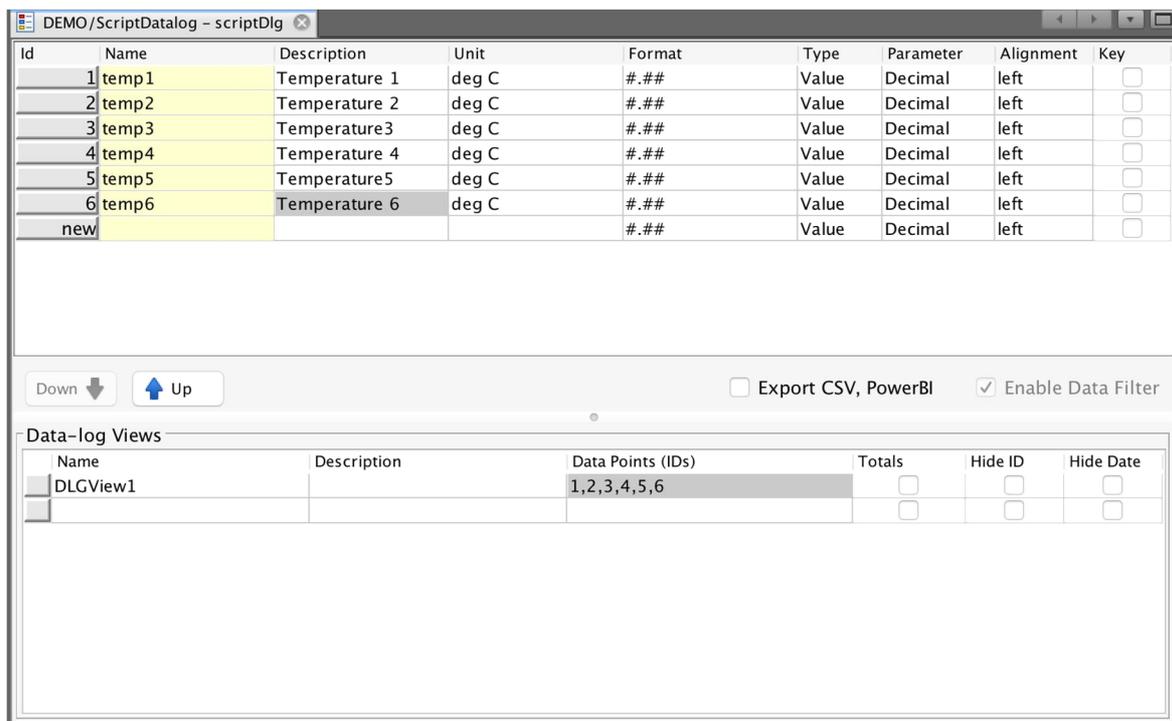
3. Fill in the variables and corresponding tags.

As you can see, we are filling tags to read/write from the PLC. On the left side, we associate the variable with each tag.

## 34.5 Script Data-logs

Script data-logs are same as the data-logs defined in your project, with one exception; mySCADA will only create data-log tables for you, but you should fill in the historical values using script. Here is a simple explanation of how to use script data-logs:

1. Create a new data-log script and open it.



2. Fill in the Items and save it. You can also create data-log views for this data-log.

Now that your data-log is created, mySCADA will automatically create a historical table, and you can fill in the data from the script.

3. Open a main.js from the Sources folder and fill in the function to log data into your script data-log.

```

1  myscada=require('./myscada');
2  myscada.init();
3
4  //periodically write values to the data-log
5  setInterval(function() {
6      // We create new data vector
7      var insertData=[];
8      //we will push in 6 values (in data-log we have defined 6 values)
9      insertData.push(10);
10     insertData.push(100);
11     insertData.push(100);
12     insertData.push(100);
13     insertData.push(100);
14     insertData.push(100);
15     //we will get current time in seconds and milliseconds
16     var time = Math.round(+new Date()/1000);
17     var timems = Math.round(+new Date()%1000);
18     //we will write our data into the history
19     myscada.insertDlg('scriptDlg',time,timems,insertData,function(status) {
20         if (!status) { /* insert error processing */
21             else {
22                 /* successful insert processing */
23             }
24         });
25     }, 1000); //Interval value in milliseconds
26
27

```

4. You are done; now every second you will get a new row of data in the history.

## 34.6 Global Variables

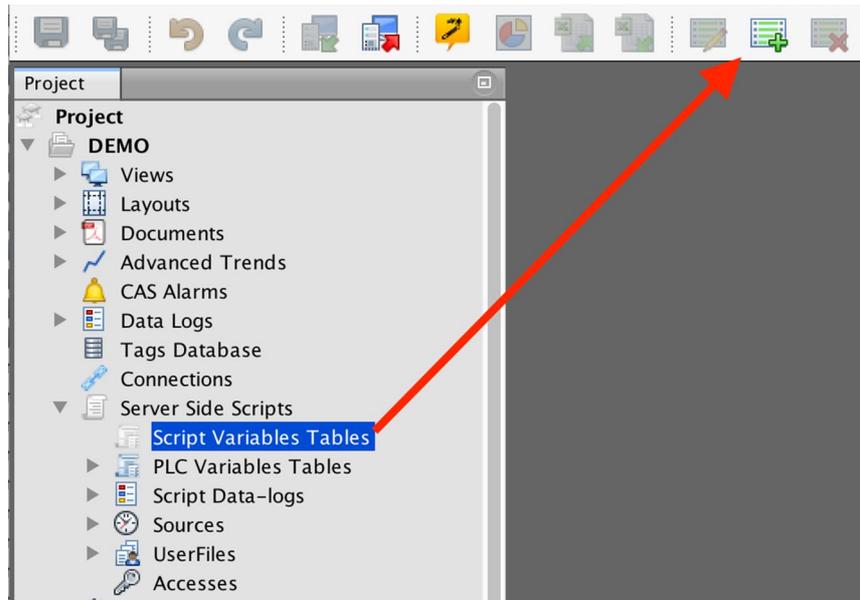
Global variables are variables you can use anywhere in your project. They have global scope. To use a global variable in your project, write it as a regular tag with an alias script.

### variable@script

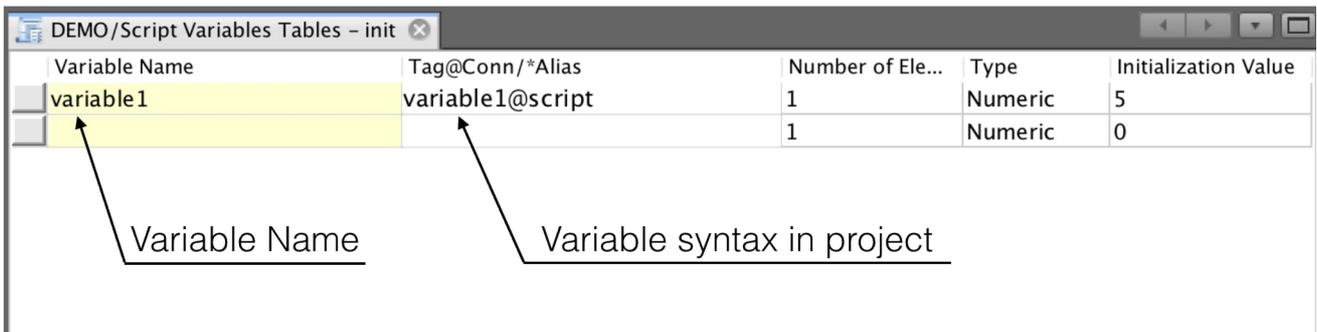
As you can see, the syntax is same as for entering tags.

### Initializing Global Variables

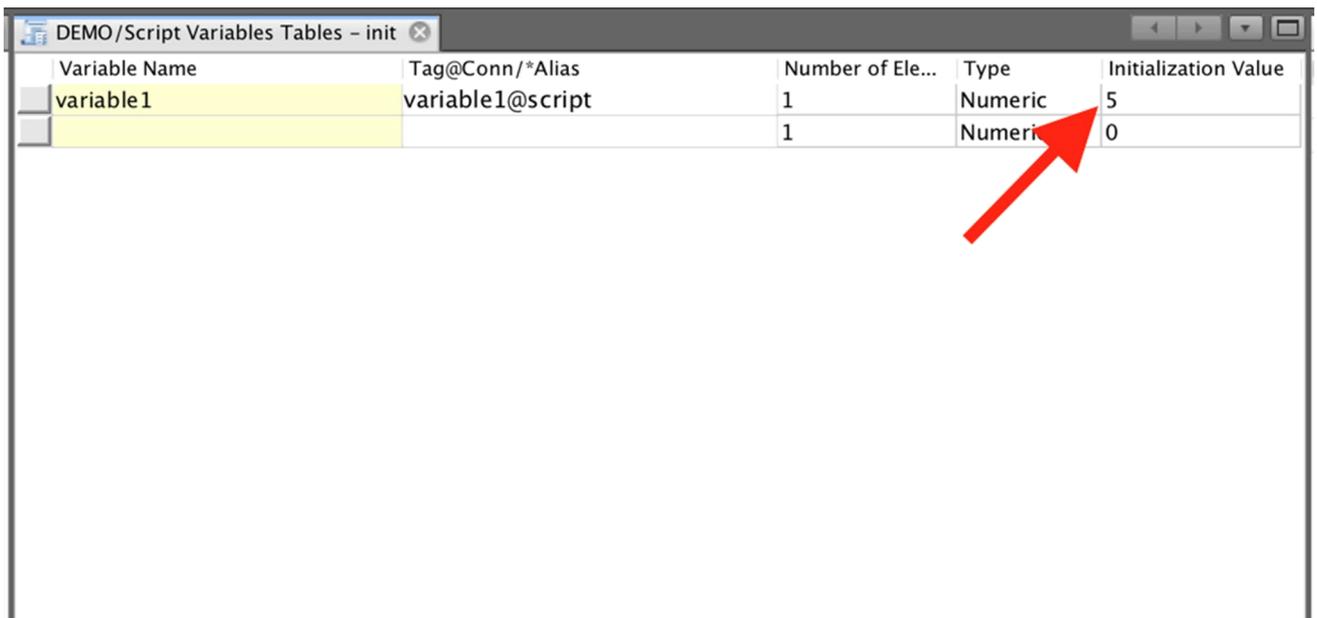
By default, when you use the global variable in the project, it is uninitialized. It is a good habit to define an initial value for the variable. To do so, navigate to the Script Variables Tables and create a new table:



Give it a name and open it.

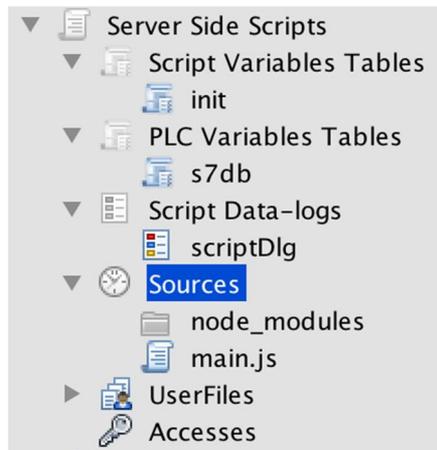


Now fill in your variable name. The Tag syntax is filled in automatically. As you can see on the right side, you can give your variable an initialization value:



## 34.7 Sources Folder

Your server side scripts project is organized into modules. You can find all modules and your main.js script in the sources folder:



In this folder, you can create a new source file or a folder containing additional sources. mySCADA follows the standard file layout for node.js projects. When writing your project, you should organize your project into independent modules.

## 34.8 Organizing Project into Modules

Let's have a look at how to organize the application containing code for a very basic HTTP server:

```
var http = require("http");
function onRequest(request, response) {
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.write("Hello World !!!");
    response.end(); }
http.createServer(onRequest).listen(port, host);
```

Now, let's turn this code into a real *NODE.JS* module that can be used by the main script.

As you may have noticed, the modules are used in the code as follows:

```
var http = require("http");
```

```
...
```

```
http.createServer(...);
```

In *NODE.JS* there is a module called *"http"* that can be used in the code by requiring and assigning the result of this requirement to a local variable. This turns the local variable into an object carrying all the public methods that the *"http"* module provides.

It is common practice to use the module name as the local variable name; however, you may choose whatever you like:

```
var foo = require("http");
...
foo.createServer(...);
```

Now it should be clear how to use internal *NODE.JS* modules.

To create and use your own modules, you do not have to change very much. Making some code for a module means that you need to export the parts of its functionality that you want to provide with scripts requiring this module.

For now, the functionality that the HTTP server needs to export is simple: scripts requiring the server module simply need to start the server.

To make this possible, put your server code into a function named *start()* and export it:

```
var http = require("http");

function start() {
    function onRequest(request, response) {
        response.writeHead(200, {"Content-Type": "text/plain"});
        response.write("Hello World !!!");
        response.end();
    }
    http.createServer(onRequest).listen(8888);
}

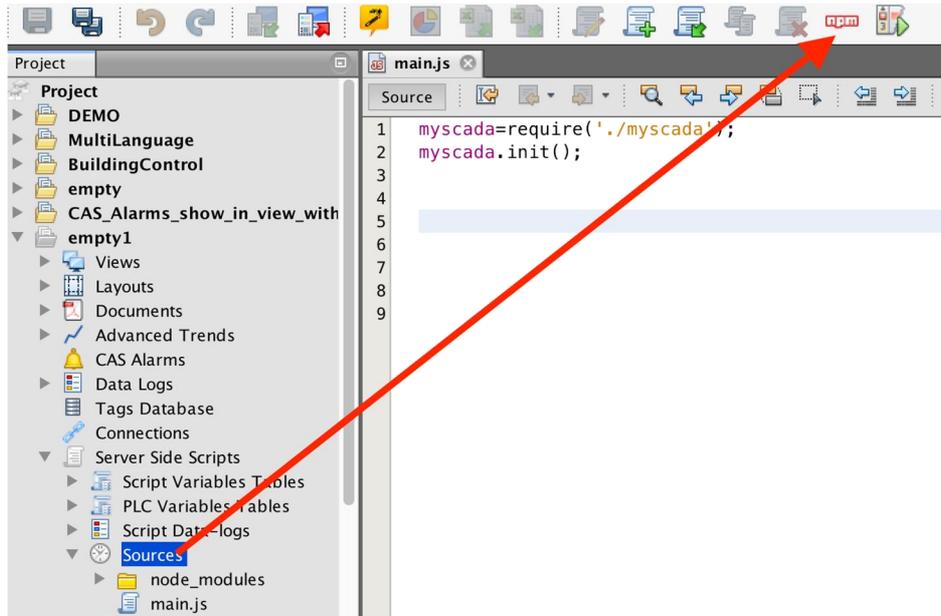
exports.start = start;
```

Now you just save this code into the new folder with name **server** and name it *server.js*. You can start the HTTP server from the main or timed scripts:

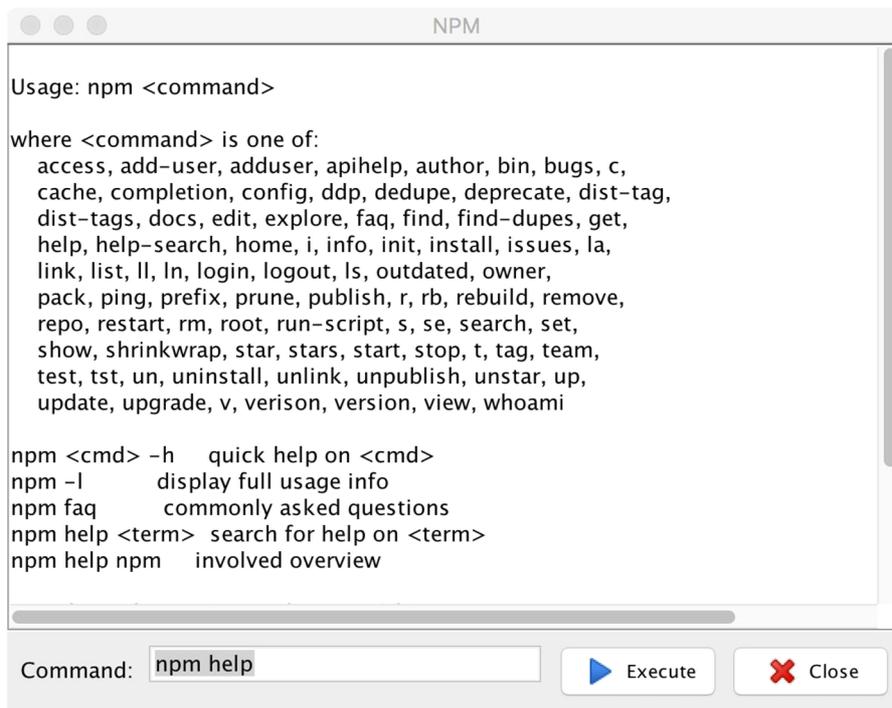
```
var server = require("server");
server.start();
```

## 34.9 Importing Modules

Node.js is a very large project actively developed by many companies. You can find a large number of modules using an integrated npm package manager. To open it, select the Sources folder in the Project tree and then click on the **NPM** icon in the main toolbar.



A new NPM window is opened. If you fill in command `npm help`, you will be presented with integrated help.



## npm install

To install a new module, just type `npm install module_name` and press the “execute” button.

## 34.10 Using the Event-driven Asynchronous Callbacks

The *NODE.JS* approach is not unique, but the underlying execution model is different from other runtime environments like *Python*, *Ruby*, *PHP*, or *Java*.

Let's take a look at the following part of the code:

```
var result = database.query("SELECT * FROM hugetable"); console.log("Hello World");
```

The first line queries the database for lots of rows, and the second line puts "Hello World" into the console.

Let's assume that the database query is really slow due to the number of rows, and it takes too long to execute.

With codes written this way, the *JavaScript* interpreter of *NODE.JS* has to read a complete result set from the database and then execute the *console.log()* function.

If this piece of code were written in PHP, for example, it would work the same way -> "read all the results at once, then execute the next line of the code". If this code were part of a web pagescript, the user would have to wait several seconds for this page to load. However, in the PHP execution model, this would not become a "global" problem (i.e. the web server starts its own PHP process for every HTTP request it receives). If one of these requests results in a slow execution of the code, it slows down the page loading only for that particular user and does not affect other users.

The execution model of *NODE.JS* is different - there is only one single process. If there is a slow database query somewhere in the process, it affects the whole process - everything comes to a halt until the slow query has finished.

To avoid this, *JavaScript* and therefore *NODE.JS* introduce a concept of event-driven, asynchronous callbacks by utilizing an event loop.

We can understand this concept by analyzing the re-written version of the problematic code:

```
database.query("SELECT * FROM hugetable", function(rows)
{
  var result = rows;
});
console.log("Hello World");
```

Instead of expecting *database.query()* to return the result directly, we pass it the second parameter - an anonymous function.

In the previous form the code was synchronous: "first do the database query, and only when this is done write to the console".

Now, *NODE.JS* can handle the database request asynchronously, provided that *database.query()* is part of the asynchronous library. It takes the query and sends it to the database, but instead of waiting for it to be finished, it makes a 'mental note' saying "when at some point in the future the database server is done and sends the result of the query, then I have to execute the anonymous function passed to *database.query()*."

## Server-side Scripts

Then, *NODE.JS* immediately executes *console.log()* and enters the event loop. It continuously cycles through this loop repeatedly, even though there is nothing else to do - events such as “*slow database query*” finally deliver their results.

*Note: This asynchronous, single-threaded, event-driven execution model is not always the only and the best option - it is just one of several models, and it too has its limitations (one being that NODE.JS is just a single process capable of running on one single CPU core). However, this model is quite approachable, because it allows for writing of applications that deal with completion in an efficient and relatively straightforward manner.*

You might want to take time to read Felix Geisendörfer's post on [“Understanding NODE.JS”](#) for additional background explanation.

## 34.11 Creating Server Side Reports

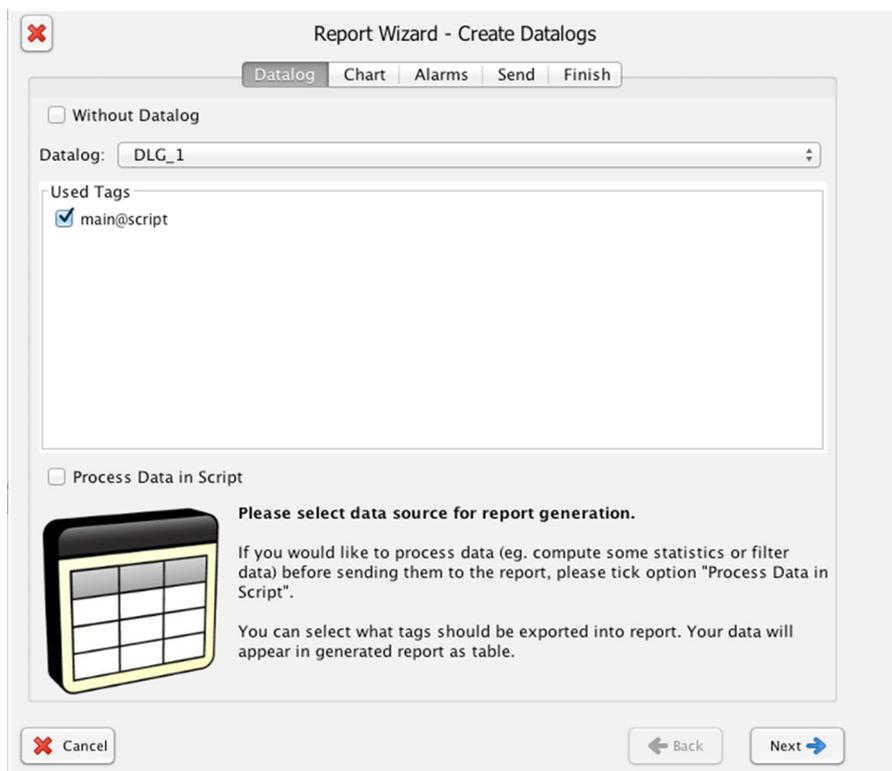
There are two ways of creating reports in server-side scripts:

- Using the *Report Wizard*
- Manual report creation

### Report Wizard

This feature helps you create a report step-by-step in the scripts, which saves you plenty of time spent creating the reports manually.

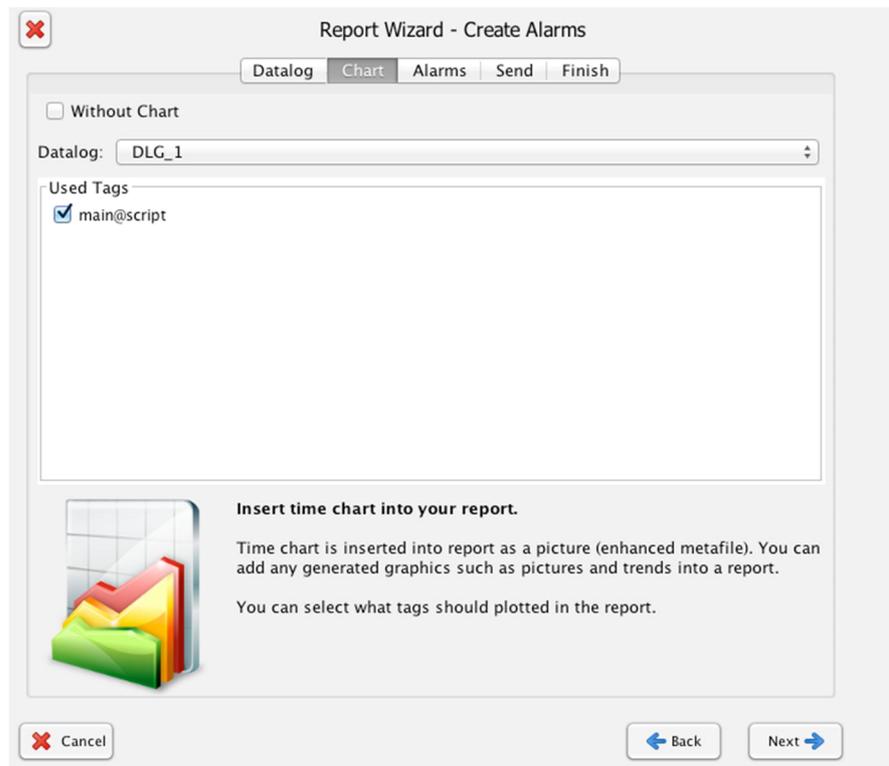
- 1) Select the **Server-Side Scripts** folder in the *Project Window* and click on the **Report Wizard** icon in the toolbar. The following window will appear:



- 2) In the tab **Data-log**, select data-logs and tags to be included in the report table. You can also activate data processing in the script - check the box *Process Data in Script*. If

you wish to create a report without data-logs check the box *Without Data-log*, then click on the **Next** button.

- 3) The following **Chart** tab allows you to include a time chart of data-logs in the report. You can select the data-logs and tags to be displayed. If you do not wish to include the charts in the report, check the box *Without Chart*. Click on the **Next** button.



- 4) In the **Alarms** tab, you can select alarms to be included in the report; then click on the **Next** button.

Report Wizard - Create Alarms

Datalog | Chart | **Alarms** | Send | Finish

Without Alarms

Online

History of alarms

Occurrence (activation count) of alarms in given time period

 **Insert alarm information into your report.**

Select online alarms to insert table of active alarms (in time of report generation).  
 Select history of alarms to insert table of alarm history.  
 Select Occurrence of alarms to get overview of number of activation of alarms in given time period sorted by activation count.

- 5) In the **Send** tab, you can select the way the report from your project will be sent. Fill in the necessary information as shown in the picture below and click on **Next**.

Report Wizard - Create Script

Datalog | Chart | Alarms | **Send** | Finish

FTP

E-mail

**FTP Setting**

host: 192.168.1.1

user: user

password: pass

port: 21

remote dir:

**E-mail Setting**

smtp: smtp.server.com

user: user

password: pass

secure: None

port: 25

from: user@server.com

to: name@server.com

cc:

use device smtp setting

 **Send report over e-mail or FTP.**

Your report is automatically stored into your device. You can retrieve it over web pages or over FTP access. If you select option e-mail, report will be automatically emailed to specified email addresses. Please do not forget to fill in SMTP setting on your device.  
 If you tick option FTP, your report will be sent over the net to the specified FTP server.

- 6) In the last tab, you will see the list of generated report template files. Now, you have to set the triggering actions for your report to be sent. Click on the button **Copy Code to Clipboard** to use a generated report script for later use and click on **Close**.

✖

## Report Wizard – Create Script

Datalog
Chart
Alarms
Send
Finish

To directory 'UserFiles' was created report template: **generatedReportTemplate002.docx**  
 To directory 'Sources' was created script: **generatedReport002**

Generated reports can be run like this:

```

// get current date in seconds
var end = Math.round(+new Date()/1000);
// compute start as now - 5 minute (e.g. 300 seconds)
var start = end-300;
// loading script file into 'report' global variable
var report = require('./generatedReport002.js');
// start report generating
report.generatedReport(start,end);
                    
```



**Your report templates are generated.**  
 Now all you need to do is to run the report on some action:

- for periodic reporting, create timed script and copy the code from this page
- trigger report generation from PLC,
  - create PLC variable table with tag inside
  - create timed script and periodically read your PLC variables table
  - when value changes generate report – copy code from this page

✖ Cancel
📄 Copy Code to Clipboard
Close ➔

### Manual report creating

Report templates are created in *MS Word* (and other compatible text editors) with simple syntax. During report processing, all defined variables inside a Word document are replaced with the script data.



*Example:*

- 1) Prepare a report in MS Word (or any compatible text editor)
  - a) To replace a single variable, put it inside single brackets: {variable}
  - b) To create a loop, you can use loop opening {#} and loop closing {/} brackets.

The image shows a Word document template for 'mySCADA technologies' with the title 'Currently Active Alarms :'. It contains a table with three columns: 'Time Sample', 'Message', and 'Value'. The table has one data row with the following content:
 

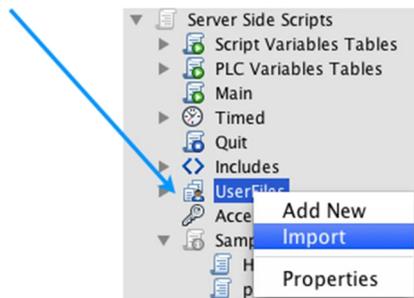
Time Sample	Message	Value
{#alarm} {atm}	{msg}	{value}{/alarm}

 Below the table, there are two lines of text: 'Generated : {aktdate}' and 'By : {name1} {name2}'. Blue arrows point to various parts of the document with annotations:
 

- An arrow points to the '{#alarm}' in the table header, with the text 'Loop over all elements in array variable alarm'.
- An arrow points to the '{/alarm}' in the table value, with the text 'End of loop'.
- An arrow points to the '{msg}' in the table, with the text 'Will be replaced by variable alarm[],msg'.
- An arrow points to the '{name1}' in the 'By' line, with the text 'Will be replaced by value of variable name1'.

 The caption below the image is 'Word Template Example'.

- c) Import the document into *UserFiles* in the *Server-side Scripts* folder:



2) Create a script to fill in the *MS Word* report with data:

*script example which reads active alarms from the system and fills table in word template*

```

1  // A) Create Replacement variable repl and fill it with
2  // some data such as Name and Surname
3  repl = {
4    "name1" : "Petr",
5    "name2" : "Svoboda"
6  };
7
8  //Read active alarms from the system
9  myscada.readActiveAlarmsForReports(0,"D.M.YYYY H:mm:ss",
10 function(status,data) {
11   //add data into replacement variable under the name alarm
12   //use same variable name as in script
13   repl.alarm=data;
14 }
15
16 //create the document engine by require docxgen
17 var mydoc = require('docxgen');
18 //load your template saved in UserFiles
19 mydoc.LoadTemplate('./UserFiles/Template.docx');
20 //perform the replacements with our repl variable
21 mydoc.LoadReplacement(repl);
22 //save newly created report into file
23 mydoc.GenOutput('./UserFiles/DEMO_REPORT.docx');
24
25 myscr.A.value=0;
26 myscada.writeTable("myscr",function(err){});

```

a) create replacement variable

b) fill it with data

c) require "docxgen"

d) open MS Word template file

e) generate report

f) save it to file

*Note: You can use the timed scripts for report generation in predefined intervals or for certain dates.*

## 34.12 mySCADA Specific Functions

Server side scripts are based on node.js. The main advantage is that you can use a standard node.js project in the mySCADA environment. However, there are SCADA-specific functions that you should use if you want to read data from PLC, process historical data, or work with any specific part of a mySCADA project. To do so, you should use a mySCADA module, which is a collection of useful functions for performing multiple tasks:

- read/write live data from PLCs
- read/write data from databases
- retrieve live data from CAS Alarms
- retrieve historical data from
  - CAS Alarms
  - Data-logs
  - User actions

To use a mySCADA module in your project, simply include it as follows:

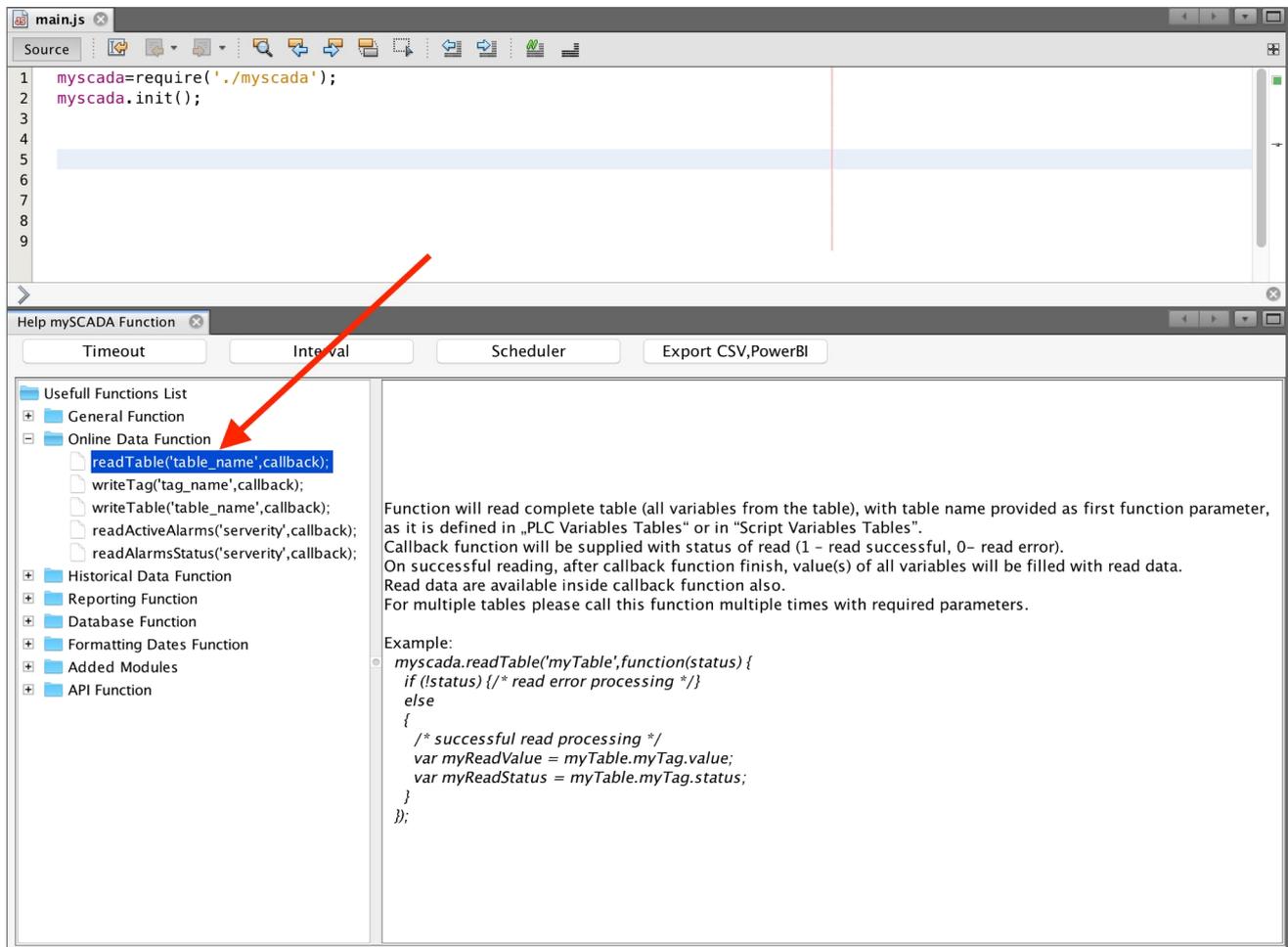
```

myscada=require('./myscada');
myscada.init();

```

NOTE: By default, when you open main.js, mySCADA is already included.

Now you can use any mySCADA specific function. List of all available functions can be found in online help located under the source code-editing window:



## 34.13 Debugging

With this feature, you can manually debug your written scripts. Debugging uses an integrated debugger.

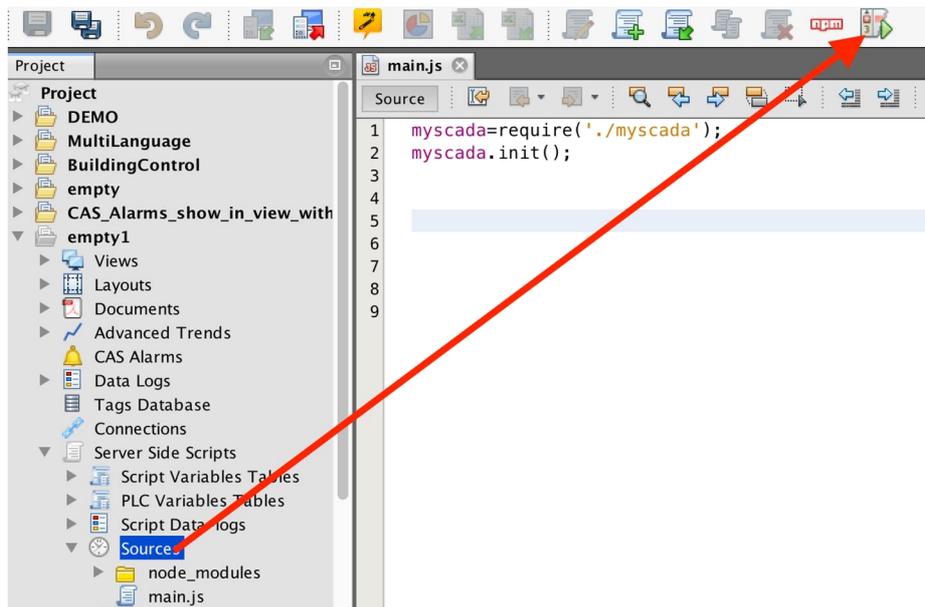
The debugger integrated into *myDESIGNER* is a powerful debugging interface.

### Main functions:

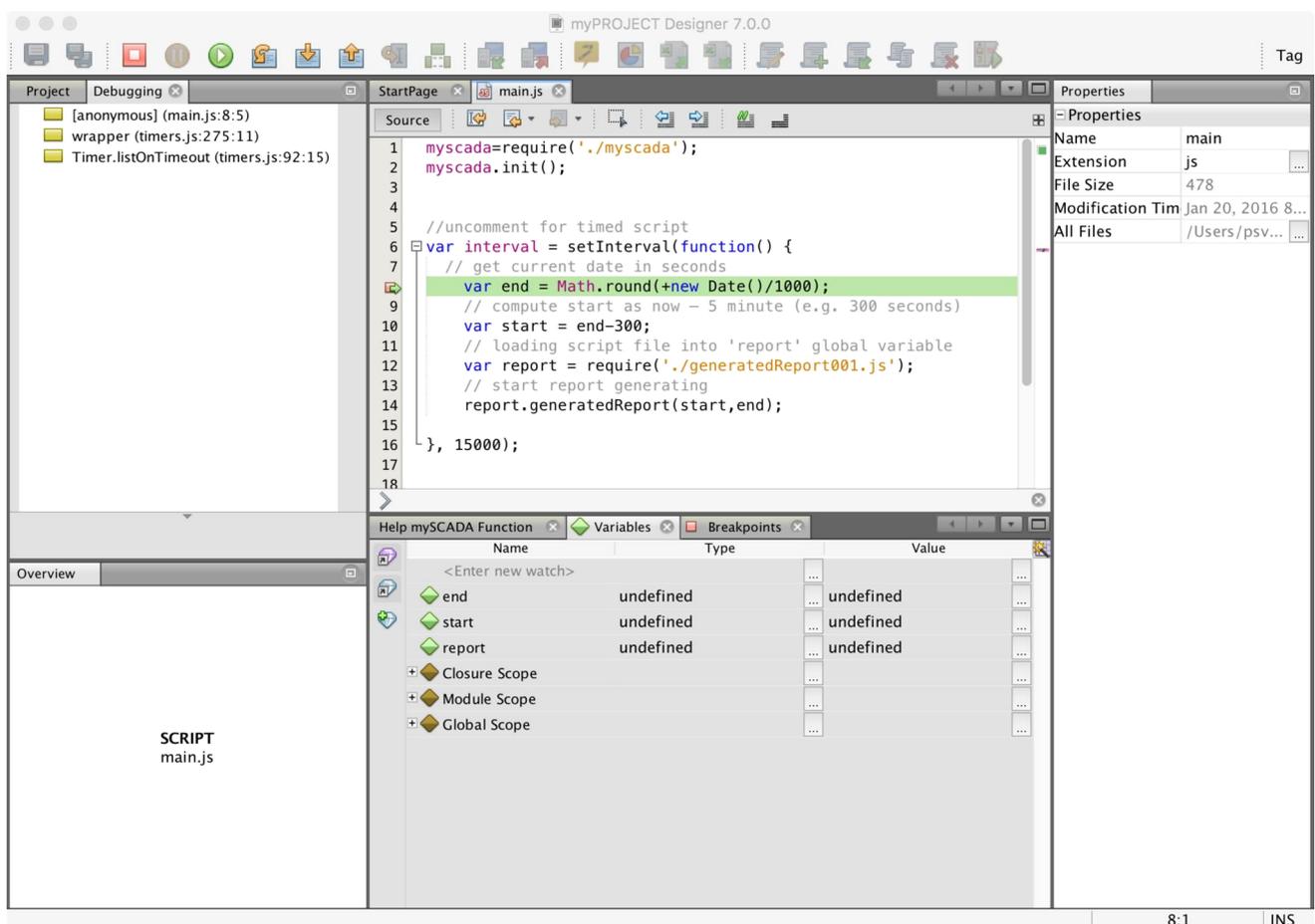
- Navigate to your source files
- Set breakpoints (and specify trigger conditions)
- Step over, step in, step out, resume (continue)
- Inspect scopes, variables, object properties
- Hover your mouse over an expression in your source to display its value in a tooltip
- Edit variables and object properties
- Continue to location
- Break on exceptions
- Disable/enable all breakpoints

**IMPORTANT:** Before you start debugging, make sure you have the same project downloaded to your device as you have in myDESIGNER.

- 1) To start debugging, click on the main.js script and then on the **Debug** icon in the main toolbar.



myDESIGNER will switch to the debug mode:



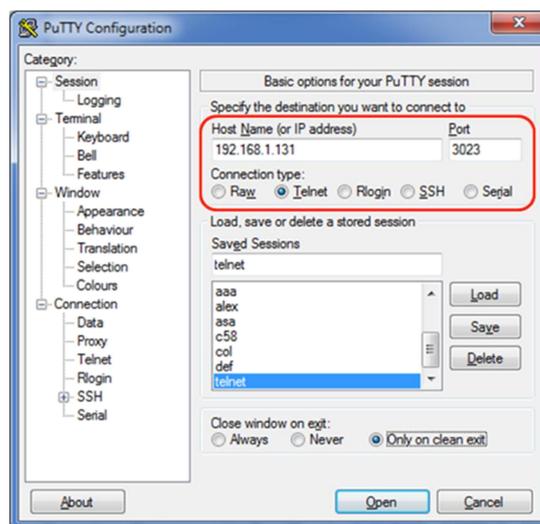
*Note: If your script generates some files, you will find them in the path specified in `UserData dir`.*

2) When you finish debugging, click on the button **STOP Debugging**.

## Console Log

For debugging of server-side scripts on *mySCADA Compact* devices, you can use the *Console log*, accessible via a web interface of *mySCADA Box* in the menu *SYSTEM* and the bookmark *STATUS*.

The console is accessible through any *mySCADA* runtime with a telnet connection on the port **11015**. The windows users can use a freeware utility *PuTTY* (see picture below), and the UNIX users (including MAC) can use a telnet command with a specific port.



Another useful tool for debugging of server-side scripts is to define a function for logging of uncaught exceptions:

```
process.on('uncaughtException', function (err) {
  console.log(err);
});
```

This function will log a description of uncaught errors into the console; you can, of course, define any action inside this function.

## 34.14 Script Status (on myBOX Devices Only)

### Scripts Status and Web Log

In the menu *SYSTEM* and the bookmark *STATUS*, you can find the scripts *status*, which can be useful for scripts troubleshooting - an example output is shown in the picture below:

Scripts status			
<b>Status</b>			
Log	<a href="#">Show log</a>		
Manual restart	<input type="button" value="Restart scripts"/>		
<b>Main script</b>			
Last start	24/09/2012 11:51:34		
Executed in	0ms		
Status	OK		
<b>Timers</b>			
Script	Last start	Executed in	Status
TS1	24/09/2012 11:51:34	1ms	OK
TS2	24/09/2012 11:51:34	60ms	ERROR
<pre>ReferenceError: pfd is not defined at Object.exports.timed.TS2 [as step] (/visudata/public/Scripts/scripts.js:27:2) at step (/home/myscript/child.js:250:11) at process. (/home/myscript/child.js:207:6) at process.EventEmitter.emit (events.js:91:17) at handleMessage (child_process.js:273:12) at Pipe.channel.onread (child_process.js:293:9)</pre>			

In the *Status* column, you can see the web log and restart all server-side scripts. You can use this log from your scripts with the function `myscada.logFile ("your text")`, which will add a time stamp to your text. This function is embedded in `NODE.JS`, so you can call it without requiring it.

In the next part, there are reports on the *Main script* and on *Timed scripts*. There are three parameters in this report:

- **Last start** – the last time when the script started
- **Executed in** – run time of the script
- **Status** – status of the script execution; if execution fails, the error log is displayed (as shown in the previous picture for the script TS2)

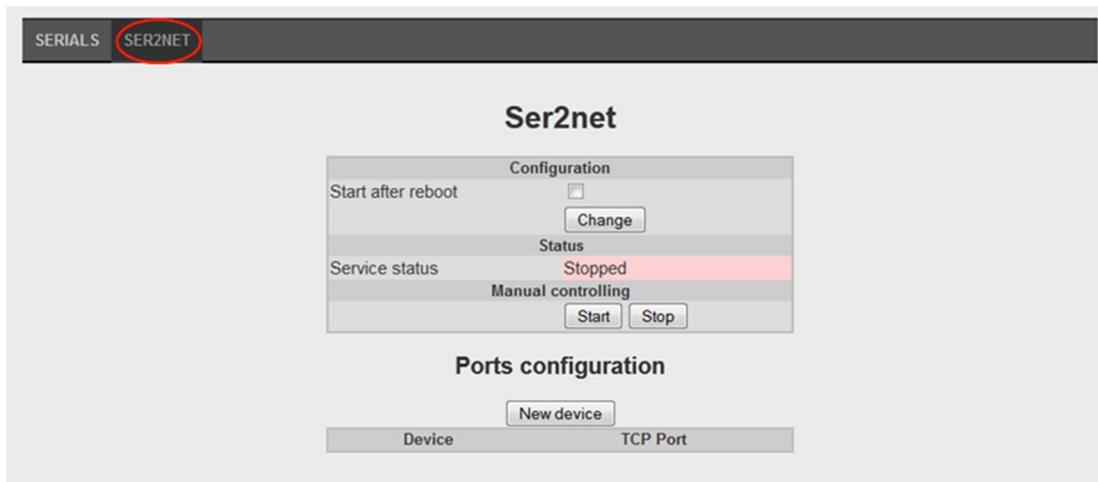
## 34.15 Ser2Net (on myBOX Devices Only)

*Ser2Net* is a very useful service that allows the use of serial ports of *mySCADA Box* in *Server-side Scripts*. At first, it is necessary to setup the parameters of serial ports in the *COM* bookmark of the *mySCADA Box* menu. The number of available serial ports depends on the hardware configuration of *mySCADA Box*. Clicking on the “edit” icons on the right can change the settings for each port.

SERIALS		SER2NET					
Name	Speed	Parity	Stopbit	Databit	XON/XOFF	HWFLOW	
P2-rs485	9600	No	1	8	Off	Off	
P2-rs232-1	9600	No	1	8	Off	Off	
P2-rs232-2	9600	No	1	8	Off	Off	
P1-rs485	9600	No	1	8	Off	Off	
P1-rs232-1	9600	No	1	8	Off	Off	
P1-rs232-2	9600	No	1	8	Off	Off	

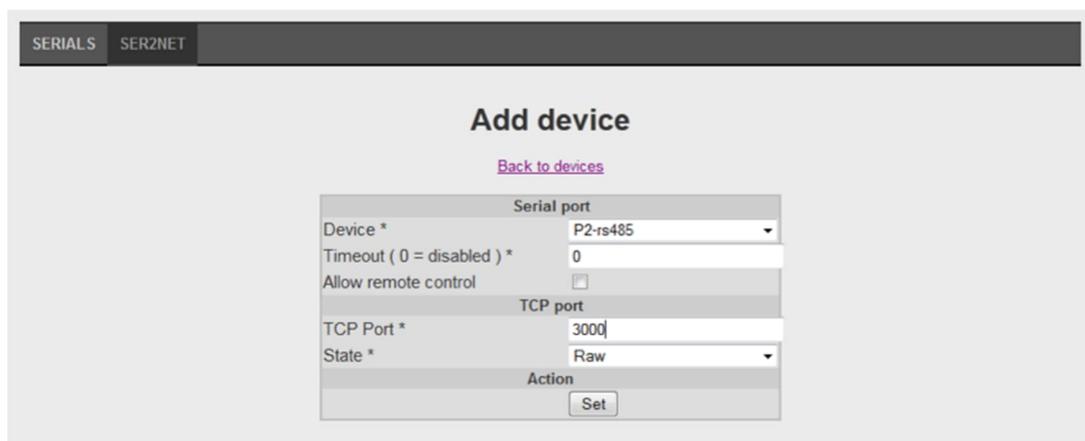
Edit Icons

After configuring the desired serial ports, you can switch to the *SER2NET* bookmark and configure the *Ser2Net* service. The *Ser2Net* configuration window is shown in the picture



below:

By default, the *Ser2Net* service is stopped and the service auto-start is disabled. To enable auto-start of *Ser2Net*, you have to check the *Start after reboot* box and click on the **Change** button to save the settings. The **Start** and the **Stop** buttons can control the *Ser2Net* service manually. To enable access to the serial port via *Ser2Net*, you have to set up a new device by clicking on **New device**. The *Add device* dialog is shown in the following picture.



**Device** – list of available serial ports

**Timeout** – time (in seconds) before the port will be disconnected if there is no activity on it.  
A zero value disables this function

**Allow remote control** – allows remote control of the serial port parameters via RFC 2217

**TCP Port** – number of the TCP/IP port to accept connections

**State** – specifies operation mode of device:

- *Raw* – enables the port and transfers all data, as it is between the TCP and the serial port
- *Telnet* – enables the port and runs the telnet protocol on the port to set up telnet parameters
- *Off* – disables the port from accepting connections

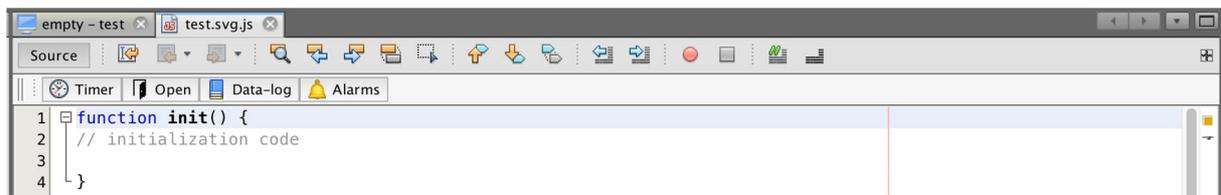
# 35 View and Server Side Scripts - Common tasks

This chapter will show you how to achieve a common tasks using the View or Server Side Scripts. You will learn how to simply create a timer, read historical data from data-logs and alarms, use open command, generate reports and more.

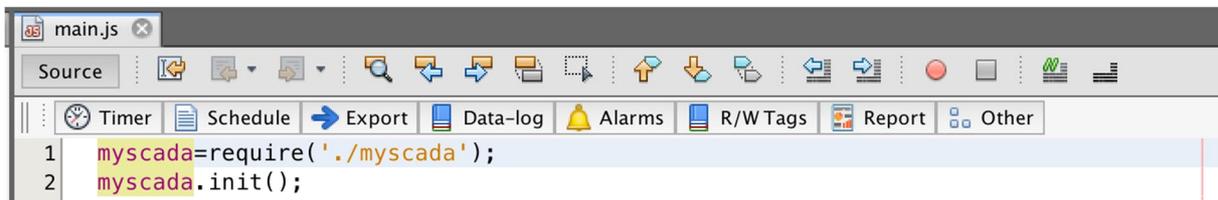
## 35.1 Graphical guides

For the all common tasks we have a prepared a simple to use graphical guides which will generate a code skeleton for you. Graphical guides are located at the top toolbar of the edited script:

### View Script Version



### Server Side Scripts Version



Now we will walk you to using each graphical guide.

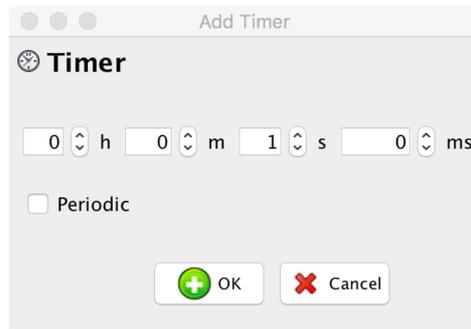
### Creating Timer

To create a timer, click on a Timer Icon in the toolbar.



## View and Server Side Scripts – Common tasks

New dialog will appear:



Select an time period and tick **repeat** if you want to trigger the timer periodically. Once you press OK, myDESIGNER will automatically create a code for you and insert it into the edited script.

```
14 | setTimeout(function() {  
15 |   //your code here!  
16 | }, 1000); //Timeout value in miliseconds
```

### Creating Scheduled Event (Server Side Scripts Only)

To create a scheduled event, you can use the CRON tool. To activate it, please press the Schedule Icon:



## View and Server Side Scripts – Common tasks

New Dialog will appear:

Cron Based Scheduler

Generate cron expression

Minutes Hourly Daily Weekly Monthly Yearly

Monday  Tuesday  Wednesday  Thursday  
 Friday  Saturday  Sunday

Start time 12 : 00

Result

Cron format 0 12 \* \* 1,3,5

Start time Thursday, March 2, 2017 12:00 PM

Order #	Date
1	Friday, March 3, 2017 12:00 PM
2	Monday, March 6, 2017 12:00 PM
3	Wednesday, March 8, 2017 12:00 PM
4	Friday, March 10, 2017 12:00 PM
5	Monday, March 13, 2017 12:00 PM
6	Wednesday, March 15, 2017 12:00 PM
7	Friday, March 17, 2017 12:00 PM
8	Monday, March 20, 2017 12:00 PM
9	Wednesday, March 22, 2017 12:00 PM
10	Friday, March 24, 2017 12:00 PM
11	Monday, March 27, 2017 12:00 PM
12	Wednesday, March 29, 2017 12:00 PM
13	Friday, March 31, 2017 12:00 PM
14	Monday, April 3, 2017 12:00 PM
15	Wednesday, April 5, 2017 12:00 PM
16	Friday, April 7, 2017 12:00 PM

OK Cancel

Now select the dates and times, when you want your code to be run and press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
17 var CronJob = require('cron').CronJob;  
18 new CronJob('0 12 * * 1,3,5', function() {  
19     //your code here  
20 }, null, true);
```

### Export to CSV and Microsoft Power BI

You can make periodic export of data-log data into CSV file. This file is easily readable by Microsoft Power BI. To do so, please click on the button **Export**.



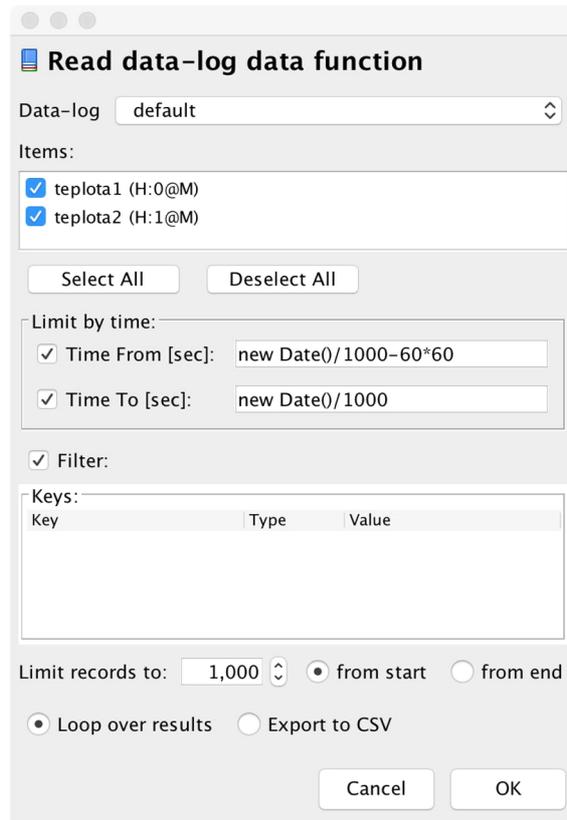
For detail description, please see section **Simple Periodic Export to CSV and Microsoft Power BI** in this manual.

### Read Historical data from data-logs

You can easily read and process historical data from data-logs. To do so, click on the Data-log Icon



New Dialog will be opened:

A dialog box titled "Read data-log data function". It has a "Data-log" dropdown menu set to "default". Below it is an "Items:" list with two entries: "teplota1 (H:0@M)" and "teplota2 (H:1@M)", both with checked checkboxes. There are "Select All" and "Deselect All" buttons. A "Limit by time:" section has two checked checkboxes: "Time From [sec]:" with the value "new Date()/1000-60\*60" and "Time To [sec]:" with the value "new Date()/1000". A "Filter:" checkbox is checked. Below it is a "Keys:" table with columns "Key", "Type", and "Value". At the bottom, there is a "Limit records to:" dropdown set to "1,000", radio buttons for "from start" (selected) and "from end", and radio buttons for "Loop over results" (selected) and "Export to CSV". "Cancel" and "OK" buttons are at the bottom right.

Key	Type	Value
-----	------	-------

In the provided dialog, please start by selecting a data-log from which you want to read data. Then select an items you want to process, or leave all selected. Other options follow:

**Limit by time:** first you can limit the data by provided time. Time is specified in UTC format in seconds since 1.1.1970. You can enter a value or provide a variable where the value is stored.

**Filter:** if the data-log contains a keyed value, you can limit results shown in your project by specifying a filter value. Again this can be a hard value or variable.

**Limit records to:** please provide limits for the number of records loaded. You can also specify if the limit is taken from the beginning eg. **“from start”** or **from end**.

**Loop over results:** if this option is checked, you will have an option to loop over the retrieved records and process them in your script.

## View and Server Side Scripts – Common tasks

**Export to CSV** (Server side scripts only): if you choose this option, the retrieved historical data will be exported into a CSV format and saved into a file. File is saved under a provided name into user data folder accessible over FTP or HTTP and HTTPS.

Loop over results  Export to CSV

Filename:  Separator:

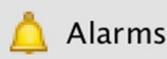
Date Format:  Append:

press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
18 //read historical data from data-log
19 var options=new Object;
20 options.dlgid=1;
21 options.tags=[1,2];
22 options.timeFrom=new Date()/1000-60*60;
23 options.timeTo=new Date()/1000;
24 options.limit=1000;
25 myscada.readDataLogData(options, function(err,result){
26     //process each row of returned data
27     for (i=0;i<result.length;i++)
28     {
29         var date=new Date(result[i].tm[0]*1000+Math.floor(result[i].tm[1]/1000));
30         var value0=result[i][1]; //teplota1 (H:0@M)
31         var value1=result[i][2]; //teplota2 (H:1@M)
32     }
33 });
```

## Reading and processing Alarms

You can read and process online and historical alarms. To do so, please click on the **Alarms** icon:



New dialog will open:

In this dialog, you can choose if to process Online or Historical alarms. We will start with **Historical alarms**:

Historical alarms dialog has several sections. We will explain in details each section.

### Columns:

Columns allows you to select all the data retrievable from the history. Please select what columns you want to process.

### Filters:

Currently, you can filter retrieved data based on time interval. Future versions will allow to extend the filter for Message, Area and Device as well.

### Aggregates:

If you need historical alarms in the form they have been stored, leave aggregates to **none**. If you want to retrieve an aggregated data based on alarm occurrence count and overall activation time, please tick the **occurrence** option.

**Limit records to:** please provide limits for the number of records loaded. You can also specify if the limit is taken from the beginning eg. **“from start”** or **from end**.

**Loop over results:** if this option is checked, you will have an option to loop over the retrieved records and process them in your script.

## View and Server Side Scripts – Common tasks

**Export to CSV** (Server side scripts only): if you choose this option, the retrieved historical data will be exported into a CSV format and saved into a file. File is saved under a provided name into user data folder accessible over FTP or HTTP and HTTPS.

Loop over results    Export to CSV

Filename:    Separator:

Date Format:    Append:    Headers

press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
16 //read historical alarm data
17 var options=new Object;
18 options.timeFrom=new Date()/1000-60*60;
19 options.timeTo=new Date()/1000;
20 options.limit=1000;
21 myscada.readAlarmsHistory(options, function(err,result)
22 {
23     //process each row of returned data
24     for (i=0;i<result.length;i++)
25     {
26         var msg=result[i].cell.msg; //Message
27         var status=result[i].cell.stat; //Alarms status
28         var severity=result[i].cell.sv; //Severity
29         var area=result[i].cell.area; //Area field
30         var acttime=new Date(result[i].cell.atm[0]*1000+Math.floor(result[i].cell.atm[1]/1000)); //Activation time
31     }
32 });
```

**Online Alarms** can be processed as well. To do so, click on the tab Online and dialog will change accordingly:

Alarms Function

**Read alarms function**

History   Online

Select Columns:

- MESSAGE    ACK TIME
- STATUS    ACK TEXT
- SEV    ACT VAL
- AREA    DEACT VAL
- DEVICE    ACK VAL
- ACT TIME
- DEACT TIME

Severity up to:

Loop over result    Export to CSV

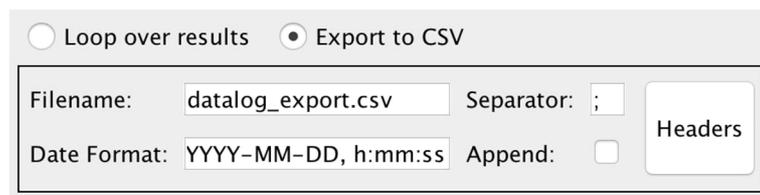
OK   Cancel

## View and Server Side Scripts – Common tasks

This function will retrieve active or non-acknowledge alarms at the time script is run. You can select which **columns** you want to process or **maximum severity** level. Other options are:

**Loop over results:** if this option is checked, you will have an option to loop over the retrieved records and process them in your script.

**Export to CSV (Server side scripts only):** if you choose this option, the retrieved historical data will be exported into a CSV format and saved into a file. File is saved under a provided name into user data folder accessible over FTP or HTTP and HTTPS.



press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
16 //read online alarm data
17 var options=new Object;
18 options.severity=999999;
19 myscada.readAlarmsOnline(options, function(err,result)
20 {
21 //process each row of returned data
22 for (i=0;i<result.length;i++)
23 {
24 var msg=result[i].cell.msg; //Message
25 var status=result[i].cell.stat; //Alarms status
26 var severity=result[i].cell.sv; //Severity
27 var acktime=new Date(result[i].cell.acktm[0]*1000+Math.floor(result[i].cell.acktm[1]/1000));
28 var value=result[i].cell.v; //Current Value
29 var actval=result[i].cell.av; //Activation value
30 }
31 });
```

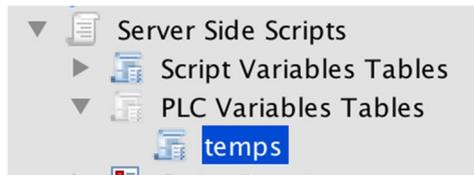
## 35.2 Read/Write data from/to PLC

To read or write data from PLC, you should use the **R/W tags** button.



To read or write tags from PLC, you must create a PLC request table first. To do so, please navigate to the **PLC Variables Tables section** and create a new table. You can call it whatever you like.

## View and Server Side Scripts – Common tasks

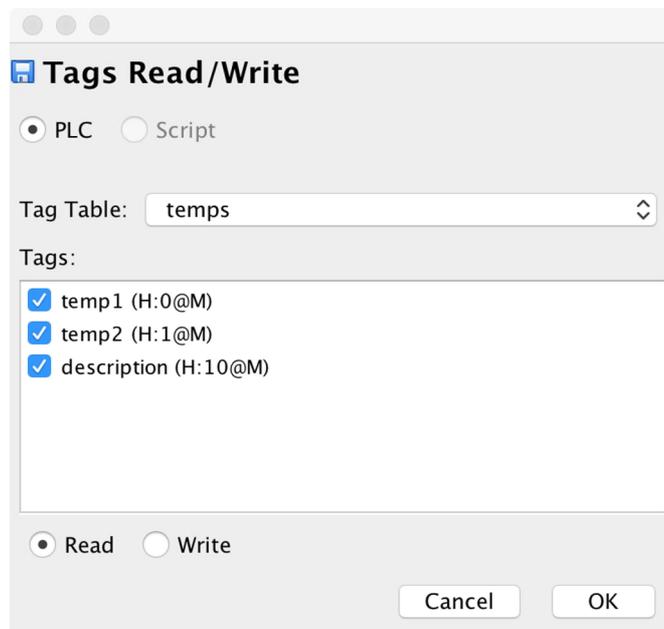


In **PLC Variables Table**, you define what tags will be read or written from the PLC.

Variable Name	Tag@Conn/*Alias	Number of Elem...	Type
temp1	H:0@M	1	Numeric
temp2	H:1@M	1	Numeric
description	H:10@M	10	String
		1	Numeric

In this example, we will be reading two numerical values and one string from the PLC with alias M.

Once you create your request, you can proceed to the script. Click on the **R/W Tags** button and you will be presented with the following dialog:



In the dialog, you can see, we have specified our created PLC variable table called **temps**. You can see all the defined tags in the **Tags** section. If you wish to read/write only some tags, please, uncheck the others. Once you are done, press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

For tags reading:

```
//Read tags for tag group temps
myscada.readTags("temps", function (err,data){
  if (!err){
    var temp1=data[0].value;
    var temp1_err=data[0].err;
    var temp2=data[1].value;
    var temp2_err=data[1].err;
    var description=data[2].value;
    var description_err=data[2].err;
  }
});
```

As you can see, myDESIGNER has prepared a code for you, which will read your tags. For each tag, you will get its value and also a status info if it was read or ended in error (for example if you missed typed the tag address)

For tags write:

```
//Write tags for tag group temps
var options={};
options['name']="temps";
options['values'] = {};
options['values']['temp1']=2.3;
options['values']['temp2']=7.7;
options['values']['description']="temperatures";
myscada.writeTags(options, function (err,data){
  if (err){
    //write error
  }
});
```

As you can see, first we create object options and we put all the values we want to write into the **options** -> **values** array. Then we call the function **myscada.writeTags**. On successful write you will get the callback to your function.

### 35.3 Generating Report

To generate a report, first create it in your project in section Reports. Then you can generate it in server side scripts. To do so, please click on the button **Report**.



For detailed explanation on how to generate report from server side scripts, please look at the **section Creating Report on Demand** in Reports chapter in this manual.

### 35.4 Other Guides

In other section you can find less common script guides. To invoke the Other guides dialog, please click on the **Other** button.

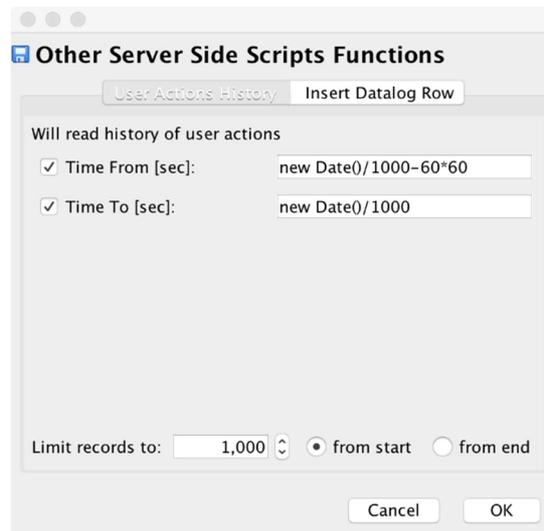


## View and Server Side Scripts – Common tasks

In Other guides dialog you are able to:

### Read history of user actions

To read history of user actions, please select the tab **User Actions History**.



**Limit by time:** first you can limit the data by provided time. Time is specified in UTC format in seconds since 1.1.1970. You can enter a value or provide a variable where the value is stored.

**Limit records to:** please provide limits for the number of records loaded. You can also specify if the limit is taken from the beginning eg. “**from start**” or **from end**.

Once you are done, press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

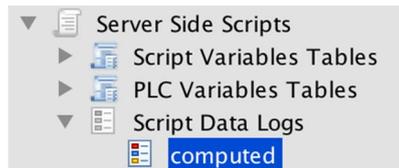
```
//Read user actions from history
var options={};
options['timeFrom']=new Date()/1000-60*60;
options['timeTo']=new Date()/1000;
options['limit']=1000;
myscada.readUserActions(options, function (err,result){
  if (!err){
    //process each row of returned data
    for (i=0;i<result.length;i++)
    {
      var date=result[i].datetime;
      var text=result[i].text;
      var user=result[i].user;
      var sev=result[i].sv;
    }
  }
});
```

## View and Server Side Scripts – Common tasks

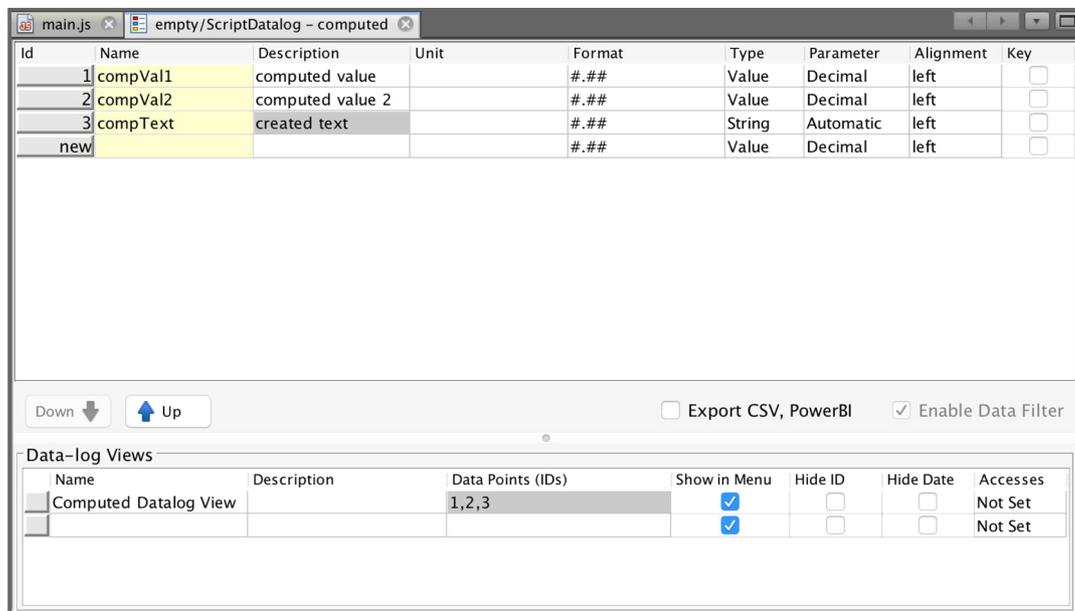
### Insert rows into the data-log

With this function, you can easily fill in data-log with data computed in server side scripts. This can be very convenient in cases when you want to log computed data instead of raw data from the PLC. We will show how to create a custom data-log and fill it with data in server side scripts.

3. Firstly, we need to create a custom data-log definition. This is an data-log which will hold our data. To do so, please navigate to Script Data Logs in server side scripts.



Here create a new data-log and give it a name.



As you can see, we have created data-log with 3 values, two numeric and one string. As you can see, Script data-log has the same definition as regular data-log. Only difference is, you don't link script data log to any data source, but you fill in its values from the server side scripts. Also, you can create a data-log views for this data-log as you would do with regular data-log. Once you are done with definition, we will proceed to data filling.

4. Filling the data-log with data. To fill our created script data-log with data, please click on the Other button and select an Insert Datalog Row tab.

## 35.5 Server Side Scripts – Examples

This section is intended to help you with basic task you may encounter when using Server Side Scripts. If you are new to node.js, we **strongly** recommend to read some general tutorial about node.js and asynchronous programming. Good tutorials can be found here:

## View and Server Side Scripts – Common tasks

<https://code.tutsplus.com/tutorials/nodejs-for-beginners--net-26314>

<http://nodeguide.com/beginner.html>

When reading the tutorials, you can skip the node.js installation and environment set up, it is already done by myDESIGNER for you. Also to install new packages, you can use embedded **npm** in myDESIGNER. It is located on the main toolbar.

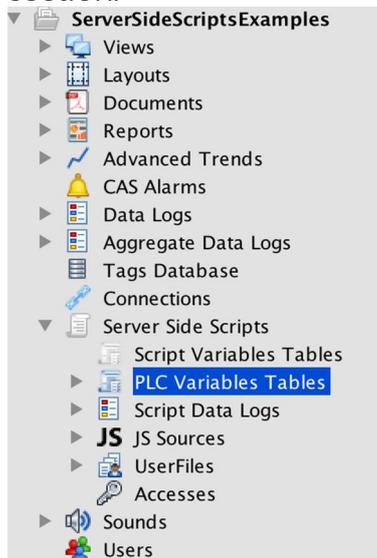
We wish you happy coding!

Did we forget to mention myDESIGNER has embedded debugger for your scripts? Just press “debug” to debug your code instantly.

### 35.6 Reading data from PLC

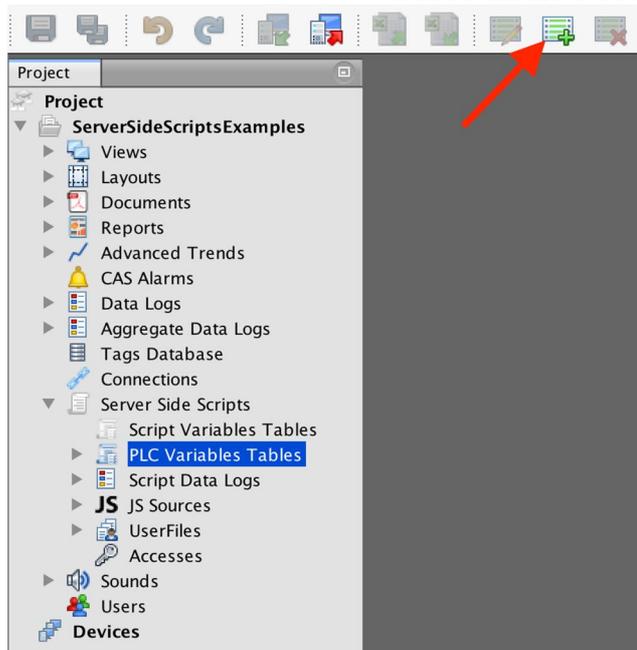
To read data from your PLC and save it into the local variables, you will use the R/W Tags function. In this example we will read 2 values from the PLC, one integer value, and a string value.

1. Firstly, navigate to the **Server Side Scripts** section and open the **PLC Variables Tables** section.

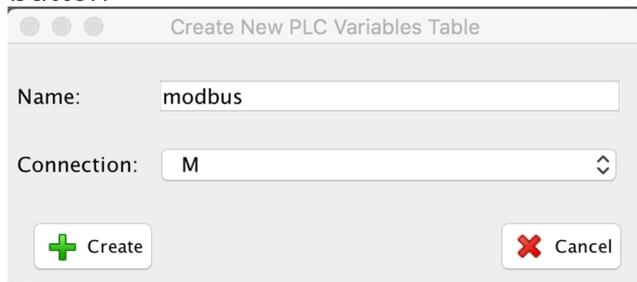


2. Now click on the “New” button in the main Toolbar

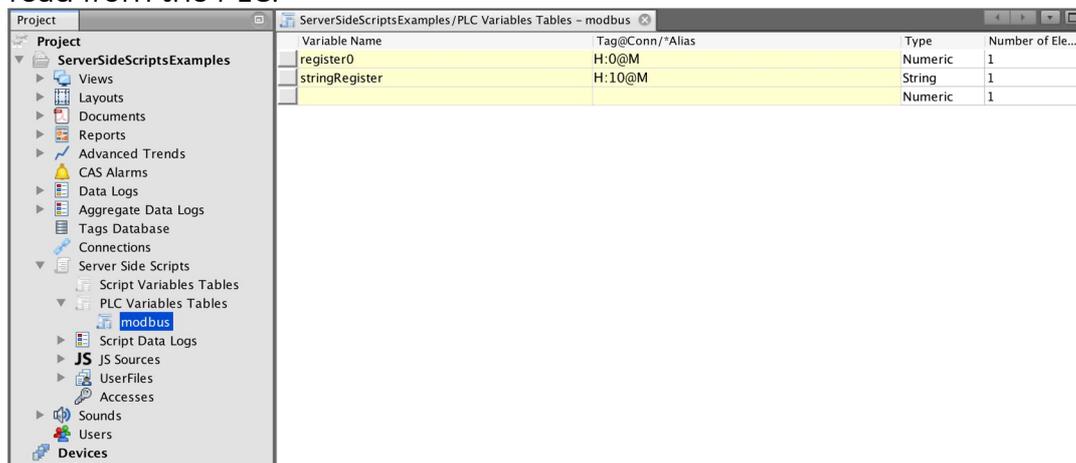
## View and Server Side Scripts – Common tasks



3. In the dialog, provide a name for the table, select connection and hit the “Create” button



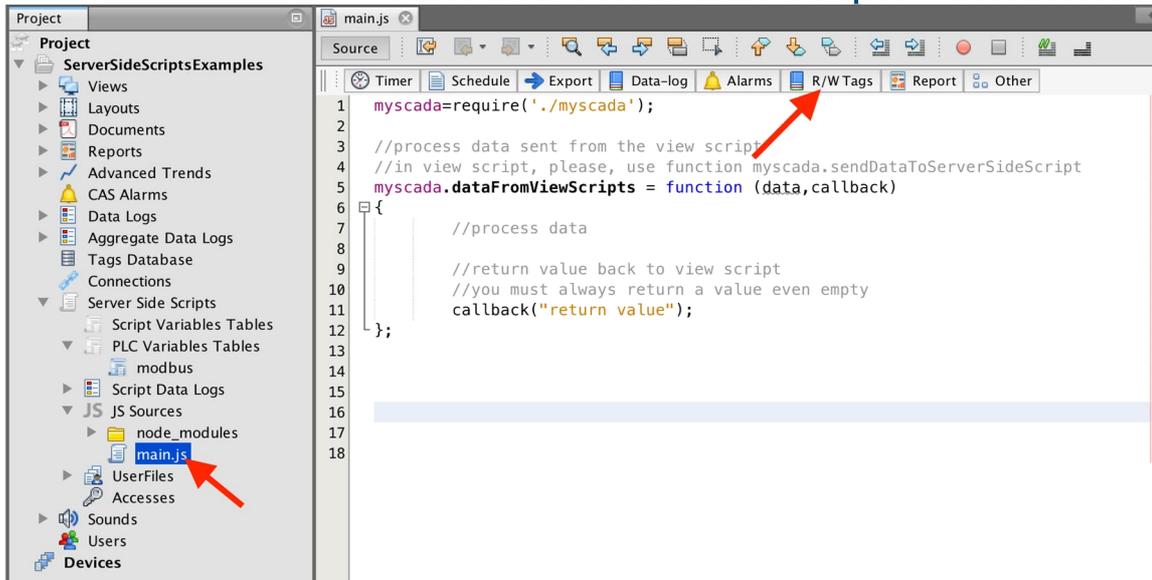
4. You will be presented with the table where you should fill all the tags you wish to read from the PLC.



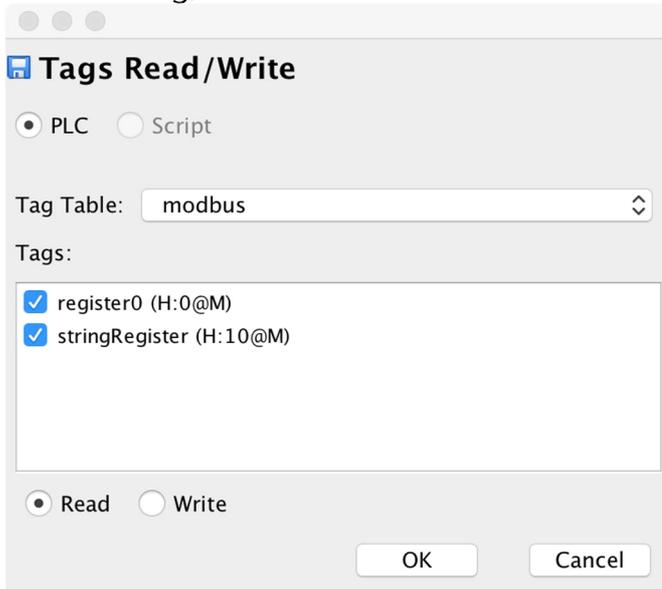
*As you can see, we have added 2 tags, one is a Numeric type and other is a String.*

5. When you are done, open the main.js script from JS Sources.
6. While in the script, click on the R/W tags button which is located in the scripts window toolbar

## View and Server Side Scripts – Common tasks



7. In the dialog, select the name of the table we have created in step 3



8. When you press the OK button, myDESIGNER will generate the code for you.

```
14 //Read tags for tag group modbus
15 myscada.readTagsSymbolic("modbus", function (err,data){
16     if (!err){
17         var register0=data['register0'].value;
18         var register0_err=data['register0'].err;
19         var stringRegister=data['stringRegister'].value;
20         var stringRegister_err=data['stringRegister'].err;
21     }
22 });
```

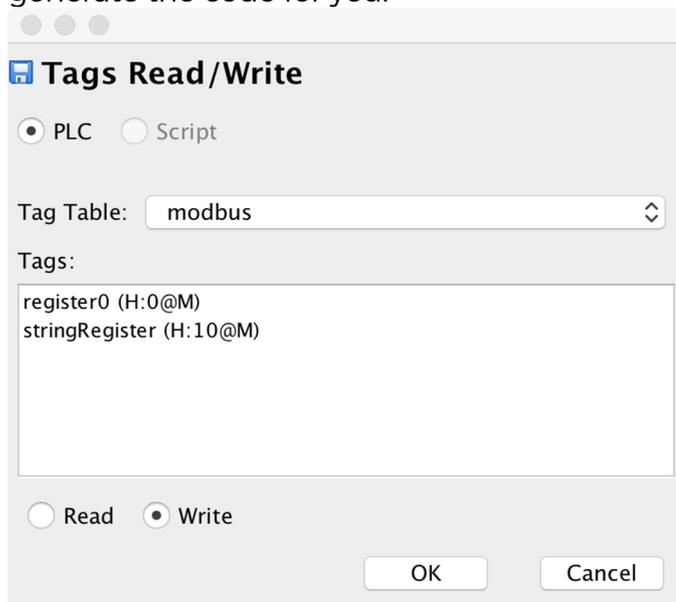
9. Once this code runs, you will receive values from the PLC and they will be stored in the local variables

As you can see from the code, you will receive the values from the PLC as well as the info if the mySCADA was able to read from the tag.

## 35.7 Writing data into PLC

To write values into the PLC, you will follow the exact same procedure as for reading tags, but, in the end, you will use a different function to write values. In this example we will write a numeric value and string into the PLC.

1. We will prepare PLC Variables Table named “modbus”, to do so, please follow steps 1. to 6. In the previous section.
2. Once you have the R/W Dialog loaded, please tick the Write on the bottom part of the dialog. When you are done, press the “OK” button, and myDESIGNER will generate the code for you.



3. When this code runs, values 10 and “String” will be written to the PLC.

```

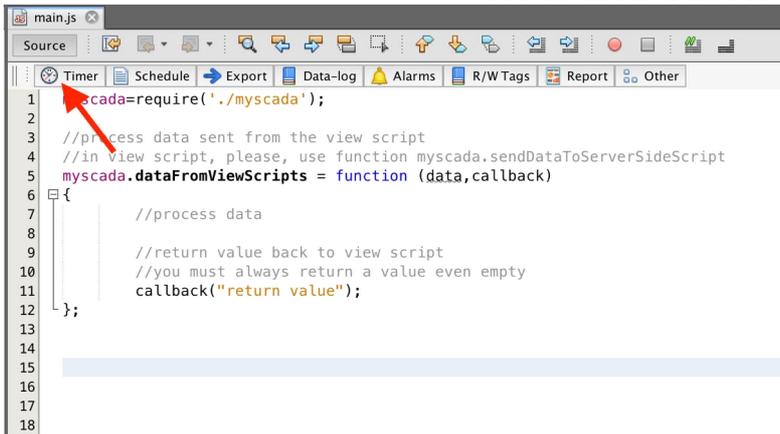
14 //Write tags for tag group modbus
15 var options={};
16 options['name']="modbus";
17 options['values'] = {};
18 options['values']['register0']=10;
19 options['values']['stringRegister']="String";
20 myscada.writeTags(options, function (err,data){
21     if (err){
22         //write error
23     }
24 });

```

## 35.8 Timers – eg. Run code in given time intervals

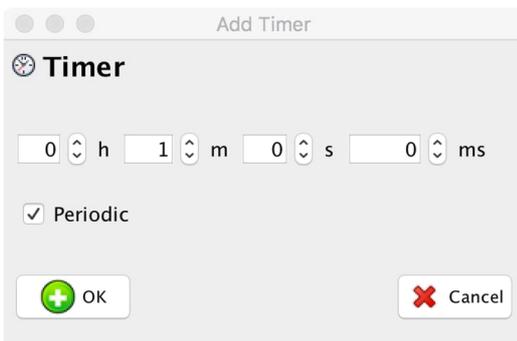
Server side scripts have a simple mechanism how to run your code in a given interval. If you want to run your code after some delay, use the **setTimeout** function. If you want to run your code repeatedly, use **setInterval**. You can use the dialog to add new timer. It is located in the script window toolbar:

## View and Server Side Scripts – Common tasks



```
1 myscada=require('./myscada');
2
3 //process data sent from the view script
4 //in view script, please, use function myscada.sendDataToServerSideScript
5 myscada.dataFromViewScripts = function (data,callback)
6 {
7     //process data
8
9     //return value back to view script
10    //you must always return a value even empty
11    callback("return value");
12 };
13
14
15
16
17
18
```

You will be presented with the Timer dialog:



0 h 1 m 0 s 0 ms

Periodic

OK Cancel

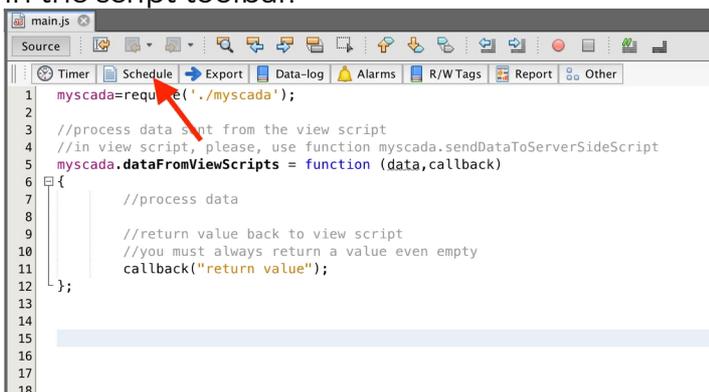
When you click on “OK”, you will have a periodic timer created.

```
17 setInterval(function() {
18     //your code here!
19 }, 60000); //Interval value in miliseconds
20
```

This code will be run periodically every minute.

## 35.9 Scheduled Execution e.g run code every Monday at 2:00 PM

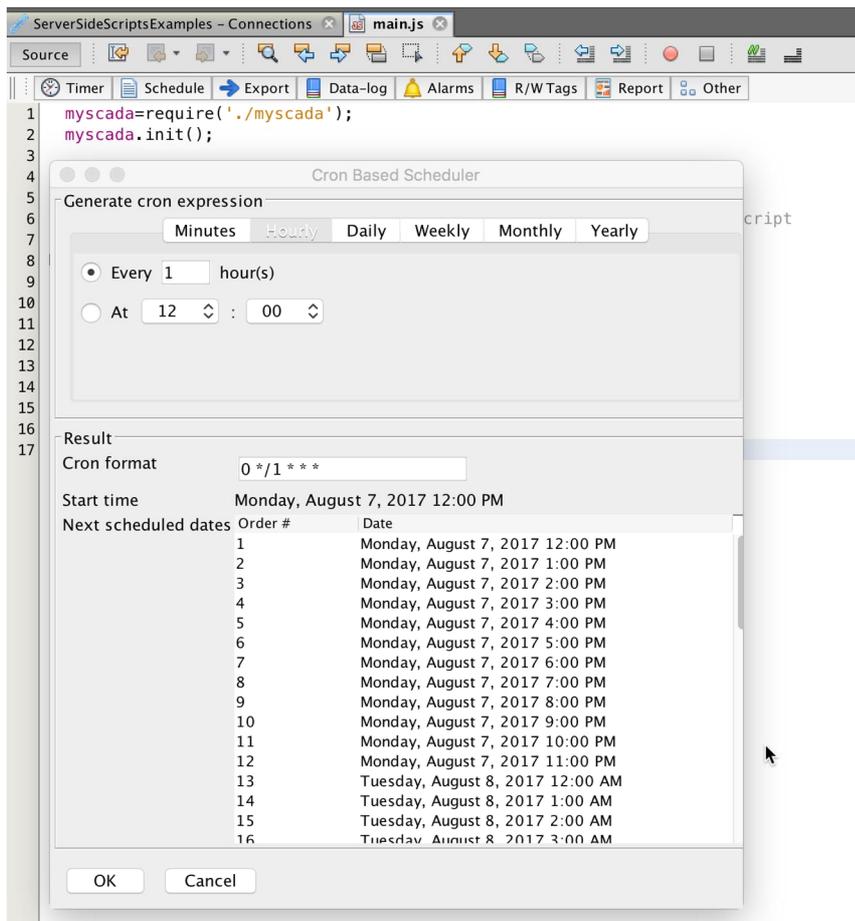
To execute a function in scheduled time intervals, please use a Cron function. It is located in the script toolbar:



```
1 myscada=require('./myscada');
2
3 //process data sent from the view script
4 //in view script, please, use function myscada.sendDataToServerSideScript
5 myscada.dataFromViewScripts = function (data,callback)
6 {
7     //process data
8
9     //return value back to view script
10    //you must always return a value even empty
11    callback("return value");
12 };
13
14
15
16
17
18
```

## View and Server Side Scripts – Common tasks

For example, to execute a script every hour, select hourly:



There is a list of times showing you when the script will run. Now press the “OK” button. A predefined function will be added to your script.

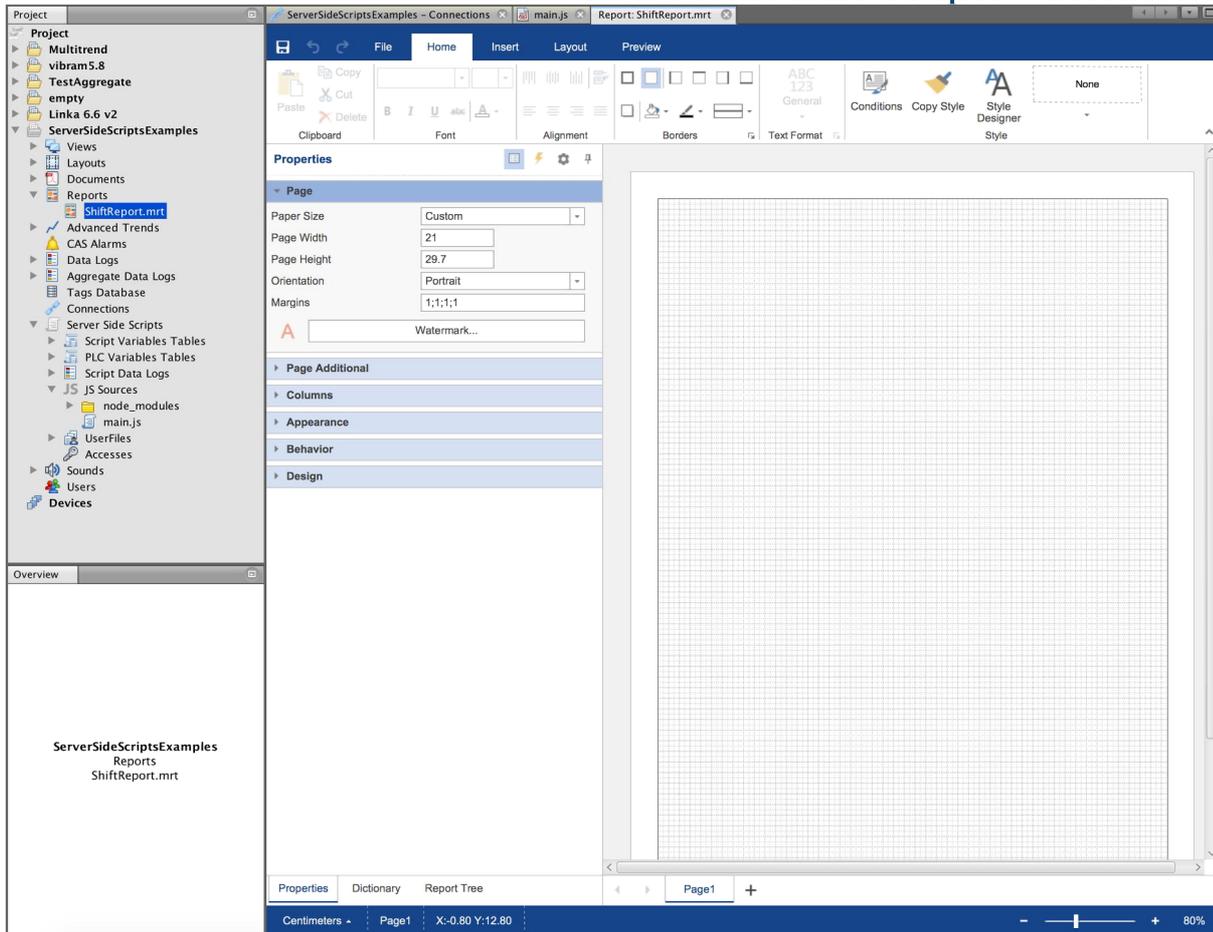
```
17 var CronJob = require('cron').CronJob;  
18 new CronJob('0 */1 * * *', function() {  
19     //your code here  
20 }, null, true);
```

### 35.10 Generating a report at given time interval

To generate a report at given time interval, use the Report function located at the script toolbar window.

Step 1: Create your report. To do so, navigate to the Reports section of your project and select “new.” You will be presented with the Reports GUI designer:

## View and Server Side Scripts – Common tasks



Step 2: Now design your report and press save.

Step 3: Use server side script to run the report in defined time intervals:

- a Create a Cron time based schedule as described in previous example:

```
17 var CronJob = require('cron').CronJob;
18 new CronJob('0 */1 * * *', function() {
19     //your code here
20 }, null, true);
```

- b Now into this function, enter the report generation. You can use the Report dialog to generate a function for you:

**Create Report function**

Report: ShiftReport.mrt

Limit by time:

Time From [sec]: new Date()/1000-60\*60

Time To [sec]: new Date()/1000

Filter:

Limits:  from start  from end

Datalog	Limit
alarms	100

Save as: ShiftReport.pdf  PDF  HTML  JSON

Language: Default

OK Cancel

After pressing “OK” button, you will get the final code:

```

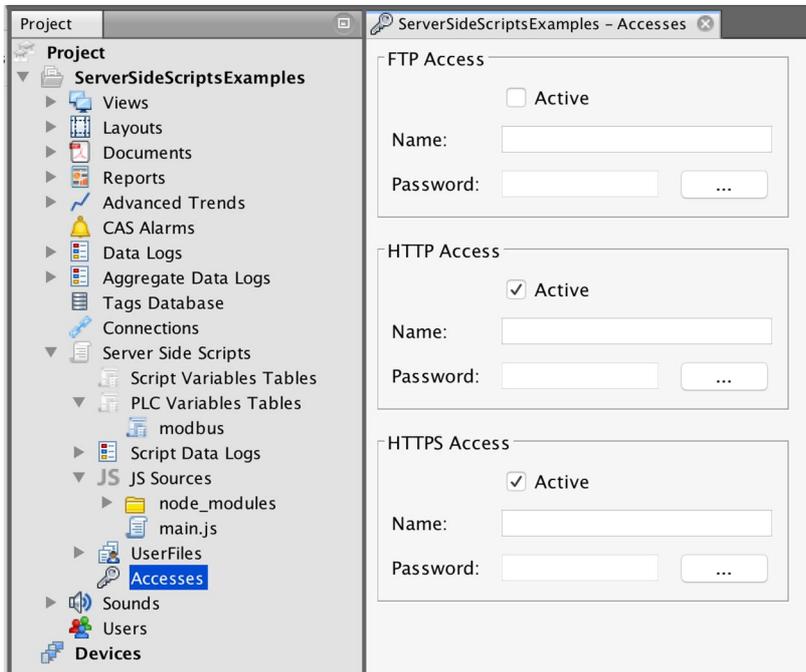
17 var CronJob = require('cron').CronJob;
18 new CronJob('0 */1 * * *', function() {
19
20 //generate report
21 var options=new Object;
22 options.report="ShiftReport.mrt";
23 options.saveAs="ShiftReport.pdf";
24 options.timeFrom=new Date()/1000-60*60;
25 options.timeTo=new Date()/1000;
26 options.exportAs="pdf";
27 options.limitEnd=false;
28 options.limits=[];
29 options.limitsAlarms=100; //limit no of records for alarm history
30 myscada.generateReport(options, function(err,report){
31
32
33 });
34
35 }, null, true);

```

Now your report will be generated every hour. You will find the generated file in the user directory accessible over www pages or over FTP.

## 35.11 Limiting Access to Generated Files

The files you generate either by using the Report Generation feature or by writing into files are located in UserFiles section. You can limit access to this section in **Accesses**.

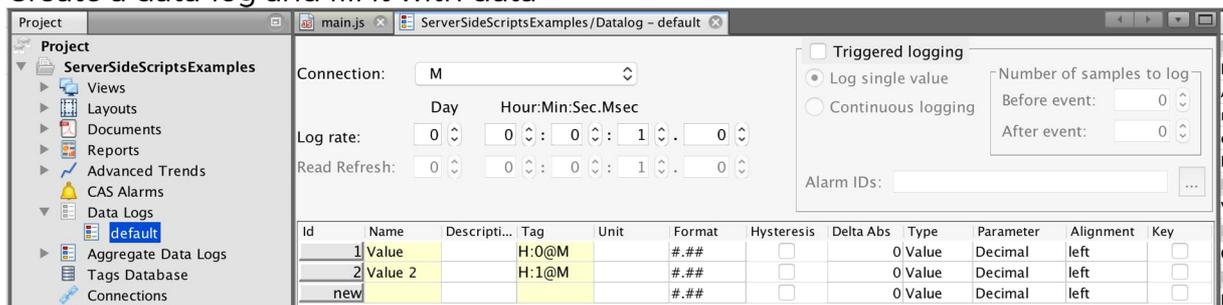


Please, don't forget to enable FTP or HTTP access to access your files!

## 35.12 Processing data-log data

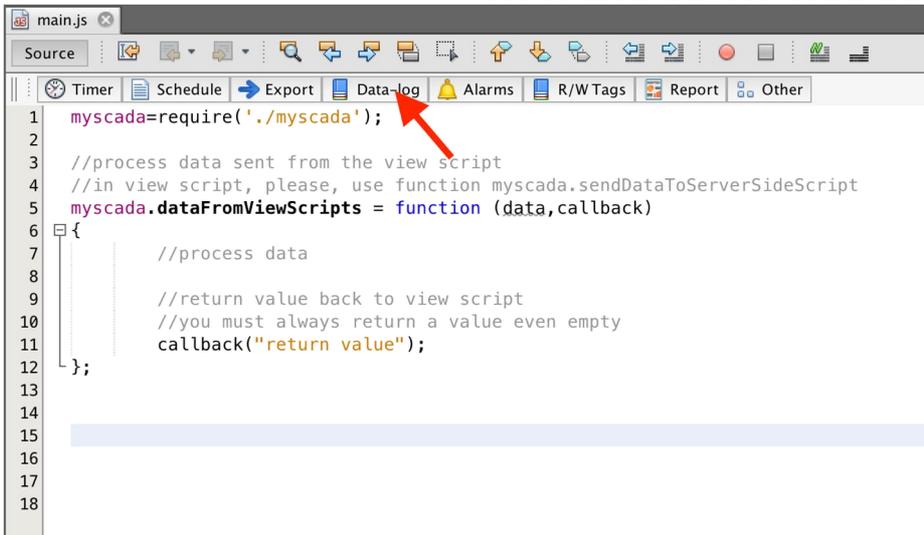
Sometimes you need to process historical data - for example, to compute the average value for the last hour. We will show you how in this simple example.

1. Create a data-log and fill it with data



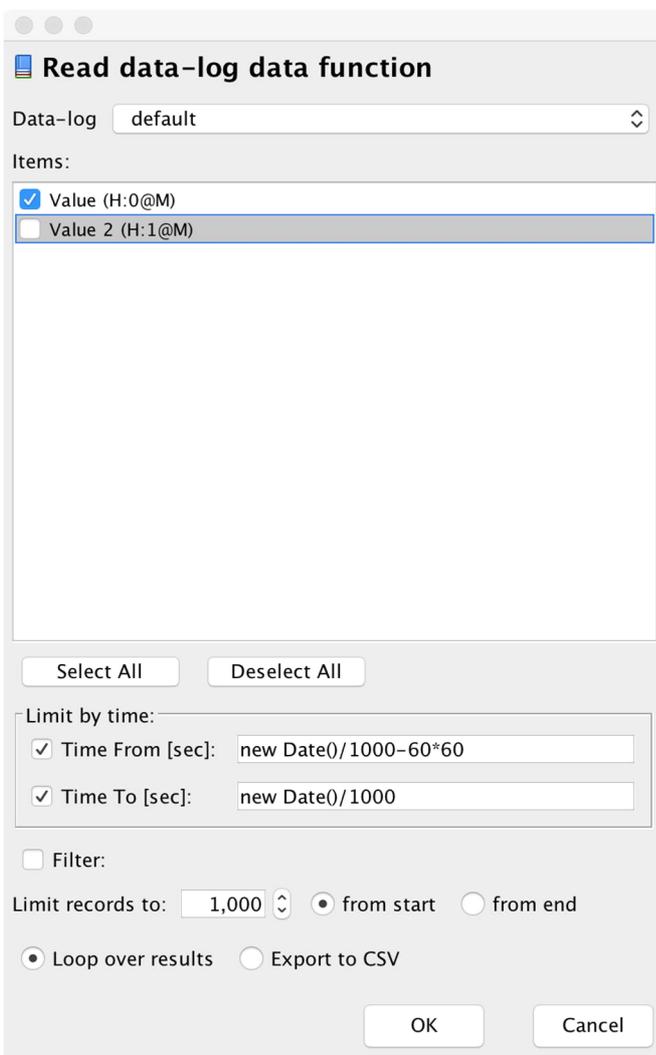
2. Use the data-log function, it is located in the toolbar of the script window

## View and Server Side Scripts – Common tasks



```
1 myscada=require('./myscada');
2
3 //process data sent from the view script
4 //in view script, please, use function myscada.sendDataToServerSideScript
5 myscada.dataFromViewScripts = function (data,callback)
6 {
7     //process data
8
9     //return value back to view script
10    //you must always return a value even empty
11    callback("return value");
12 };
13
14
15
16
17
18
```

3. You will be presented with the Read data-log data function



**Read data-log data function**

Data-log: default

Items:

- Value (H:0@M)
- Value 2 (H:1@M)

Select All    Deselect All

Limit by time:

- Time From [sec]: new Date()/1000-60\*60
- Time To [sec]: new Date()/1000

Filter:

Limit records to: 1,000     from start     from end

Loop over results     Export to CSV

OK    Cancel

Select just the first value as we need to calculate the average from one value only.

After pressing OK, myDESIGNER will generate the code for you including the loop.

## View and Server Side Scripts – Common tasks

```
15 //read historical data from data-log
16 options={};
17 options['dlgid']=1;
18 options['tags']=[1];
19 options['timeFrom']=new Date()/1000-60*60;
20 options['timeTo']=new Date()/1000;
21 options['limit']=1000;
22 myscada.readDataLogData(options, function(err,result){
23 //process each row of returned data
24   for (i=0;i<result.length;i++)
25   {
26     var date=result[i].datetime;
27     var value0=result[i][1]; //Value (H:0@M)
28   }
29 });
```

We can modify the code to compute the average:

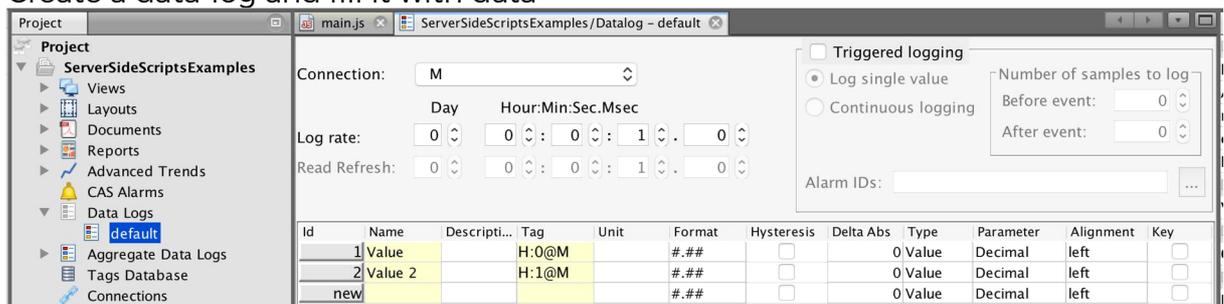
```
15 //read historical data from data-log
16 options={};
17 options['dlgid']=1;
18 options['tags']=[1];
19 options['timeFrom']=new Date()/1000-60*60;
20 options['timeTo']=new Date()/1000;
21 options['limit']=1000;
22 myscada.readDataLogData(options, function(err,result){
23 //process each row of returned data
24   var sum=0;
25   for (i=0;i<result.length;i++)
26   {
27     var value0=result[i][1]; //Value (H:0@M)
28     var sum=sum+value0; //first we will add up all values
29   }
30   var average=sum/result.length; //and now from sum we will compute average
31 });
```

Now you have the computed average from last hour, you can do with it whatever you need.

### 35.13 Exporting data-log data into CSV files

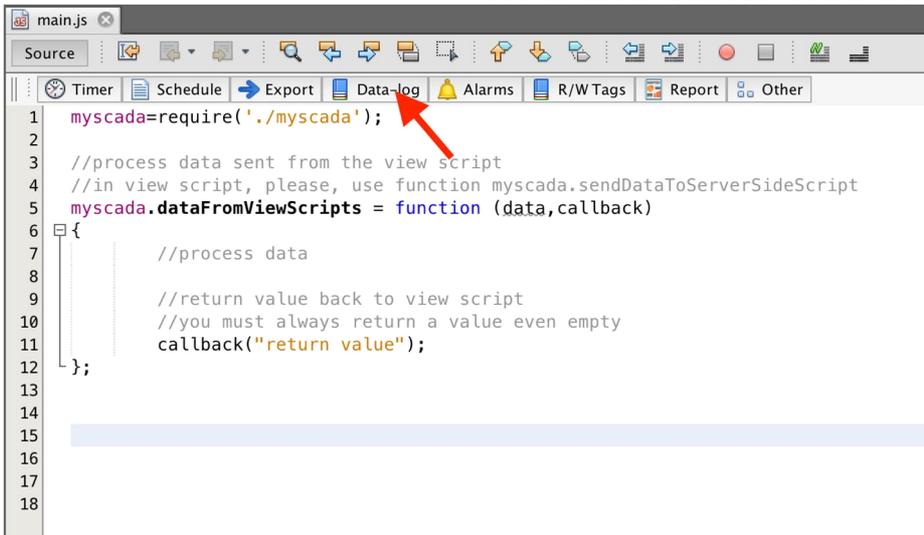
Very often, you need to export data from data-logs into the CSV files. Doing it in mySCADA is very easy and straightforward. All you have to do is:

#### 4. Create a data-log and fill it with data



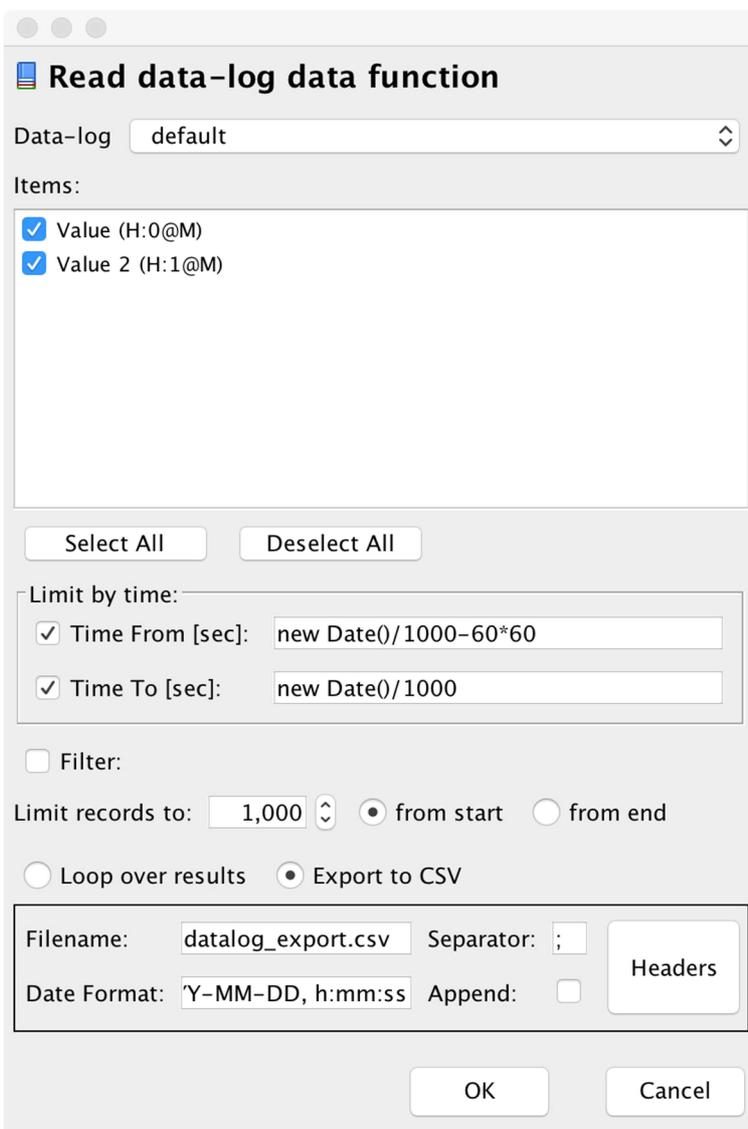
#### 5. Use the data-log function, it is located in the toolbar of script window

## View and Server Side Scripts – Common tasks



```
1 myscada=require('./myscada');
2
3 //process data sent from the view script
4 //in view script, please, use function myscada.sendDataToServerSideScript
5 myscada.dataFromViewScripts = function (data,callback)
6 {
7     //process data
8
9     //return value back to view script
10    //you must always return a value even empty
11    callback("return value");
12 };
13
14
15
16
17
18
```

6. You will be presented with the Read data-log data function



**Read data-log data function**

Data-log: default

Items:

- Value (H:0@M)
- Value 2 (H:1@M)

Select All    Deselect All

Limit by time:

- Time From [sec]: new Date()/1000-60\*60
- Time To [sec]: new Date()/1000

Filter:

Limit records to: 1,000     from start     from end

Loop over results     Export to CSV

Filename: datalog\_export.csv    Separator: ;    Headers

Date Format: Y-MM-DD, h:mm:ss    Append:

OK    Cancel

7. Tick the “Export to CSV” option and Press “OK” to generate the code:

## View and Server Side Scripts – Common tasks

```
19 //read historical data from data-log
20 options={};
21 options['dlgid']=1;
22 options['tags']=[1,2];
23 options['timeFrom']=new Date()/1000-60*60;
24 options['timeTo']=new Date()/1000;
25 options['limit']=1000;
26 options['export']={};
27 options['export']['type']='CSV';
28 options['export']['file']='datalog_export.csv';
29 options['export']['separator']=';';
30 options['export']['dateformat']='YYYY-MM-DD, h:mm:ss';
31 options['export']['headers']=["date","Value","Value 2"];
32 myscada.readDataLogData(options, function(err,result){
33
34 });
```

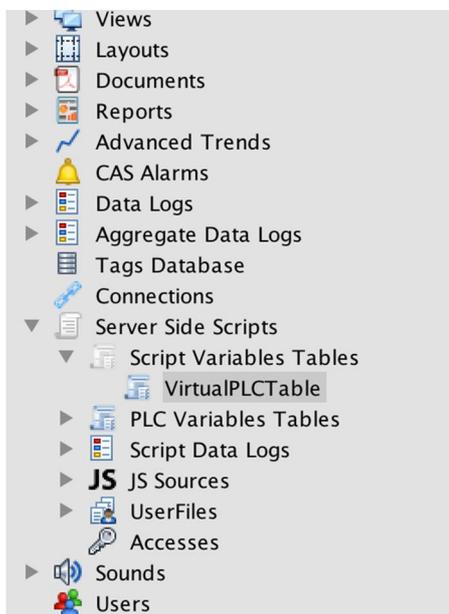
When this code is run, mySCADA will read values for the previous one hour (specified by timeFrom) and save them into the file called datalog\_export.csv

If the file exists, it is overwritten. If you want to append data to this file instead, please use option “append.” This way when you run the script, new data will always append to the end of the file.

### 35.14 Using Virtual PLC

You can create virtual tags in virtual PLC to show and use in your project, eg. Views, trends, alarms etc. To do so, first define the PLC table in Server Side Scripts section.

1. We have created the “VirtualPLCTable”



2. And now we will create some tag inside. You can also create multiple tags.

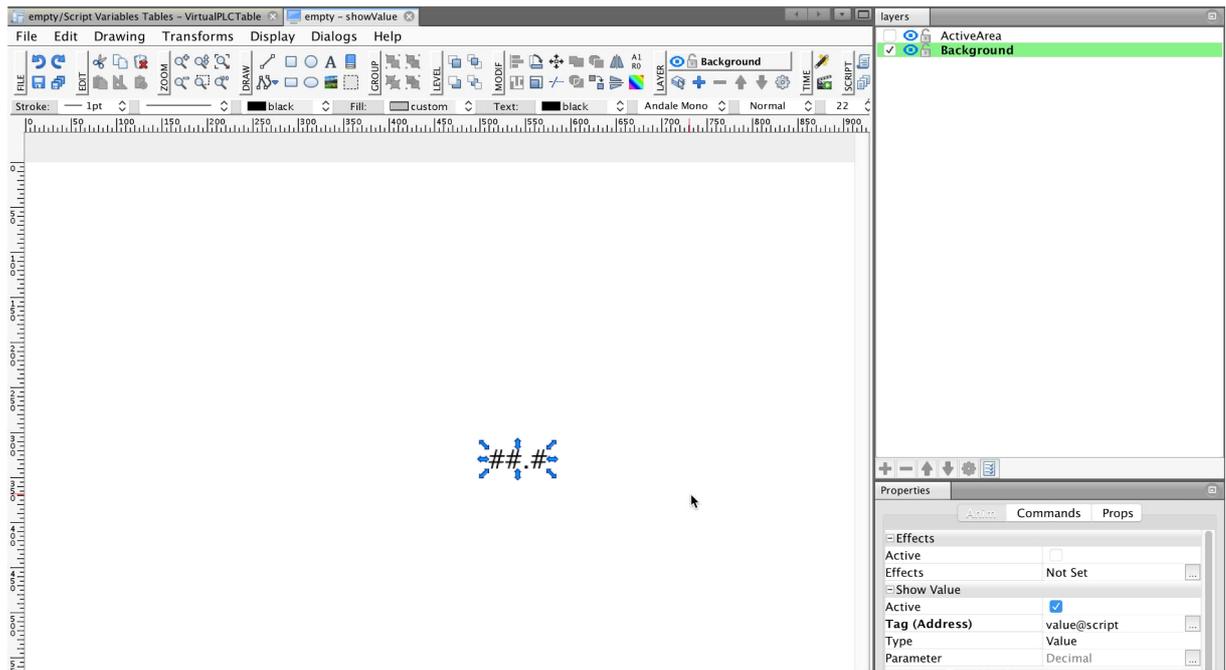


Variable Name	Tag@Conn/*Alias	Type	Number of Elem...	Initialization Value
value	value@script	Numeric	1	10
		Numeric	1	0

Tag has name **value@script** and it has initialization value of 10

## View and Server Side Scripts – Common tasks

- Now you can use the tag anywhere in your project. So let's put it into the view:

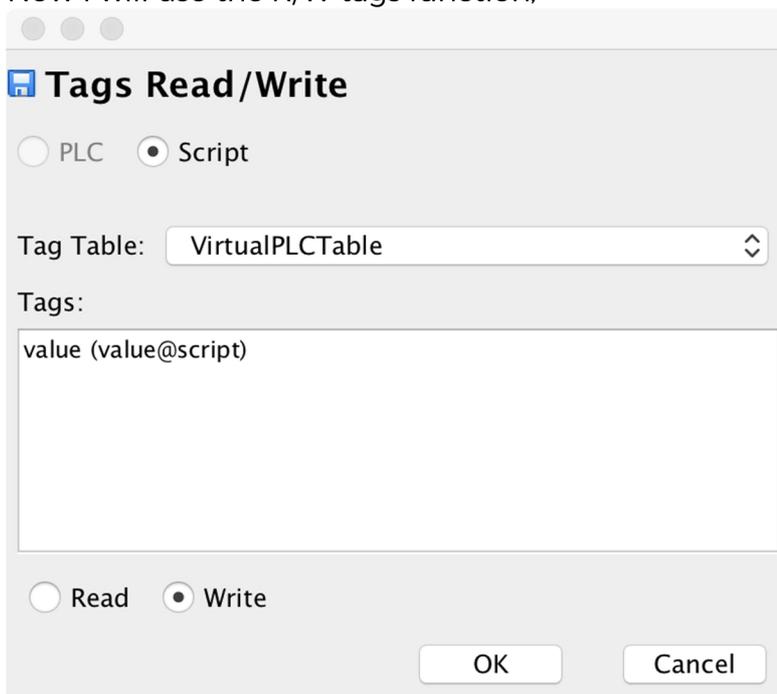


We have created the text element, and we have used the Get Value animation where we used our tag **value@script**

- Next step will be to periodically update the value from scripts. To do so, please use the function R/W Tags located in the toolbar of the script window.
  - Create a timer (you can use the Timer button in the toolbar)

```
16  setInterval(function() {  
17  
18  
19  
20  }, 1000); //Interval value in milliseconds
```

- Now I will use the R/W tags function,



## View and Server Side Scripts – Common tasks

After hitting “OK,” the generated code will look like this

```
16  global.counter=0;
17  setInterval(function() {
18      global.counter++;
19      //Write tags for tag group VirtualPLCTable
20  var options={};
21      options['name']="VirtualPLCTable";
22  options['values'] = {};
23  options['values']['value']=global.counter;
24  myscada.writeTags(options, function (err,data){
25      if (err){
26          //write error
27      }
28  });
29
30  }, 1000); //Interval value in miliseconds
```

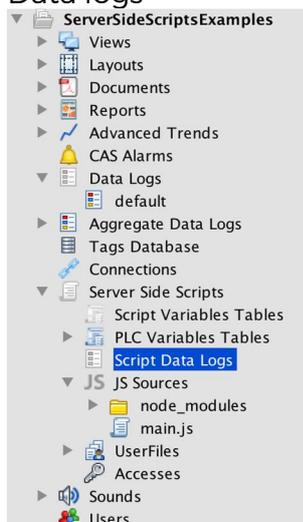
We are using a global variable **counter** to increase the value of value@script each 1 second.

TIP: analogically, you can read values from virtual PLC using your scripts. To do so, navigate again to the **R/W Tags** function and select **read** instead of write.

### Populating data-log from Server Side Scripts

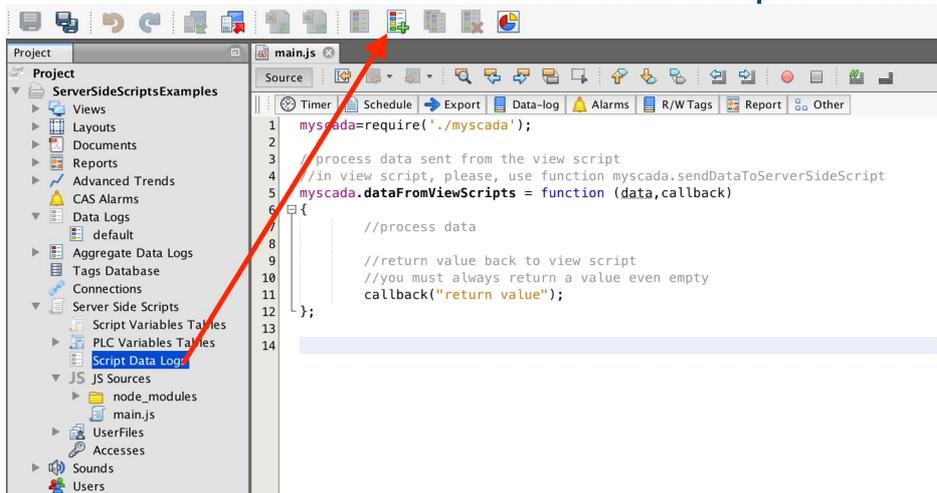
Often you need to compute something in the scripts and save results into the history – eg. Into data-log. To do so, we will present you with a simple example where we will every 10 seconds increase the value by one and save it into the datalog. When the value equals 10, we will start counting again. In the result you will have a datalog populated by values from 0 to 10 each 10 seconds. Let's get started

1. Create a Script Data log. To do so, navigate to Server Side Scripts -> Script Data logs

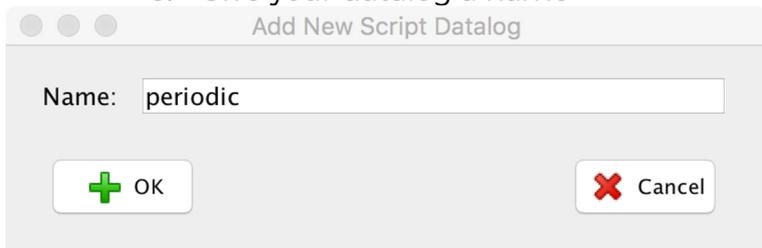


2. Select New in the main toolbar

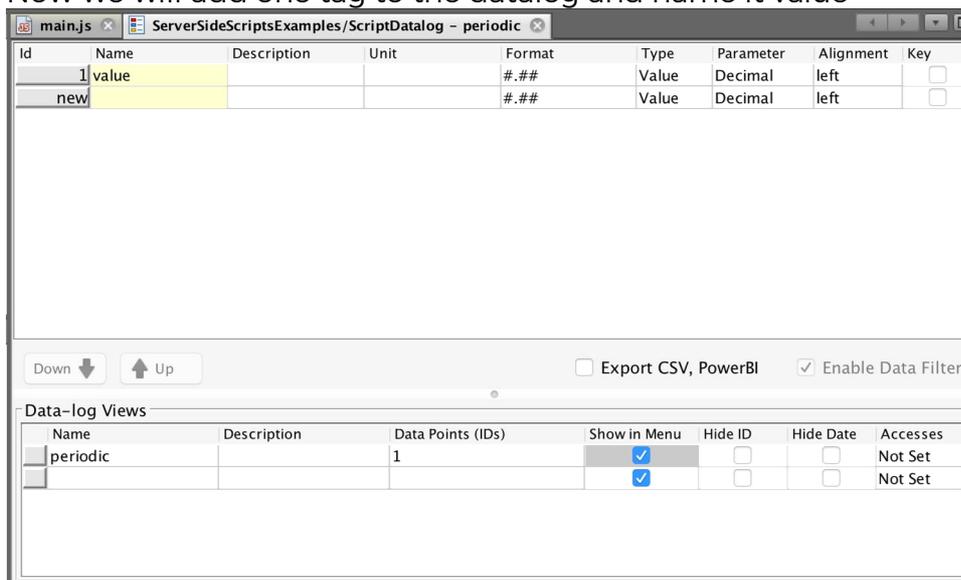
## View and Server Side Scripts – Common tasks



3. Give your datalog a name



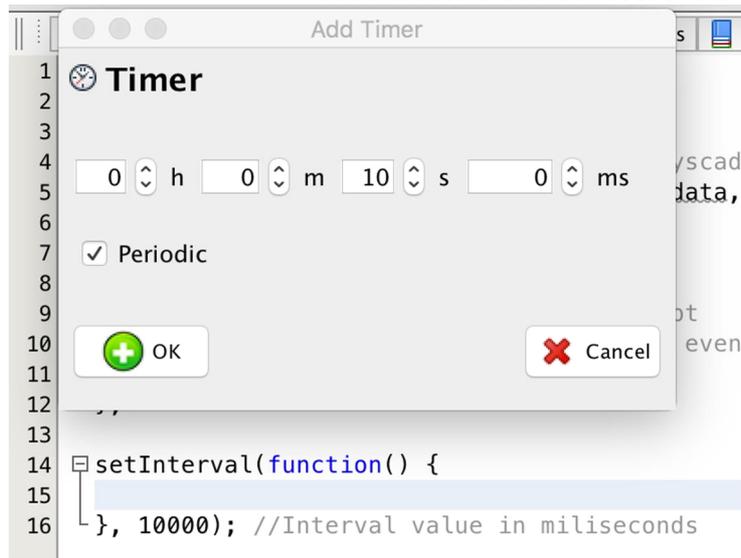
4. Now we will add one tag to the datalog and name it value



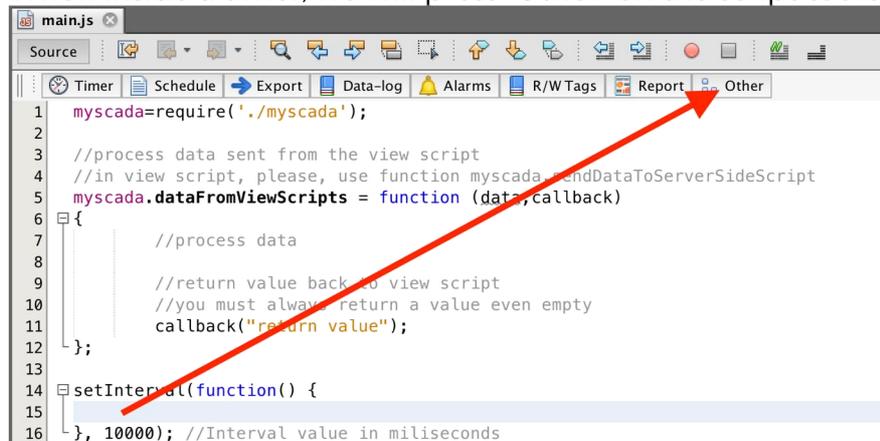
5. And now let's navigate to the main.js script.

6. Create a timer and set a repeat to 10 seconds

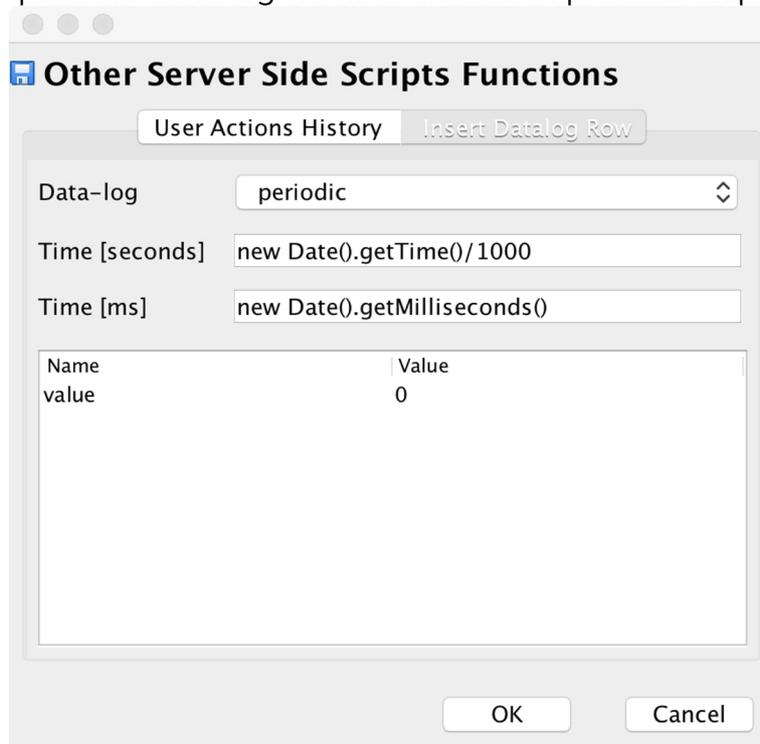
## View and Server Side Scripts – Common tasks



7. When inside a timer, we will press “Other” on the script toolbar



8. In the following dialog, we will select “Insert data data-log row” and select “periodic” data-log we have created in previous steps.



9. When we press “OK” button, we will see following code:

## View and Server Side Scripts – Common tasks

```
14  setInterval(function() {
15    //Insert row into data-log periodic
16    var options=new Object;
17    options['time']=new Date().getTime()/1000;
18    options['timeMs']=new Date().getMilliseconds();
19    options['dlgID']=2;
20    options['data']=new Array();
21    options['data'][0]=0; //value
22  myscada.insertRowIntoDataLog(options, function (err,data){
23    if (!err){
24      //insert was successfull
25    }
26  });
27
28  }, 10000); //Interval value in miliseconds
```

10. We will modify it to get the incremented value:

```
14  global.val=0;
15  setInterval(function() {
16    global.val++;
17    if (global.val>10) global.val=0;
18    //Insert row into data-log periodic
19    var options=new Object;
20    options['time']=new Date().getTime()/1000;
21    options['timeMs']=new Date().getMilliseconds();
22    options['dlgID']=2;
23    options['data']=new Array();
24    options['data'][0]=global.val; //value
25  myscada.insertRowIntoDataLog(options, function (err,data){
26    if (!err){
27      //insert was successfull
28    }
29  });
30
31  }, 10000); //Interval value in miliseconds
```

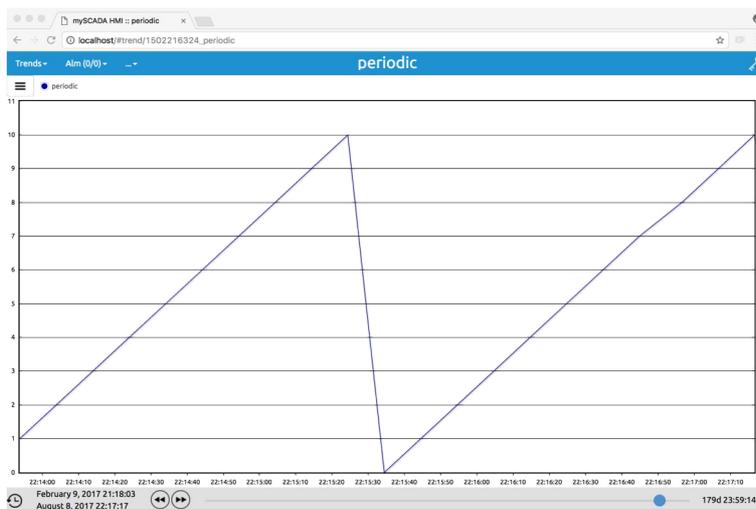
As you can see, we have added the `global.val` variable which we increase with each timer tick. This variable is written to the data-log using the function `insertRowIntoDataLog`.

11. When your project runs, it will automatically write the value to the data-log every 10 seconds

## View and Server Side Scripts – Common tasks



#	TIME	value
1	August 8, 2017 22:17:16	10.00
2	August 8, 2017 22:17:06	9.00
3	August 8, 2017 22:16:56	8.00
4	August 8, 2017 22:16:44	7.00
5	August 8, 2017 22:16:34	6.00
6	August 8, 2017 22:16:24	5.00
7	August 8, 2017 22:16:14	4.00
8	August 8, 2017 22:16:04	3.00
9	August 8, 2017 22:15:54	2.00
10	August 8, 2017 22:15:44	1.00
11	August 8, 2017 22:15:34	0.00
12	August 8, 2017 22:15:24	10.00
13	August 8, 2017 22:15:14	9.00
14	August 8, 2017 22:15:04	8.00
15	August 8, 2017 22:14:54	7.00
16	August 8, 2017 22:14:44	6.00
17	August 8, 2017 22:14:34	5.00
18	August 8, 2017 22:14:24	4.00
19	August 8, 2017 22:14:13	3.00



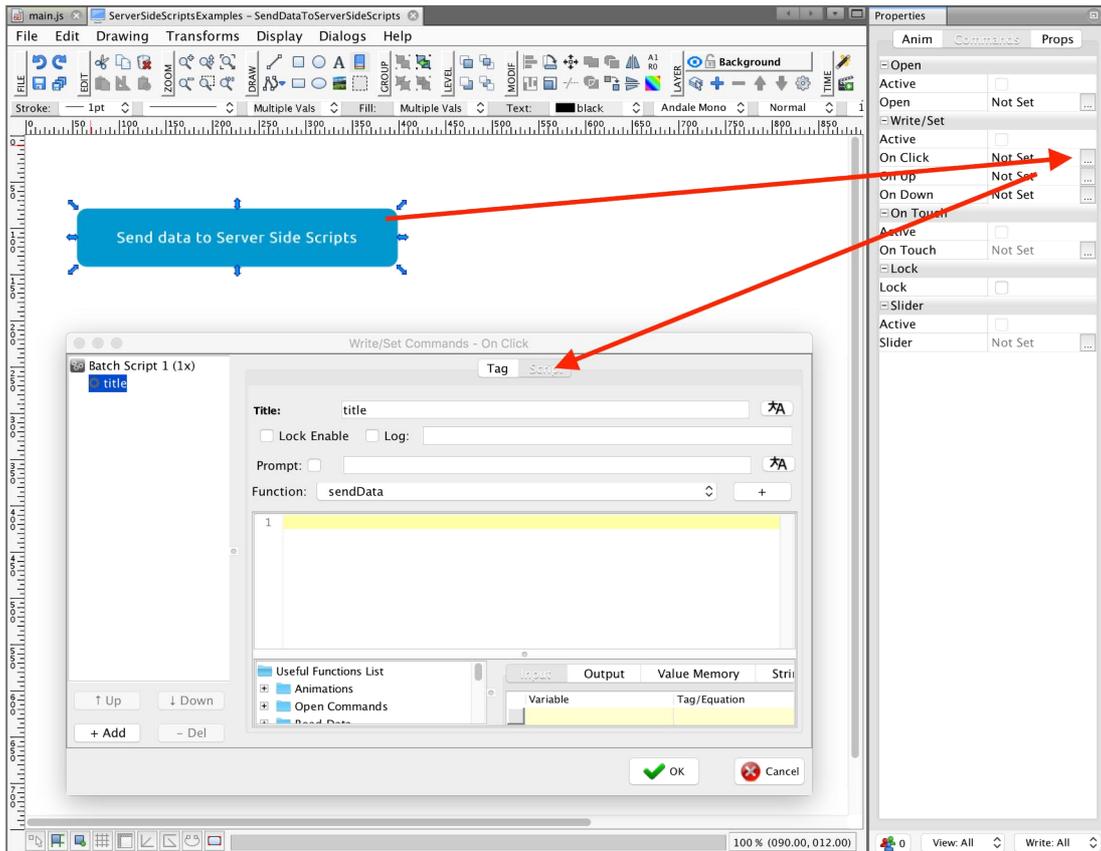
### 35.15 Sending Data from View Script into Server Side Scripts

You can easily send some data from View Scripts into the Server Side Scripts. When is this useful? For example, user can select some item on the screen, you can send this selection into the server side scripts, retrieve some data, write a recipe to the PLC or do whatever action you need to do.

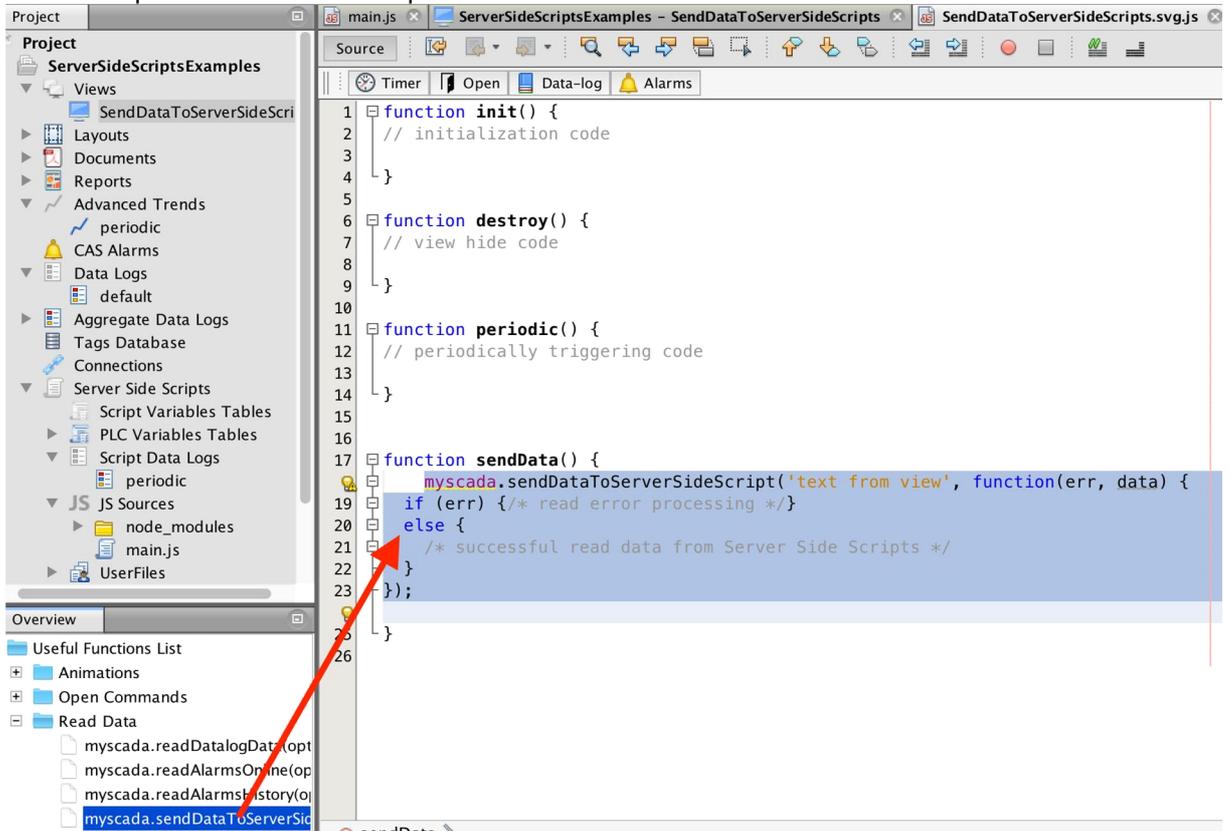
We will show you a simple example on how to send a javascript object into the server side scripts and get a javascript object as the reply back.

## View and Server Side Scripts – Common tasks

1. We will create a view with a button. Click on the button, in properties select **Commands** and **On Click** command. In On Click command select Script and create a new function `sendData`.



2. Now open a view script and fill in `sendData` function. You can use drag and drop from Help as shown on the picture:

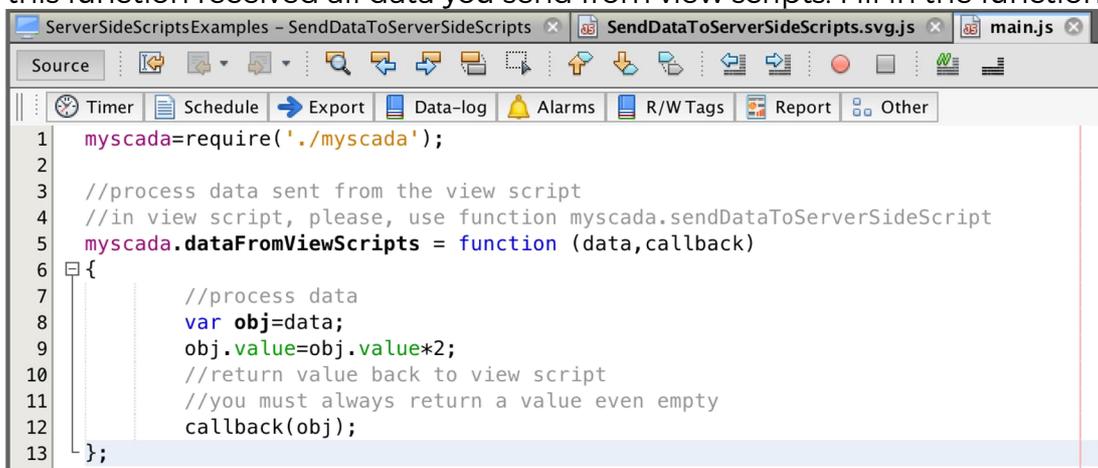


## View and Server Side Scripts – Common tasks

- Now we will create a javascript object and send it to the server side script. You can of course send a string or number as well.

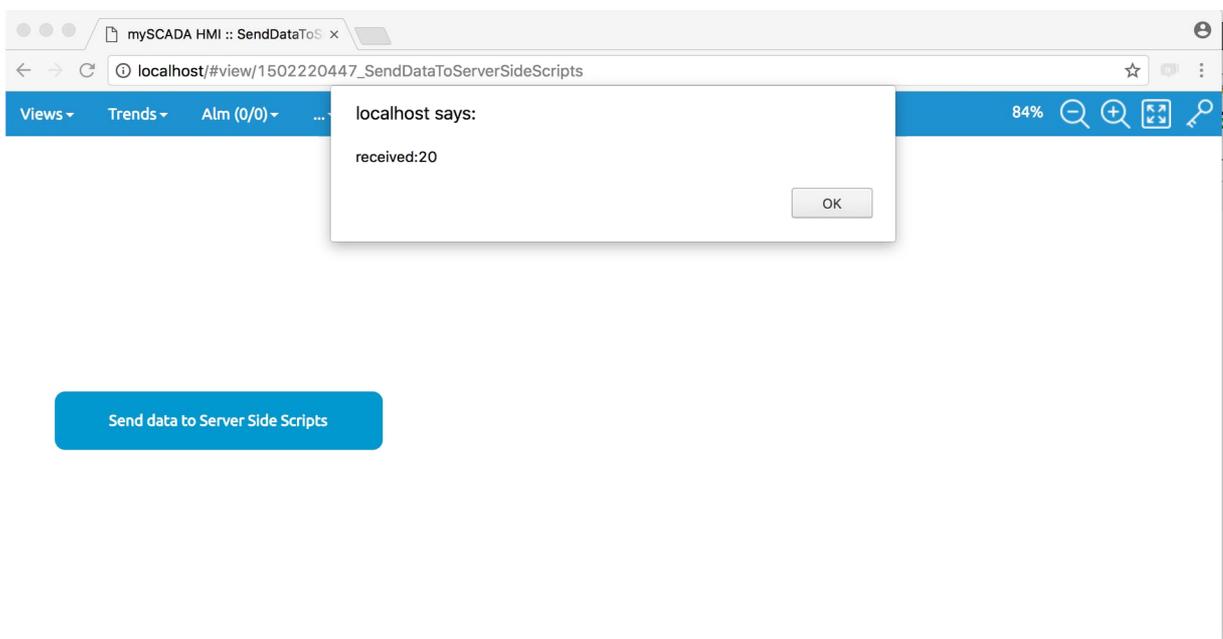
```
17 function sendData() {
18     var obj=new Object;
19     obj.value=10;
20     myscada.sendDataToServerSideScript(obj, function(err, data) {
21         if (err) { /* read error processing */
22         else {
23             /* successful read data from Server Side Scripts */
24             alert("received:"+data.value);
25         }
26     });
27 }
```

- Once we have our part in view script ready, let's go to our server side script. Open main.js
- You can see there is already prefilled function called `myscada.dataFromViewScripts` this function received all data you send from view scripts. Fill in the function:

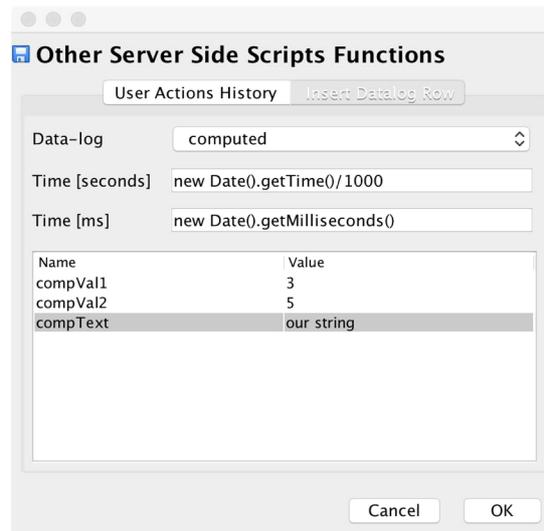


```
ServerSideScriptsExamples - SendDataToServerSideScripts x SendDataToServerSideScripts.svg.js x main.js x
Source
Timer Schedule Export Data-log Alarms R/W Tags Report Other
1 myscada=require('./myscada');
2
3 //process data sent from the view script
4 //in view script, please, use function myscada.sendDataToServerSideScript
5 myscada.dataFromViewScripts = function (data,callback)
6 {
7     //process data
8     var obj=data;
9     obj.value=obj.value*2;
10    //return value back to view script
11    //you must always return a value even empty
12    callback(obj);
13 }
```

- You can see that our function receives data, multiplies the value in received object by 2 and sends the object back.
- Now when you load your project into myPRO or myBOX, when you press a button, your object is sent to the server, received by the function `myscada.dataFromViewScripts` and then sent back to view script.



## View and Server Side Scripts – Common tasks



In the provided dialog, please select our created data-log by its name. Then fill in the time stamp of the record. Under this time, the record will be saved. Time has two parts, first is UTC time in seconds (since 1.1.1970) and second part is the number of milliseconds. If you leave the default values, new record will be logged at the time when your script is called.

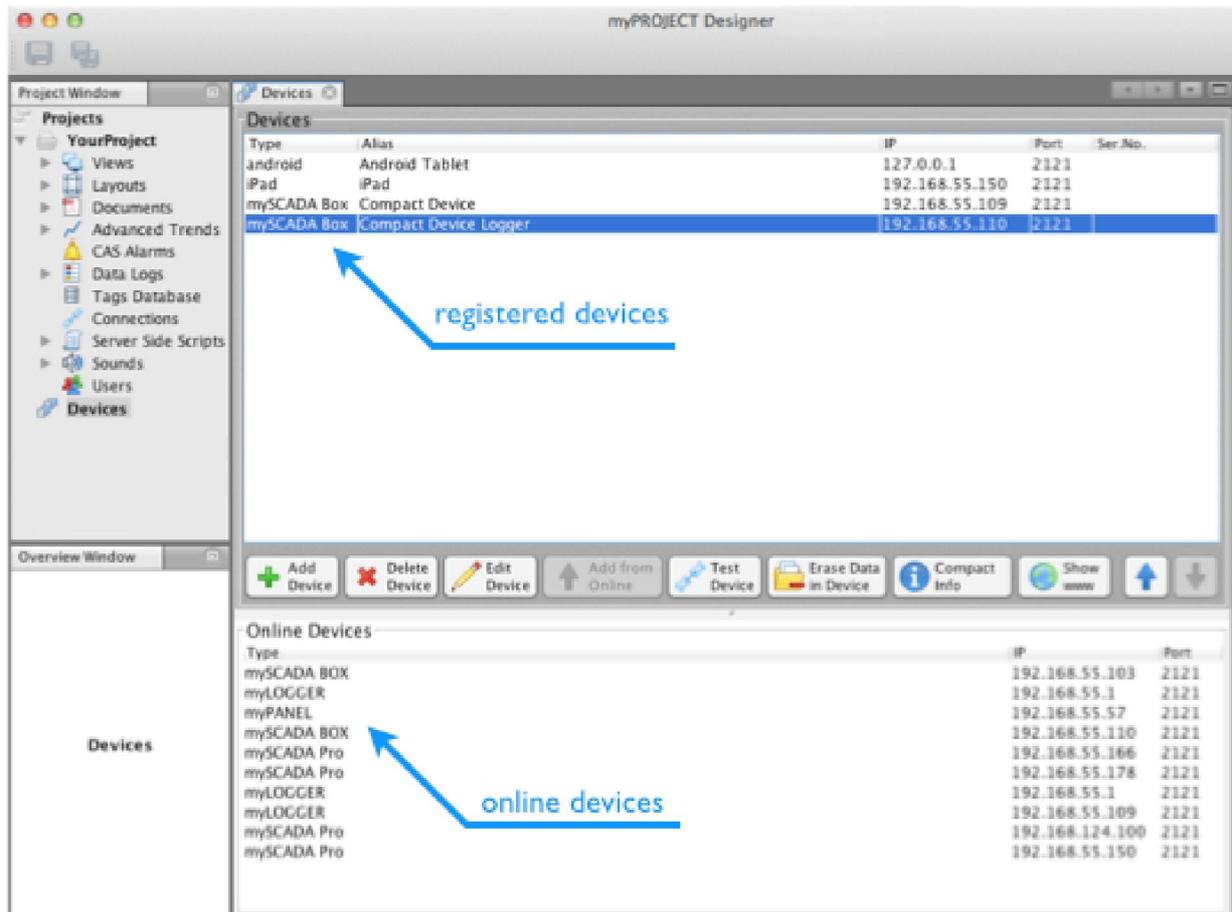
Finally, fill in the values for each defined tag in data-log. Those can be hard coded values or variables.

Once you are done, press **OK** button. myDESIGNER will automatically create a code for you and insert it into the edited script.

```
//Insert row into data-log computed
var options=new Object;
options['time']=new Date().getTime()/1000;
options['timems']=new Date().getMilliseconds();
options['dlgID']=2;
options['data']=new Array();
options['data'][0]=3;
options['data'][1]=5;
options['data'][2]="our string";
myscada.insertRowIntoDatalog(options, function (err,data){
  if (!err){
    //insert was successfull
  }
});
```

## 36 Devices

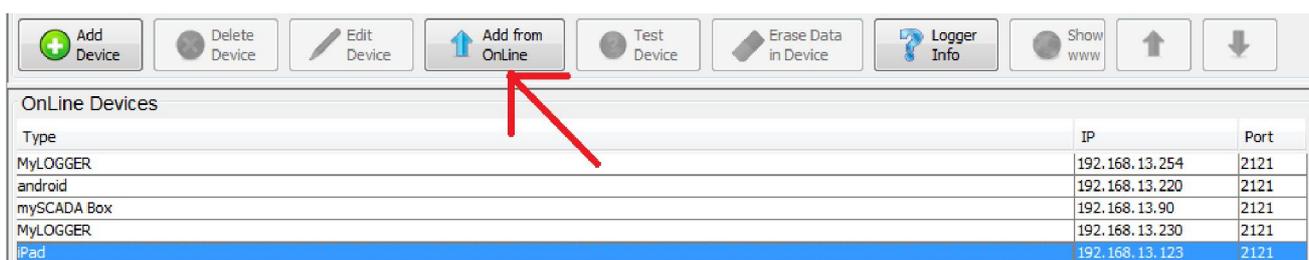
This module allows a quick overview of all available *mySCADA* device connections manually added and connected to the local network.



The main window of the *Devices* window is split into two parts:

- Upper part displays all permanently defined devices
- Lower part displays the list of all compatible devices in the local network (iOS or Android devices with the *mySCADA* app running in the foreground, *mySCADA* boxes, and *myPRO* computers)

You can easily store online devices by selecting them from the *Online Devices* list and clicking on the **Add from OnLine** icon.



- 1) Click on the **Add Device** button to add a device manually and fill in the dialog window:



The screenshot shows a dialog window titled "Add New Device". It contains the following fields and controls:

- Type:** A dropdown menu with "iPhone" selected.
- Alias:** An empty text input field.
- IP:** A text input field containing "192.168.1.1".
- Port:** A text input field containing "2121".
- Buttons:** A green "+" button labeled "OK" and a red "X" button labeled "Close".

- 2) Select the device type first
- 3) Set the device parameters:

*Alias* - name describing the device

*IP* - fill in the IP address of the devices

*Ser.No.* - serial number of the device (myBOX/myLOGGER only). You can find the serial number in the **System-Status** menu of the device.

## 37 EtherNet/IP Driver

The user is not forced to enter the data type during the screen drawing. The proper data type is fetched during the data reading from the PLC. There might be some limitations, as only certain data types are supported. The following paragraph specifies all supported data types for a given PLC type.

The EtherNet/IP driver supports all numeric data types:

Tag Data Type	Description
DINT	An atomic data type consisting of a DWORD used for storing a 32-bit signed integer value (-2,147,483,648 to +2,147,483,647).
SINT	An atomic data type that stores an 8-bit signed integer value (-128 to +127).
INT	An atomic data type consisting of a word used for storing a 16-bit signed integer value (-32,768 to +32,767).
REAL	An atomic data type that stores a 32-bit IEEE floating-point value.
BOOL	The BOOL data type is an atomic data type consisting of a single bit.

The access to user-defined data types, like structures or arrays, is supported. To read or write the elements inside the structured tags, you should use the “dot” notation, e.g.

*tank[1].volume* is proper syntax to read the volume of the 1<sup>st</sup> tank.

### Reading

### Controller

Tags:

To read a controller tag, just specify the tag name.

### Reading

### Program

Tags:

To read a program tag, you have to specify the program name before the tag name. You must use the following syntax:

Program:

*name*

-

program

name.tag

name

*tag* - tag name

So if you want to communicate with tag named valve from a program named control, you will have the following syntax: Program:control.valve

## Limitation of the BOOL arrays in ControLogix

The different syntax to access the elements in the BOOL arrays is used in the *myDESIGNER*, compared to the programs stored in the controller. In *myDESIGNER*, you have to address the word and bit position separately. The word consists of the 32 bits - BOOL elements and the range of the BOOL array is split into multiple words.

### Example:

If you have a BOOL array TestBool with 54 elements, you can access its elements using the following address:

3 <sup>rd</sup> element (bit)	TestBool[0].3
34 <sup>th</sup> element (bit)	TestBool[1].2
54 <sup>th</sup> element (bit)	TestBool[1].22
Out of range, bad address	TestBool[2].0
Out of range, bad address	TestBool[50]

## Recommendation on improvement of communication speed

The communication with the *ControLogix* controller has been designed in such way that reading arrays together is faster than reading each element separately. This means that if you use the tags that can be arranged into the arrays, you will notice a significant increase of speed.

For example, to read the status of levels in all 10 tanks, you should design your data types so that all levels are stored in an array with 10 elements. Below you will see an example of how to implement the faster method:

Level[1]	
Level[2]	FAST
Level[3]	

If you prefer structures, you can use the following way; however, addressing of this type will slow down the communication. Therefore, we do not recommend the use of this type of data addressing.

BigTank[1].Level	
BigTank[2].Level	SLOW
BigTank[3].Level	

## 38 MicroLogix and SLC Driver

For *MicroLogix* PLC, the data types are defined by means of the used file types. In PLC programs, you can use many different file types, but only the types specified in the following table can be read or written by the *mySCADA* application.

File Type	Description	Use with these controllers
N - Integer	16-bit long numbers, signed, this file is used for storing numeric values or bit information, read/write support.	SLC and MicroLogix
B - Binary	16-bit long numbers, signed, this file stores internal relay logic, read/write support.	SLC and MicroLogix
F - Float	This file stores numbers with a range of 1.1754944e-38 to 3.40282347e+38, read/write support.	SLC and MicroLogix
L - Long	32-bit long numbers, signed, also called double word, read/write support.	MicroLogix 1100, 1200, 1400 and 1500
O - Output	This file stores the state of output terminals for the controller, read only.	SLC and MicroLogix
I - Input	This file stores the state of input terminals for the controller, read only.	SLC and MicroLogix

The other file types like *Strings*, *Timer*, *Counter*, and *Control* are not supported.

There is a limitation that certain outputs can be read-only. To set outputs, you should use the other file type and then adjust the program design.

To access the file elements, you should specify the element position after the colon, i.e. to read or write to the third element of the integer file no. 100, use the following syntax:

*N100:3*

You can access each bit directly by defining its position after the slash, i.e. to access the seventh bit of the third element in the integer file no. 100 use the following syntax:

*N100:3/7*

## 39 Modbus Driver

The *Modbus* communication protocol describes the interactions of each device on a *Modbus* network. In the protocol description, you can find all the details about establishing addresses, device recognitions, and all other important parts of *Modbus* communication. Here we will concentrate on protocol usage by *mySCADA* application.

### 39.1 Tag name syntax

In the following table, the available syntaxes of the tag names are summarized.

Tag	Meaning	
Rn:e	16-bit signed integer stored in input register at address e	Read access only. Read function: 4
R:e	16-bit unsigned integer stored in input register at address e	Standard address range: 30001-39999
H:e	16-bit unsigned integer stored in holding register at address e	Read/write access. Read function: 3 Write function: 16 Standard address range: 40001 - 49999(extended to 499999 by some manufacturers)
Hn:e	16-bit signed integer stored in holding register at address e	
Hf:e	Float number stored in holding register at address e	
Hfs:e		
Hfsb:e		
Hfsw:e		
Rf:e	Float number stored in input register at address e	
Rfs:e		
Rfsb:e		
Rfsw:e		
Hd:e	32-bit integer stored in holding register at address e	
Hds:e		
Hdsb:e		
Hdsw:e		
Rd:e	32-bit integer stored in input register	

Rds:e	at address e	
Rdsb:e		
Rdsw:e		
I:e	Discrete input, Boolean value	Read access only. Read function: 2 Standard address range: 10000-19999
O:e	Discrete Output Coils	Read/write access: Read function: 1 Write function: 15 Standard address range: 1-9999

Many manufacturers use addresses instead of tags when describing *Modbus* communication. *mySCADA* currently does not support direct access by address, so you have to convert such addresses to tags, according to the provided table above. Be aware that specific read/write functions are needed to access each range, so the correct choice of tag type is important.

## 39.2 32-bit registers in Modbus

The *Modbus* protocol was designed to operate with devices of 16-bit register length. Consequently, special considerations are required when implementing 32-bit data elements. Most implementations use two consecutive 16-bit registers to represent 32 bits of data, or 4 bytes of data. Within these 4 bytes, single-precision, floating-point data can be encoded into a Modbus RTU message.

*Modbus* itself does not define floating-point data types, but it is widely accepted that it implements 32-bit floating-point data using the IEEE-754 standard. However, the IEEE standard has no clear definition of the byte order. Therefore, the most important consideration when dealing with 32-bit data is that the data be addressed in the correct order.

The following table shows two adjacent 16-bit registers conversion to a 32-bit floating point or 32-bit integer values:

Register suffix	Swap mode	16-bit registers	32-bit floating point or integer
-	N/A	[a b][c d]	[a b c d]
s	Byte and word swap	[a b][c d]	[d c b a]
sb	Byte swap	[a b][c d]	[b a d c]
sw	Word swap	[a b][c d]	[c d a b]

The following section describes the 32-bit data types implementation to the *mySCADA* application.

### 39.3 Floating point numbers

Floating point data type is possible for use in both input and holding registers, and all possible byte swap combinations are supported.

Register	Mapping	Swap mode	Bytes in 2 16-bit registers	Resulting 32-bit floating point
Hf	Holding registers	N/A	[a b][c d]	[a b c d]
Hfs	Holding registers	Byte and word swap	[a b][c d]	[d c b a]
Hfsb	Holding registers	Byte swap	[a b][c d]	[b a d c]
Hfsw	Holding registers	Word swap	[a b][c d]	[c d a b]
Rf	Input registers	N/A	[a b][c d]	[a b c d]
Rfs	Input registers	Byte and word swap	[a b][c d]	[d c b a]
Rfsb	Input registers	Byte swap	[a b][c d]	[b a d c]

Rfsw	Input registers	Word swap	[a b][c d]	[c d a b]
------	-----------------	-----------	------------	-----------

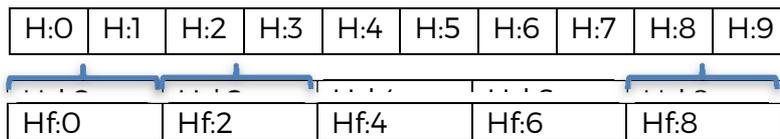
### 39.4 32-bit Integers

The long integers are implemented in the same manner as the floating-point numbers.

Register	Mapping	Swap mode	Bytes in 2 16-bit registers	Resulting 32-bit integer
Hd	Holding registers	N/A	[a b][c d]	[a b c d]
Hds	Holding registers	Byte and word swap	[a b][c d]	[d c b a]
Hdsb	Holding registers	Byte swap	[a b][c d]	[b a d c]
Hdsw	Holding registers	Word swap	[a b][c d]	[c d a b]
Rd	Input registers	N/A	[a b][c d]	[a b c d]
Rds	Input registers	Byte and word swap	[a b][c d]	[d c b a]
Rdsb	Input registers	Byte swap	[a b][c d]	[b a d c]
Rdsw	Input registers	Word swap	[a b][c d]	[c d a b]

### 39.5 Address mapping

On the *Modbus* server-side, only 16-bit long holding and input registers are used. The 32-bit long data types are just the interpretation of the two adjacent registers. The concept of address mapping is clearly shown in the following figure. Three tables are shown there. All of them address the same place in the memory: holding registers. In the first table the holding registers are shown, in the second the 32-bit integers are shown and, in the last table, the floating-point data types are shown.



### 39.6 Signed and unsigned numbers

Signed and unsigned integers can be stored in the registers. The unsigned numbers are read from or written to the *Modbus* device using simple addressing, like H or R to holding and input registers. To use signed integers, add the suffix “n” to the address. Using tag Hn:5 means that access the holding register at address 5 and the number will be interpreted as the signed integer.

In 32-bit data type numbers, the signed integers are used by default, and the suffix “n” is not used.

**Example:**

An example of advanced *Modbus* functionality can be seen on *ModbusDemo* screen, which can be downloaded as a part of the demo project from <http://www.myscada.org/downloads>.

The current values of inputs and outputs are visible in the upper part of the screen. Further, the current values of holding registers H:0 and H:1 and their data interpretation if the stored numbers are signed integers, float numbers, or 32-bit integers. To set the register, click on the **Set** or **Toggle** buttons. Figure 46 shows the storage of a floating point number 376.455 stored using an Hf:0 register, and the floating-point value 2.456 is stored in the input register Rf:0. z

click text of values or buttons to change current values

Inputs	I:0	I:1	I:2	I:3	I:4
	1	0	0	1	0

Outputs	O:0	O:1	O:2	O:3	O:4
	0	1	1	0	0
	<input type="button" value="Toggle"/>				

### Holding registers

all 32 bit input numbers are stored in 2 h. registers, they are mapped to the same address (0) in this demo

- unsigned integer	H:0	17340	<input type="button" value="Set"/>	H:1	14909	<input type="button" value="Set"/>
- signed integer	Hn:0	17340	<input type="button" value="Set"/>	Hn:1	14909	<input type="button" value="Set"/>
- 32 bit integer, no swap	Hd:0	1136409149				<input type="button" value="Set"/>
- 32 bit float, no swap	Hf:0	376.455				<input type="button" value="Set"/>
- 32 bit integer, swap words and bytes	Hds:0	1027259459				<input type="button" value="Set"/>
- 32 bit float, swap words and bytes	Hfs:0	0.046				<input type="button" value="Set"/>
- 32 bit integer, swap words	Hdsw:0	977093564				<input type="button" value="Set"/>
- 32 bit float, swap words	Hfsw:0	0.001				<input type="button" value="Set"/>
- 32 bit integer, swap bytes	Hdsb:0	-1136444102				<input type="button" value="Set"/>
- 32 bit float, swap bytes	Hfsb:0	-0.012				<input type="button" value="Set"/>

### Input registers

all 32 bit numbers are stored in 2 input registers, they are mapped to the same address (0) in this demo

- unsigned integer	R:0	16413		R:1	12059
- signed integer	Rn:0	16413		Rn:1	12059
- 32 bit integer, no swap	Rd:0	1075654427			
- 32 bit float, no swap	Rf:0	2.456			
- 32 bit integer, swap words and bytes	Rds:0	456072512			
- 32 bit float, swap words and bytes	Rfs:0	0.000			
- 32 bit integer, swap words	Rdsw:0	790315037			
- 32 bit float, swap words	Rfsw:0	0.000			
- 32 bit integer, swap bytes	Rdsb:0	490740527			
- 32 bit float, swap bytes	Rfsb:0	0.000			

## 40 Siemens S7family PLCs Driver

**IMPORTANT:** mySCADA is using a direct memory access when communicating with Siemens S7 PLCs. Make sure you uncheck the “optimized block access” checkbox in your TIA portal project.

Standard S7-200/300/400/1200 LOGO! item Syntax

### Address Syntax

#### Input, Output, Flag Memory Types

<memory type><S7 data type><address>

<memory type><S7 data type><address><.bit>

#### DB Memory Type

DB<num>,<S7 data type><address>

DB<num>,<S7 data type><address><.bit>

where <num> ranges from 1 to 65535.

### 40.1 Memory types

Memory Type	Description	Address Range	Data Type	Access
<b>I</b> <b>E</b>	Inputs	Dependent on S7 Data Type (see table below)		Read/Write
<b>Q</b> <b>A</b>	Outputs			Read/Write
<b>M</b> <b>F</b>	Flag Memory			Read/Write
<b>V</b>	Variable Memory			Read/Write
<b>DB</b>	Data Blocks			Read/Write

*Note: The Variable Memory is available only for S7-200 and LOGO!.*

### 40.2 S7 Data types

S7 Data Type	Description	Address Range	Data Type
<b>X</b>	Bit	Xo.b-X65534.b .b is Bit Number 0-7	<b>Boolean</b>
<b>B</b> <b>BYTE</b>	Unsigned Byte	Bo-B65535 BYTEo-BYTE65535	<b>8-bit unsigned integer</b>
		Bo.b-B65535.b BYTEo.b-BYTE65535.b .b is Bit Number 0-7	Boolean

<b>C CHAR</b>	Signed Byte	Co-C65535 CHARo-CHAR65535  Co.b-C65535.b CHARo.b-CHAR65535.b .b is Bit Number 0-7	<b>8-bit signed integer</b>  Boolean
<b>W WORD</b>	Unsigned Word	Wo-W65534 WORDo-WORD65534  Wo.b-W65534.b WORDo.b-WORD65534.b .b is Bit Number 0-15	<b>16-bit unsigned integer</b>  Boolean
<b>I INT</b>	Signed Word	Io-I65534 INTo-INT65534  Io.b-I65534.b INTo.b-INT65534.b .b is Bit Number 0-15	<b>16-bit signed integer</b>  Boolean
<b>D DWORD</b>	Unsigned Double Word	Do-D65532 DWORDo-DWORD65532  Do.b-D65532.b DWORDo.b- DWORD65532.b .b is Bit Number 0-31	<b>32-bit unsigned integer</b>  Boolean
<b>DI DINT</b>	Signed Double Word	DIo-DI65532 DINTo-DINT65532  DIo.b-DI65532.b DINTo.b-DINT65532.b .b is Bit Number 0-31	<b>32-bit signed integer</b>  Boolean
<b>REAL</b>	IEEE Float	REALo-REAL65532	<b>Float</b>

*Note: Be cautious while modifying WORD, INT, DWORD, and DINT types, as each address starts at a byte offset within the device. Therefore, words MWO and MWI overlap at byte 1. Writing to MWO will also modify the value held in MWI. Similarly, DWORD and long types can also overlap. It is recommended that these memory types should be used in such way that overlapping does not occur. As an example, DWORD MD0, MD4, MD8 ... etc. can be used to prevent bytes from overlapping.*

### 40.3 S7 1200/1500 notes

To access the DB in S71200/1500, some additional settings in PLC-side are needed.

1. Only global DBs can be accessed.
2. The optimized block access must be turned off.
3. The access level must be "full" and the "connection mechanism" must allow GET/PUT.

Set the previous in TIA Portal (shown version V12)

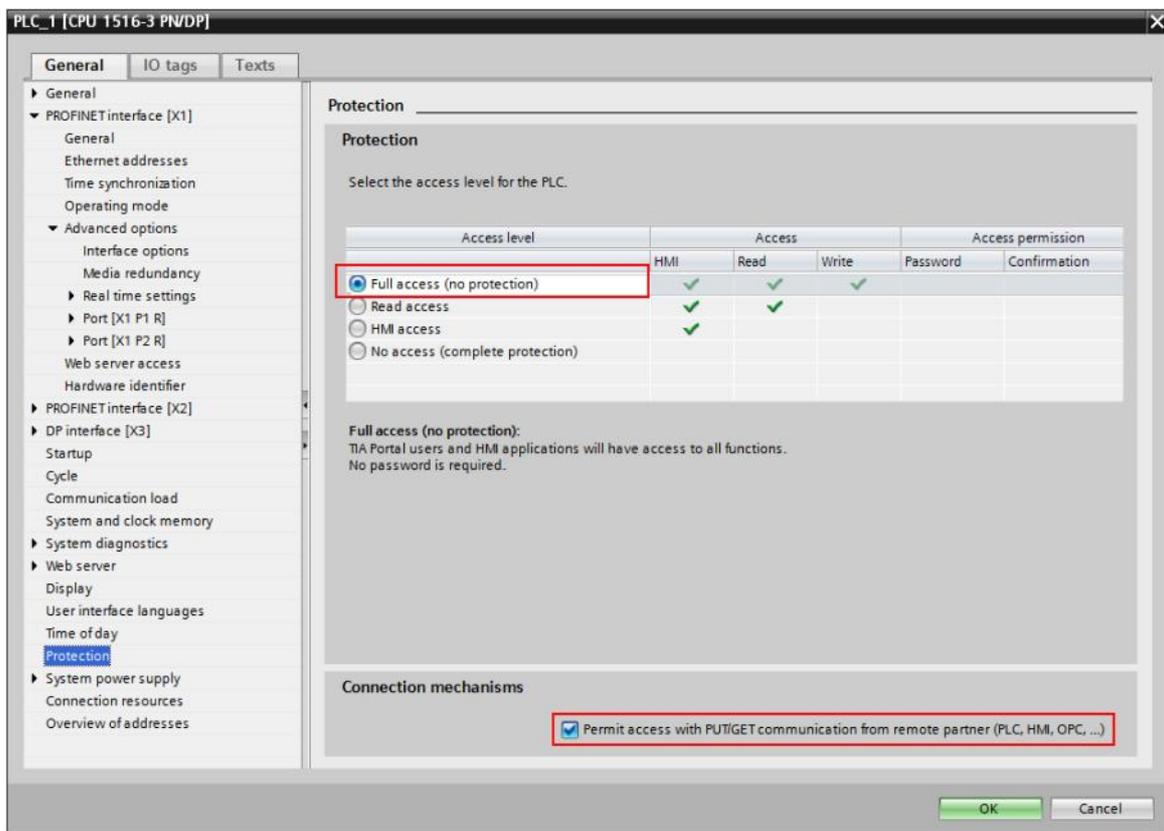
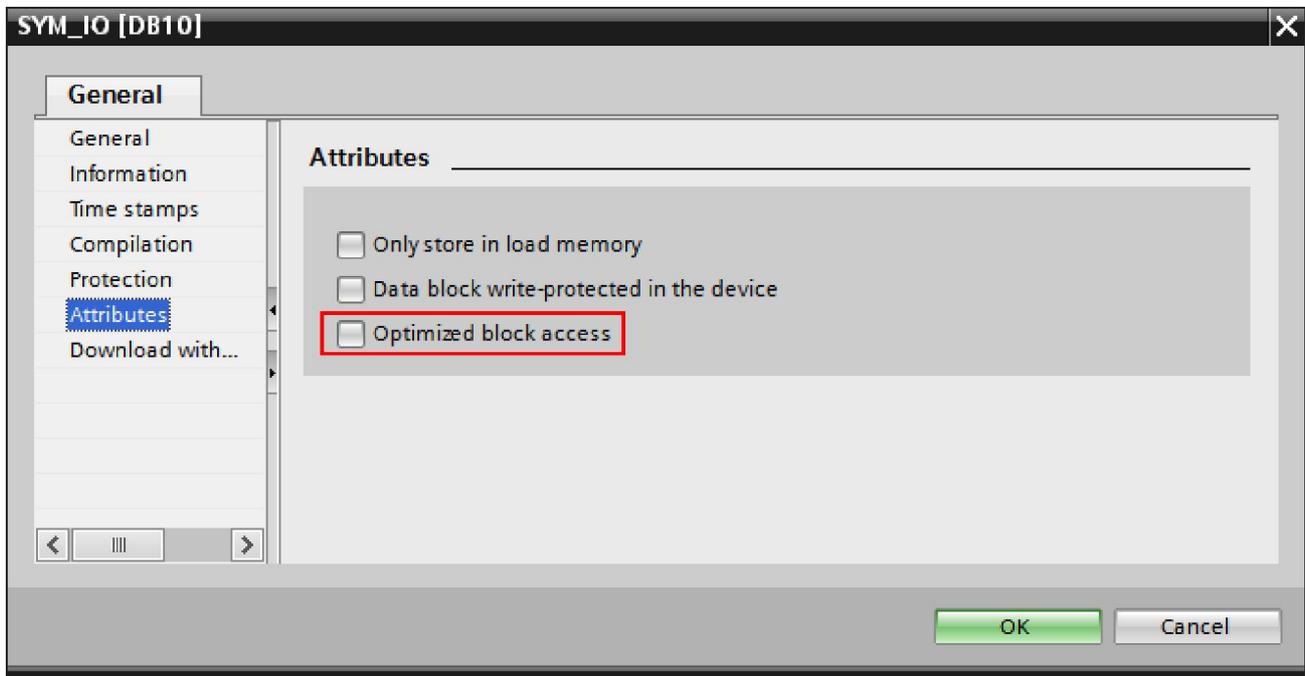
Select the DB in the left pane under "Program blocks" and press **Alt+Enter** (or in the contextual menu select "Properties...").

Uncheck Optimized block access (by default it is checked).

## 40.4 Protection

Select the CPU project in the left panel and press **Alt+Enter** (or in the contextual menu select “Properties...”)

In the item *Protection*, select “Full access” and Check “Permit access with PUT/GET ....”, as pictured.



## 40.5 LOGO! OBA7/OBA8 configuration

Configuring the **server connection** allows you to connect *LOGO!* with *mySCADA* devices for reading and writing to the memory, just like an HMI panel would do.

- 1) Select the **Ethernet Connections** item in the **Tools** menu.
- 2) Right-click on the **Ethernet Connections** and select *Add connections* to add a connection.
- 3) Double-click on the new connection created and edit its parameters by selecting **Server Connection**.
- 4) Confirm the dialog, close the connection editor and **download** the configuration into LOGO!

*Note:*

*The Local TSAP corresponds to the Remote TSAP of LOGO! and vice-versa. **This is a key concept for the S7 connections!** If you uncheck "Accept all connections," you must specify the mySCADA device address. The "Connect with an operator panel" checkbox can be checked or unchecked.*

## 40.6 S7-200 (via CP243-1) configuration

The configuration of *S7 200* is very similar with *LOGO!*

- 1) Select **Internet Wizard** in the **Tools** menu.
- 2) Set up the CP243 position and IP configuration; in the next window, select at least one *Peer-to-Peer* connection and set it up the same way as with *LOGO!*

## 4.1 OPC UA Driver

---

OPC Unified Architecture is an interoperability standard developed by the OPC Foundation. It is the successor to OLE for process control (OPC). Although developed by the same organization, OPC UA differs significantly from its predecessor, i.e. the older DCOM-based version is not supported. Using this protocol has two main advantages:

- **Security** - configures a secure message interchange based on contemporary cryptography
- **Scalability** - OPC server can communicate with a different vendor's infrastructure

You should keep in mind that when using OPC you are not communicating with PLCs directly. The OPC server stands in the middle, and all communication is dependent on the OPC server configuration.



*mySCADA* implementation of the OPC UA standard supports only the binary protocol - *opc.tcp://Server*. We do not support the web version - <http://Server>.

### 4.1.1 Connection configuration

With *myDESIGNER*, you can set connections to the *OPC* servers and the following parameters.

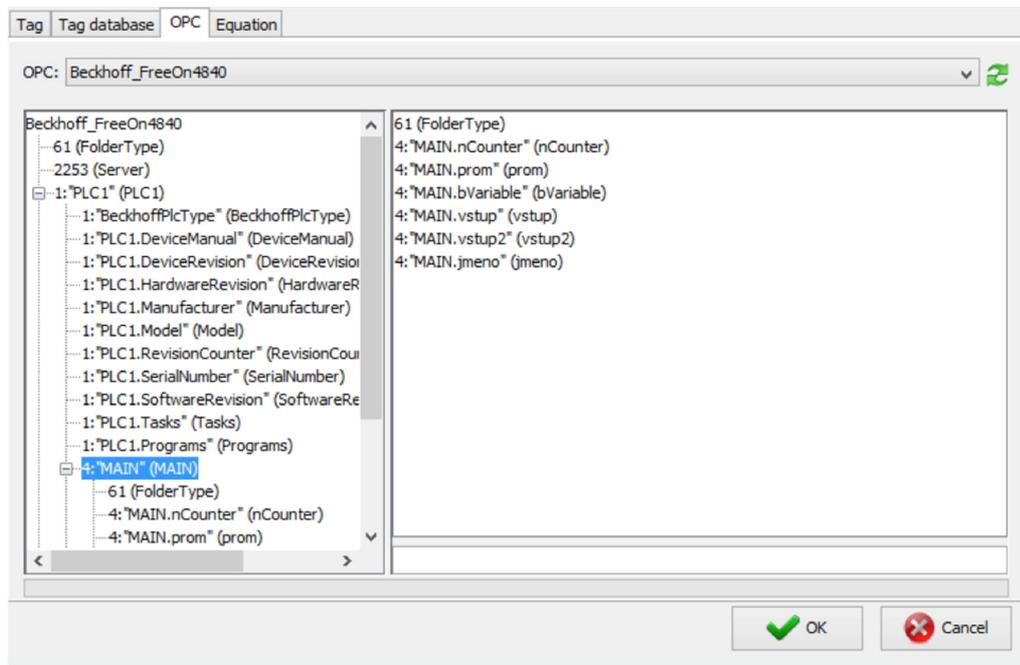
The screenshot shows a configuration dialog for an OPC UA driver. The fields are as follows:

- Type: OPC UA
- Alias: Ignition\_55.58\_userpass
- IP: 192.168.55.58
- Port: 4096
- Password
  - User: opcuauser
  - Psw: [masked]
- Path: opc.tcp://192.168.55.58:4096/None/None
- Security police: None
- MessSecMode: None
- Advanced Options
  - Optimisation Window: 50
  - Separate Writes

- **IP Address, Port** – IP address and port of the *OPC UA* server
- **User, Password** (Optional) – credentials to the *OPC UA* server, if configured on the server
- **Security Police** (Optional)
  - **Basic128Rsa15** - uses 128-bit cryptography
  - **Basic256** - uses 256-bit cryptography
  - **None** (Default)
- **Message Security Mode** (Optional)
  - **SignEnc** - Signs and encrypts the messages
  - **Sign** - Signs the messages
  - **None** (Default) - No security
- **Certificates** (Optional) - you should provide a private key and associated certificate in the PEM format; you can easily recognize the PEM format that the certificate file will begin with “-----BEGIN CERTIFICATE-----” and end with the sentence “-----END CERTIFICATE-----”.

## 41.2 Tag Name Syntax

The tag name is any string that is supported and accepted by the connected *OPC UA* server. You can use the browse function in the designer to get the list of tags at your disposal. You can use the dialog to select the proper tag; later, after selecting, you can edit the tag name. Usually, the subsequent edition is not needed, but sometimes it is, e.g. to target the proper tag address by adding the suffix.



## 42 MELSEC-Q Driver

Currently, *mySCADA* supports only a part of *MELSEC-Q* protocol, namely 3E type of packets, originally intended for E71 type adapters. In the table below, you can find the list of both basic (original) tag syntaxes and some extensions introduced by our team.

### 42.1 Tag name syntax

In the following table, the available syntax of the tag names is summarized.

Tag	Meaning	Address range (decimal)
SM (91h)	Special relay, 1-bit value	0000 to 2047
SD (A9h)	Special register, 16-bit value	0000 to 2047
X (9Ch)	Input, 1-bit value	0000 to 8191
Y (9Dh)	Output, 1-bit value	0000 to 8191
M (90h)	Internal relay, 1-bit value	0000 to 8191
L (92h)	Latch relay, 1-bit value	0000 to 8191
F (93h)	Annunciator, 1-bit value	0000 to 2047
V (94h)	Edge relay, 1-bit value	0000 to 2047
B (A0h)	Link relay, 1-bit value	0000 to 8191
D	Data register, 16-bit value, unsigned	000000 to 012287

(A8h)		
W (B4h)	Link register, 16-bit value, unsigned	0000 to 8191
TS/TC (C1h)/(C0h)	Contact/Coil timer, 1-bit value	0000 to 2047
TN (C2h)	Current value timer, 16-bit value, unsigned	0000 to 2047
SS/SC (C7h)/(C6h)	Contact/Coil retentive timer, 1-bit value	0000 to 2047
SN (C8h)	Current value retentive timer, 16-bit value, unsigned.	0000 to 2047
CS/CC (C4h)/(C3h)	Contact/Coil counter, 1-bit value	0000 to 1023
CN (C3h)	Current value counter, 16-bit value, unsigned.	0000 to 1023
SW (B5h)	Link special register, 16-bit value, unsigned.	0000 to 2047
SB (A1h)	Link special relay, 1-bit value	0000 to 2047
DX (A2h)	Direct input, 1-bit value	0000 to 8191
DY (A3h)	Direct output, 1-bit value	0000 to 8191

Our implementation of MELSEC protocol also supports the following extension tags:

Tag	Meaning	Address range (decimal)
DS (A8h)	Data register, 16-bit value, signed	000000 to 012287
DD (A8h)	Data register, 32-bit value, signed	000000 to 012286 Use regular D registers, address increased by 2
FL (A8h)	Data register, 32-bit float value	000000 to 012286 Use regular D registers, address increased by 2
FD (A8h)	Data register, 64-bit float value	000000 to 012284 Use regular D registers, address increased by 4
WS (B4h)	Link register, 32-bit signed value	0000 to 8191 Use regular W registers, address increased by 2

All custom tags consist of multiple regular 16-bit tags in the *least to the most significant* word order.

## 42.2 Connection Settings

**IP Address** – IP address of the PLC.

**Port** – port for the TCP connection to the PLC; please note that all *Mitsubishi* PLCs support only one connection / port. For multiple connections to the same PLC, set a different port for each *mySCADA* device.

**MultiBatch Optimized** – enable this function if the PLC supports “Multiple block batch read/write” functions (codes 0406/1406). Enabling this option for a PLC that does not support such functions will cause a communication error.

**CPUTimer** – set this parameter if you expect an extensive communication with the PLC or communicate with a heavily loaded PLC to increase the response preparation time on the PLC side.

**Optimization Window** – this setting influences how “aggressive” the optimization of your request will be. If it is set to 0, each tag will be requested in a separate packet (slow, but helps to find errors in the syntax). With a default value of 1, all “adjusted” tags (D0000 and D0001, for example) will be requested in one packet. Setting it to 2 allows optimization to skip one tag address to the count tags as “adjusted” (for example, D0000 and D0002 will be requested in one packet with such a setting). Setting it to 3 allows it to skip 2 addresses, etc.. The maximal value for this setting is 20. Please note that if an error returns in response to the request, all tags included in this request will not be read. Use this feature carefully.

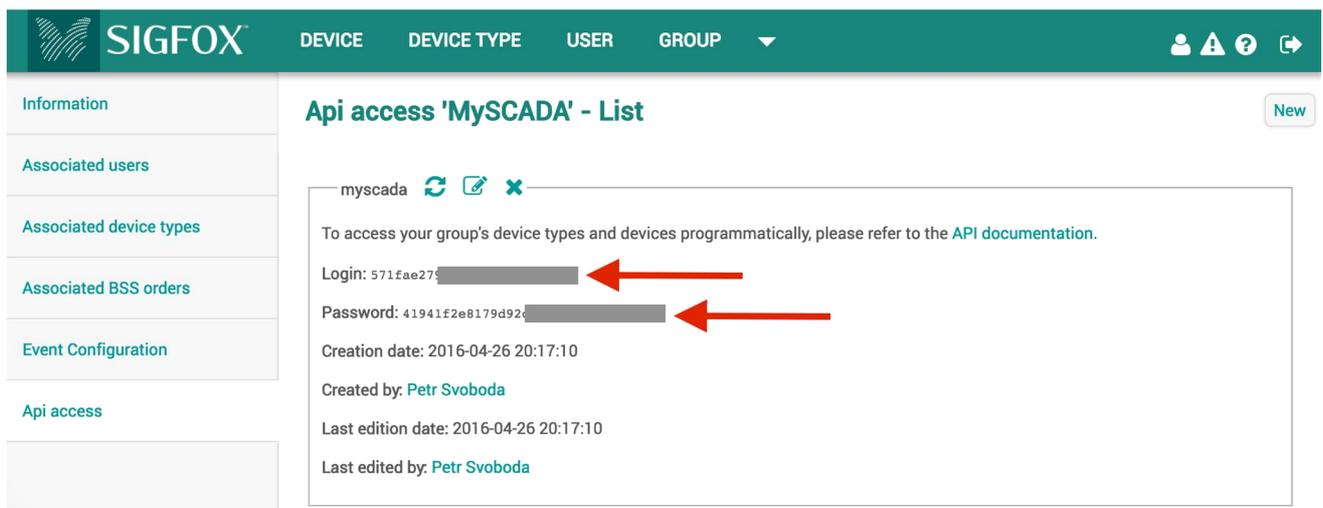
The following block of settings is only required for hierarchy access within the *MELSEC* network of multiple PLCs - for a direct (local) communication, leave the settings at their default values.

- **PC Number** – sets number of requesting station
- **Network number** – sets the network number to send the request to
- **Request dest.Module No.** – sets the number of requested modules for multi-CPU PLC connection
- **Request dest.Module I/O** - sets the number of requested modules for connecting via multi-drop

## 43 SigFox Driver

SIGFOX is a cellular-like system that connects remote devices using Ultra Narrow Band (UNB) technology. It is mainly targeted at low data rate applications. A large number of devices can be interconnected using SIGFOX technology. mySCADA provides a direct connection to the SIGFOX portal. You can retrieve current data from all of your devices registered in SIGFOX portal and use them freely in mySCADA to visualize, log, and use for alarming.

1. First, log into the SIGFOX portal at <https://backend.sigfox.com>, select your group, and enable "Api access." You will be presented with login details as in the picture below:



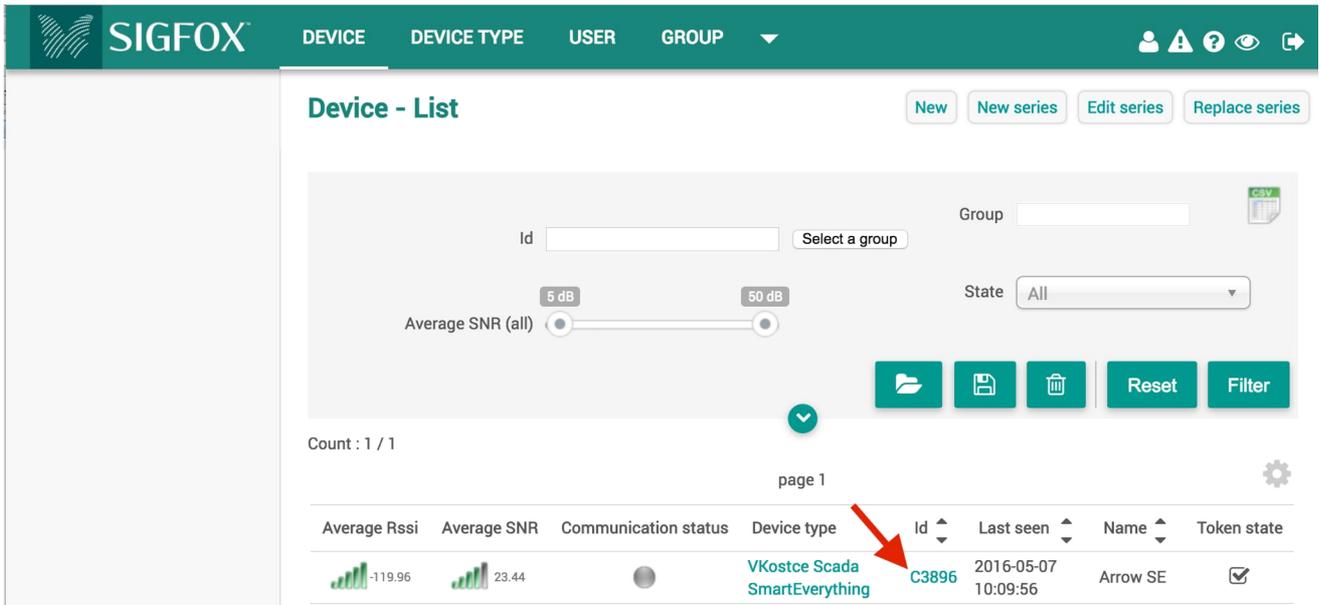
The screenshot shows the SIGFOX portal interface. The top navigation bar includes 'DEVICE', 'DEVICE TYPE', 'USER', and 'GROUP'. The main content area is titled 'Api access 'MySCADA' - List'. A sidebar on the left contains menu items: 'Information', 'Associated users', 'Associated device types', 'Associated BSS orders', 'Event Configuration', and 'Api access'. The main content area displays the following information for the 'mySCADA' connection:

- mySCADA (with refresh, edit, and delete icons)
- To access your group's device types and devices programmatically, please refer to the [API documentation](#).
- Login: 571fae275 [redacted] (indicated by a red arrow)
- Password: 41941f2e8179d924 [redacted] (indicated by a red arrow)
- Creation date: 2016-04-26 20:17:10
- Created by: Petr Svoboda
- Last edition date: 2016-04-26 20:17:10
- Last edited by: Petr Svoboda

2. Create a new connection in the mySCADA project and enter the Login and Password from the previous step.

3. Now please add all the devices you wish to use the SIGFOX portal.

Reading data from devices: To read data from your devices connected to SIGFOX network, all you need to know is the ID of your device. You can find the ID in the device list:



When you know ID of your device, you can read various pieces of information from the device: for example, latitude and longitude, time of delivered message, or the message itself. To read the message, you have several options of how to retrieve values from the message automatically. To do so, please look at the format on how to do it.

This is the format on how to retrieve a value from your device:

## deviceID.code.datatype[index].bit

**deviceID** is the ID of your device as registered in SigFox portal; it is a five-digit hexadecimal number.

**code** specifies which data you want to read from the device. Possible options are:

- data - message from the device (up to 12 bytes)
- lat - latitude
- lng - longitude
- time - time of received message in UTC unix timestamp format
- tap - TAP ID of the base station that received the message

*TIP: if you just want to read values from the message, you can omit writing "data" and directly write: **deviceID.datatype[index].bit***

### 43.1 datatype

The system can automatically convert most of the data types from your message. The following types are possible:

code	type	Length [byte]
uchar	unsigned char	1
char	signed char	1
usint	unsigned short int	2
sint	signed short int	2
uint	unsigned int	4

Int	signed int	4
float	floating point number	4
Chars	return char array	Length of message
String	read and convert the message into HEX string format	Length of message * 2

**IMPORTANT:** By default, all data are represented in Little Endian format, typical for embedded devices. If you want to represent data in Big Endian format, add “**be**” before the data type.

Example: reading integer from devices message

- In little endian format                   C3896.int (or C3896.leint)
- In big endian format                    C3896.beint

## 43.2 Reverse order

By default, the system reads bytes from your message from left to right. You can simply reverse the order (e.g. read bytes of your message from right to left) by adding “r” before the data type. Example

Message from device as seen in SIGFOX portal:

7044b200000031a4fb41

command **C3896.int** will read 7044b200 and convert to int value

command **C3896.rint** will read 14bf4a13 and convert to int value

## Index

If you don't specify index, values are read from the beginning (that means from the zero index). You can specify at which byte of the message your data are located, e.g. where to start reading from. Therefore, for example, C3896.int[2] will start reading data from the third byte of the message and will read 4 bytes, converting them to integers. The index always starts from 0.

## Bits

For integer data types (char, sint, int) you can also extract bits. To do so add .BitNo after the data type. For example, this command C3896.char[1].2 will read the second character from the message and extract the third bit. Therefore, you will get 0 or 1 as a result, depending on the bit status.

### Advanced Settings

Moreover, you can control some advanced settings concerning your device by adding a JSON type string after the tag. The syntax is as follows

**deviceId.code.datatype[index].bit{“timeout”:3600}**

Currently, only the option timeout is supported; in the future, more options will be added based on customer requirements. Timeout specifies the maximum age of the message to consider device communication. The value is in seconds.

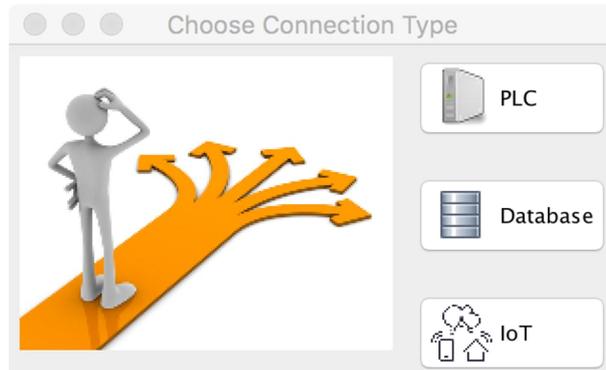
## Examples

C3896.char	read first char of the message
C3896.char[10]	read 11th character of the message (indexing is always from zero)
C3896.int	read first four bytes of message and convert them to integers
C3896.int[2]	read four bytes of the message starting at the third character and convert them to integers
C3896.char[1].0	read the second character of the message and return 1st bit of it

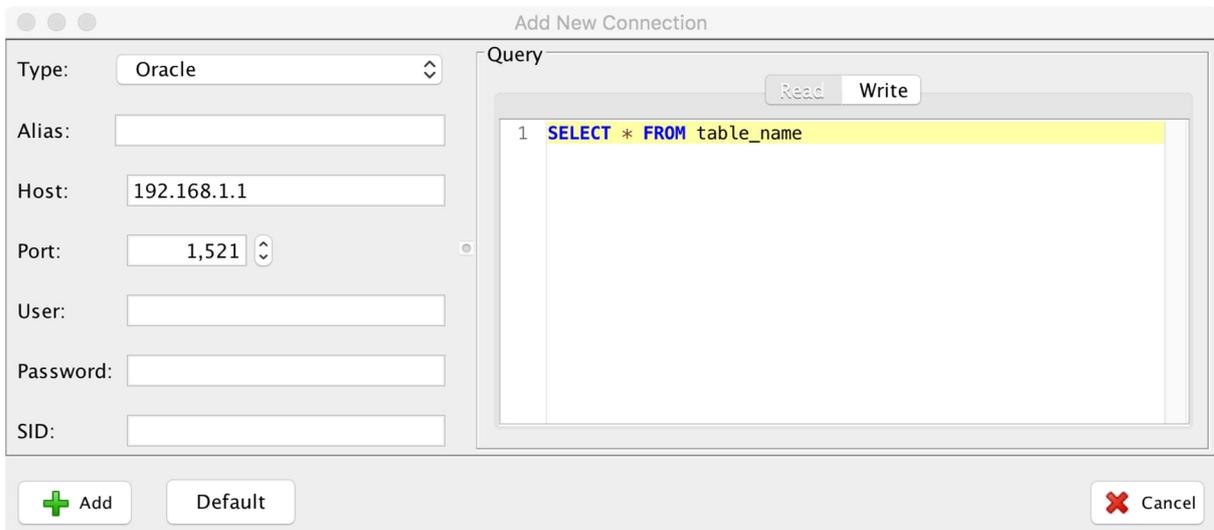
## 44 Databases Driver

mySCADA has embedded communication drivers for most of the common database systems, namely Oracle DB, Microsoft SQL, PostgreSQL, MySQL, and SQLite. mySCADA also enables generic connections to any database supporting a standardized ODBC driver.

To connect to the database, start by adding a connection to your project. First, go to connections, then click on add connection. A new dialog will be shown:



Please select Database, and you will be provided with connection details.



Please fill in necessary details such as username, password, and Host address. It can be either the IP address or host name.

In the *Query -> Read* section, fill in your query to retrieve data.

Then Click on *Add* to add this connection.

## 44.1 Reading values from SQL database

Once you have added the connection, you can now use values from the database anywhere in the mySCADA project. To do so, please follow the described syntax.

Imagine the following SQL query

```
SELECT ID,COLUMN1, COLUMN2, COLUMN3 FROM table_name
```

To retrieve data from this query you can:

1. Specify directly column name

```
COLUMN1@alias
```

Will retrieve values as array from column with name COLUMN1.

2. Specifying a column position with keyword **COL**

```
COL[1]@alias
```

Will retrieve values as array from first column.

3. Specifying column name and row index

```
COLUMN2[5]@alias
```

Will retrieve value from column with name COLUMN2 on 5<sup>th</sup> row.

4. Specifying column name with a condition  
(supported operators are: =,<,>)

```
COLUMN1[ID>1]@alias
```

Will retrieve values from first column; only rows with column named id is greater than 1 will be processed.

5. Specifying a row position with keyword **ROW**

```
ROW[1]@alias
```

Will retrieve first row of values and return them as array

6. Specifying a row with a condition  
(supported operators are: =,<,>)

```
ROW[id=5]@alias
```

Will retrieve values from first column; only rows with column named id is greater than 1 will be processed.

## 44.2 Examples

(tag data are marked in green)

Reading tag COLUMN1@alias :

ID	COLUMN1	COLUMN2	COLUMN3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53

Reading tag COL[1]@alias :

ID	COLUMN1	COLUMN2	COLUMN3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53

Reading tag COLUMN2[5]@alias :

ID	COLUMN1	COLUMN2	COLUMN3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53

Reading tag COLUMN3[ID>2]@alias :

ID	COLUMN1	COLUMN2	COLUMN3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53

Reading tag ROW[1]@alias :

ID	COLUMN1	COLUMN2	COLUMN3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53

Reading tag ROW[COLUMN1=31]@alias :

ID	COLUMN1	COLUMN2	COLUMN3
1	11	12	13
2	21	22	23
3	31	32	33
4	41	42	43
5	51	52	53

*Note: Access to Oracle, PostgreSQL, and MySQL databases uses a native driver provided by the corresponding companies.*

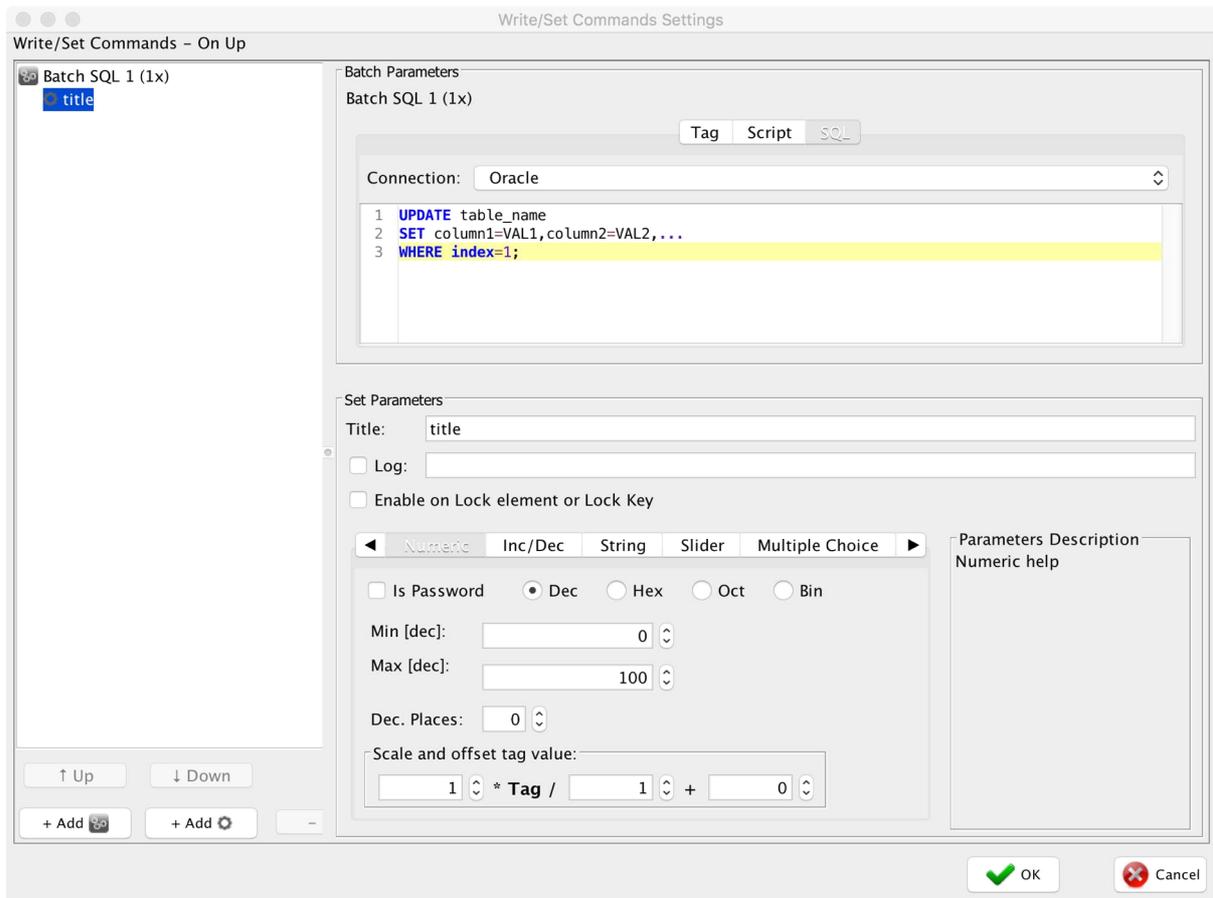
Tip: You can have multiple connections to single or even multiple databases. This way you can use multiple tables and optimize your queries.

### 44.3 Writing values to SQL database

You can also use mySCADA to write new values or update existing ones in the SQL database. To do so, please follow these steps:

1. In the connection parameters, please fill in the **SQL → Write Query**. This query is a template for your write query; you can modify it anytime when specifying a new set command. However, it is convenient to prepare this template query so you don't have to write it all over again.

## 2. Specify a set command



In set command, you can see your write query. This query will write values to the database. As you can see in the query, there is the keyword VAL; whenever you use VAL followed by a number, this VAL will be replaced with the real value provided as user input at runtime.

If you put multiple set commands into one Batch, they will be all processed as one query; in this case, you should have as many VAL keywords in your query as you have your sets in your batch.

For detailed information on how to work with set command and batches, please look at the corresponding chapter in this manual.

## 44.4 Using SQL connectivity in Server Side Scripts

You can use SQL connectivity in your scripts. This way you actually have limitless options on how to link mySCADA with your database. You can, for example, read data from PLCs, process them, and send processed data to the SQL database.

To work with SQL in Server Side Scripts, please use the corresponding mySCADA functions in the section SQL. You can also look at the provided examples.

# 45 Download/Upload from/to Device

---

Watch video describing this functionality: <https://www.youtube.com/watch?v=TJ7Qot2iItY>

## 45.1 Download to Device

When you have finished designing and setting your project, you are ready to download it to your operating device.

- 1) Select the project you wish to download to the device from the *Projects* folder in the *Project Window* - this will render the **Upload/Download** icons in the toolbar.

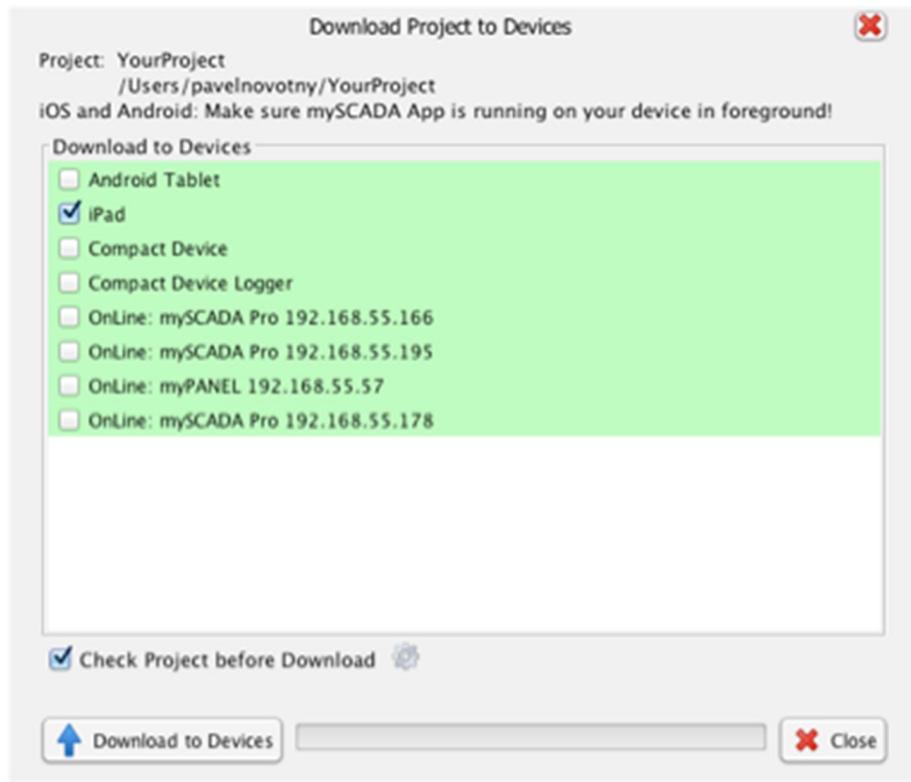


- 2) Click on the **Download to Devices** button to load the selected project to the device.



You can also right-click on the project from the list and select *Download* from the menu.

- 3) In the following dialog window, you will see the list of all defined devices and all available **On-line** devices you can load your project into (except *mySCADA Box* and *myLOGGER* - these need a serial number for communication).



- 4) Before downloading, check the box **Check Project before Download** -> if there are any errors found the following dialog will show up:



*Note: You can download your project to multiple devices at once. To do that, check multiple devices.*

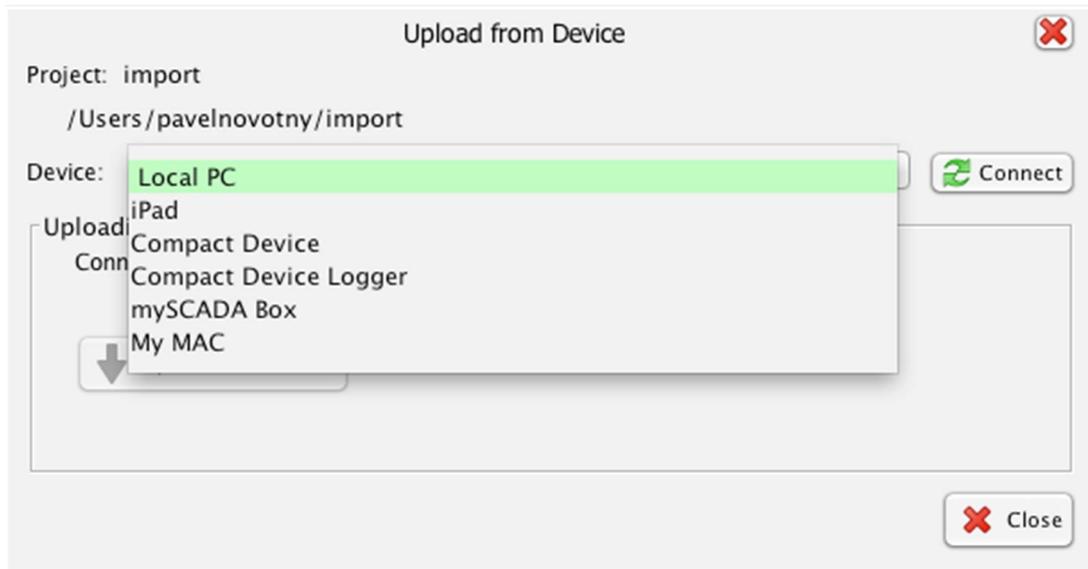
**Warning:** when loading a new project to your device **all existing projects loaded on the device will be overwritten!** Thus, be sure to back-up all important project information from the device before loading a new project to it.

## 45.2 Upload from Device

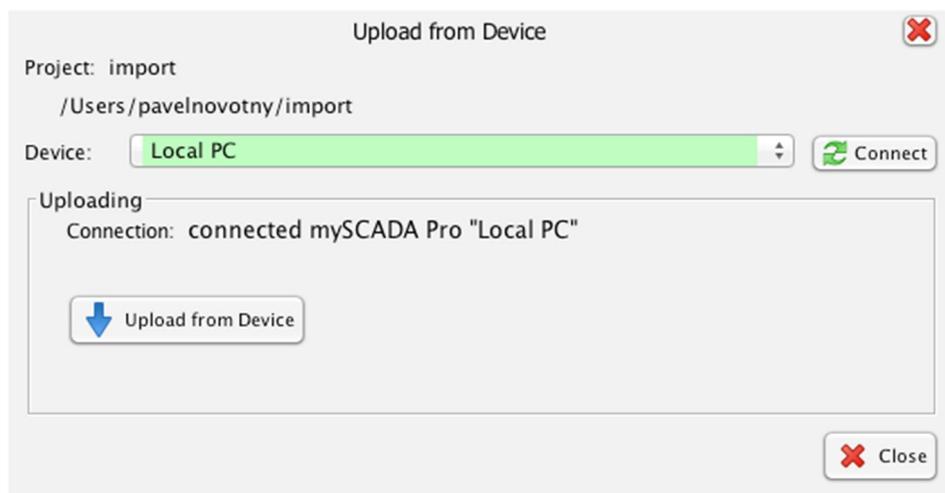
You can also upload a *mySCADA* project from your device to *myDESIGNER*.

Select one of the existing projects:

- 1) Click on the **Upload from Device** button  or right-click on the **Projects** folder in the *Project Window* and select *Upload from Device* from the context menu. The following dialog window will show up:

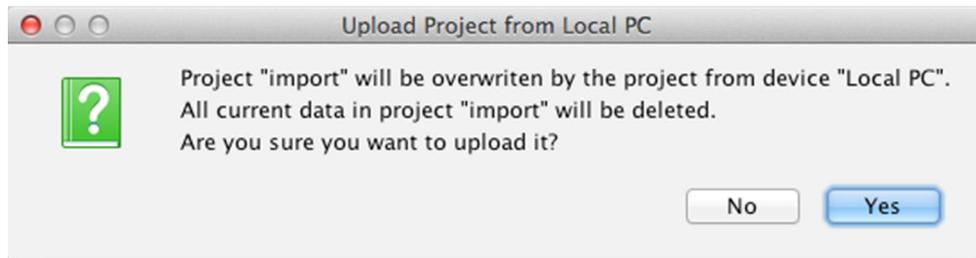


- 2) Select the device you want to upload the project from and click on the **Connect** button. If the connection is successful the following message will be displayed:



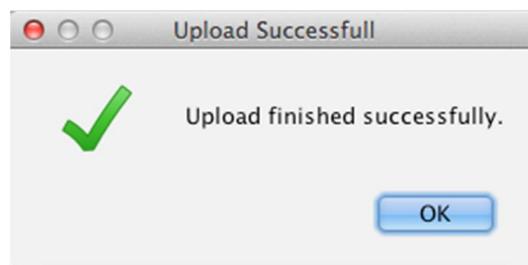
- 3) Click on the **Upload from Device** button 

The warning dialog notifying you of possible project data loss will appear.



**Warning: All data saved in the project folder will be lost!**

4) After successful project uploading, the confirmation dialog will be shown.



Now start using your created project and enjoy working with *mySCADA*.

THANK YOU!

No part of this document or of the program may be reproduced or transmitted in any form without the express written permission of mySCADA Technologies s.r.o.

Information in this document is subject to change without notice and is not binding in any way for the company mySCADA Technologies s.r.o.