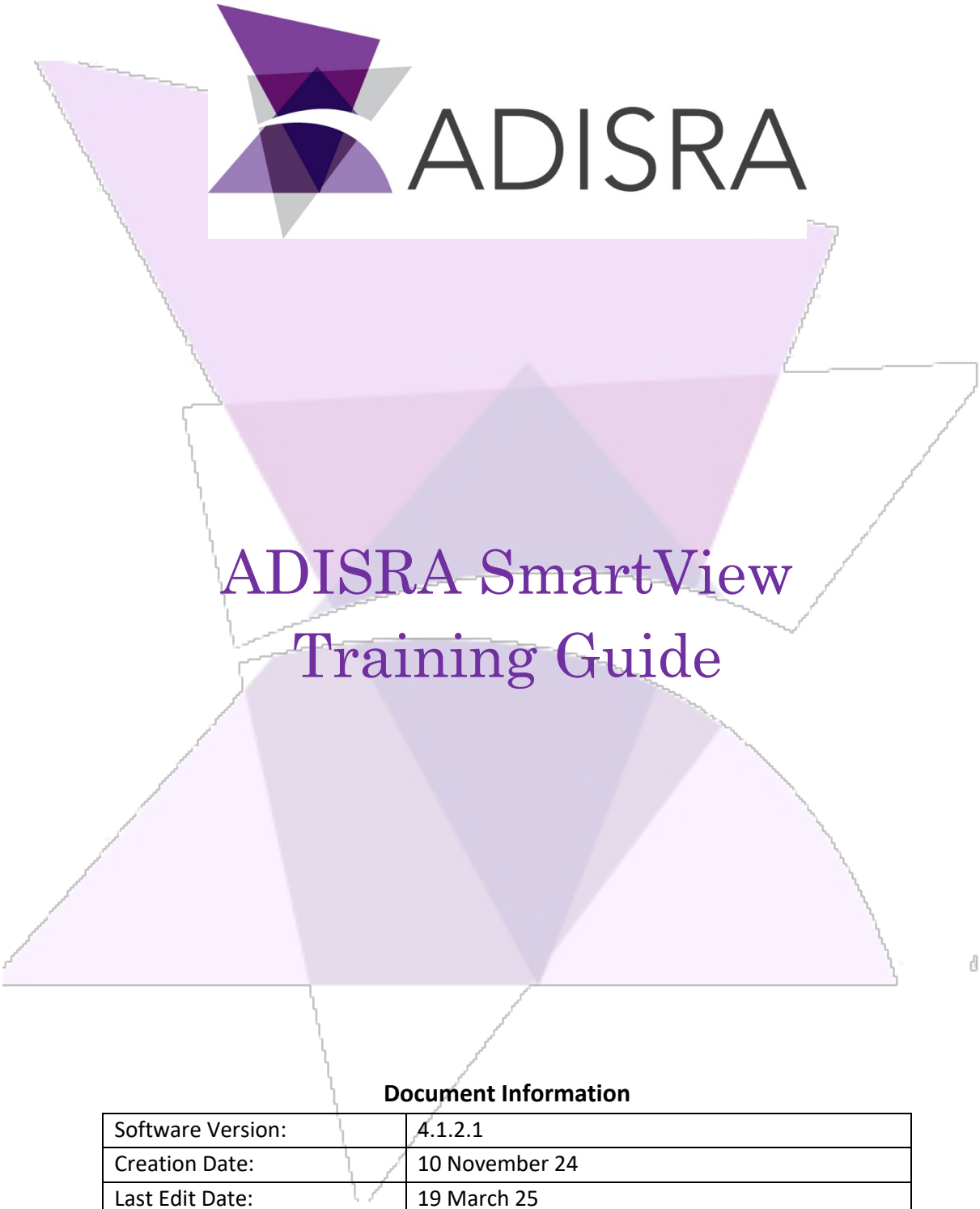


ADISRA SmartView Training Guide

Document Information

Software Version:	4.1.2.1
Creation Date:	10 November 24
Last Edit Date:	19 March 25
Version:	V 1.1

ADISRA · 6001 W. Palmer Lane, Suite 350-285 · Austin, TX 78727
Phone: 1-833-5ADISRA (1-833-523-4772) ·. 1-512-993-2949
www.ADISRA.com

Table of Contents

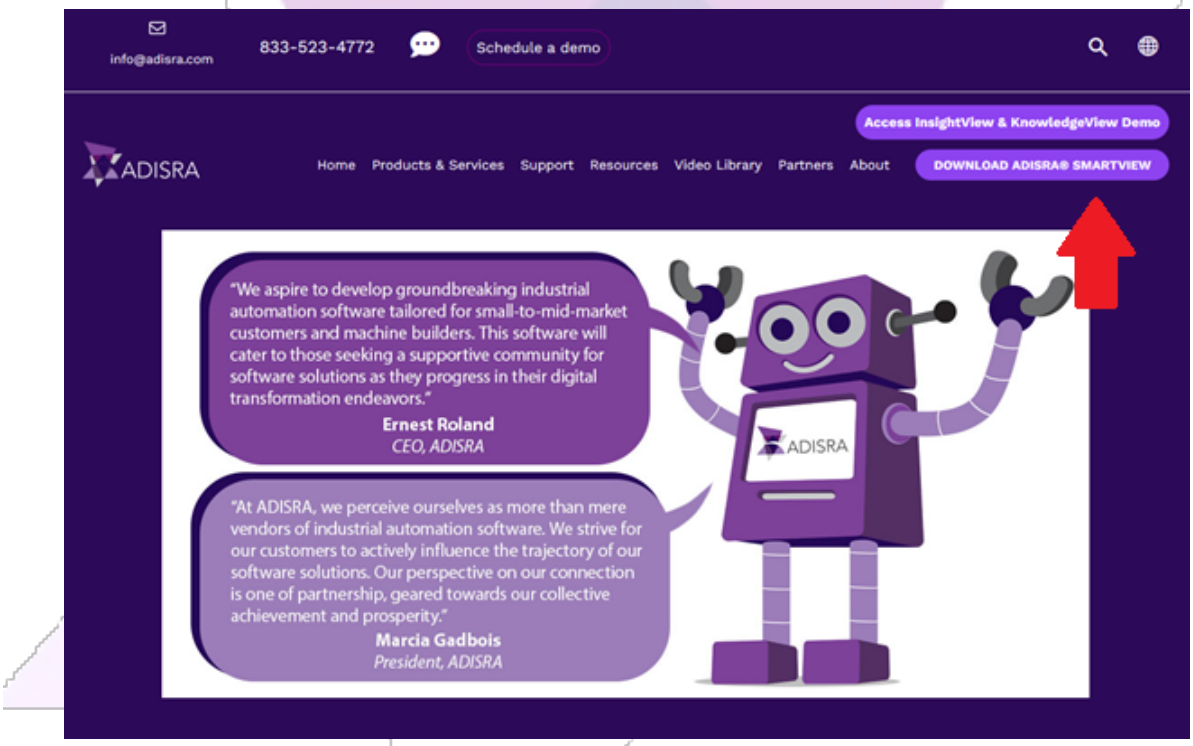
1. Summary

This document explains how to download, install, register, develop, and run a SCADA application using the main resources available in the ADISRA SmartView software engineering environment.

Any doubt, suggestion or criticism of this training is welcomed by the development team. Therefore, please contact us at support@adisra.com.

2. Downloading ADISRA SmartView

ADISRA SmartView Software is available for download from https://adisra.com/.



2.1. Download Step by Step

1. Go to https://adisra.com/
2. In the menu, click on the "DOWNLOAD ADRIRA® SMARTVIEW" button
3. Fill in the form with the requested data then click on the "Submit" button
4. Click on the "ADISRA SmartView 4.1.X.X" link for the appropriate OS

5. After the browser completes the download, extract the software from the downloaded ZIP file.
6. Select the file, right-click and select Extract Files. Choose the location for the extracted software files.

3. Installing ADISRA SmartView

To install ADISRA SmartView, we recommend that the user's system meets the following requirements to optimize its performance.

3.1. Hardware Requirements

Minimum:

- CPU: 1.44 GHz
- RAM: 4 GB
- Hard Disk (min): 8 GB

Recommended:

- CPU: 1.7 GHz
- RAM: 6 GB
- Hard Disk(min): 20 GB

.

3.2. Software Requirements

3.2.1. Windows OS

- Windows 11
- Windows 10
- Windows 8.1
- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012 (64-bit edition)
- Windows Server 2022

3.2.2. Linux OS

- Arch Linux (CLI)
- Elementary OS (GUI)
- Zorin OS (GUI)
- Lubuntu (GUI) - Ubuntu-based
- Pop!_OS (GUI) - Ubuntu-based
- Alpine Linux (CLI)
- Alpine Standard (CLI)
- Alpine Virt (CLI)
- CentOS (Both CLI and GUI)
- Debian (GUI)

- Linux Mint (GUI)
- Ubuntu Desktop (GUI)
- Ubuntu Server (CLI)

3.2.3. NET Runtime

- The application requires the .NET 6 runtime
- .NET 6 is already self-contained in the ADISRA SmartView Linux installer, so there is no need to install it
- If the machine on which the software is being installed does not have the ".NET 6" prerequisite, it will be necessary to install it manually. See "How to Install ADISRA SmartView Linux RunTime" document for further information.

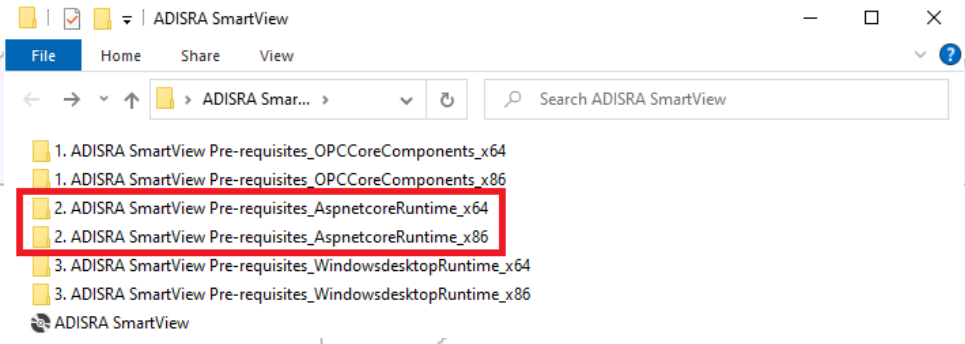
3.2.4. OPC

- The ADISRA SmartView installer™ will verify that the machine on which the software is being installed has the OPC Core component. If any software prerequisites need to be installed (except Windows operating systems), the installer will install the necessary requirements.

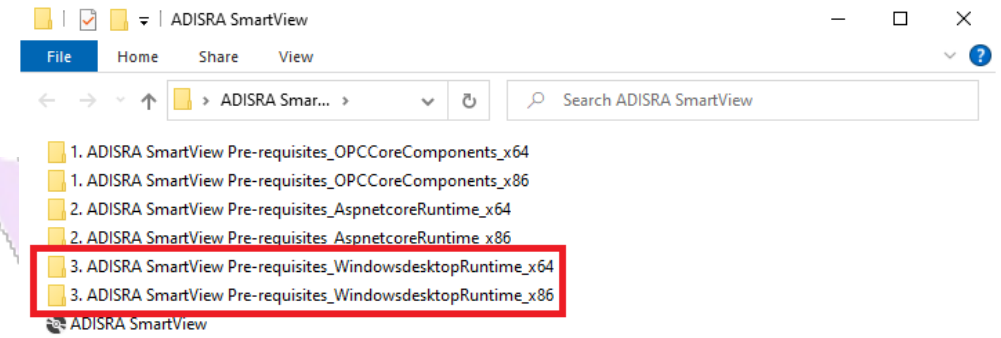
3.3. Installation Step by Step for Windows

The Installation Step by Step below is for a Windows OS. For the Linux OS Installation Step by Step please see "How to Install ADISRA SmartView Linux RunTime" document for further information.

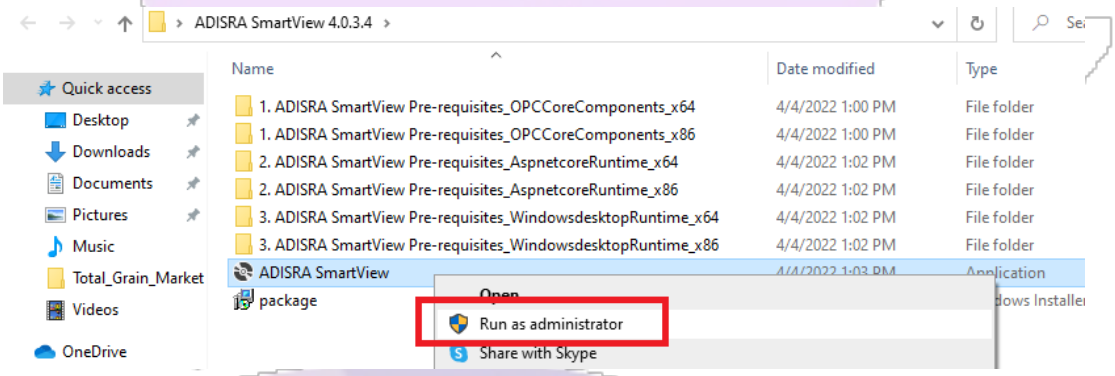
1. Install the ASPNET Core Runtime (x64 or x86) located in the installation folder;



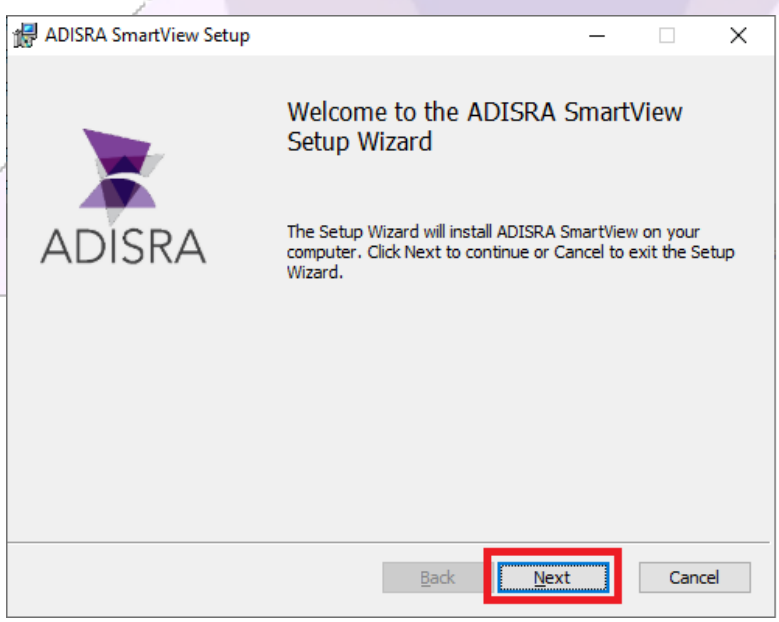
2. Install Windows Desktop RunTime (x64 or x86) located in the installation folder;



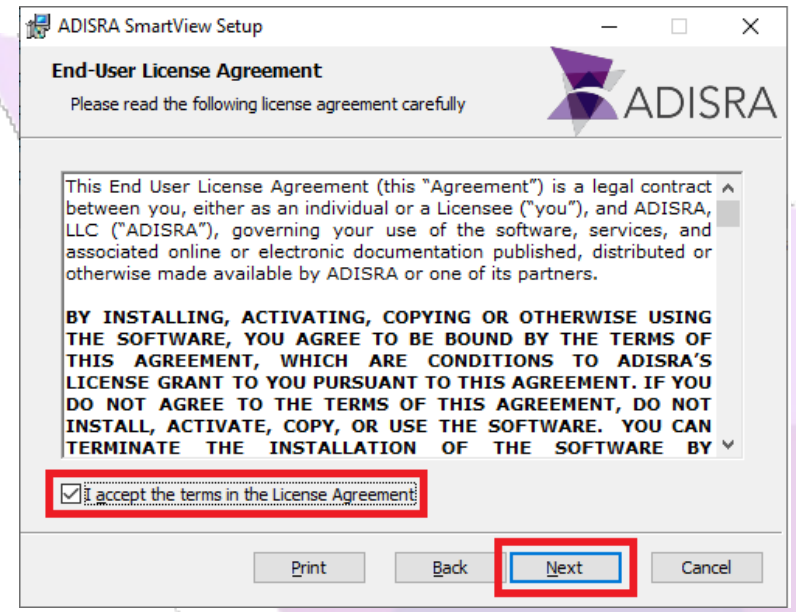
3. After installing the above frameworks, click the "ADISRA SmartView.exe" executable to install ADISRA SmartView. Run the installer as an administrator;



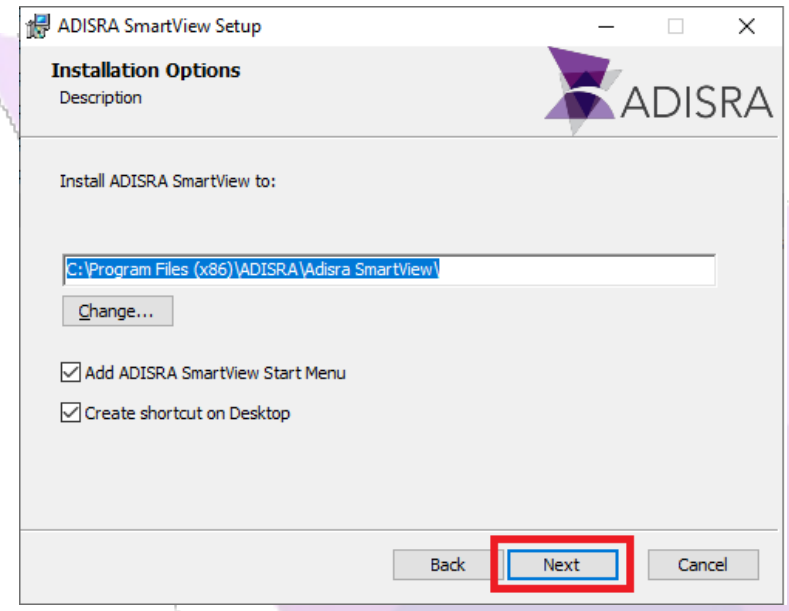
4. Click “Next”;



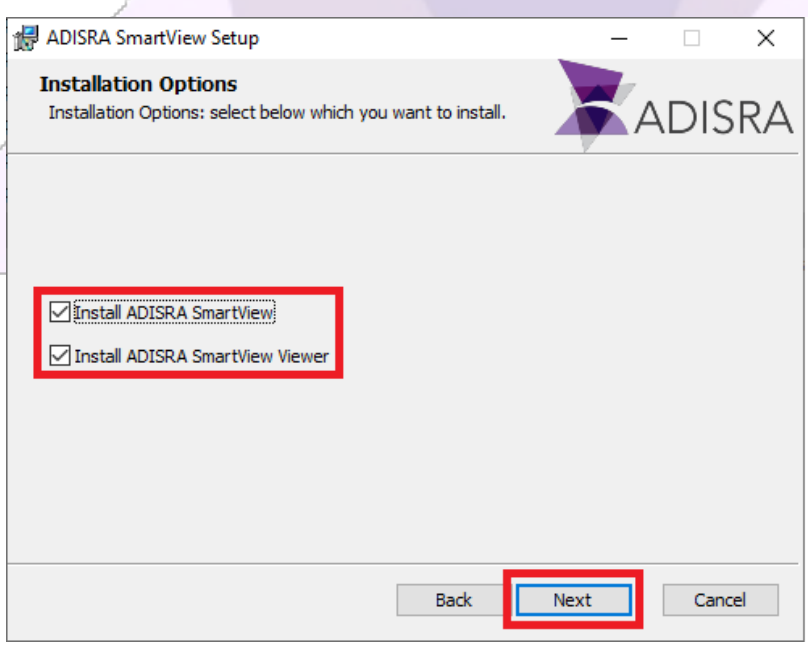
5. Read the terms. If you agree, select the option “I accept the terms in the License Agreement” and click “Next”;



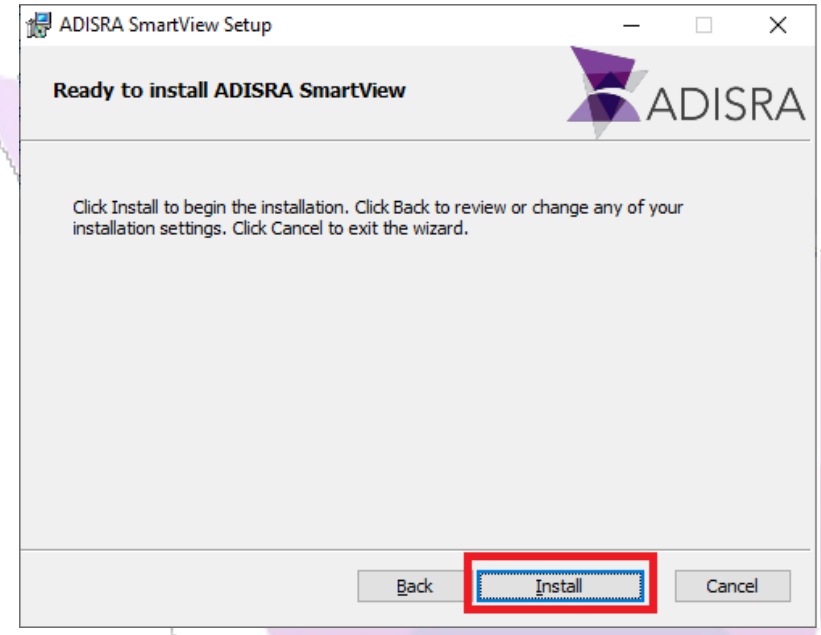
7. Choose the location where the ADISRA SmartView will be saved. We recommend keeping the path and checkboxes at the default.



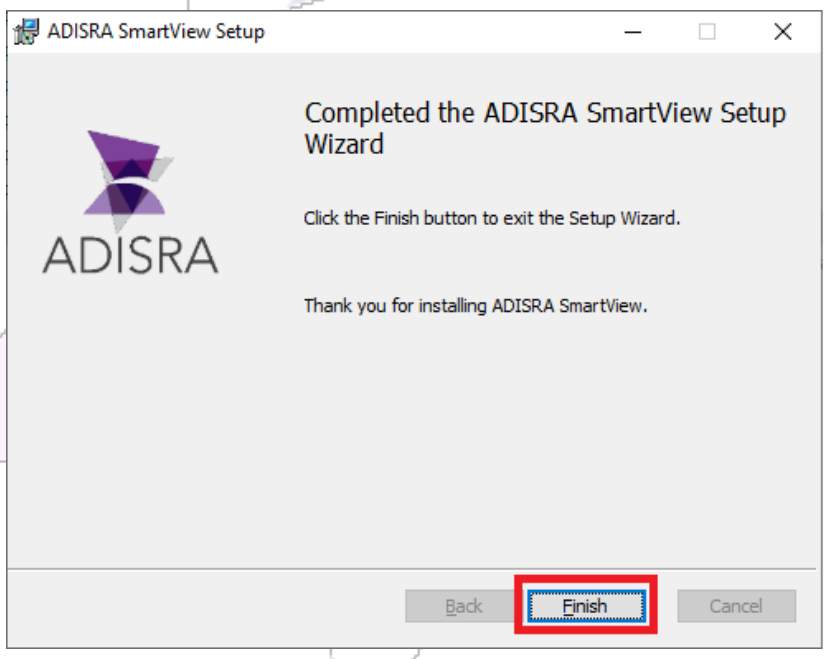
8. Choose which modules will be installed. For the RunTime machine, we recommend keeping the checkboxes as default. Therefore, the "Install ADISRA SmartView" option will install the "RunTime" and "Engineering" module and the "install ADISRA SmartView" option will install the "Viewer" module. Then click on "Next".



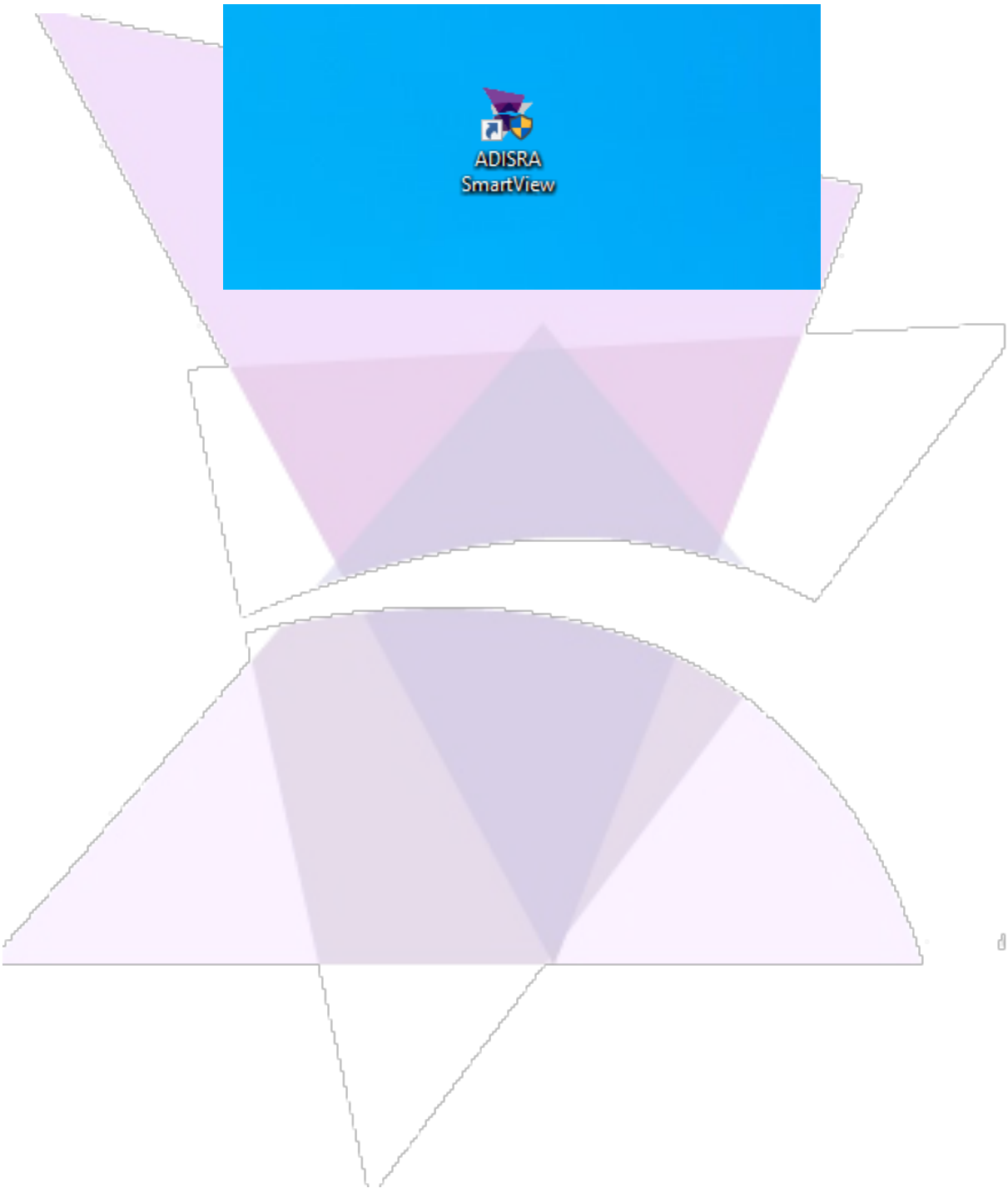
9. Click “Install” to start the installation.



10. Installation completed! Click "Finish" to close the installation wizard.



11. The ADISRA SmartView shortcut has been created on the desktop. This shortcut will open the engineering environment, which will be the module used to develop the application.



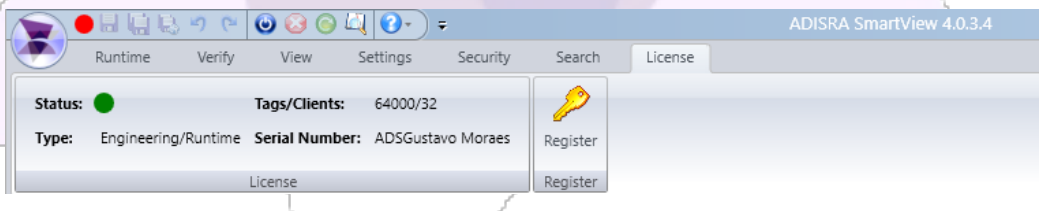
4. Registering an ADISRA SmartView License

The ADISRA SmartView license authorizes a single machine to execute the ADISRA SmartView software. The single license cannot be used on multiple machines since it is associated with the unique components of the specific machine that issued the licensing code.

The license code is generated using a machine's hardware identification, which is based on 3 hardware components: CPU, HDD, and Mother Board. The component-based license ensures that a single license will only enable the software to work on a specific machine. If the user plans to change any of the three hardware components, we recommend the user contact ADISRA to deactivate the license and then have it reactivated after the hardware component has been replaced.

The ADISRA team securely generates each license based on a license code generated by the customer. The license generation steps are explained in detail later in this document. A License can be deactivated at any time, but if it is deactivated, the customer must contact ADISRA to reactivate the license.

Information about your license is available on the License ribbon. The information available is the license status, license type, the maximum clients and tags number, and the license serial number. The license status is represented by a circle color: green means activated, yellow means trial version, and orange means there are five days remaining until trial expiration.



4.1. Types of Licenses

The License System has four license types:

Trial License (TRIAL)

This license is provided upon installation of ADISRA SmartView. After 30 days, the license will expire, requiring at least one of the licenses below. This

license allows access to the ADISRA SmartView engineering environment™ to develop the application using all available resources. After the application is developed, it is possible to run the application in RunTime mode. However, there is a time limit of 2 hours, that is, the RunTime mode will be closed if it exceeds the time of 2 hours.

Engineering License (ENG)

This license allows access to the ADISRA SmartView engineering environment to develop the application using all available resources. After the application is developed, it is possible to run the application in RunTime mode. However, there is a time limit of 2 hours, that is, the RunTime mode will be closed if it exceeds the time of 2 hours.

Engineering License and RunTime (ENG+RT)

This license allows access to the ADISRA SmartView engineering environment to develop the application using all available resources. After the application is developed, it is possible to run the application in RunTime mode with unlimited time.

RunTime License (RT)

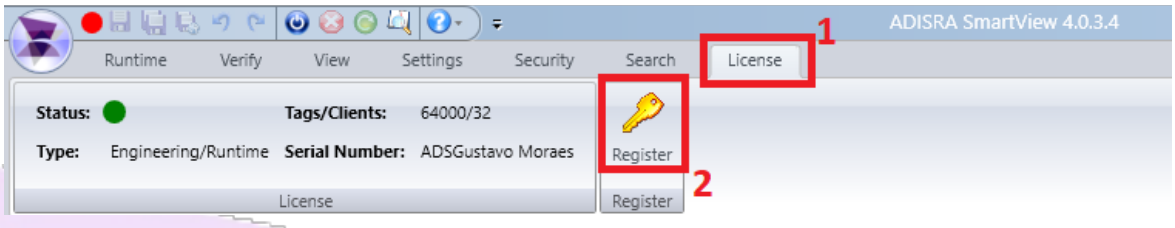
This license does not allow access to the ADISRA SmartView engineering environment to develop the application, just run the application in RunTime mode with unlimited time.

4.2. Registering a License

ADISRA SmartView gives the user a 30-day development trial when it is first installed. When the 30 days expire, a permanent license will need to be purchased to run the software. The instructions below will show the steps to register a new license:

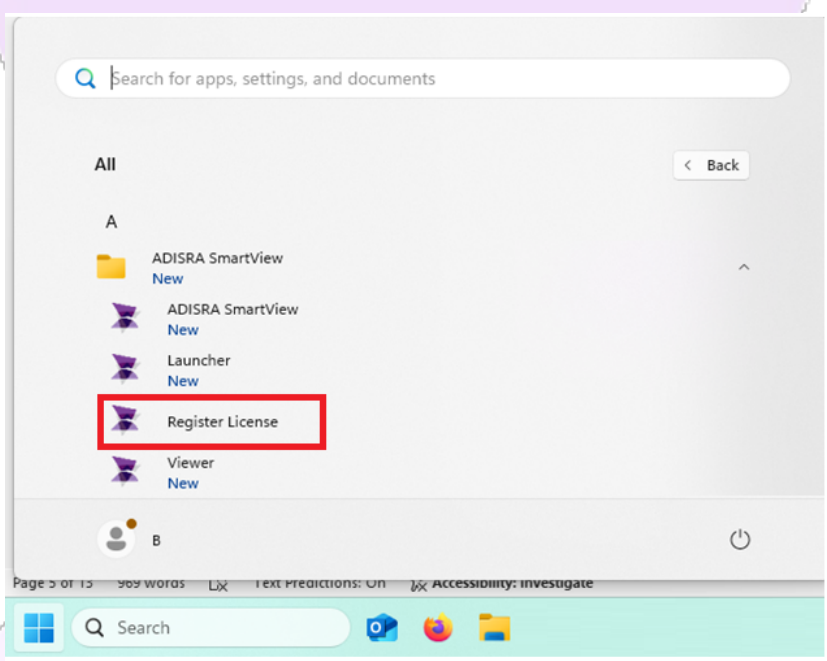
There are 2 ways to open the "License Register" window.

1. Select the License tab in the ADISRA SmartView banner and then the Register button:

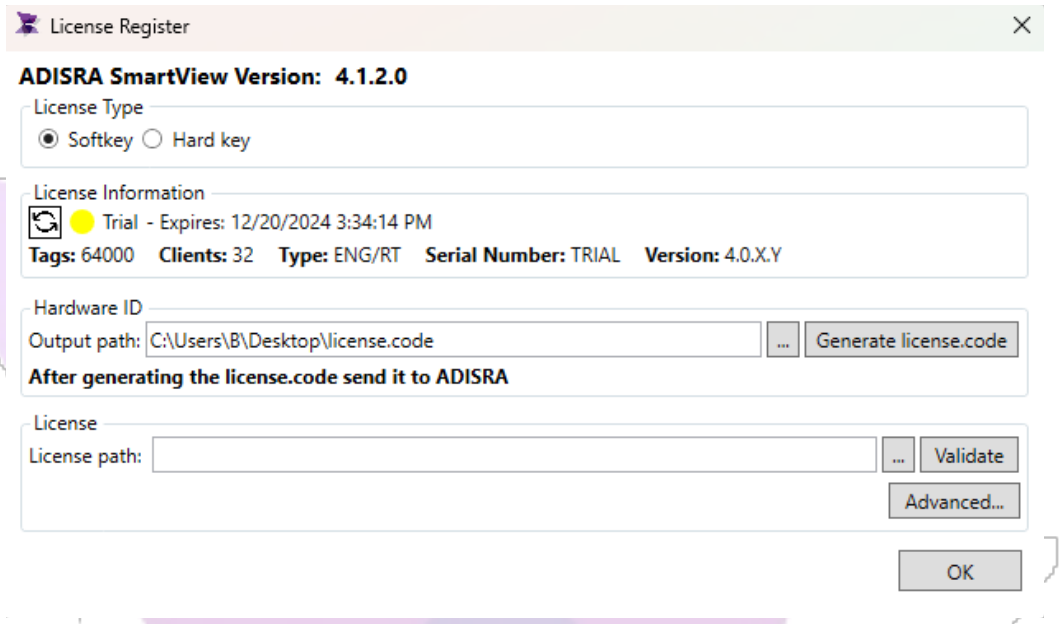


Or alternatively,

2. Go to the Windows toolbar: Start > All > ADISRA SmartView > Register License:



By using either method 1 or 2, the following window will appear.

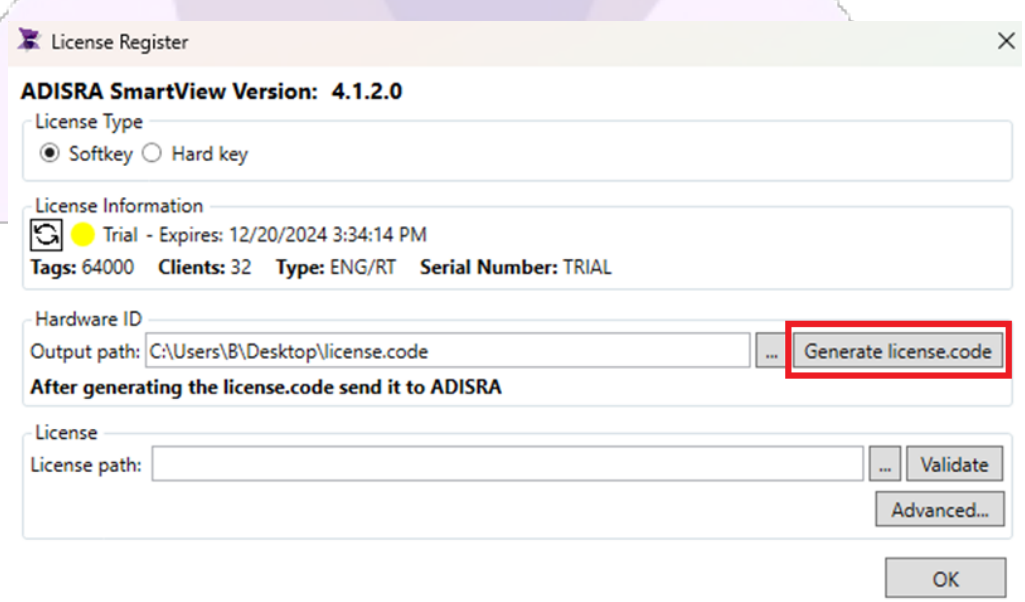


There are two license types: a Softkey or a Hard Key.

4.2.1. Softkey

The softkey involves receiving a digital license, a ".lic" file, from ADISRA. Follow the steps below to generate and validate the softkey license.

1. Click the button labeled "Generate license.code" to generate file containing this machine's unique Hardware ID information. The file location will be displayed in the text box called Output Path.



note: *The user can change the Output Path by clicking on the ... button next to the Output Path text box. Please do not change the file name.*

2. Attach the text file named "license.code" to an email and send it to info@adisra.com.

The ADISRA will verify your purchase and send back a License Key File that matches the Hardware ID. Download and save the license file to your storage drive. It is important to remember the license file location.

3. Enter the license key file location in License Path text box or browse to locate it by clicking on the ... button next to the License path and then click the Validate button.
4. You will be prompted to confirm the operation once the program accepts or validates your License Key.

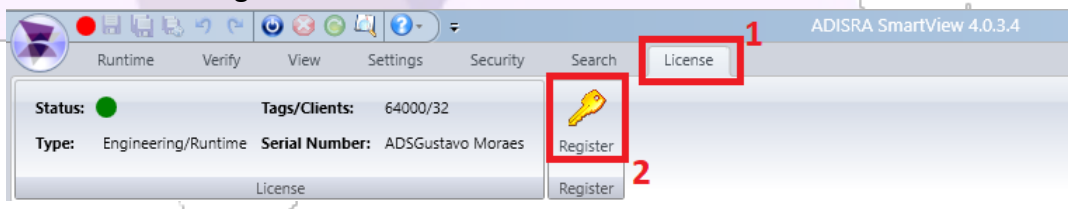
Your machine is now licensed and ready to use ADISRA SmartView

Close the Register License window and run ADISRA SmartView again.

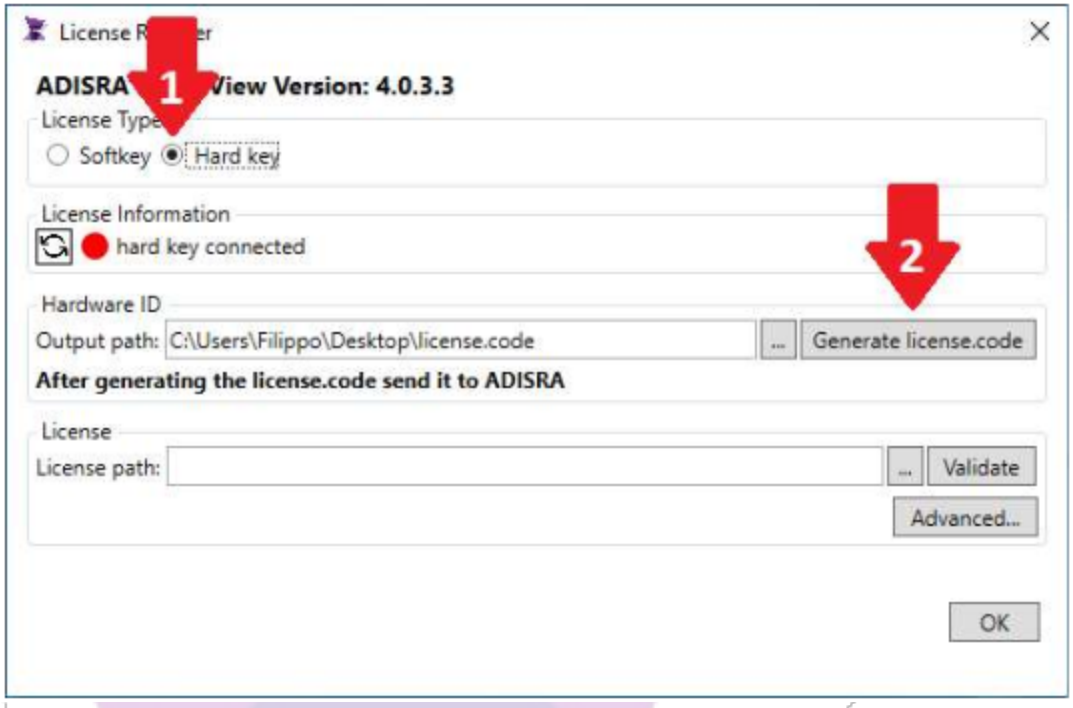
4.2.2. Hard key

The hard key license is used for a license located on an external removable device called a dongle. Follow the steps below to generate and validate the hardkey license.

1. Connect the dongle to the computer's USB port;
2. Launch ADISRA® SmartView and in the top menu, click on the "License" tab, then click the "Register" button.

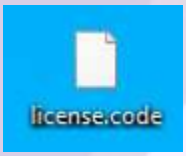


3. In the "License Type" field, select the "Hard Key" option and then click on the "Generate license.code" button.



note: *If you do not have a dongle attached to the computer the License Information will show "No hard key connected".*

4. A file will be generated on the desktop called "license.code"

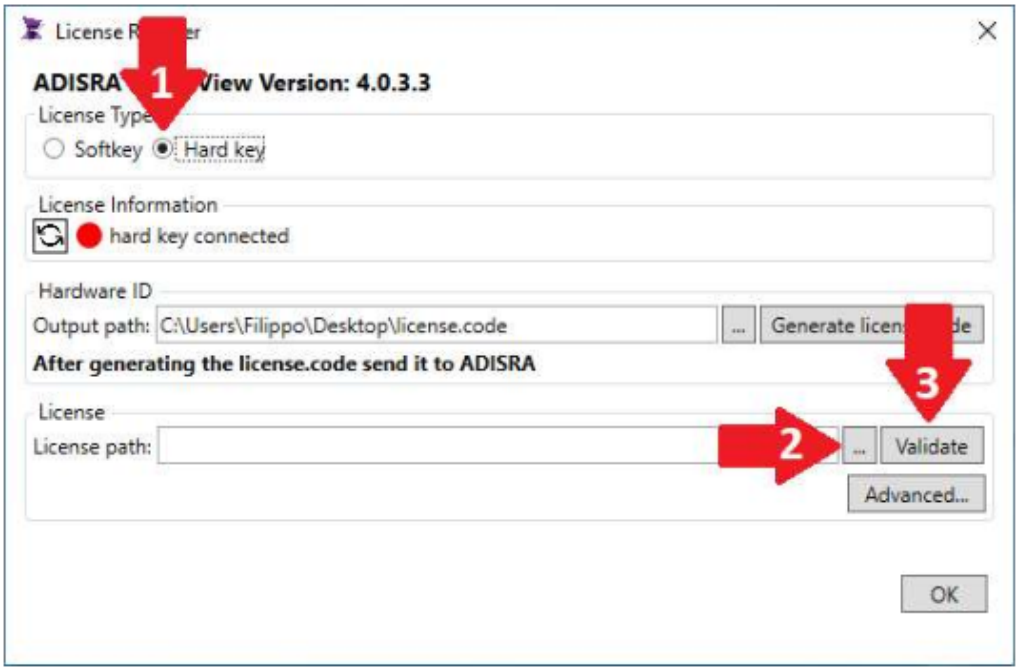


5. Send an email to "info@adisra.com" attaching the Hard key activation file (license.code). In the body of the email, identify the current version of ADISRA SmartView and the Serial Number of the license, which can be found in the certificate of authenticity. (see section 5)

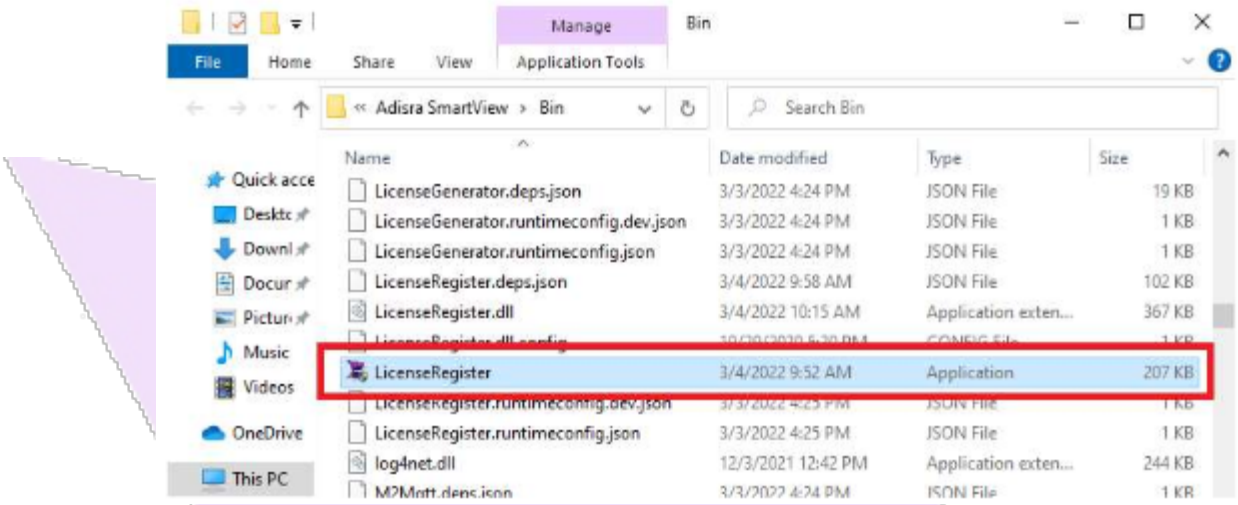
6. Once this is done, the ADISRA team will activate the Hard Key license. A file called "license.lic" will be sent to your email, in which you have the Hard Key activation code.

7. Now with the "license.lic" file in your possession, open ADISRA SmartView again. A window will inform you that the current license is not valid as the software is not licensed. Click "Yes" to continue.

8. In the “License Type” field, select the Hard Key option. Then click on the “...” button and point to the “license.lic” file. After clicking the "Validate" button, your license is ready to use. You have successfully validated your Hard Key.



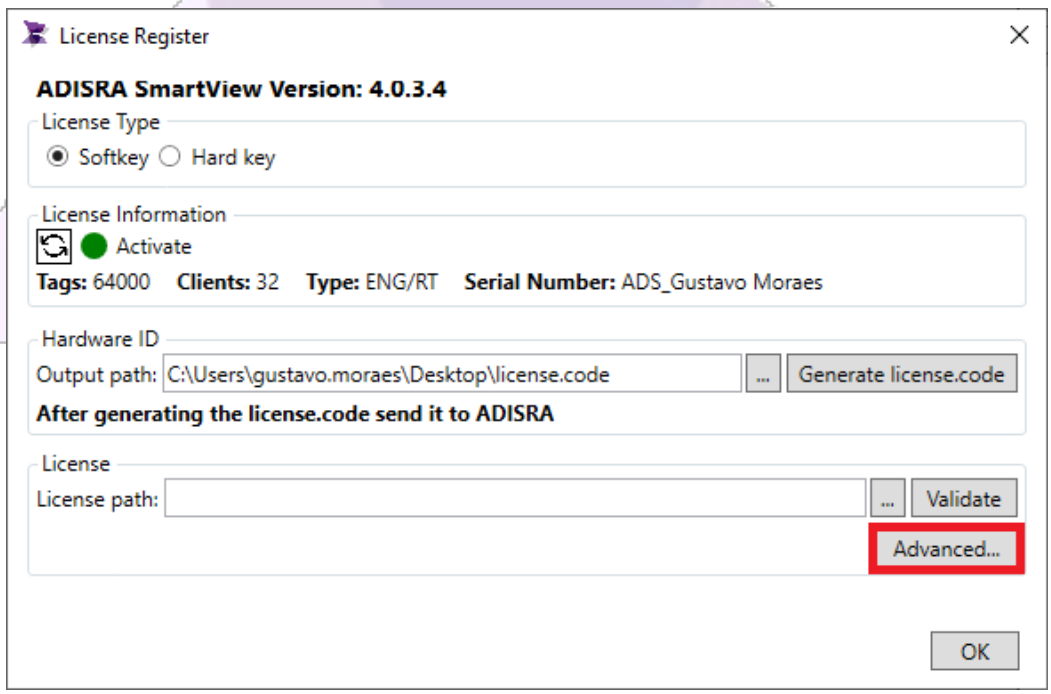
note: *It is also possible to access the above window through the executable “LicenseRegister.exe”, located in the “bin”.*



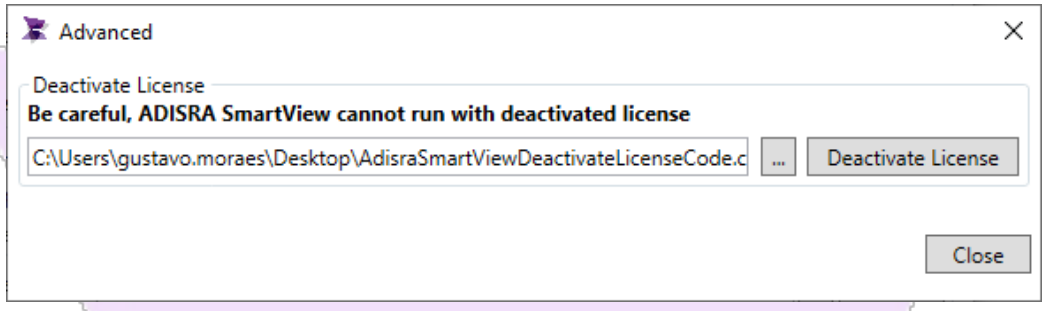
4.3. Deactivating a License

Each ADISRA SmartView Softkey license is registered to a specific machine. In order to use the Softkey license on another machine, it will be necessary to deactivate the current license. To execute the deactivation procedure, please follow the steps below:

1. In the License Register window, press the "Advanced…" button:



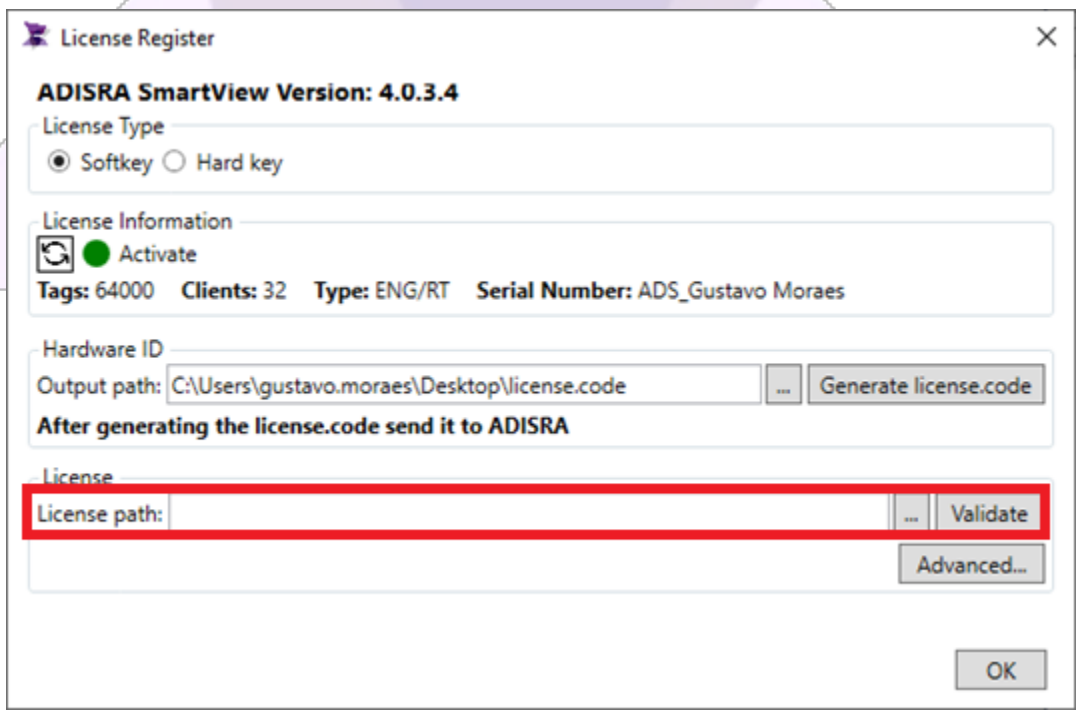
2. It will open a new window so you may decide where the deactivated license file will be generated:



note: The user can change the deactivated license file location by clicking on the ... button next to the "Deactivate License" button.

3. Click the button "Deactivate License" to generate the file.
4. Once deactivated, ADISRA SmartView will ask for the license again and will not work until a regenerated license is validated again.

To validate a regenerated license, send the deactivated license file to info@adisra.com and ADISRA will send a regenerated license back which you can save to the file location in License Path text box or browse to locate it by clicking on the ... button next to the License path and then click the Validate button.



4.4. Certificate of authenticity

When the user purchases a license for ADISRA SmartView, the software vendor/ADISRA will generate a certificate of authenticity for that license, this certificate contains the owner's name, or the company name, the email, along with the license's unique: serial number and the hardware ID.

See below an example of a certificate of authenticity:

note: *For our training, it will not be necessary to register any license, as the TRIAL license has all the permissions for the development of the Hypothetical Project.*

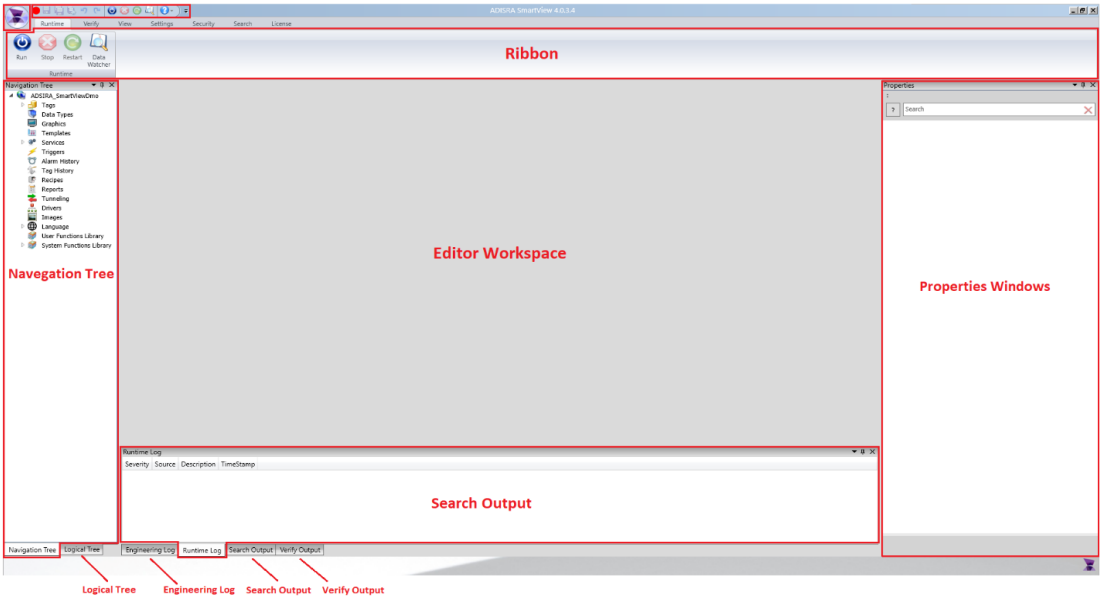
5. Knowing the Engineering Environment

To access the engineering environment of the ADISRA SmartView, run the shortcut "ADISRA SmartView" located on the desktop.

> ***note:*** *If this is the first time ADISRA SmartView has opened, the Engineering Environment will open the demo application. Otherwise, it will open with the last project that was used.*

5.1. Details of the Engineering Environment

The Engineering Environment of ADISRA SmartView is a user-friendly, easy-to-understand interface with many features. The Engineering Environment is where the user can configure the application development interface, using the most varied software resources.



The image above is a layout of the ADISRA SmartView engineering environment™ pattern. All the highlighted areas will be explained throughout the training.

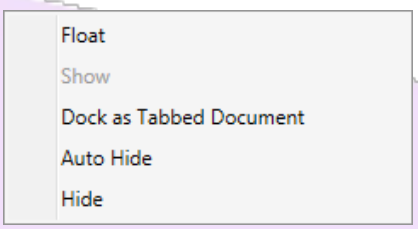
5.2. Organizing the Layout

The highlighted windows are part of the ADISRA SmartView interface, in which they can float above the program, be dragged beyond the edges of the program, or remain fixed in one of several locations. (By default, all windows are docked.)

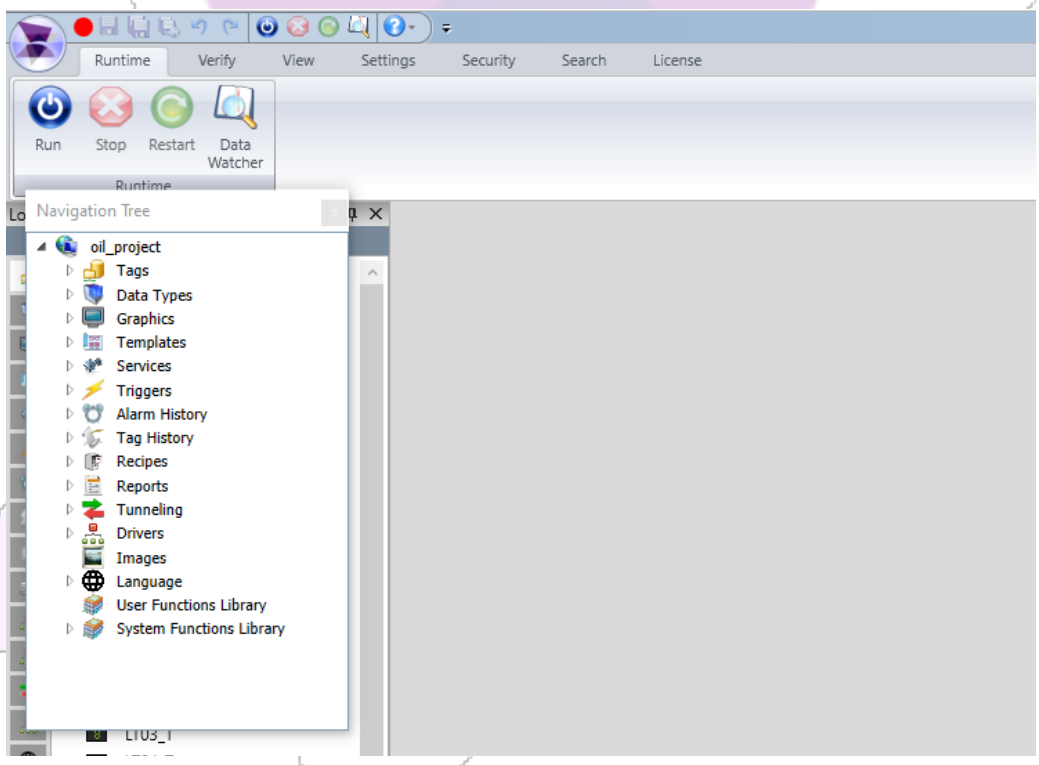
5.2.1. Floating Window

There are two ways to float a window in ADISRA SmartView:

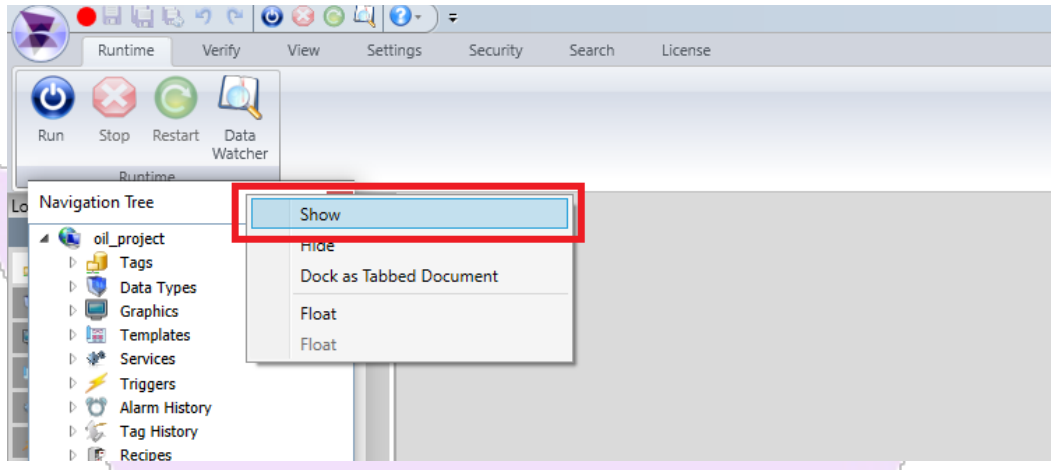
1. Click on the panel title bar and drag the panel;
2. Right-click on the title bar and select “Float”.



When selecting the option “Float”, the window will be separated from the main window of the software. The user can then drag the floating window to any location on the screen by clicking on the panel's title bar and moving the mouse.



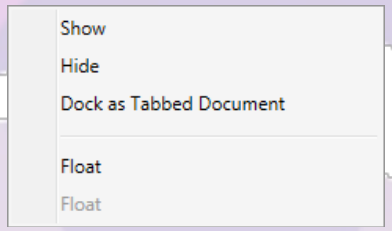
To return a floating window to its last docked location on the desktop, right-click on the title bar and select “Show”.



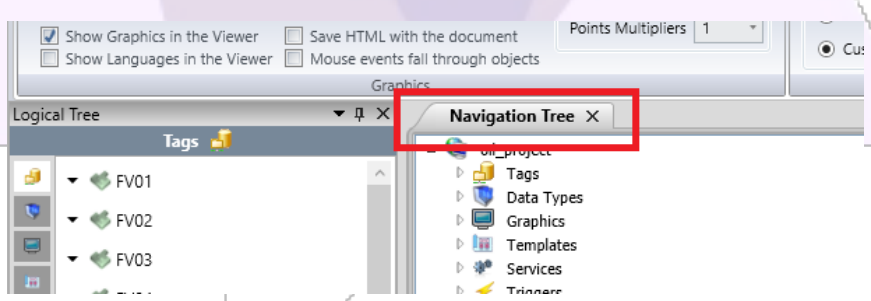
5.2.2. Docked Window

There are two ways to dock a window in ADISRA SmartView:

1. Right-click on the title bar and select “Show”. This will return the window to its last docked position.

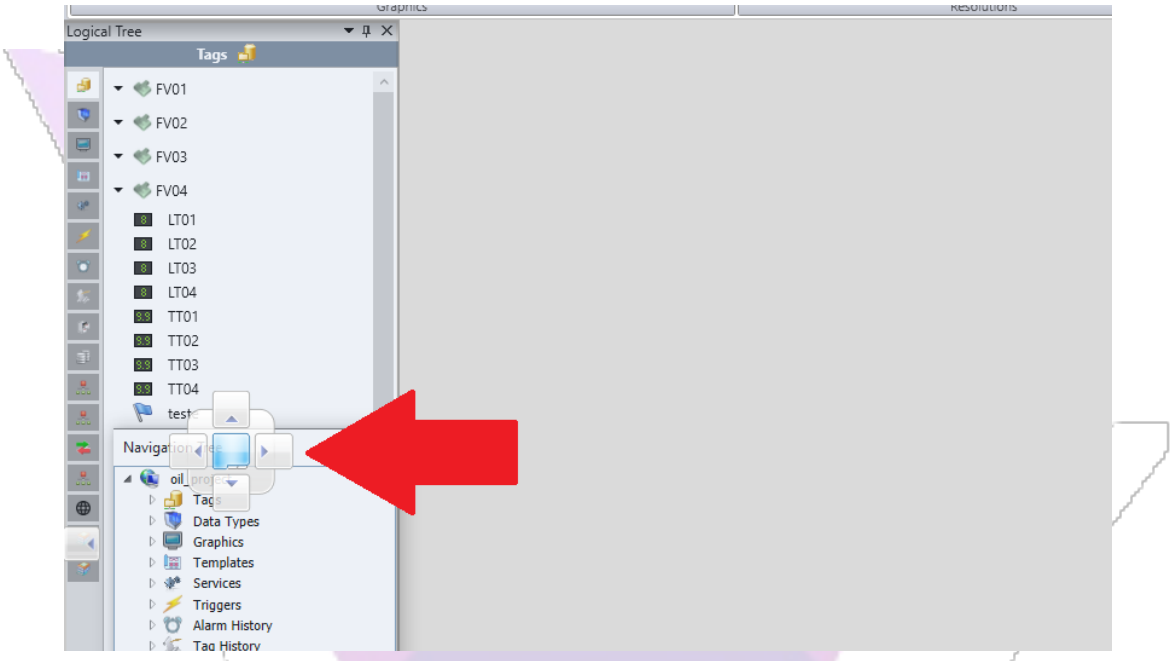


2. Right-click on the title bar and select “Dock as Tabbed Document”. This will return the window to the middle workspace as a tabbed document window (see below).



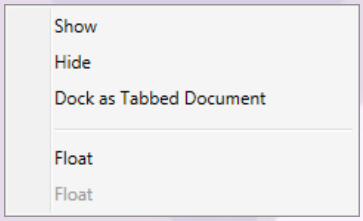
3. At any time, the user can move a window to any location using the “Docking Navigator”. To use the anchor browser, right-click the window’s title bar and

drag it to one of the locations indicated by the anchor browser that appears on the screen.

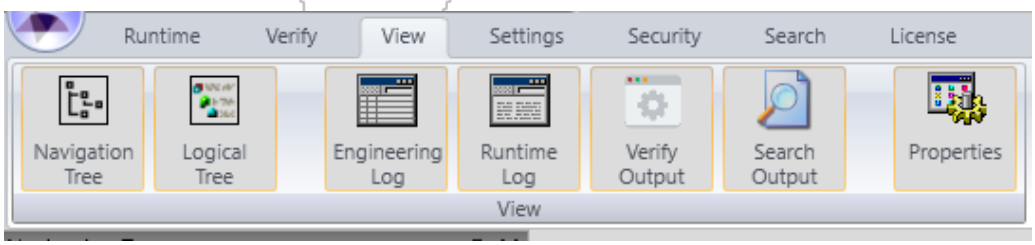


5.2.3. Hidden Window

To hide a window in ADISRA SmartView, right-click the title bar and select "Hide".



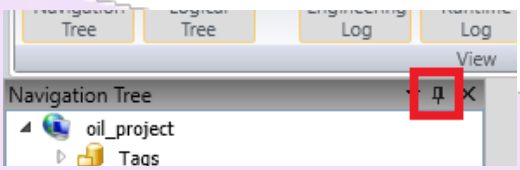
Whenever a window (Navigation Tree, Logical Tree, Engineering Log, Build Output, Runtime Log, Search Output, Properties) is hidden and the user wants it to be visible in the workspace again, he can select View from the top menu and click in the appropriate hidden window to view it again in the Preview Ribbon.



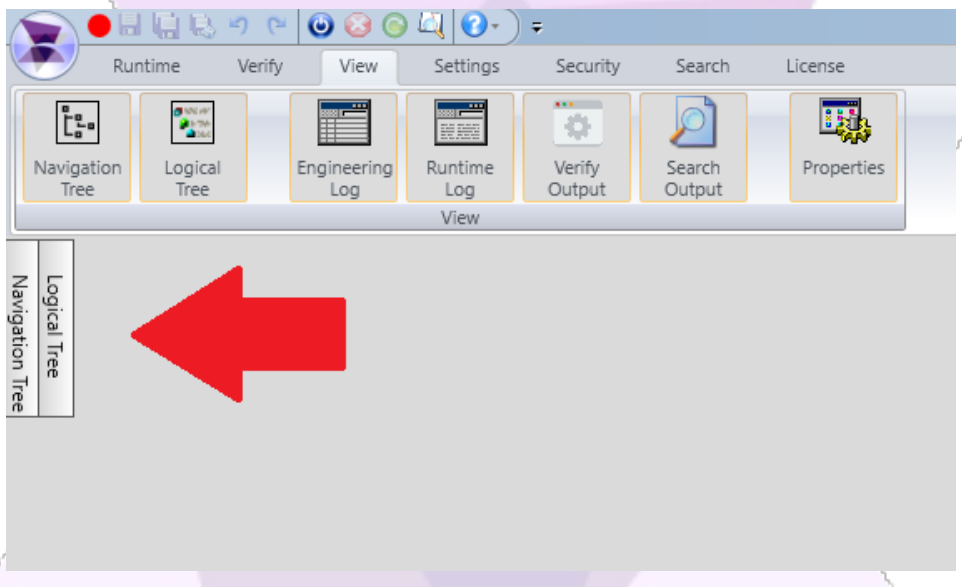
5.2.4. Auto-Hiding Window

There are two ways to automatically hide a window:

1. Right-click on the window's title bar and select Auto-Hide;
2. Click the thumbtack icon in the title bar.



Auto-hide a window reduces it to a tab to maximize the workspace as shown in the image below.



6. Introduction to Training

6.1. Overview

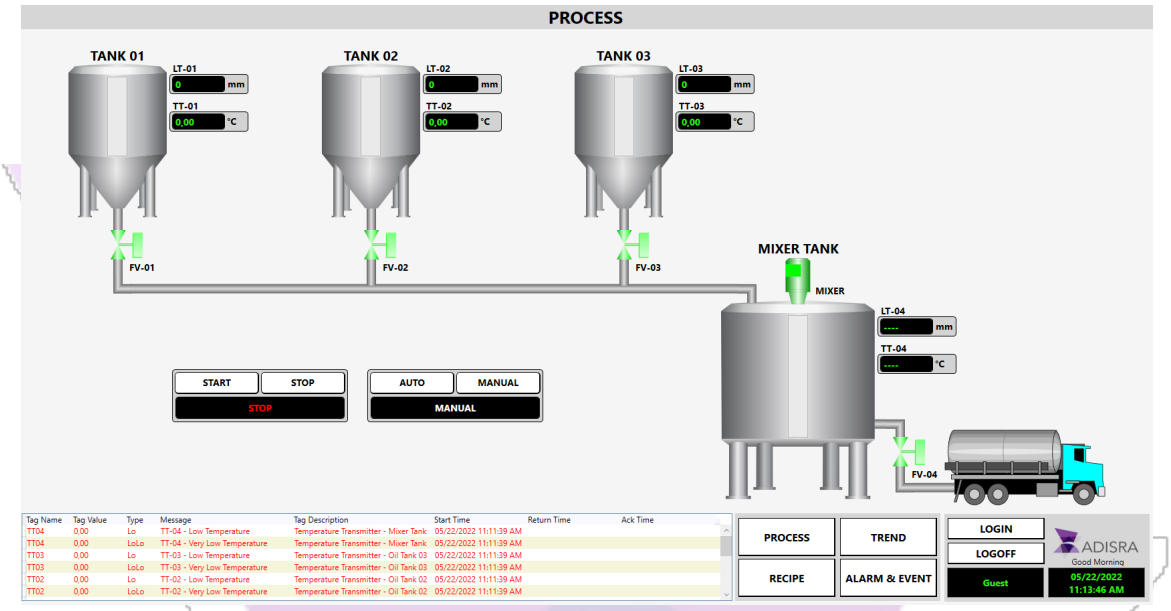
The following training aims to show and guide a student on how to use the main features of the ADISRA SmartView software for the development of SCADA Software. The training will be based on a hypothetical project, which will simulate a real application in the oil industry.

The oil industry is composed of 3 storage tanks and 1 mixing tank. At the outlet of each tank, there is a check valve, totaling 4 valves. Each tank will provide Level and Temperature information.

The following are the general screens of the SCADA software which we will develop using ADISRA SmartView.

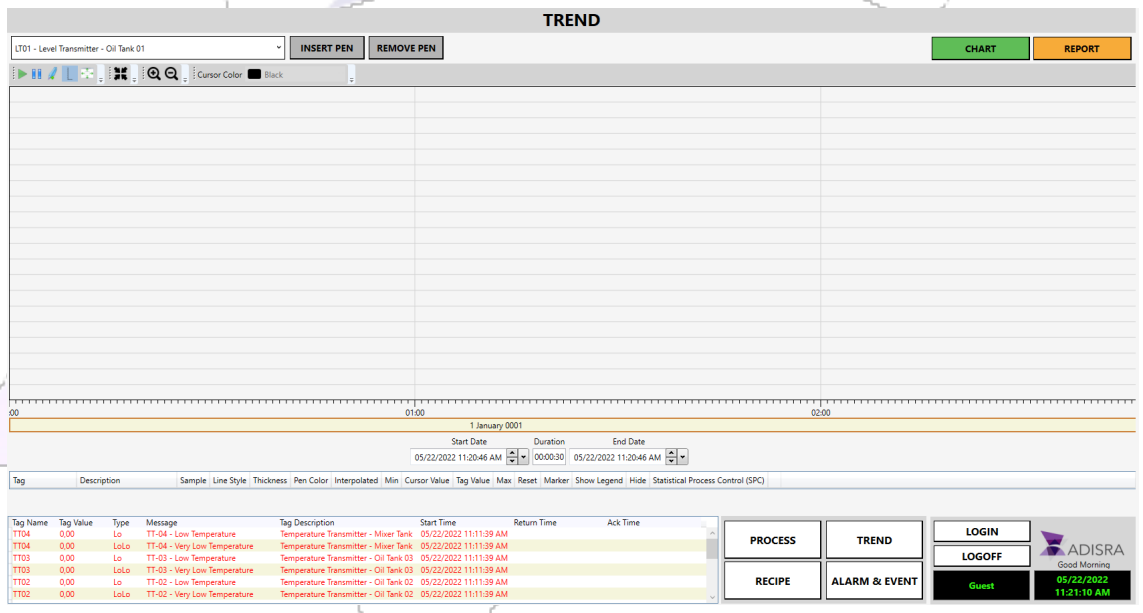
6.1.1. Process Screen

This screen will be responsible for showing the operator the main variables of the industrial process, such as level and temperature of each tank, open or closed valve status, mixer motor status, process status, and activity in Automatic or manual. At the top of the screen, the title of the current screen is shown. At the bottom, the summary of alarms, navigation buttons, user login and logoff and system date and time are shown.



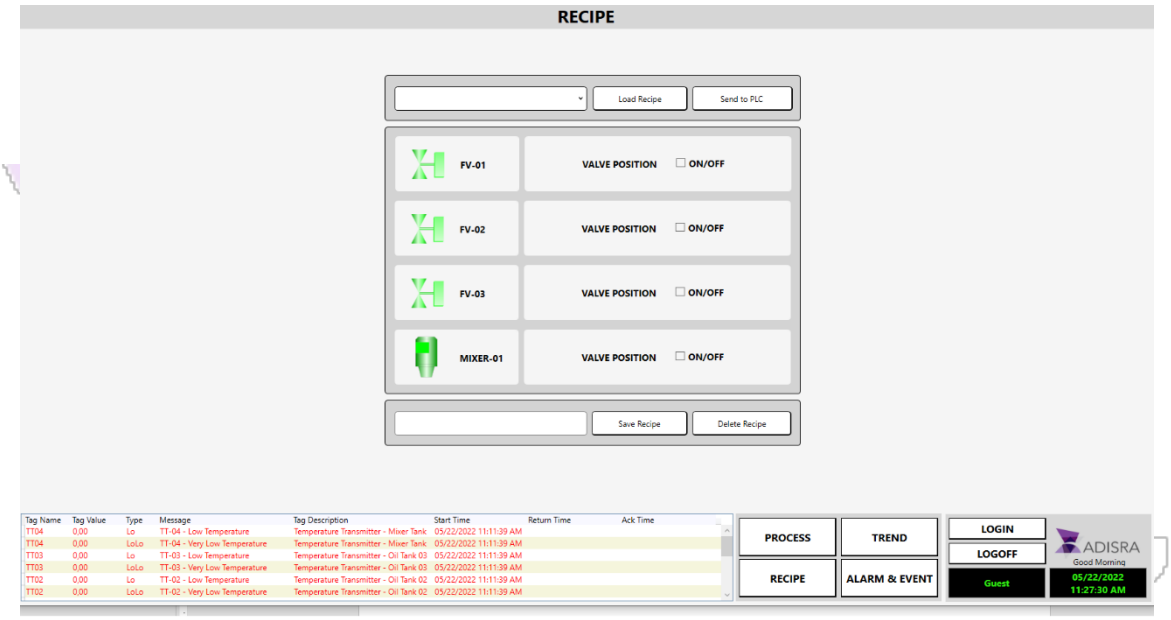
6.1.2. Trend Screen

This screen will be responsible for showing the operator, graphically, the main variables as a function of time.



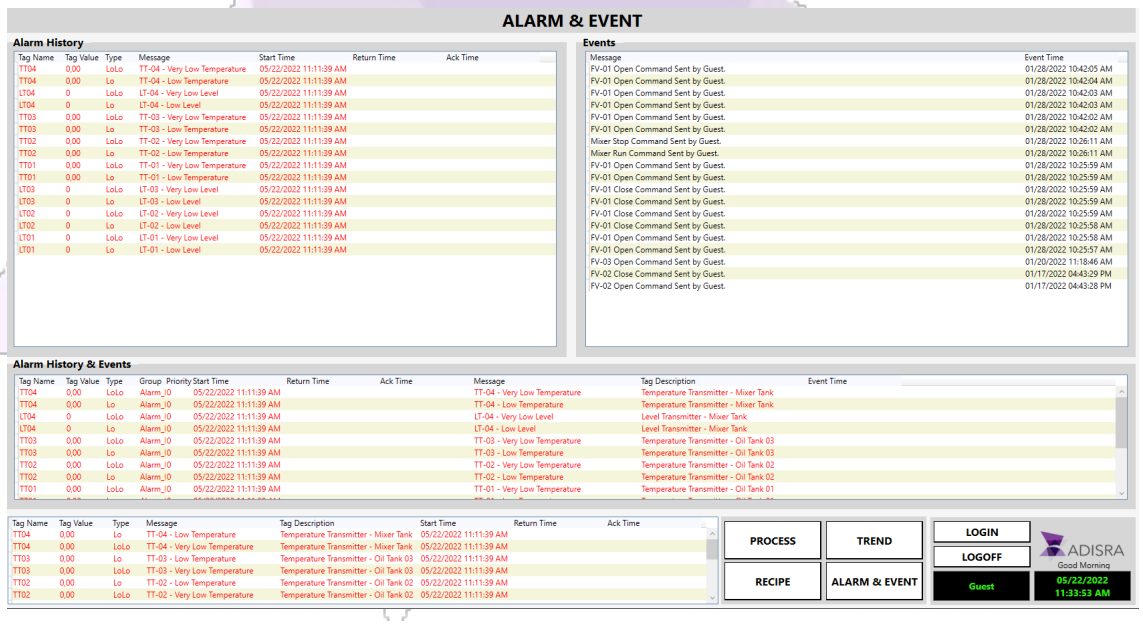
6.1.3. Recipe Screen

This screen will be responsible for allowing the operator to load and save recipes for the system's PLC.



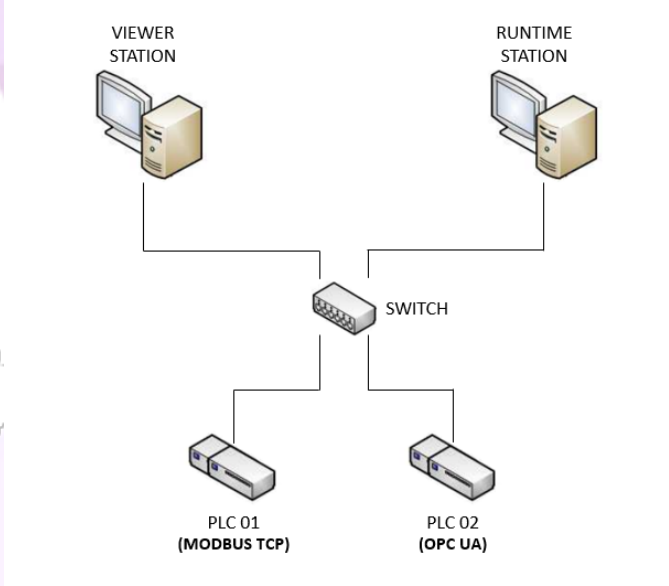
6.1.4. Alarm & Event Screen

This screen will be responsible for showing the operator the alarms and events of the application.



6.1.5. Application Architecture

The SCADA architecture of the Hypothetical Project contains two PLCs (simulated) collecting information from the industrial environment (Sensors and Actuators), in which the PLC 01 will communicate with the RunTime Station machine using the Modbus TCP protocol and the PLC 02 will communicate using the OPC UA protocol. The entire SCADA application will be developed in an engineering environment, and after the end of the development, the ADISRA SmartView must be installed on the RunTime machine where the application will be executed. On the Viewer Station machine, it will be necessary to install only the ADISRA SmartView Viewer.



6.1.6. Tag List

As with any industrial automation project, there is a document called the “List of Tags” or “List of Inputs and Outputs”, which aims to show all input and output variables (Analog or Digital) of the industrial application. For our hypothetical project we will have a list of tags containing all the communication tags with the simulated PLCs.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TCP	40001	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TCP	40002	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TCP	40003	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TCP	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TCP	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TCP	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TCP	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TCP	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TCP	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TCP	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TCP	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TCP	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TCP	7	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TCP	8	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TCP	9	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA	-	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_04_FAIL	(Fail) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

The tag list has the following columns:

PLC: Identifies which PLC the tag belongs to;

TAG: Identifies the logical name of the tag;

Description: Identifies the description of the tag;

Protocol: Identifies the communication protocol used;

Modbus Address: Identifies the tag's modbus address;

Type: Identifies the data type, which can be (Boolean, Float, or Integer);

Range: Identifies the tag measurement range;

Eng. Un.: Identifies the tag engineering unit;

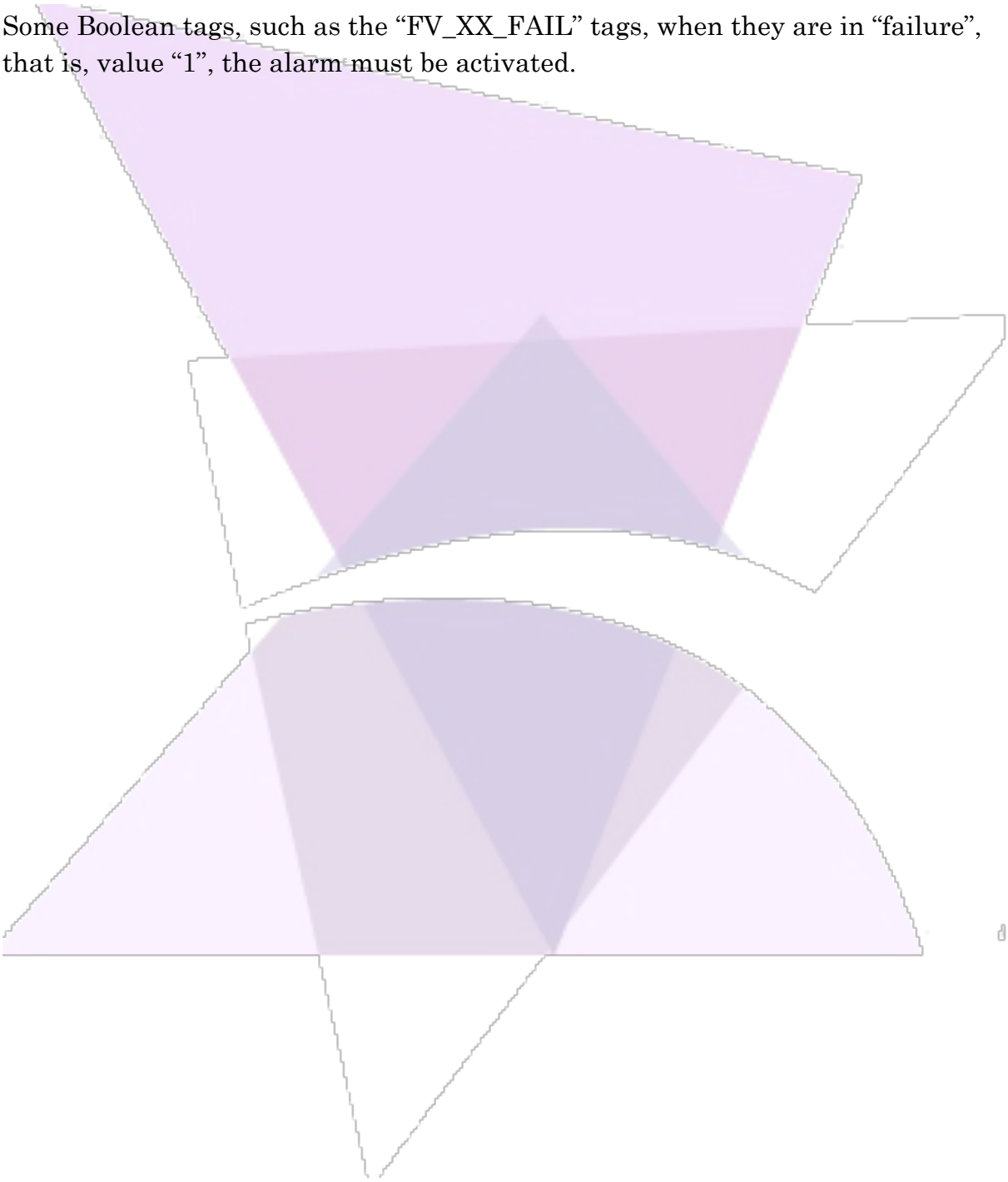
Alarms: Identifies the trigger values for alarm conditions (LL, L, H, and HH).

Example:

The LT_01 tag is a Level Transmitter, which belongs to PLC 01 (Modbus TCP Protocol), the modbus address is 40001 and is of the Integer type. The LT_01

measurement range is from 0 to 3000mm, and the alarm conditions must be configured (LowLow = 500, Low = 700, High = 2300 and HighHigh = 2800).

Some Boolean tags, such as the “FV_XX_FAIL” tags, when they are in “failure”, that is, value “1”, the alarm must be activated.



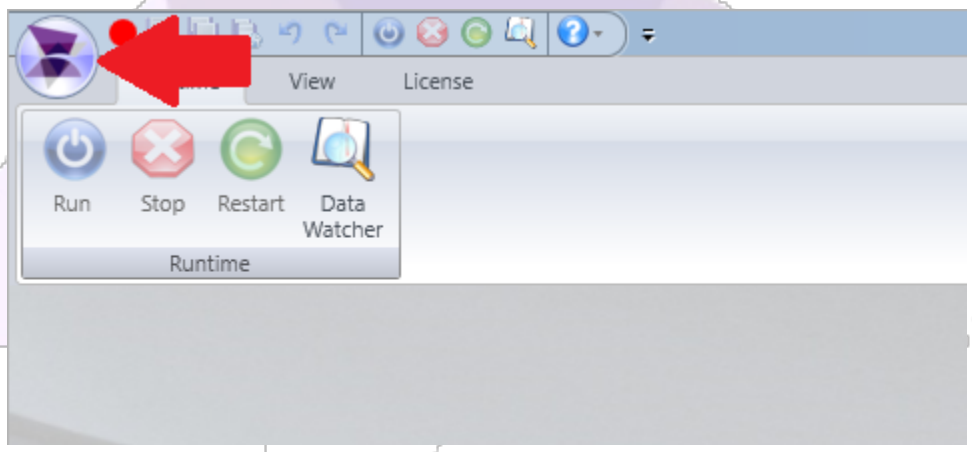
7. Starting the Project

7.1. Creating a New Project

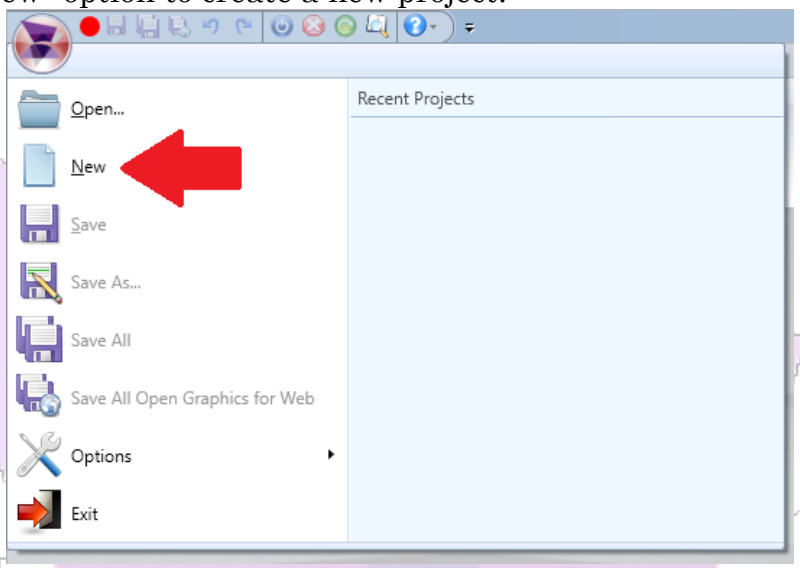
Now that we are familiar with the ADISRA SmartView Software, let's create our application based on the presented hypothetical project.

1. Click the “ADISRA SmartView” shortcut to open the development environment.

2. Click on the “Menu” button located in the upper left corner.

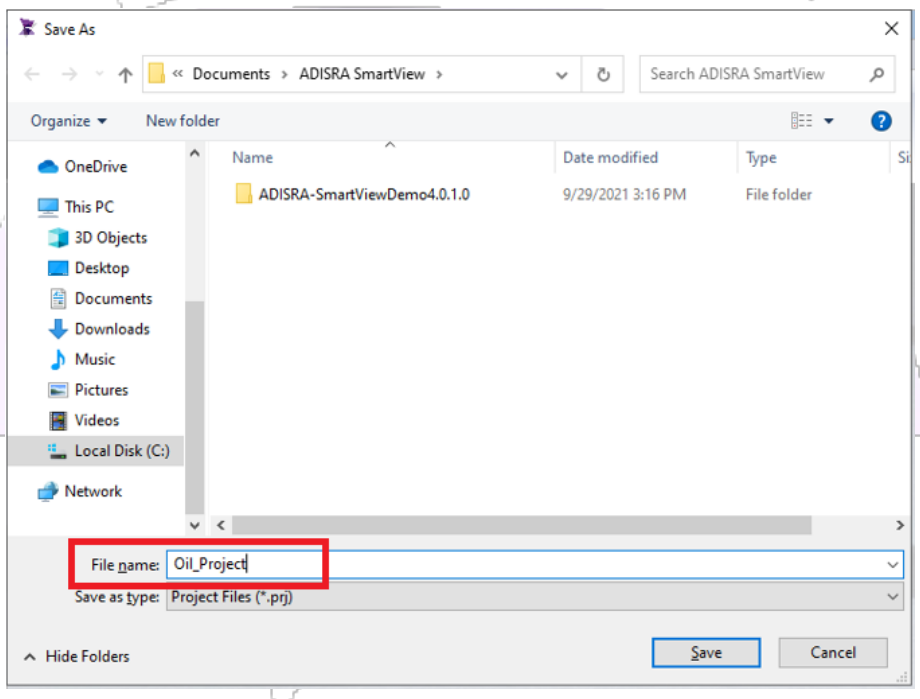


Click on the "New" option to create a new project.

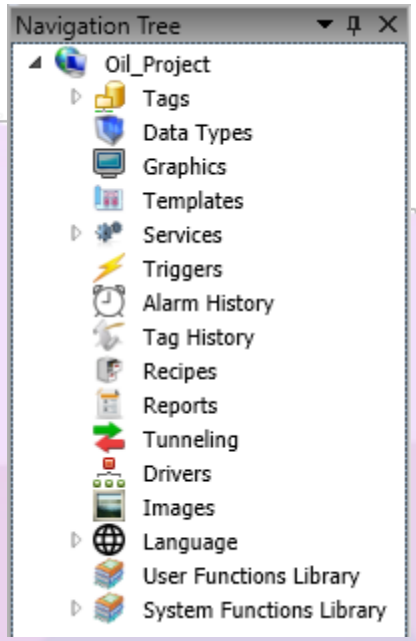


3. Choose the location where the project will be saved. Put the project name as "Oil_Project" as shown in the image below. By default, the project will be created with the extension (.prj). After naming the project, click on the "Save" button.

   note: *The project name cannot contain special characters or spaces.*



To start the project, the user must familiarize himself with the layout of the development environment. ADISRA SmartView displays the project structure in the Tree Navigation as shown below.

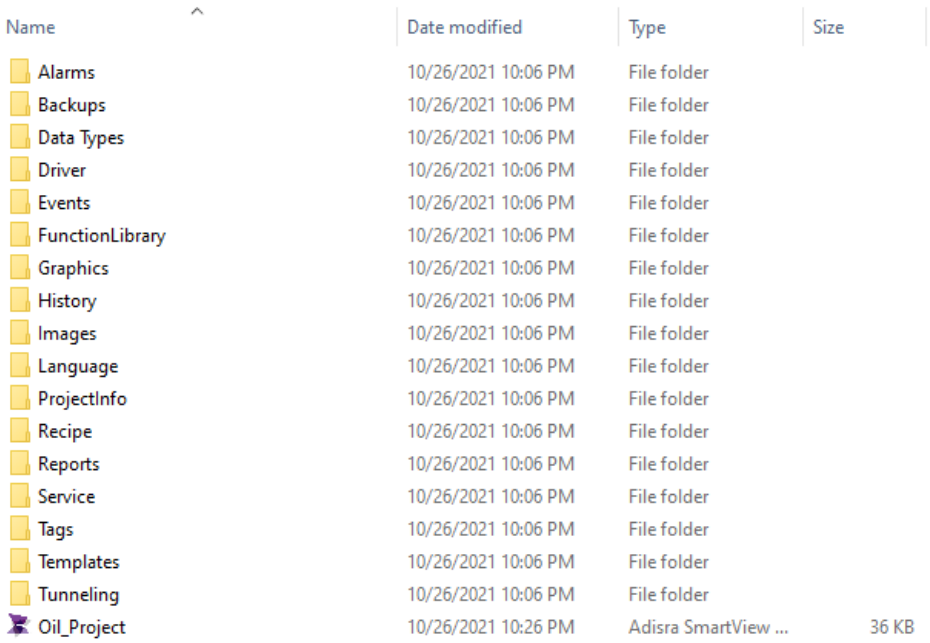


As can be seen in the image of the navigation tree, some documents are created automatically, such as Tags, Services, alarms, graphics, library of system functions documents and others. The image represents the basic structure of the tree navigation for any project.

We can start using these documents or create new documents and customize them.

The image below shows the project folder containing subfolders of each document, such as Tags, services, etc. As well as the startup file, called "Oil_Project.prj".

note: *If you followed the examples the project folder will be located in the "C:\Users\Documents\ADISRA SmartView\oil_project" directory.*



Name	Date modified	Type	Size
Alarms	10/26/2021 10:06 PM	File folder	
Backups	10/26/2021 10:06 PM	File folder	
Data Types	10/26/2021 10:06 PM	File folder	
Driver	10/26/2021 10:06 PM	File folder	
Events	10/26/2021 10:06 PM	File folder	
FunctionLibrary	10/26/2021 10:06 PM	File folder	
Graphics	10/26/2021 10:06 PM	File folder	
History	10/26/2021 10:06 PM	File folder	
Images	10/26/2021 10:06 PM	File folder	
Language	10/26/2021 10:06 PM	File folder	
ProjectInfo	10/26/2021 10:06 PM	File folder	
Recipe	10/26/2021 10:06 PM	File folder	
Reports	10/26/2021 10:06 PM	File folder	
Service	10/26/2021 10:06 PM	File folder	
Tags	10/26/2021 10:06 PM	File folder	
Templates	10/26/2021 10:06 PM	File folder	
Tunneling	10/26/2021 10:06 PM	File folder	
Oil_Project	10/26/2021 10:26 PM	Adisra SmartView ...	36 KB

7.2. Opening a New Project

If you need to reopen the “Oil_Project” project, follow the steps below:

1. Click on the “Menu” button.
2. Click on the “Open” option or press CTRL+O.
3. A new window will be opened. Navigate to the project folder and select the file with the “.prj” extension.
4. Then click “Open” button.

.

.

8. Document Tags

8.1. Overview

In ADISRA SmartView, a Tag refers to a unique identifier for a data point within the application, i.e. it is used to receive and write values in "RunTime" to and from a data source. In the Tags document, the user can create, remove and store the tags used in the project. Each tag has a property list, in which some properties are fixed and others editable.

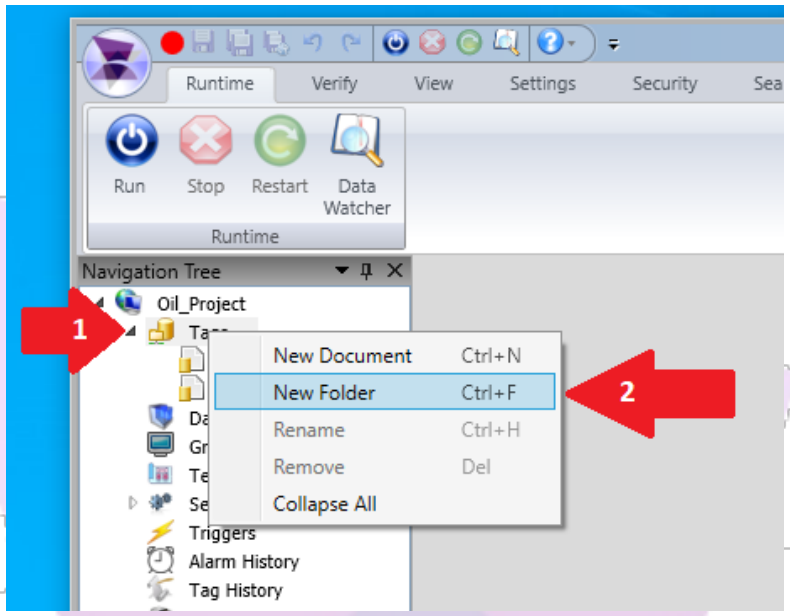
8.2. Creating Tags

Now that we have our application created, let's add the tags of our hypothetical project presented by the "Tag List" document as mentioned in the "Introduction to Training" chapter.

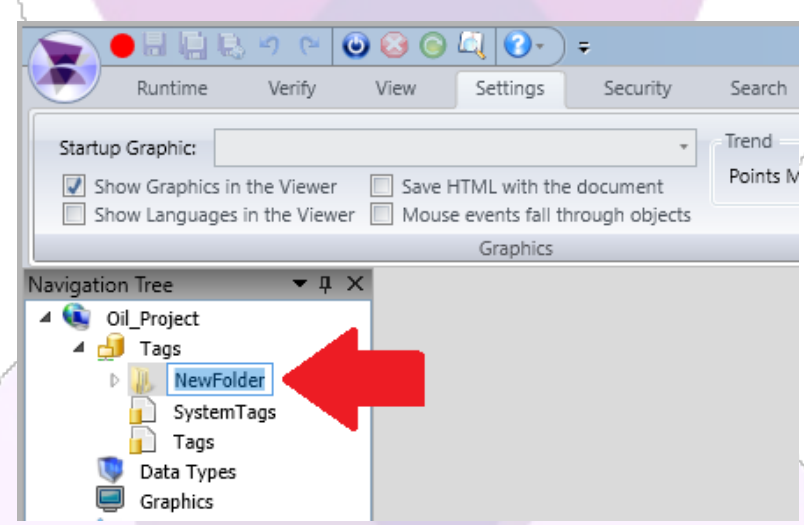
First, we will add the tags referring to PLC 01.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TCP	40001	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TCP	40002	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TCP	40003	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TCP	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TCP	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TCP	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TCP	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TCP	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TCP	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TCP	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TCP	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TCP	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TCP	7	BOOL	(Fail = 1 / Ok = 0)	-		Active Alarm		
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TCP	8	BOOL	(Fail = 1 / Ok = 0)	-		Active Alarm		
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TCP	9	BOOL	(Fail = 1 / Ok = 0)	-		Active Alarm		
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA	-	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_01_FAIL	(Fail) Flow Valve 01 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-		Active Alarm		
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-		Active Alarm		
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

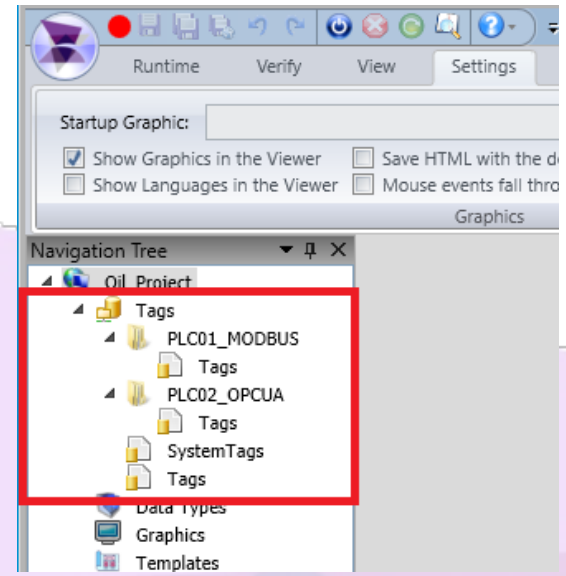
1. Right-click on the "Tags" Document and then on the "New Folder" option.



2. Change the name of the new folder to “PLC01_MODBUS”.

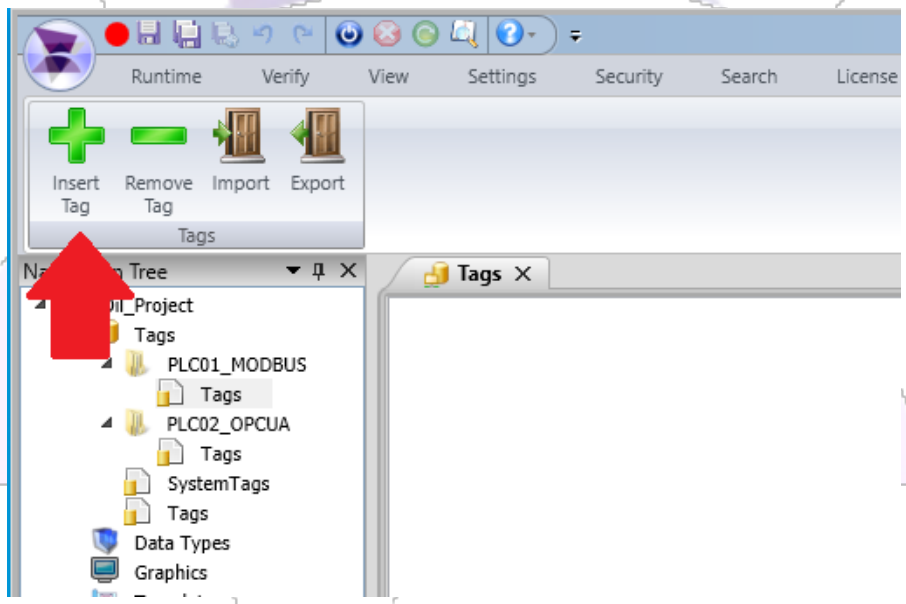


3. Create another new folder called “PLC02_OPCUA”. The organization of the folders should be as shown in the image below.

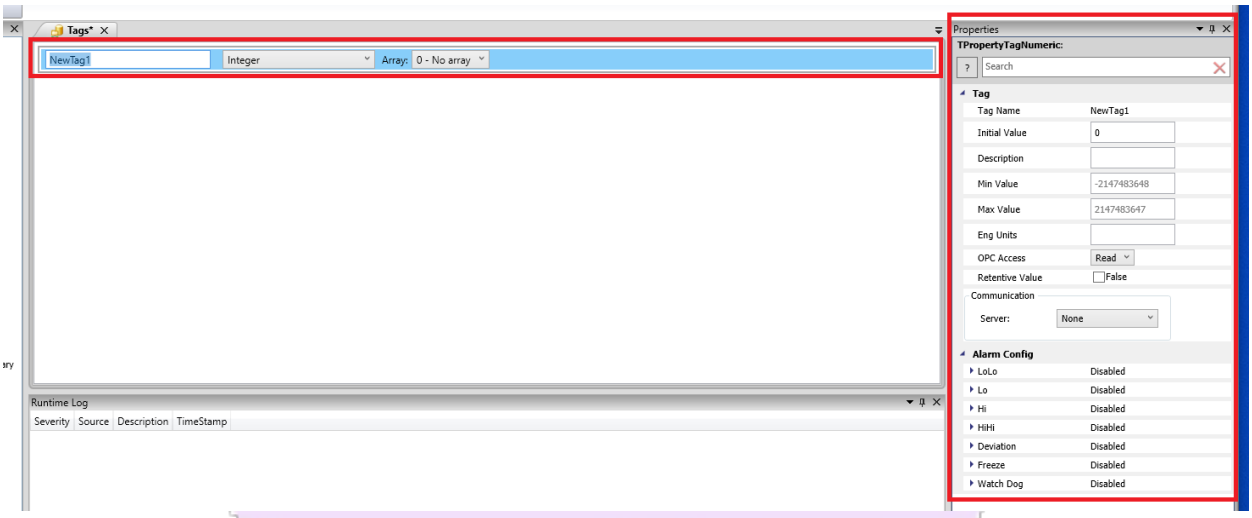


note: *After creating the folders, a group of tags called "Tags" is automatically created for each folder.*

4. Now, open the tag group in the "PLC01_MODBUS" folder and click on "Insert Tag" to insert the first PLC 01 tag.



5. Note that after clicking on "Insert Tag", a new line was inserted in the group, containing the tag name, tag type and array. On the right, we can see the properties of this tag. So, let's configure it according to the project's tag list. We will start with the tag "LT_01".

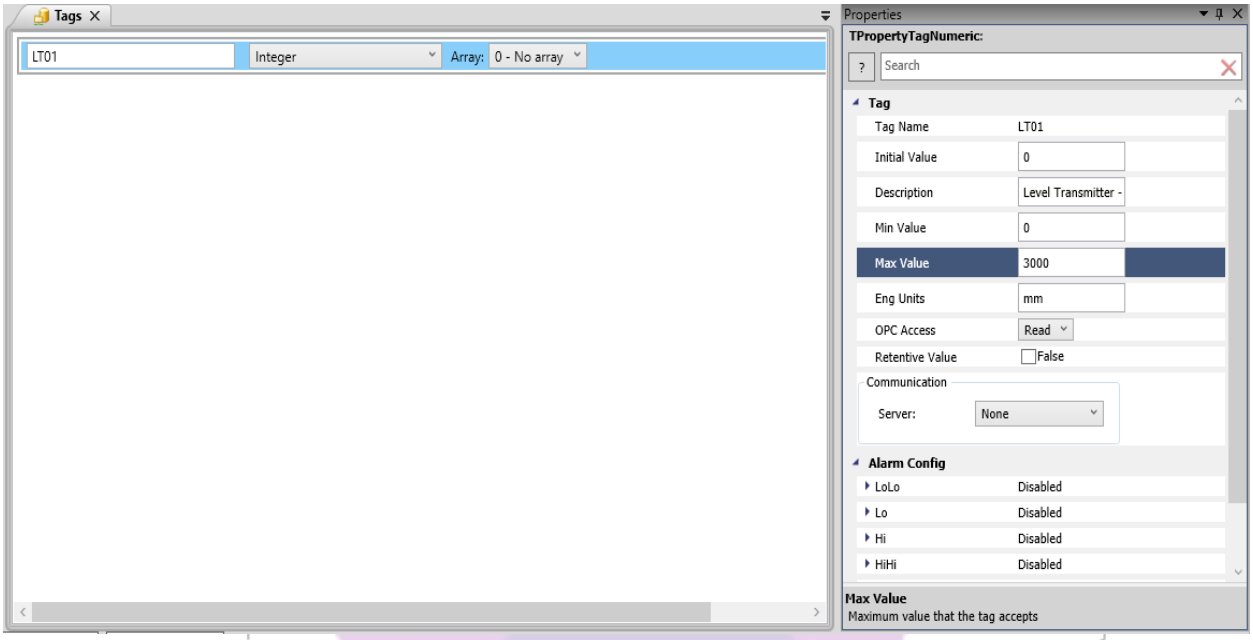


6. In this first tag called "LT01", the fields must conform to the information in the tag list.

- **Name:** LT01
- **Type:** Integer
- **Array:** 0 – No array
- **Description:** Level Transmitter - Oil Tank 01
- **Min Value:** 0
- **Max Value:** 3000
- **Eng Unit:** mm

note: *The Alarm fields will be configured in the alarms chapter.*

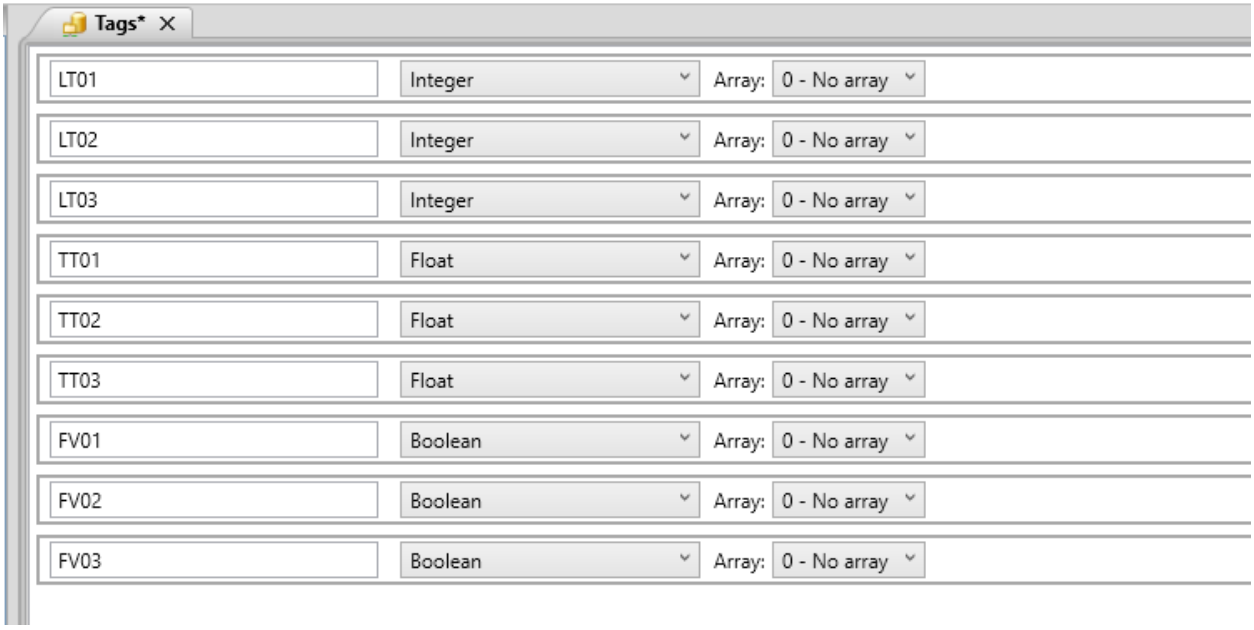
7. Check if the tag is configured according to the image below



8. Now, insert all tags referring to PLC01. The Tag Group should look like the image below.

Heads up: *The FV01, FV02 and FV03 valve tags must be created, but not configured, as shown in the image below. Because in the next chapter we will use the DataType feature. This feature is intended to facilitate the creation of repetitive tags.*

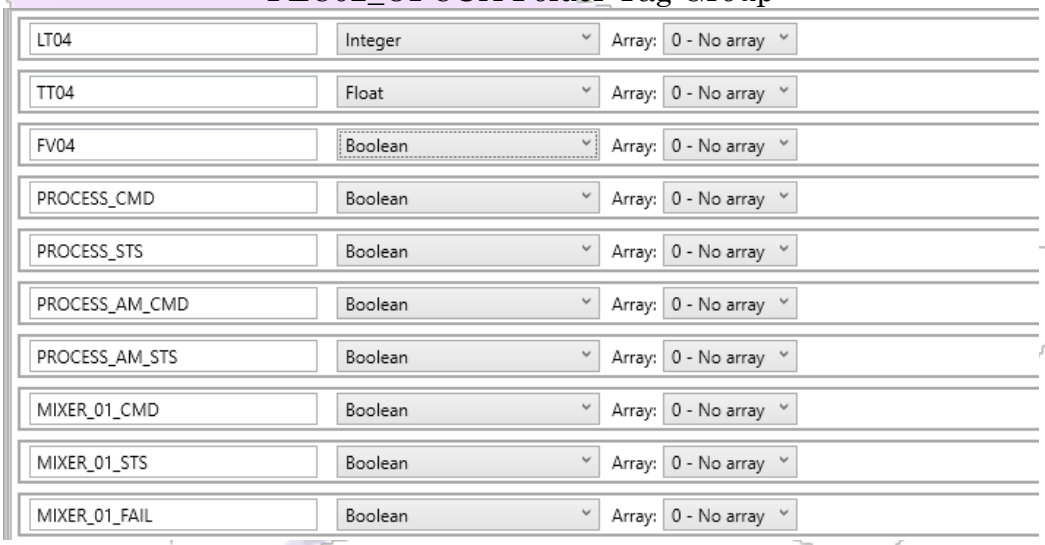
PLC01_MODBUS Folder Tag Group



9. Now, do the same step by step for the PLC02 tags. See the Tag List to create them.

Heads up: *The FV04 valve tag must be created, but not configured, as shown in the image below. Because in the next chapter we will use the DataType feature. This feature is intended to facilitate the creation of repetitive tags.*

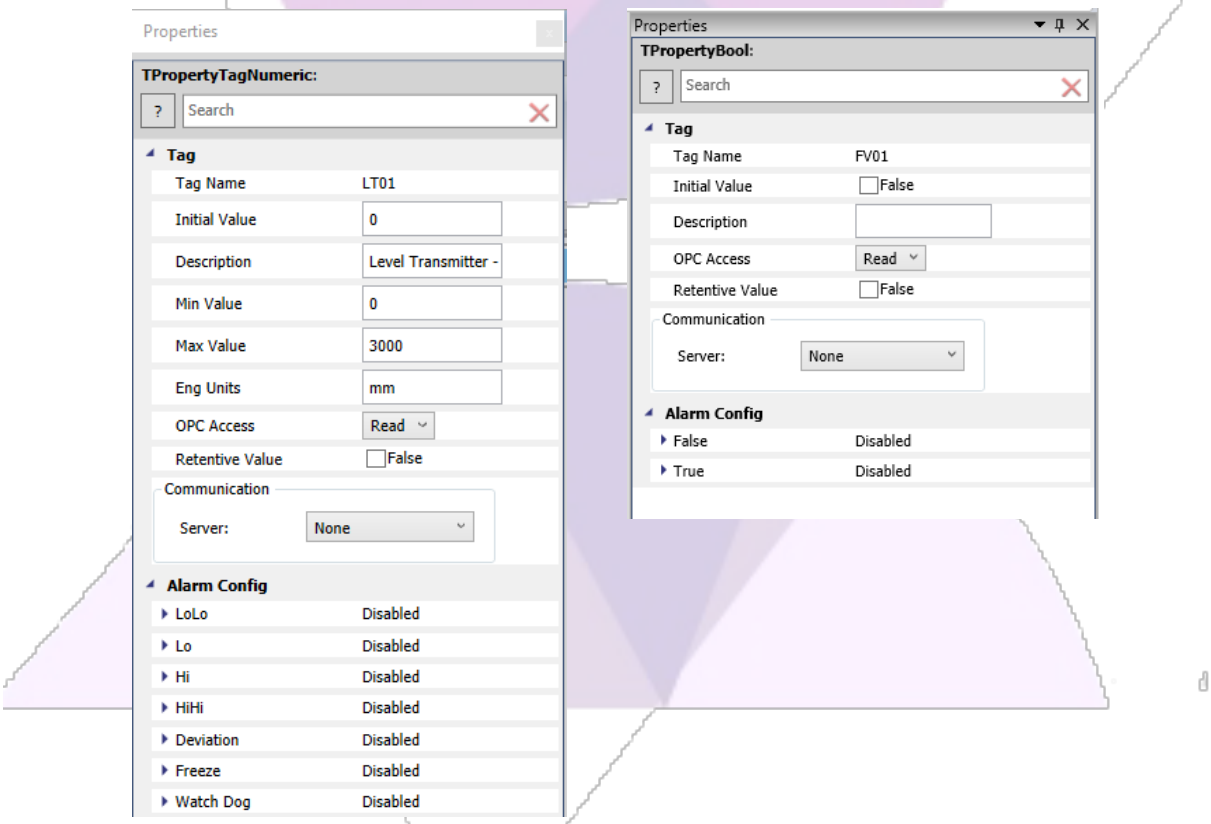
PLC02_OPCUA Folder Tag Group



8.3. Ownership of Tags

In ADISRA SmartView, a Tag refers to a unique identifier for a data point within the application, i.e. it is used to receive and write values in "RunTime" both to and from a data source. In the Tags document, the user can create, remove and store the tags used in the project. Each tag has a property list, in which some properties are fixed and others editable.

Every tag created in ADISRA SmartView is composed of several properties. These properties can be configured in the development environment, as seen in the properties list, used in a C# script and changed in RunTime Mode. Below we will know the main properties of a tag.



- **Tag Name:** Name of the created tag.
- **Description:** Tag Description.
- **Min Value:** Minimum acceptable value in the tag.
- **Max Value:** Maximum acceptable value in the tag.
- **Eng Units:** Engineering Unit
- **OPC Access:** It configures the type of tag interaction with the OPC Driver.
  Read: Allows only reading on the tag.

Write: Allows only writing to the tag.
Read/Write: Allows reading and writing to the tag.
None: Does not allow reading and writing.

- **Retention Value:** Configures whether the tag will keep the value after the RunTime goes to "Stopped" mode.
- **Communication – Server:** Enables tag value simulation.

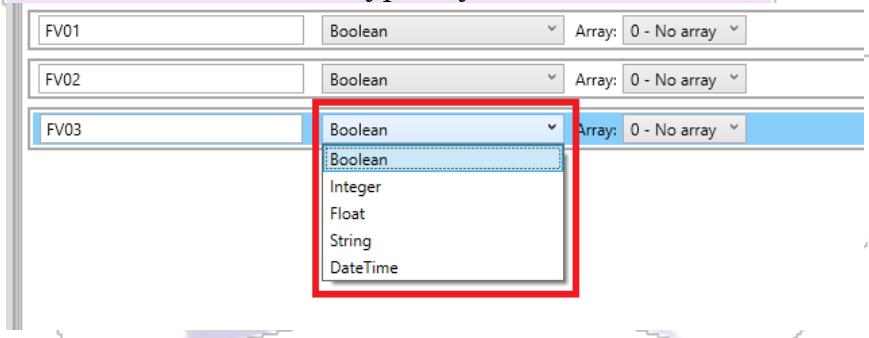
note: *The alarm properties will be discussed in the alarms chapter.*

9. Data Type Document

9.1. Overview

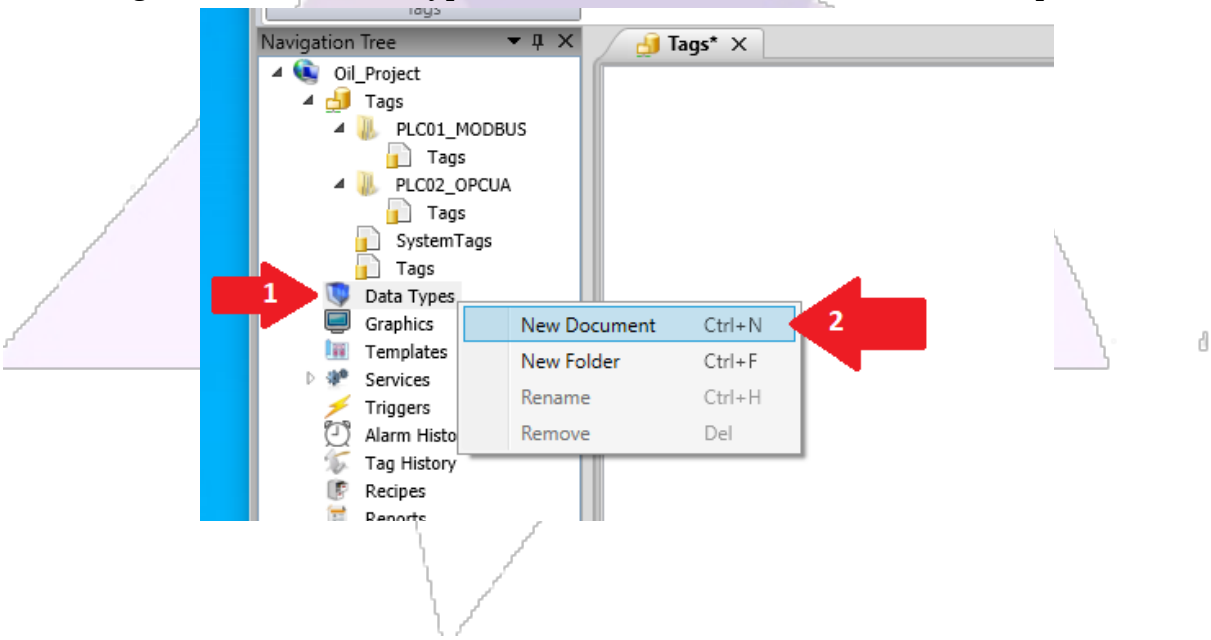
The Data Type Document allows the user to create a structure to be used in the creation of Tags, when the inserted tags are similar, thus facilitating the development of the application. When we insert a new tag, we can determine if the tag type is Boolean, Integer, Float, String, or DataTime. So, after creating a new Data Type, the new type will be available in the list of Data Types in the tags document.

Datatypes by default



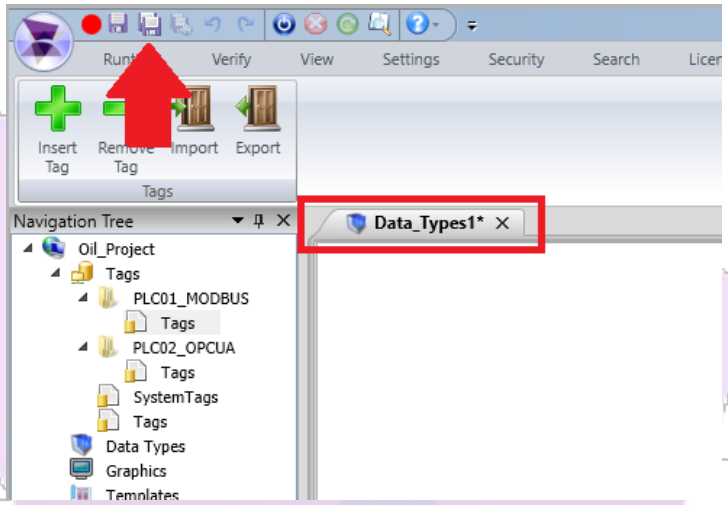
9.2. Creating DataType

1. Right-click on “Data Types” and then on the “New Document” option.

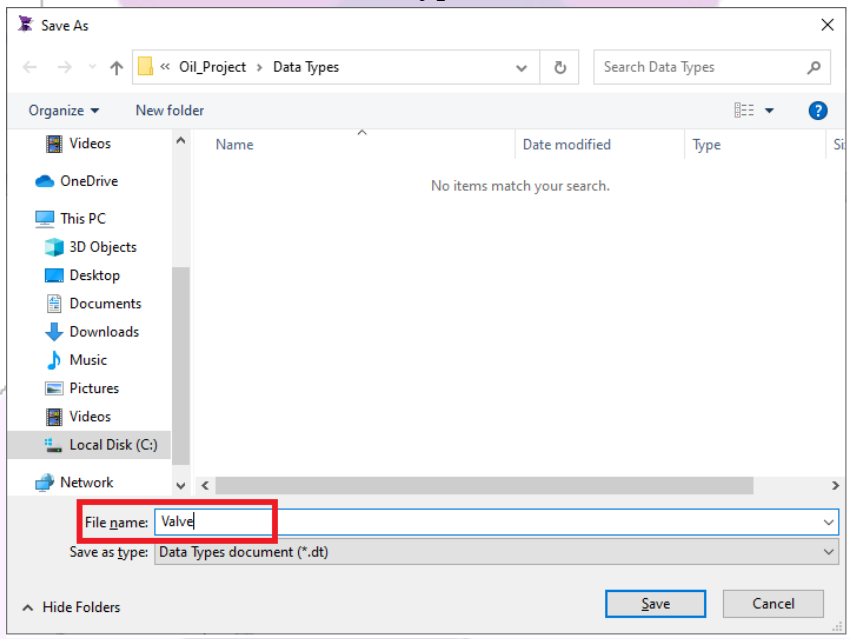


2. Once this is done, the new document will be opened on the right, called “Data_Types1”. In order to organize our project, let's rename the new data

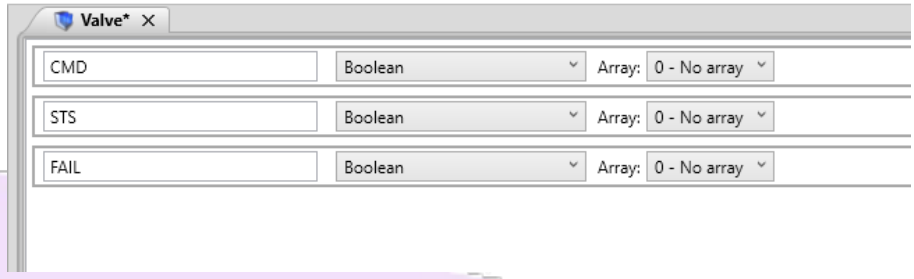
type to “Valve”. Then, click on the “Save as” option as shown in the image below.



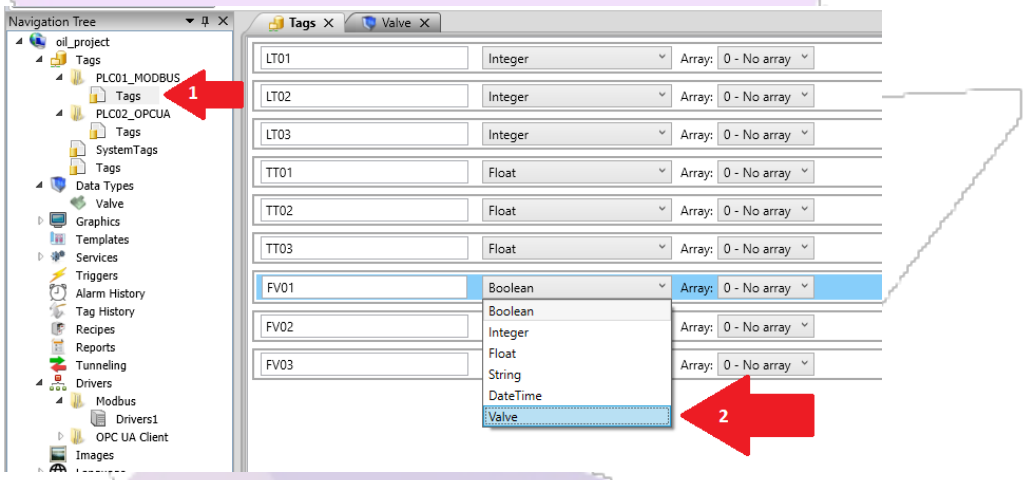
3. Change the name of the new data type to “Valve” and then click “Save”.



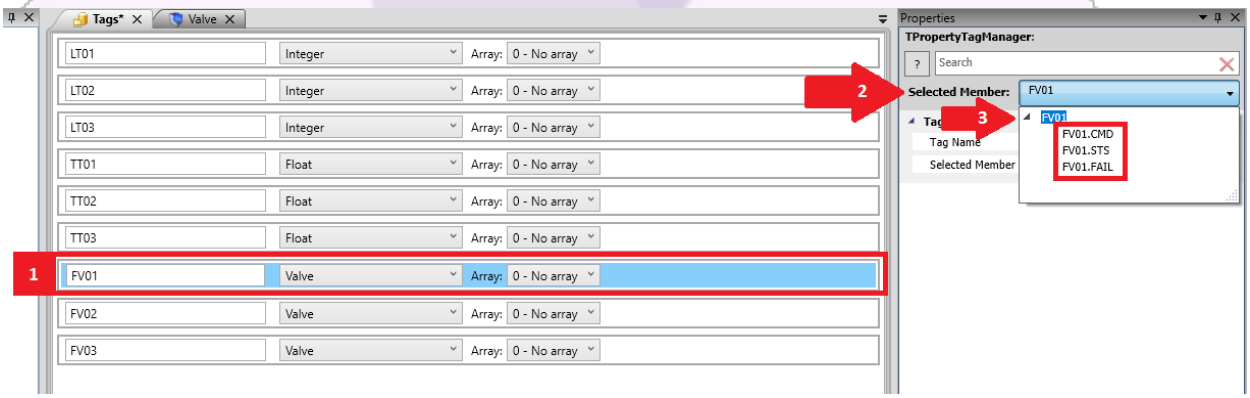
4. After creating the new data type, let's add the new sub tags. This new data type will be used for valve tags. As the valves have the same types of tags, such as Command, Status and Fault, we will create a single structure in which it can be reused for all valves. Therefore, click on “Insert tag” to add the new sub tags. The data type should be as shown in the image below.



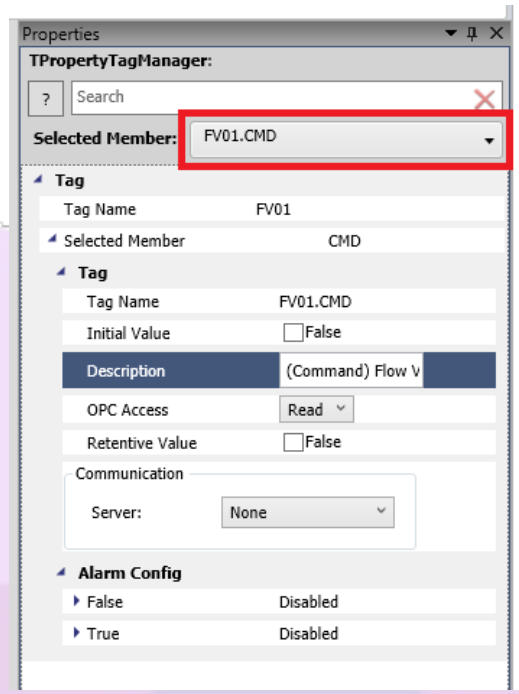
5. After inserting the sub tags of the data type "Valve", go back to the document "Tags" of the PLC 01_Modbus", and change the datatype of FV's 01,02 and 03 to "Valve".



6. Once this is done, with the tag selected, it is possible to observe that the tag "FV01" and the others in which they are associated with the datatype "Valve", have a structure with the sub-tags CMD, STS and FAIL. Now, the tags referring to FV01 are called FV01.CMD, FV01.STS and FV01.FAIL. Therefore, with the use of this "Data Type" resource, the creation of tags referring to valves becomes much more efficient.



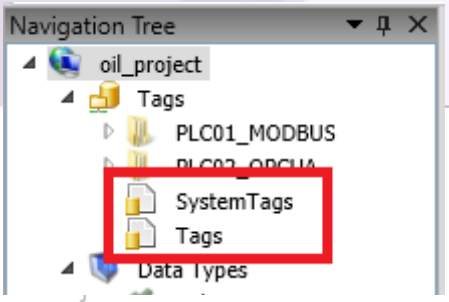
7. Finish configuring the sub tags as per the project tag list.



Heads up: *Repeat all this configuration for the FV04, located in the tags document, in the "PLC02_OPCUA" folder. After configuring the tags, do not forget to save the application by clicking on "Save as".*

9.3. System Tags

By default, ADISRA SmartView always creates a new application with two groups of tags. The group "System Tags" and "Tags". The "System Tags" group has tags that can be used in the application, such as system date and time, software version, activated software time, project name, machine IP and various other information. The other group called "Tags" is created blank. As we've already organized the project's tags into folders, we're not going to use this tags document.



9.4. Deleting Tags and Data Types

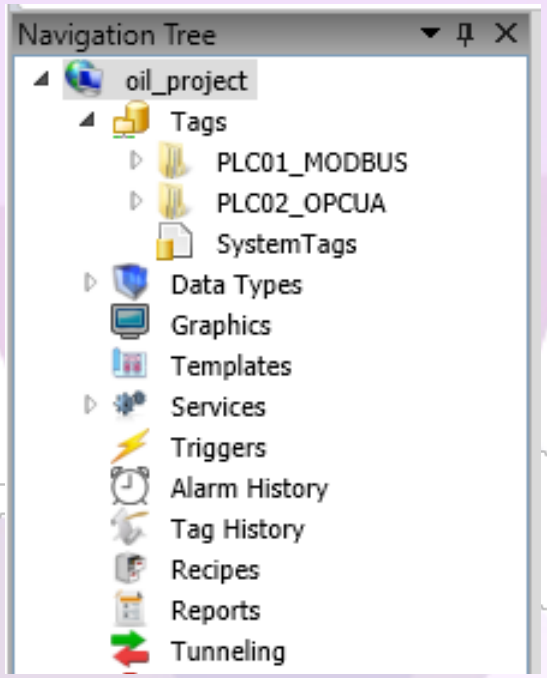
To delete a document, follow the steps below:

1. Click the right mouse button on the document, in this case, we will delete the document "Tags" in which it was created by default.
2. Then click on the "Delete" option.

note: *This step can be used to delete any document.*

Heads up: *To delete any document in the navigation tree, the document must be closed.*

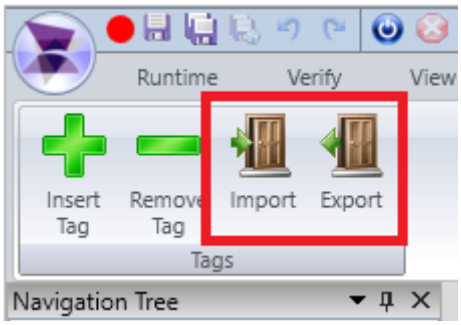
The navigation trees should look like the image below.



9.5. Deleting Tags and Data Types

ADISRA SmartView allows you to export and import tags and data types, but always respects the structure.

To import or export tags and data types, use the buttons below within each document.

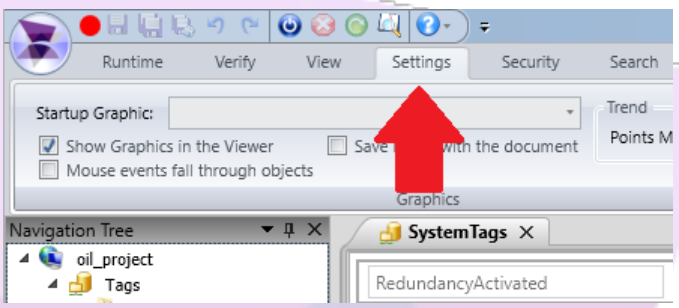


9.6. Tag counter

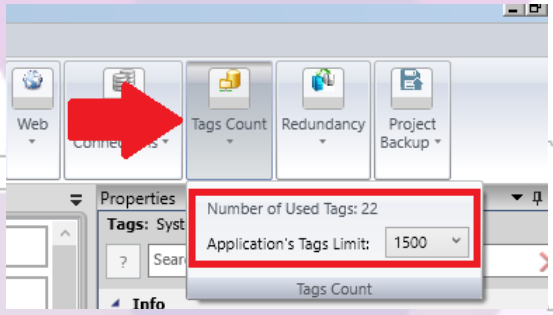
ADISRA SmartView has a tag counter, which can be used to know how many tags are being used in the application and the tag limit according to the software license.

To view the tag counter, follow the steps below:

1. In the top menu, click on the “Settings” option;



2. In the right corner, click on the “Tags Counts” option.



10. Starting RunTime

10.1. Getting to Know RunTime Mode

RunTime Mode is when the developed application is executed on the local computer. This mode is very important for the developer to see if the application is working as planned and if there are any errors.

1. The image below shows the status and main buttons of the RunTime mode.

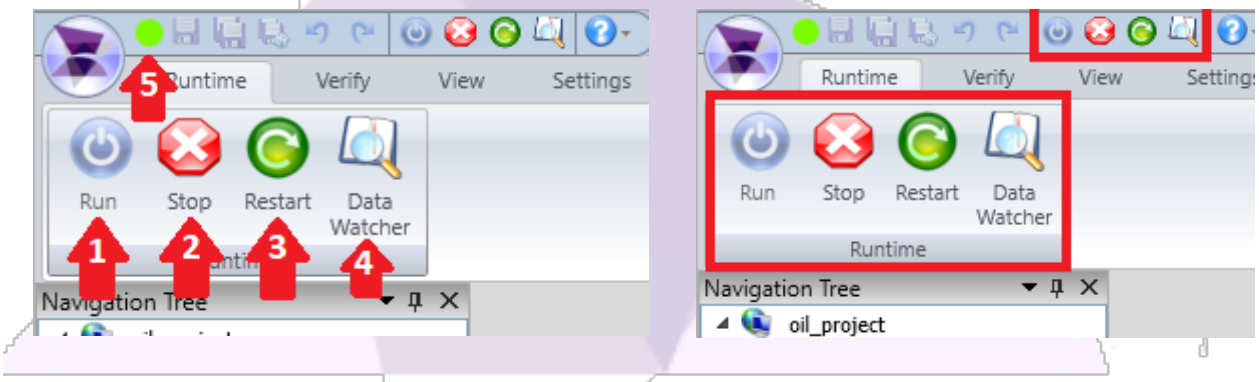
   1) Run button (Changes the state of the application from Stop to Run);

   2) Stop button (Changes the state of the application from Run to Stop);

   3) Restart button (Restarts the application);

   4) Data Watcher button (Opens the Data Watcher);

   5) Application Status (Green = Run / Red = Stop).

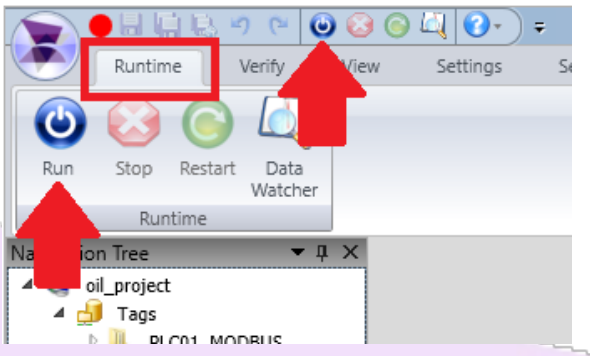


note: *On the image to the right, it is possible to notice the Run, Stop, Restart and Datawatcher buttons also represented in miniature.*

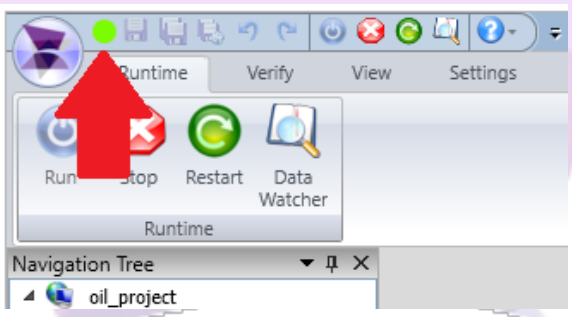
10.2. Running the Application

To start the application, and view the current values of all tags created, we must put the application in **RUN** mode.

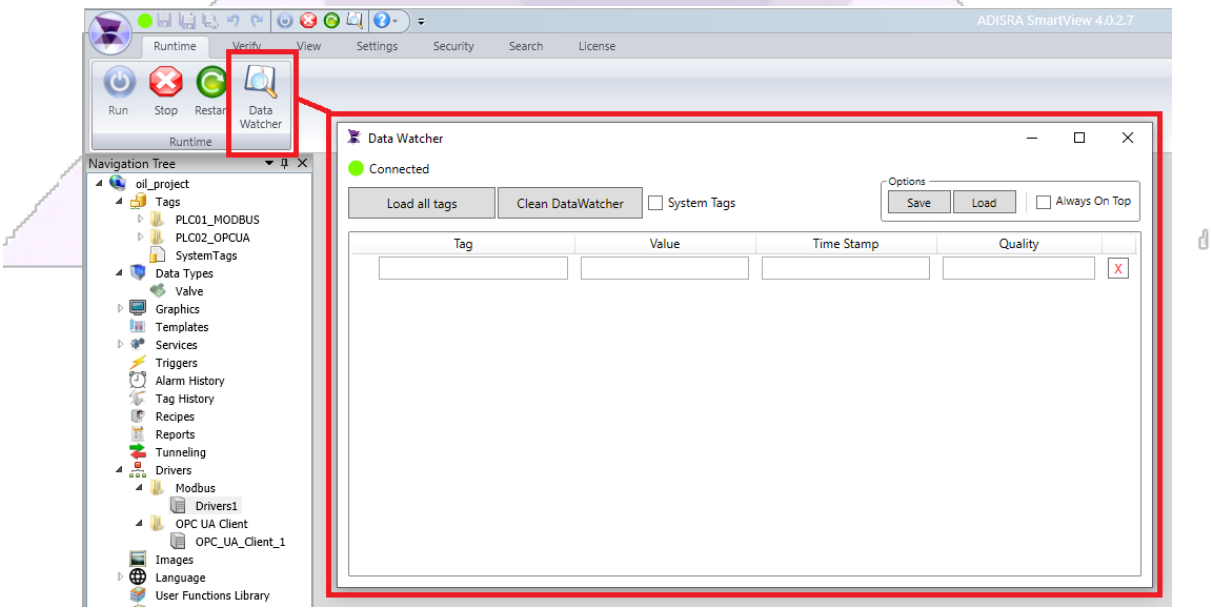
1. In the menu, click on the "RunTime" tab and then on the button located above the menu or on the options of the "RunTime" tab, as shown in the image below.



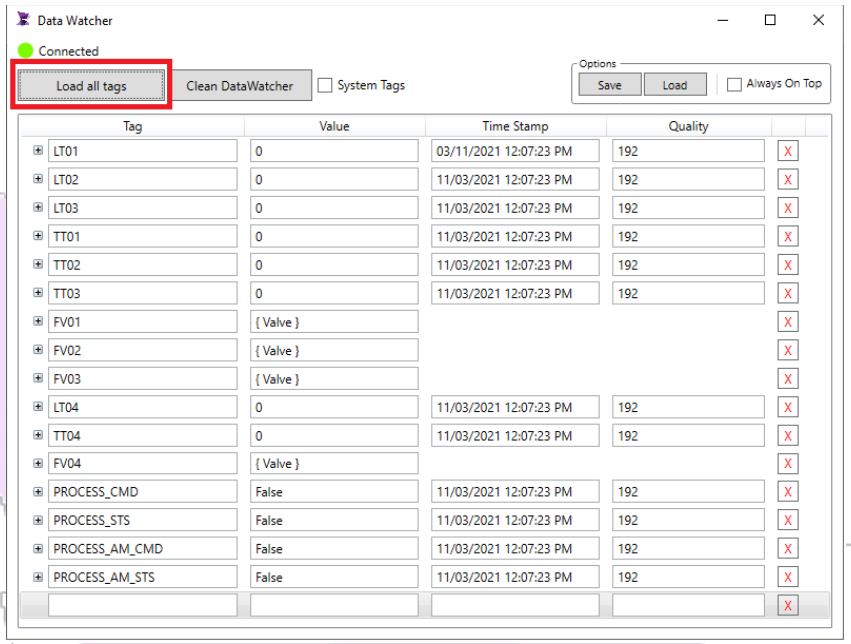
2. Note that after clicking the button, the application status changes from red to green, showing that the application is now in **RUN** mode.



3. As we haven't created any graphical screens yet, apparently nothing will be displayed on the screen, even though the viewer module has started along with the other modules. However, we can view the status of all application tags using the "Data Watcher" tool, also located on the "RunTime" tab. So let's open it.

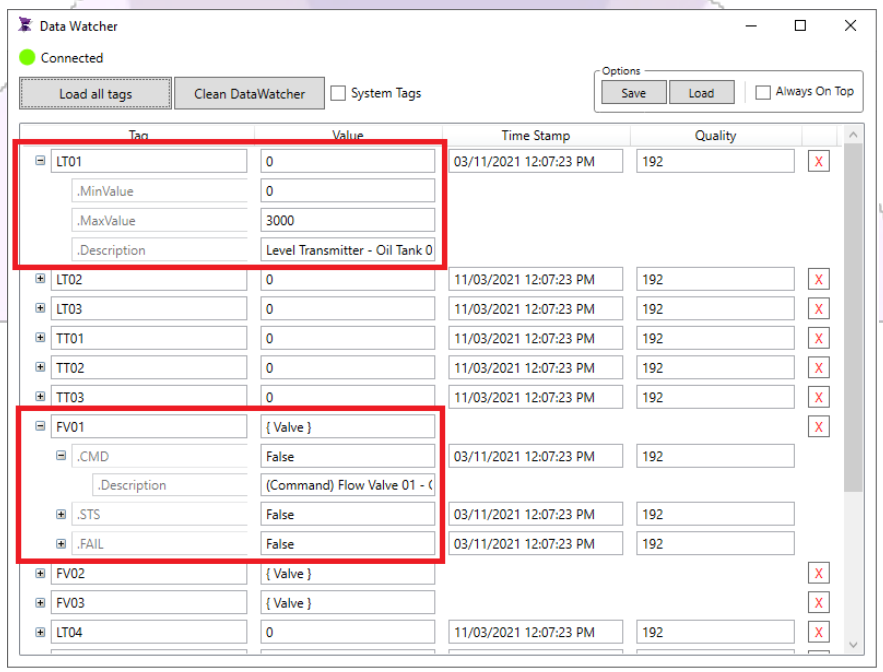


4. Then click on the "Load all tags" button to visualize all the tags of the application.



note: *To view a particular tag, simply fill in the "tag" column line and a list will appear.*

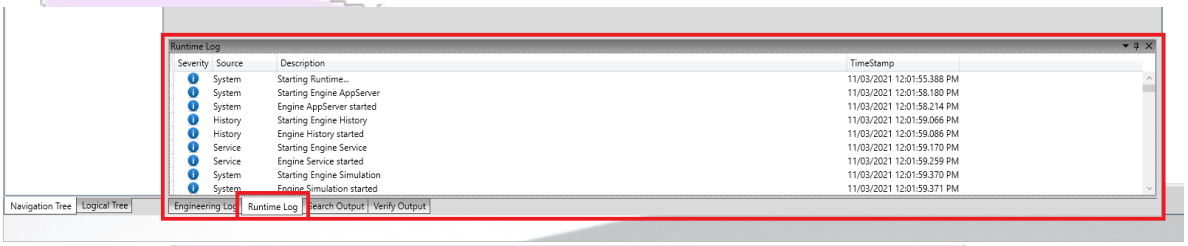
5. In the image below, you can see the status of each previously created tag. By clicking the "+" button located to the left of each tag, the properties of each tag will appear according to the datatype. Through the Data Watcher it is possible to visualize the tag name, the current value, the TimeStamp identifying the date and time of the last update and the quality. We can also change the tag value by clicking on the "Value" field.



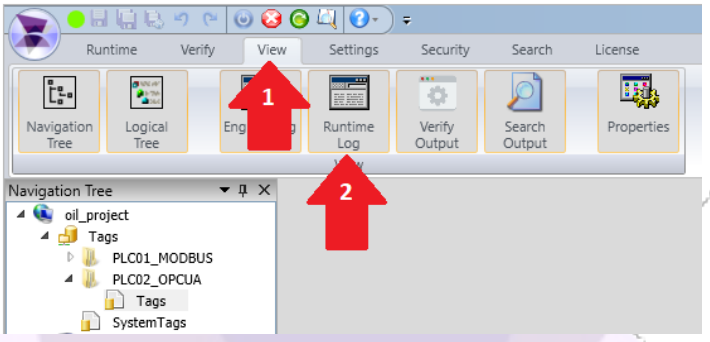
10.3. RunTime Log

The RunTime log displays all application events when in RUN mode. Therefore, it is possible to view events such as the application status (Run or Stop), the status of each module (Run or Stop), logged user status, commands and communication readings, errors, etc.

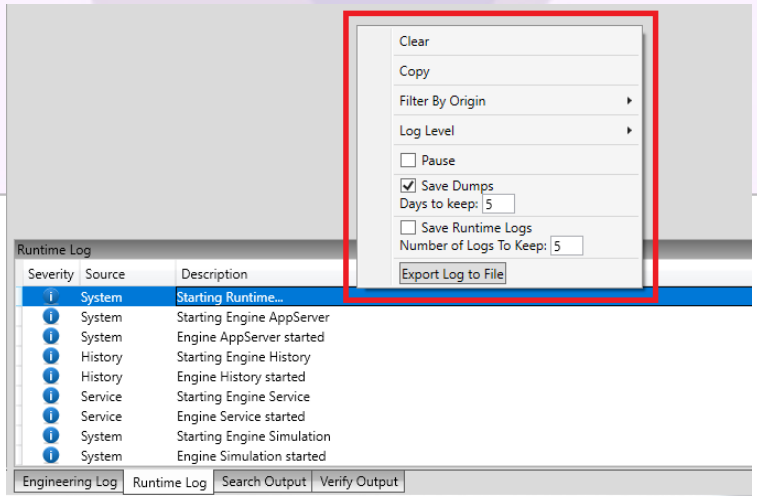
1. By default, the RunTime Log window will be visible in the split below the application.



2. If you want, you can hide the RunTime Log. To do so, click on the "View" tab located on the menu and then uncheck the RunTime Log option. But for our training, we will make the RunTime Log visible.

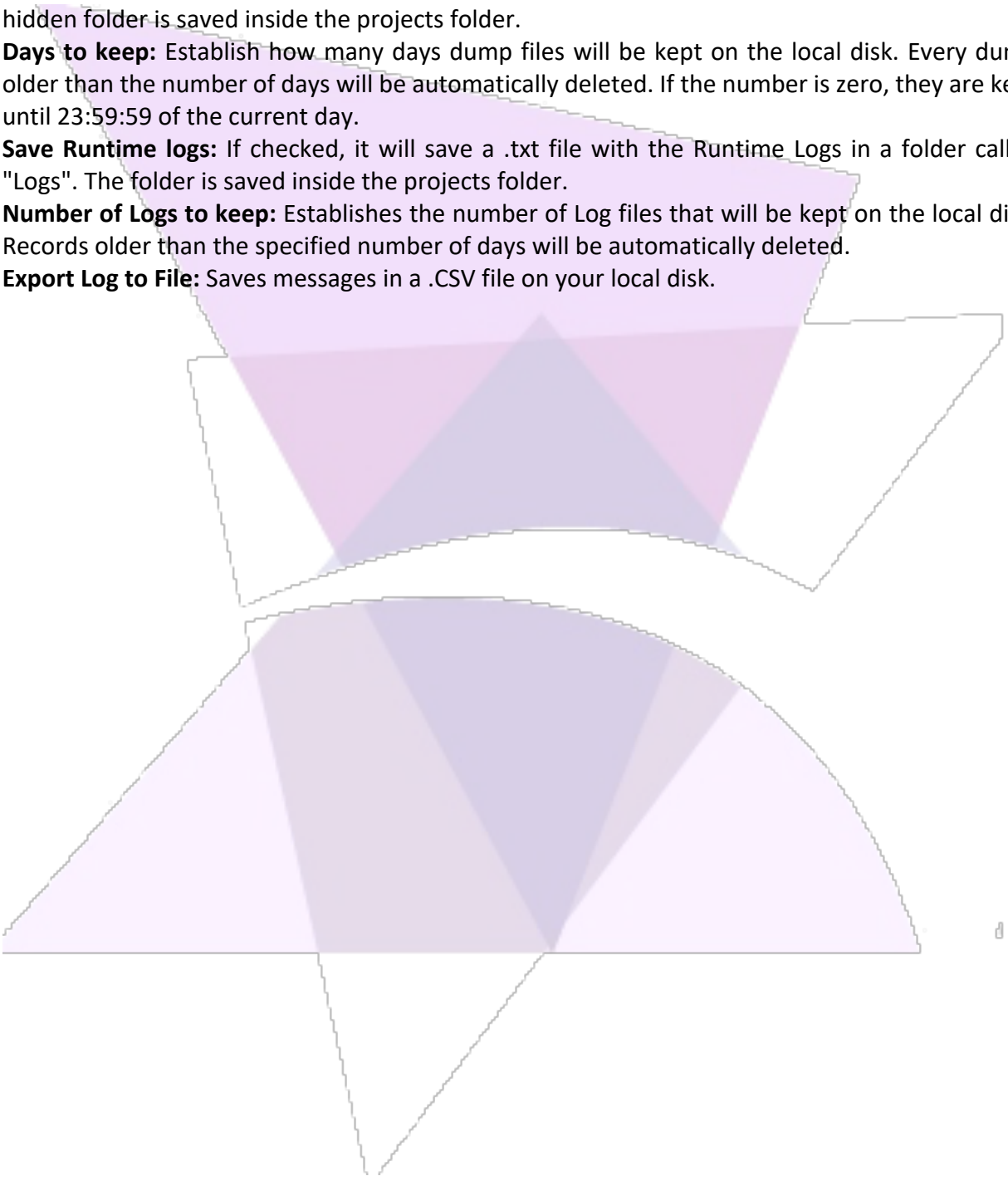


3. By right-clicking on any area of the RunTime Log, we can access some settings.



Clear: Clear all messages in the window
Copy: Copy the record to the clipboard

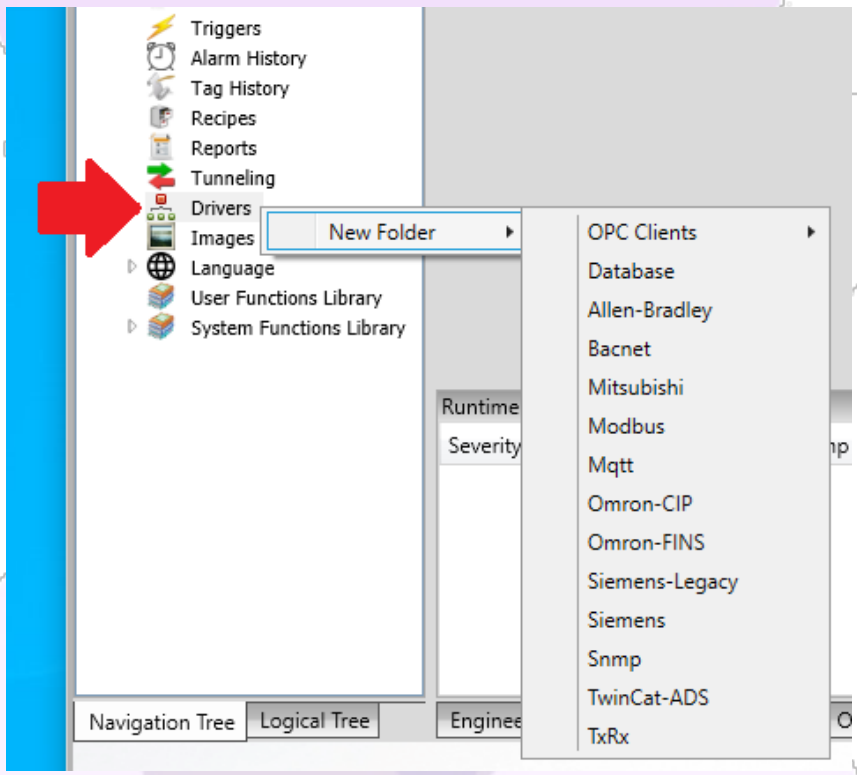
Filter By Origin: Sets different filters depending on the origin of the message
pause: Pauses the engineering log, so new messages are not visible in the window.
Save Dumps: If checked, it will save a .dump file with all errors in a hidden folder called "DumpFiles", which contains more detailed information about application errors or crashes. The hidden folder is saved inside the projects folder.
Days to keep: Establish how many days dump files will be kept on the local disk. Every dump older than the number of days will be automatically deleted. If the number is zero, they are kept until 23:59:59 of the current day.
Save Runtime logs: If checked, it will save a .txt file with the Runtime Logs in a folder called "Logs". The folder is saved inside the projects folder.
Number of Logs to keep: Establishes the number of Log files that will be kept on the local disk. Records older than the specified number of days will be automatically deleted.
Export Log to File: Saves messages in a .CSV file on your local disk.



11. Communication Drivers

11.1. Overview

Communication Drivers are responsible for the communication between ADISRA SmartView and field devices, such as PLC, sensors, and any other hardware that needs to communicate, enabling writing and reading between software and hardware. It is also possible, from the communication driver, to connect to a database. It is important to mention that database communication behaves like driver communication as it allows the user to link a tag to a database table for a specific unique key.



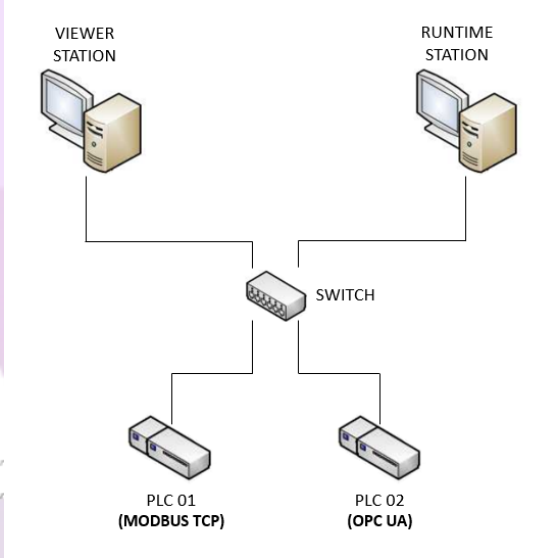
The “Drivers” Document can be found in the navigation tree. By right-clicking, it is possible to insert a new document according to the communication protocol to be used to communicate with the Field Device. ADISRA SmartView has several communication drivers to communicate with several protocols used by the main manufacturers, such as OPC Client (UA, DA and HDA), OPC Server, Allen-Dradley, Bacnet, Mitsubishi, Siemens, TwinCat and among others.

If the field device does not use the above protocols, it is possible to create a unique protocol per document (TxRx).

Well, now that we have the tags inserted and configured, let's learn how to add, configure and associate the communication drivers with the tags created earlier.

According to the SCADA architecture of our hypothetical project, the SCADA software ADISRA SmartView should communicate with the PLC 01 using the communication protocol (MODBUS TCP) and with the PLC 02 using the communication protocol (OPC-UA).

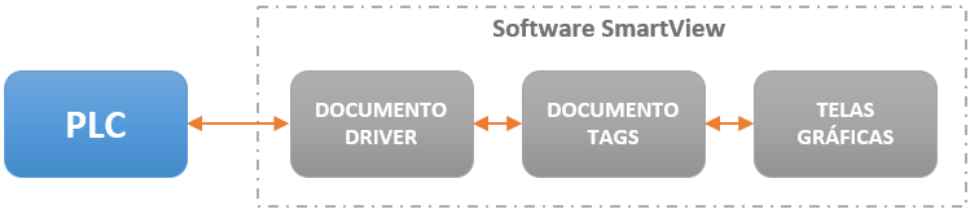
Heads up: *As we do not have the PLCs physically, we will use simulators in which each PLC will be represented.*



11.2. Communication Flowchart

Below is how the field device communication flows with ADISRA SmartView works.

1- The PLC will provide the addresses to be read or written;
2- The driver document will communicate the ADISRA SmartView with the PLC, that is, the PLC communication addresses will be associated with the tags created in the tags document. A tag can only be associated with a single communication address.
3- The associated tags can be used in graphic screens (Front-End) or in scripts, services, etc. (Back-End).

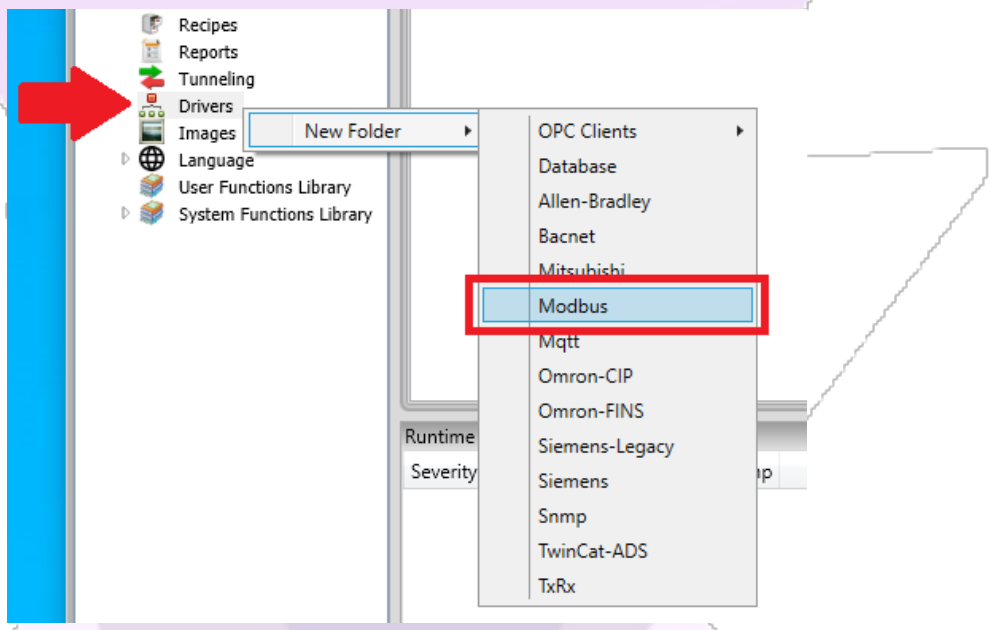


11.3. Modbus TCP Driver

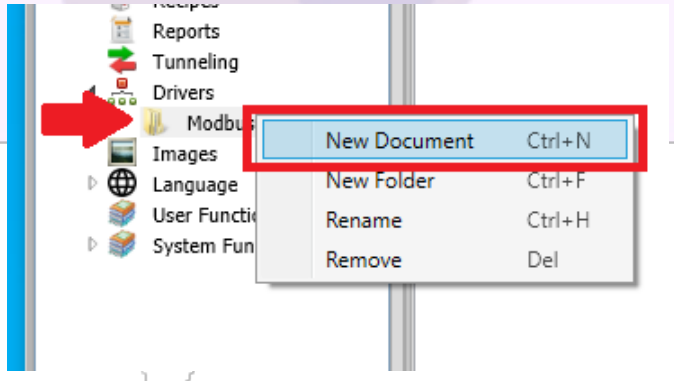
We will start by creating and configuring the Modbus TCP Driver and associating the tags with the PLC addresses. Next, it will be shown how the Modbus Simulator works.

11.3.1. Configuring the Driver

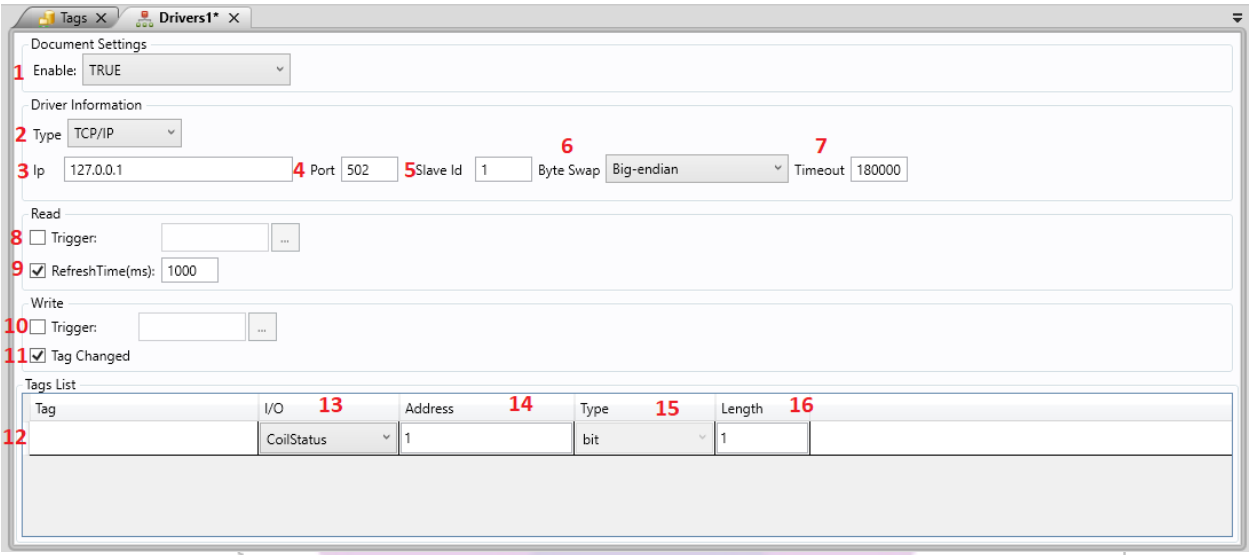
1. In the navigation tree, right-click on the “Driver” document, then on “New Folder” and select the “Modbus” option;



2. Now, right-click on the “Modbus” folder and select the “New Document” option. After that, save the application by clicking on the “Save All” button.

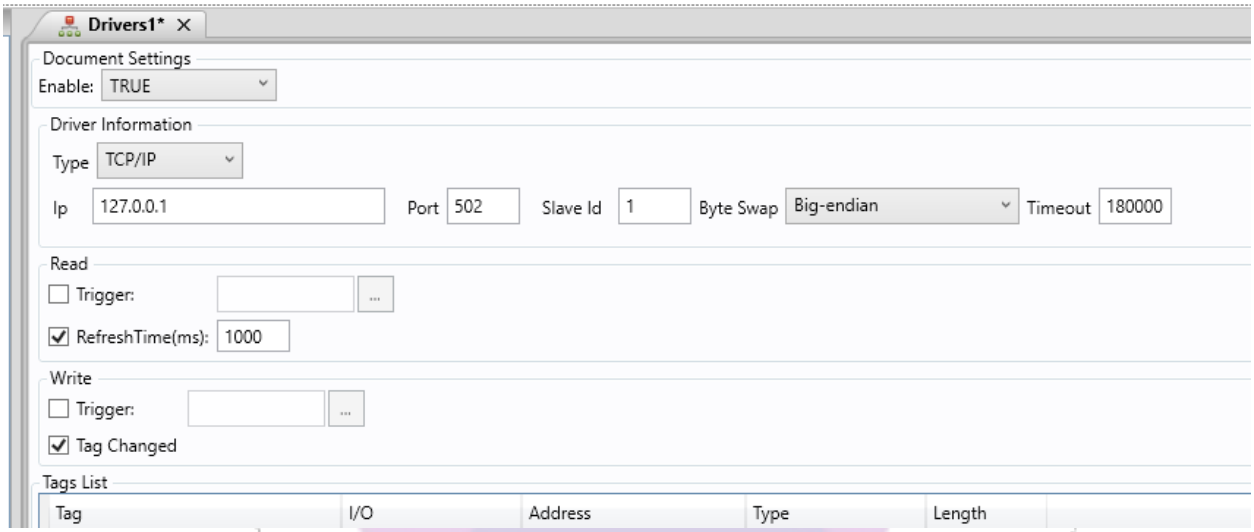


Before continuing the driver configuration step by step, let's get to know each Modbus Driver configuration field.



1- Enables or disables the Modbus Driver Document;
2- Selects the communication type (TCP/IP, UDP or Serial);
3- Controller IP (PLC);
4- Controller communication port (PLC);
5- Configures the Slave ID of the Controller (PLC);
6- Configure Byte Swap mode;
7- It configures the time limit (in milliseconds) that ADISRA SmartView will try to communicate with the Driver before closing the connection;
8- If checked, it will read all addresses as soon as a tag configured in the field to the side is changed;
9- If checked, it will read all addresses every specified time, in milliseconds;
10- If checked, it will write to all addresses whenever a tag configured in the field to the side is changed;
11- If checked, it will write to all addresses as soon as any tag in the list below is changed;
12- Field to associate the tag created in the “Tags” document;
13- Selects the communication address function;
14- Field to add a specific communication address;
15- Select the type of address;
16- Field to add the length of the message to be transferred.

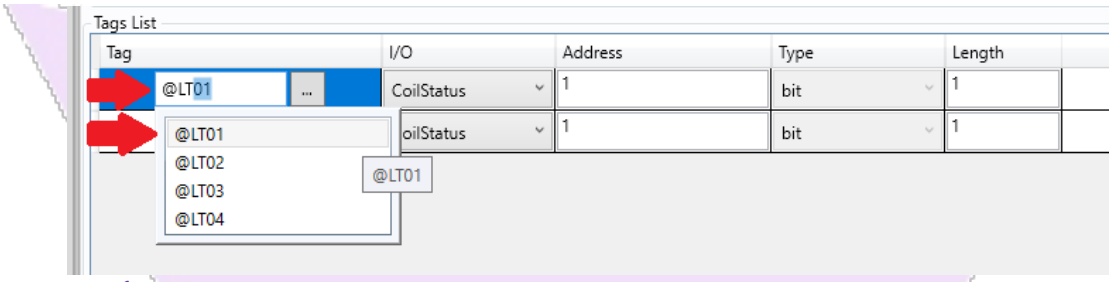
3. Now that we know each field of the Modbus driver, let's start the configuration, and the association of ADISRA SmartView tags with PLC addresses. First, configure the Driver fields according to the image below. The PLC IP will be 127.0.0.1, because we will use our simulator.



4. The addresses of the PLC variables are available in the project's Tag List. As shown in the image below, the variables highlighted in the red rectangle represent the PLC01 MODBUS TCP variables, and each variable has a modbus address, highlighted in the green rectangle.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TC	40001	NTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TC	40002	NTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TC	40003	NTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TC	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TC	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TC	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TC	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TC	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TC	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TC	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TC	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TC	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TC	7	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TC	8	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TC	9	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA		NTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_01_FAIL	(Fail) Flow Valve 01 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

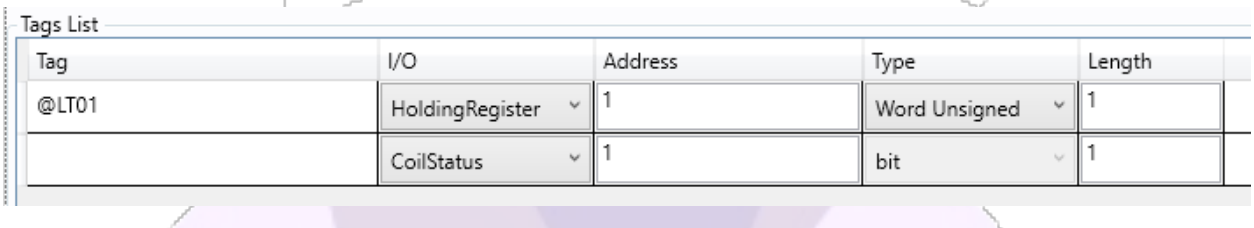
5. Let's associate the PLC addresses with the ADISRA SmartView tags. To do so, in the "Tag" field, double-click and type the "@" symbol, and then a list will appear with all the created and system tags. Therefore, we can bind to any ADISRA SmartView tag.



note[1]: *It is possible to find the tag without adding the "@" symbol. For example, typing only "LT01".*

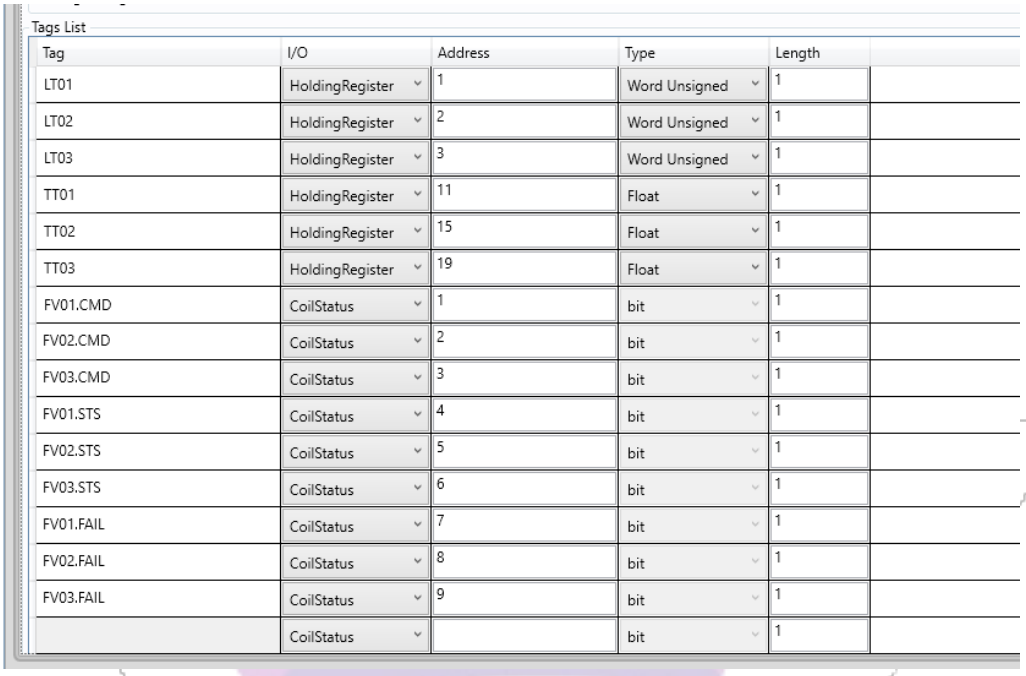
note[2]: *It is possible to create a tag by simply typing the name of the new tag in the "Tag" field and clicking enter, and then a new window will open asking for Name, Document, type and array of the new tag.*

6. After adding the ADISRA SmartView tag, we will associate and configure it with its modbus address according to the Project Tag List. The configuration should look like this:



note: *It is not necessary to write the entire modbus address, for example, the LT01 tag is associated with the address 40001, but we only insert the number 1.*

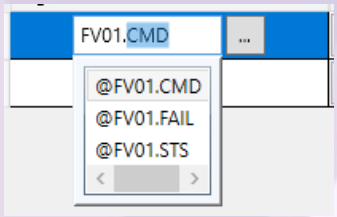
7. Now, repeat the same configuration process for the other tags in the project. The list of Driver tags should look like this:



Tags List

Tag	I/O	Address	Type	Length
LT01	HoldingRegister	1	Word Unsigned	1
LT02	HoldingRegister	2	Word Unsigned	1
LT03	HoldingRegister	3	Word Unsigned	1
TT01	HoldingRegister	11	Float	1
TT02	HoldingRegister	15	Float	1
TT03	HoldingRegister	19	Float	1
FV01.CMD	CoilStatus	1	bit	1
FV02.CMD	CoilStatus	2	bit	1
FV03.CMD	CoilStatus	3	bit	1
FV01.STS	CoilStatus	4	bit	1
FV02.STS	CoilStatus	5	bit	1
FV03.STS	CoilStatus	6	bit	1
FV01.FAIL	CoilStatus	7	bit	1
FV02.FAIL	CoilStatus	8	bit	1
FV03.FAIL	CoilStatus	9	bit	1
	CoilStatus		bit	1

note: *To add valve tags, which have a DataType of type "Valve", type the name of the tag, for example "FV01" and then type ".".*

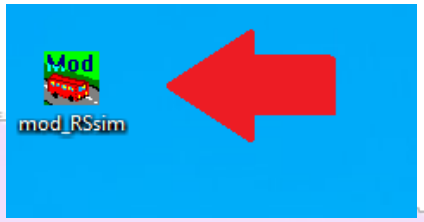


11.3.2. Configuring the Simulator

1- Click on the link below and download the Modbus simulator.

Link: https://adisra.com/wp-content/uploads/2025/03/ADISRA-SmartView-GettingStarted-DriverSimulator.zip

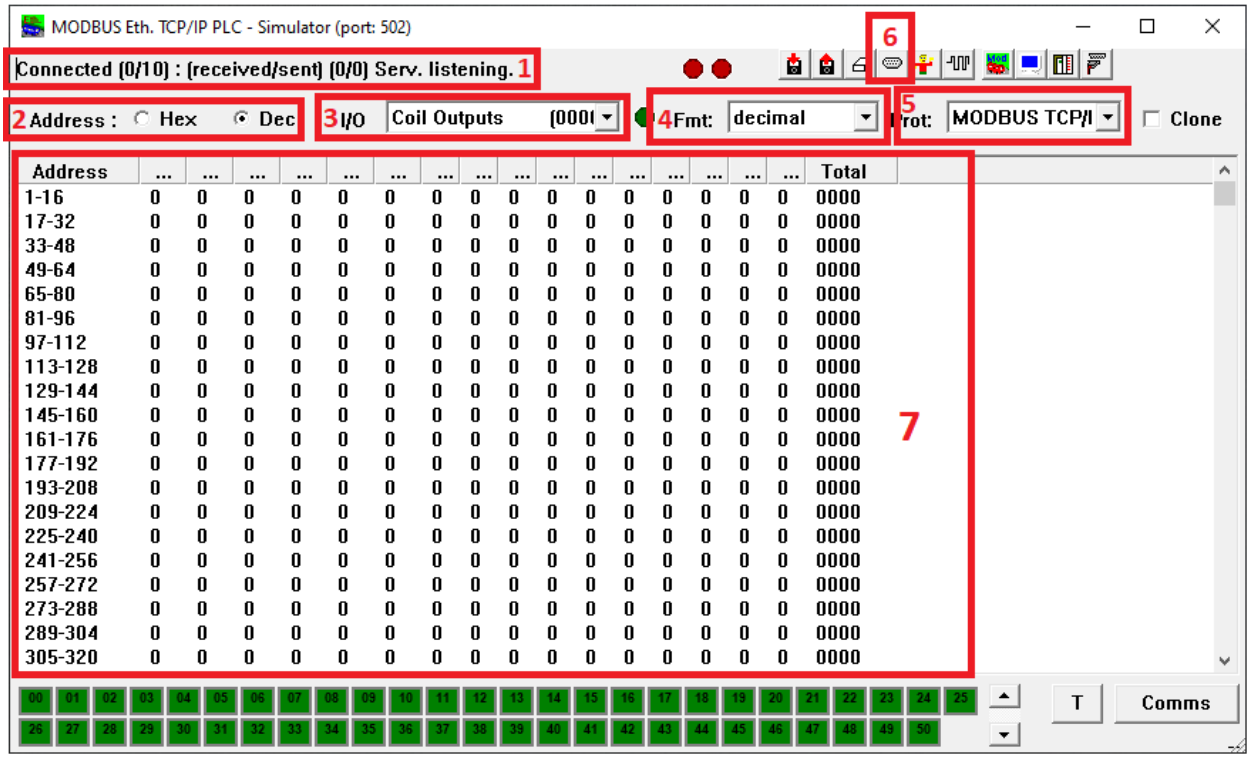
2- After downloading, extract the file and click on the “mod_RSsim” icon to open the simulator.



Heads up: *This simulator is demo. Then, every 45 minutes a message will appear informing you that the usage time has been exceeded, and communication will be interrupted. Therefore, you will need to reopen the simulator.*

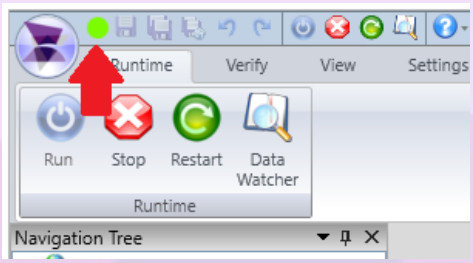
Now, let's get to know the simulator a little.

1- Simulator communication status with ADISRA SmartView;
2- Address format displayed in the mailing list. Hexadecimal or Decimal;
3- Field to choose the type of address shown in the list below;
4- Field to choose which format the value of the addresses in the list below should be presented;
5- Field to choose the communication protocol;
6- Opens a window with the Simulator's communication settings;
7- Modbus mailing list. According to the image below, the Coil-type addresses are being presented.

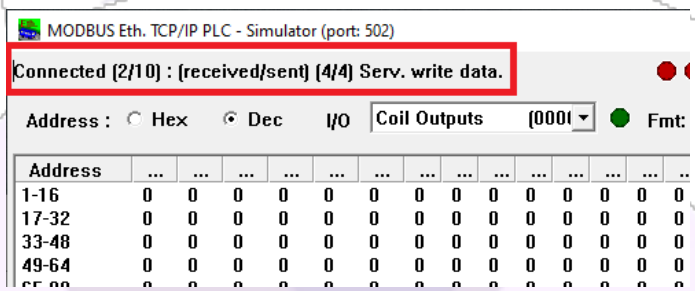


After getting to know the simulator's interface a little, let's learn how to communicate Simulator X ADISRA SmartView. Basically, it will not be necessary to configure the communication parameters of the simulator, as we have already configured ADISRA SmartView according to the parameters of the simulator. Therefore, we only need to verify that the communication protocol in the simulator is Modbus TCP/IP.

1- With the Modbus simulator open, in the “Prot:” field, select the Modbus TCP/IP protocol.
2- Go back to ADISRA SmartView and verify that the application is running RunTime. To establish communication, the Runtime must be active.



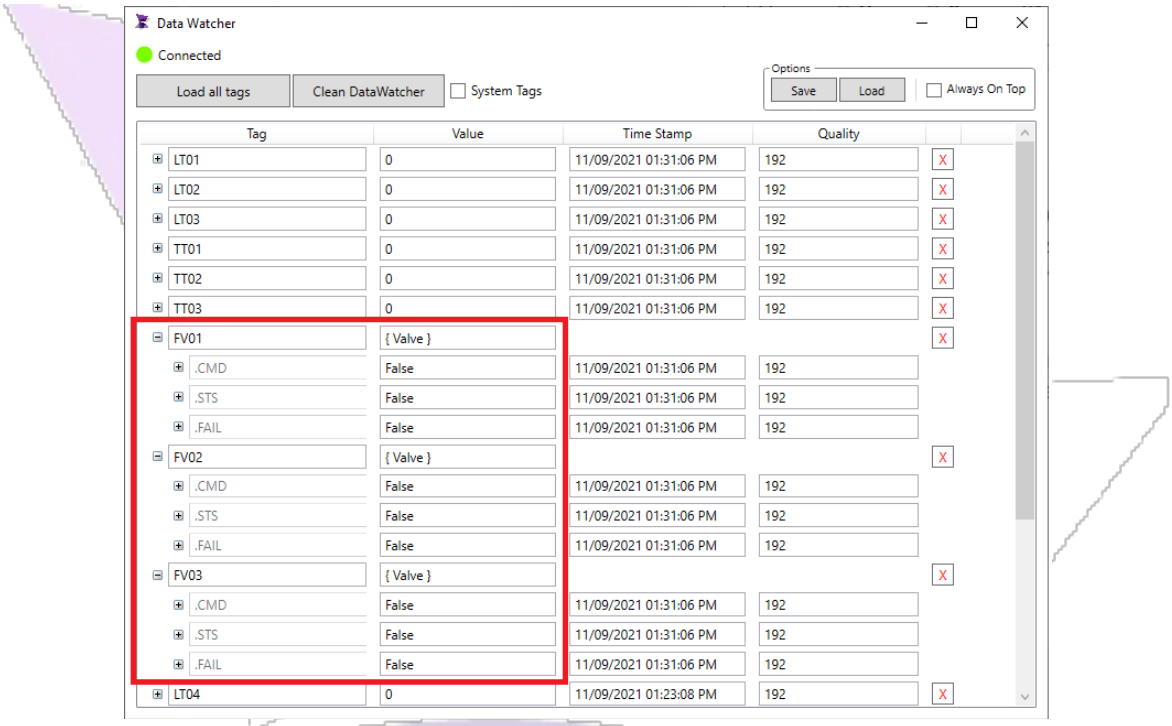
3- After running RunTime, go back to the simulator and check if the communication status is as shown in the image below:



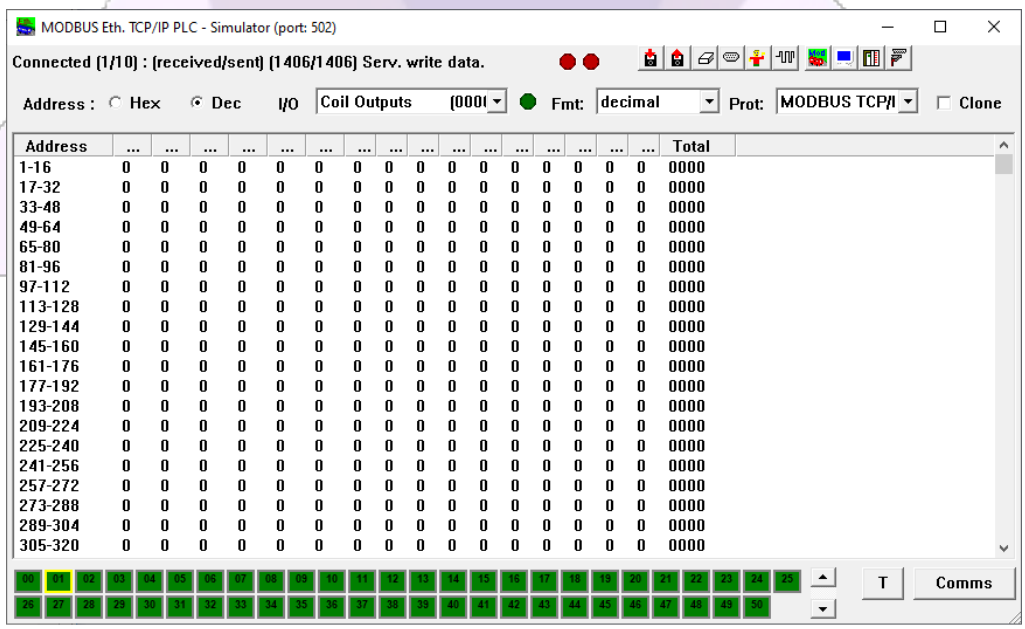
note[1]: *The status will be increasing all the time, for example: (4/4), (8/8), (10/10) and so on.*

note[2]: *If communication does not establish between the Simulator and ADISRA SmartView, go back to the engineering environment and verify that the Modbus Driver is “Enable”, then restart the RunTime by clicking on*

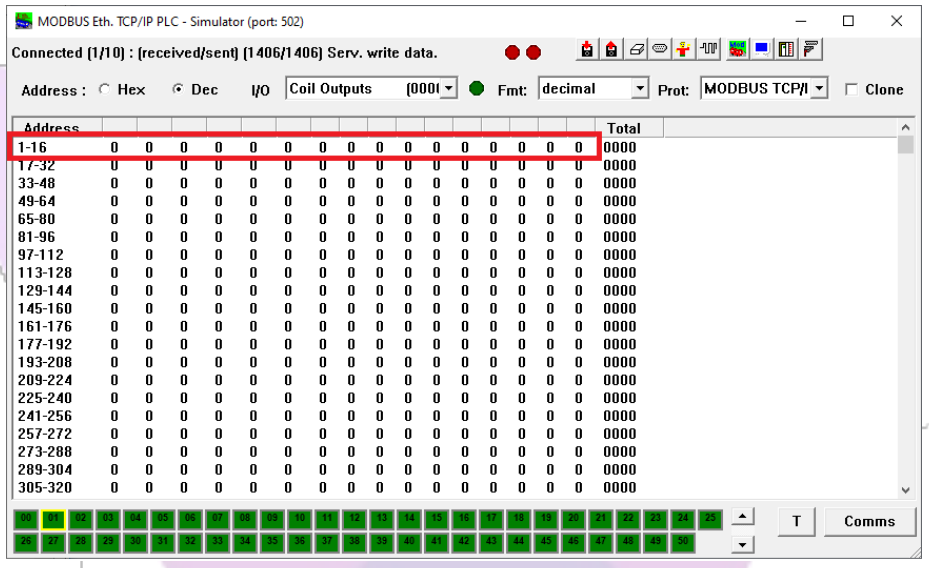
4- With the communication established, let's start reading and writing the status and command addresses of the valves. Then, to read and write these addresses, open the ADISRA SmartView Data Watcher, click on “Load All Tags” and expand each valve tag.



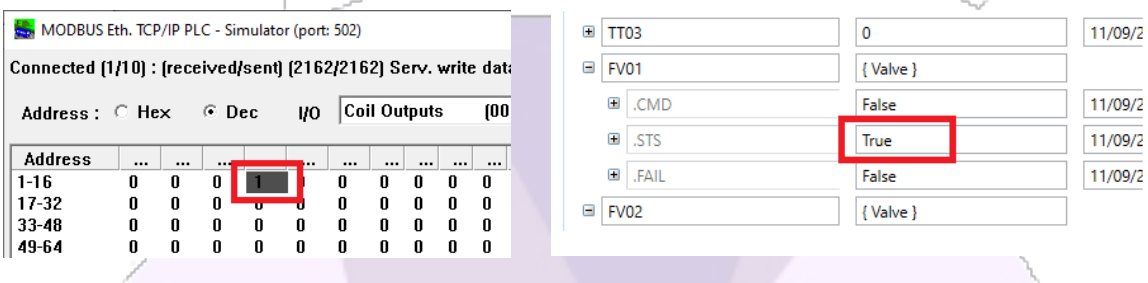
5- Now, go back to the simulator, keep the Address field in “Dec”, the I/O field in “Coil Outputs” and the Fmt field in “Decimal”. The simulator screen should look like the image below:



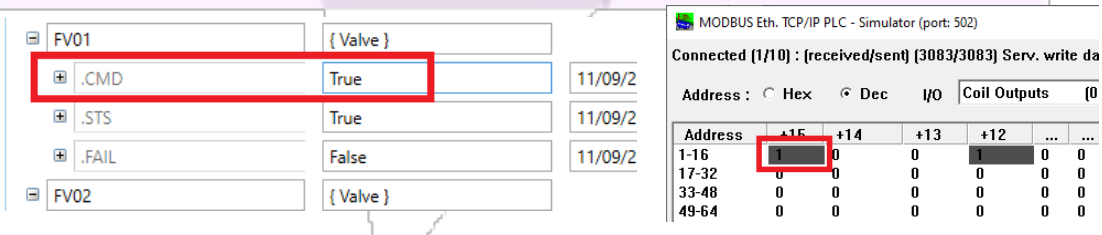
6- Note that each line is made up of 16 coil-type addresses, so all addresses we are monitoring are on the first line.



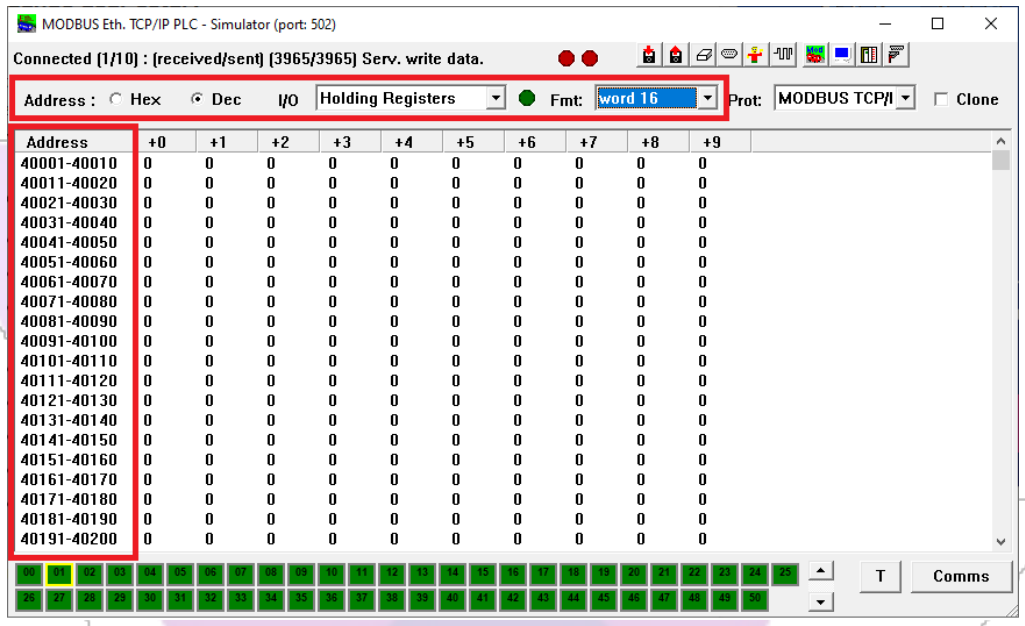
7- Let's change the status of FV01.STS from “False” to “True”. According to the tag list, the address of FV01.STS is “4”. Double-click on the value “0” in the fourth column. Then open Data Watcher and check if the value of FV01.STS has changed.



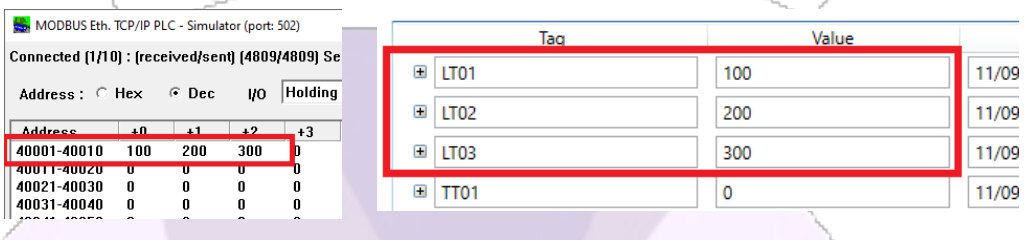
8- Now in DataWatcher, let's change the value of the FV01.CMD tag to “True”. To do this, click on the tag value field, type “True” and press enter. Then open the Simulator and check if the address “1” has been changed to “True”.



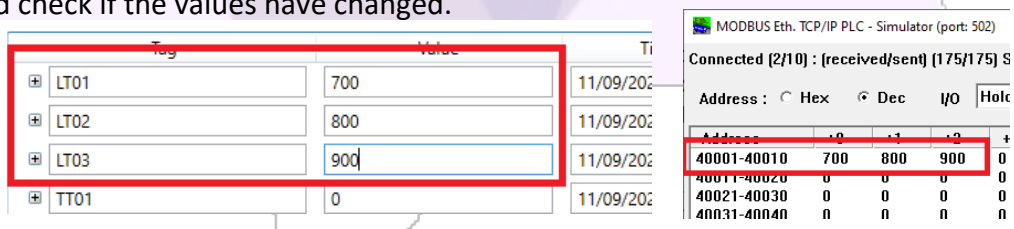
9- In addition to valve status and commands, we will read and write to the “LT01, LT02 and LT03” tags. To do this, keep the Address field in “Dec”, the I/O field in “Holding Registers” and the Fmt field in “Word 16”. The simulator screen should look like the image below:



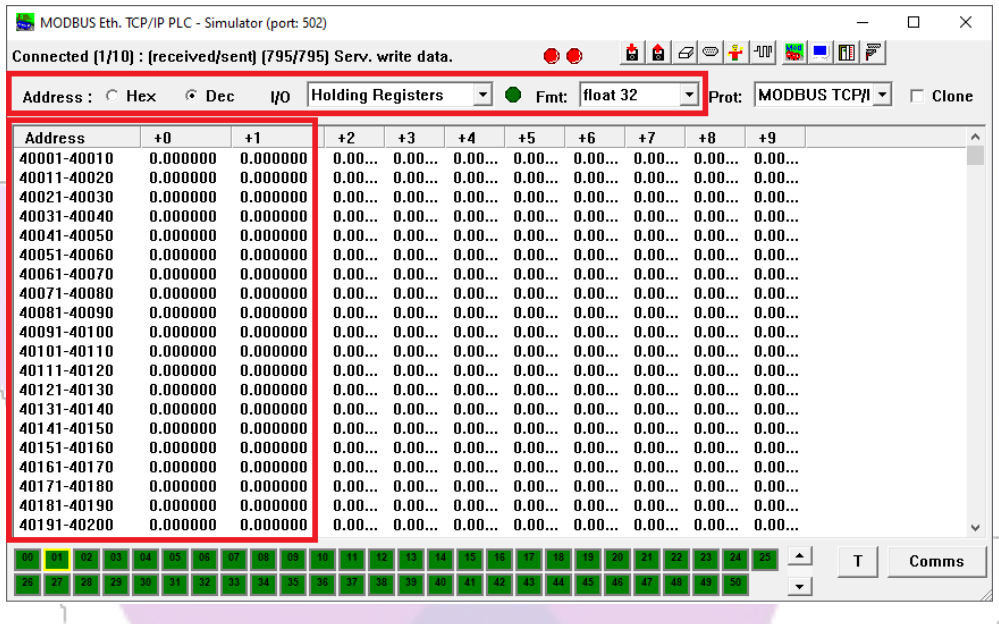
10- Now let's change the value of each LT. According to the project's tag list, the LTs have the following addresses: 40001, 40002 and 40003 respectively. In the simulator, change the value of each LT to: LT01 = 100, LT02 = 200 and LT03 = 300 as shown in the image below. Then open DataWatcher and check if the values have changed.



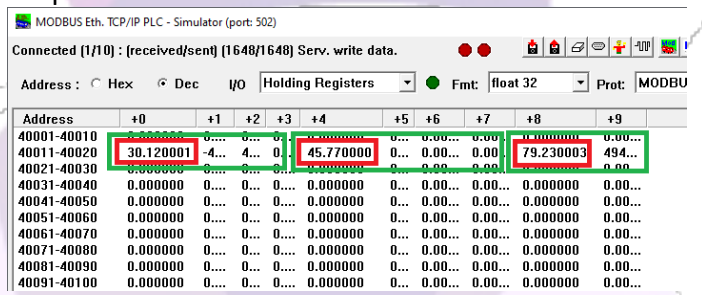
11- Then test the writing on the tag. Change the value of each LT in DataWatcher to: LT01 = 700, LT02 = 800 and LT03 = 900 as shown in the image below. Then open the Simulator and check if the values have changed.



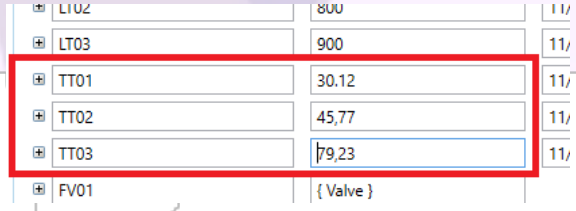
12- In addition to the valves, LT01, LT02 and LT03, we will read and write the tags TT01, TT02 and TT03. To do so, keep the Address field in “Dec”, the I/O field in “Holding Registers” and the Fmt field in “Float 32”. The simulator screen should look like the image below:



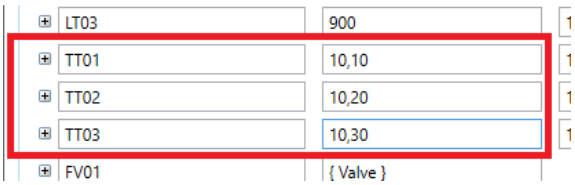
13- Now let's change the value of each TT. According to the project's tag list, the TT's have the following addresses: 40011, 40015 and 40019 respectively. In the simulator, change the value of each TT to: TT01 = 30.12, TT02 = 45.77 and TT03 = 79.23 as shown in the image below. Then open DataWatcher and check if the values have changed.

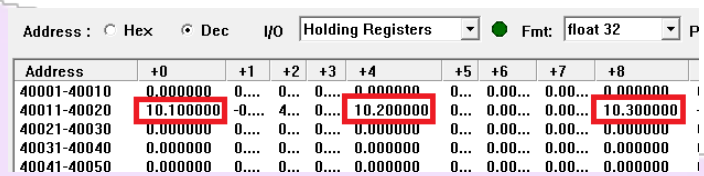


note: *As the TT's addresses are of the "Float" type, 4 bytes are used. So the addressing jumps from 4 to 4 modbus addresses.*



14- Then test the writing on the tag. Change the value of each TT in DataWatcher to: TT01 = 10.10, TT02 = 10.20 and TT03 = 10.30 as shown in the image below. Then open the Simulator and check if the values have changed.





If the entire test was carried out according to the steps above, the Modbus TCP/IP communication between the Simulator and ADISRA SmartView was successful.

11.4. OPC UA Driver

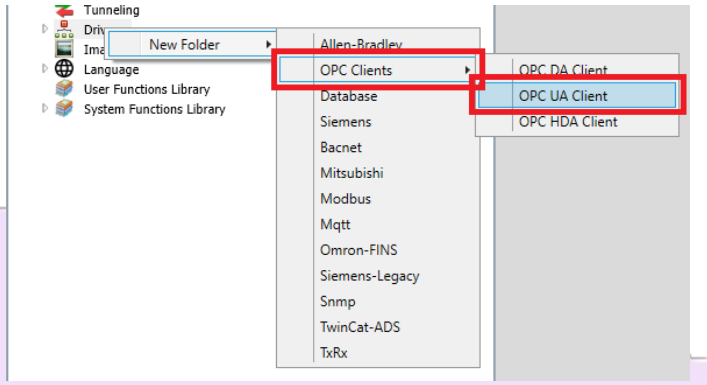
After creating, configuring and testing the Modbus TCP communication, let's start configuring the OPC-UA Driver and associating the tags with the PLC addresses.

ADISRA SmartView can be a client or a server in OPC-UA communication. For our project, we will use an OPC-UA Server simulator. Therefore, the ADISRA SmartView must be configured as a Client.

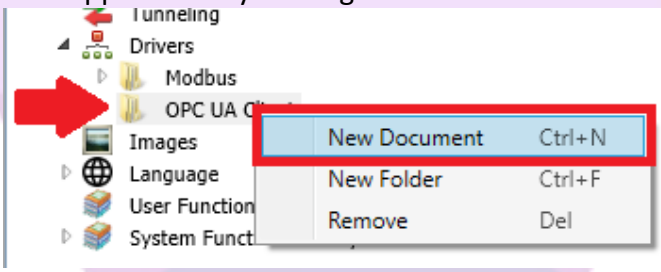
note: *If you need to use ADISRA SmartView as a Server, the configuration must be done through the "Settings" menu, then click on "OPC Server", and then enable the "Enable OPC Server" option. However, for our project, we will use ADISRA SmartView as a client. Therefore, the configuration must be done using the "Driver" document.*

11.4.1. Configuring the Driver

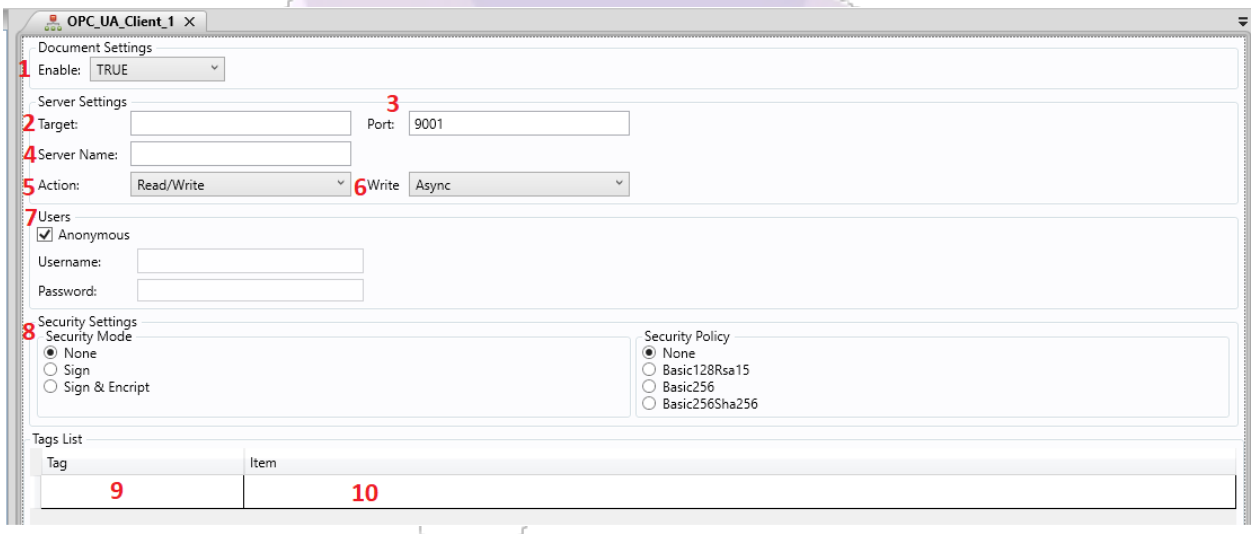
1. In the navigation tree, right-click on the "Driver" document, then on "New Folder" and select the "OPC Clients" option and then click on "OPC UA Client".



2. Now, right-click on the “OPC UA Client” folder and select the “New Document” option. After that, save the application by clicking on the “Save All” button. .



Before continuing with the driver settings step by step, let's get to know each configuration field of the OPC-UA Driver.



1- Enables or disables the Document Driver OPC-UA;
2- Field to enter the URL of the OPC Server;
3- OPC UA Server communication port;
4- Field to enter the hostname, if necessary;
5- Selects the type of tag reading;

6- Selects Asynchronous or Synchronous writing mode;
7- Enables the use of “Username” and “password” to establish communication;
8- Enables the use of a security algorithm in the exchange of messages;
9- Field to insert the ADISRA SmartView tag;
10- Field to insert the OPC-UA Server Simulator tag.

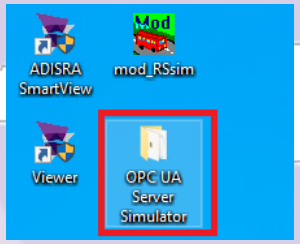
note: *To configure the OPC-UA Driver, we will first need to install and open the OPC-UA Simulator to extract the configuration information. Therefore, the OPC-UA Driver configuration will continue after the simulator configuration.*

11.4.2. Configuring the Simulator

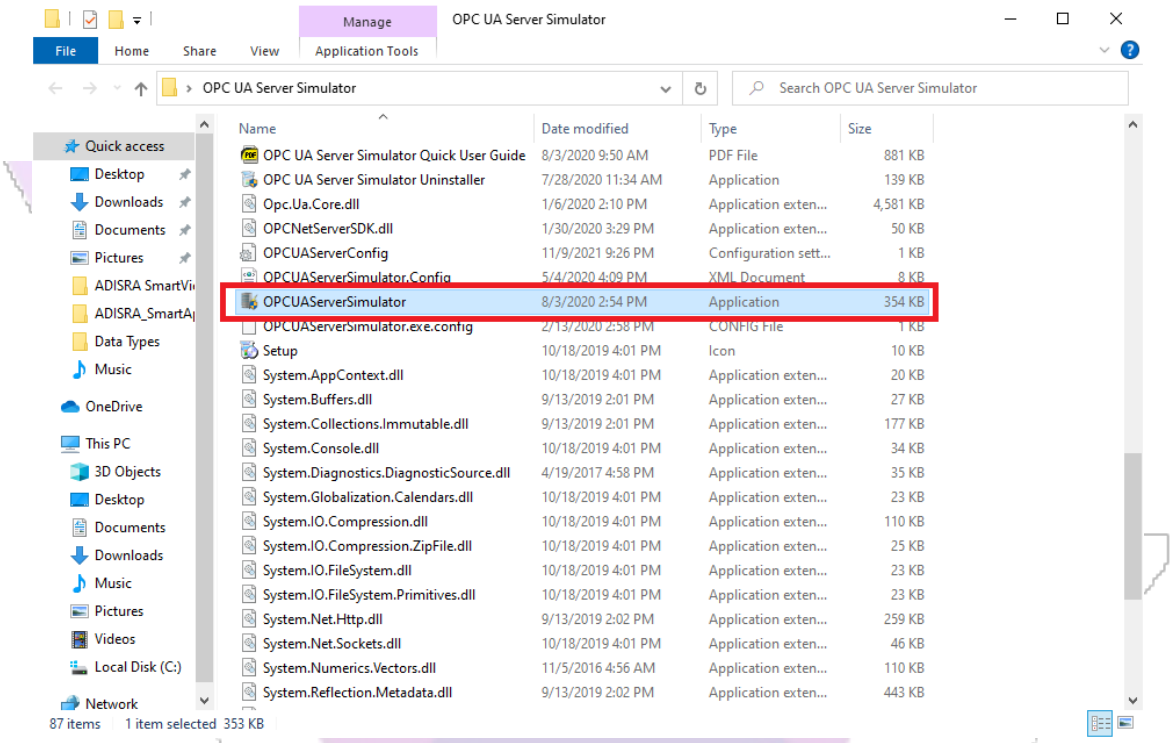
1. Click on the link below and download the OPC-UA simulator.

Link: https://adisra.com/wp-content/uploads/2025/03/ADISRA-SmartView-GettingStarted-Integration%20Objects.zip

2. After downloading, extract the “OPC UA Server Simulator” folder to the desktop.

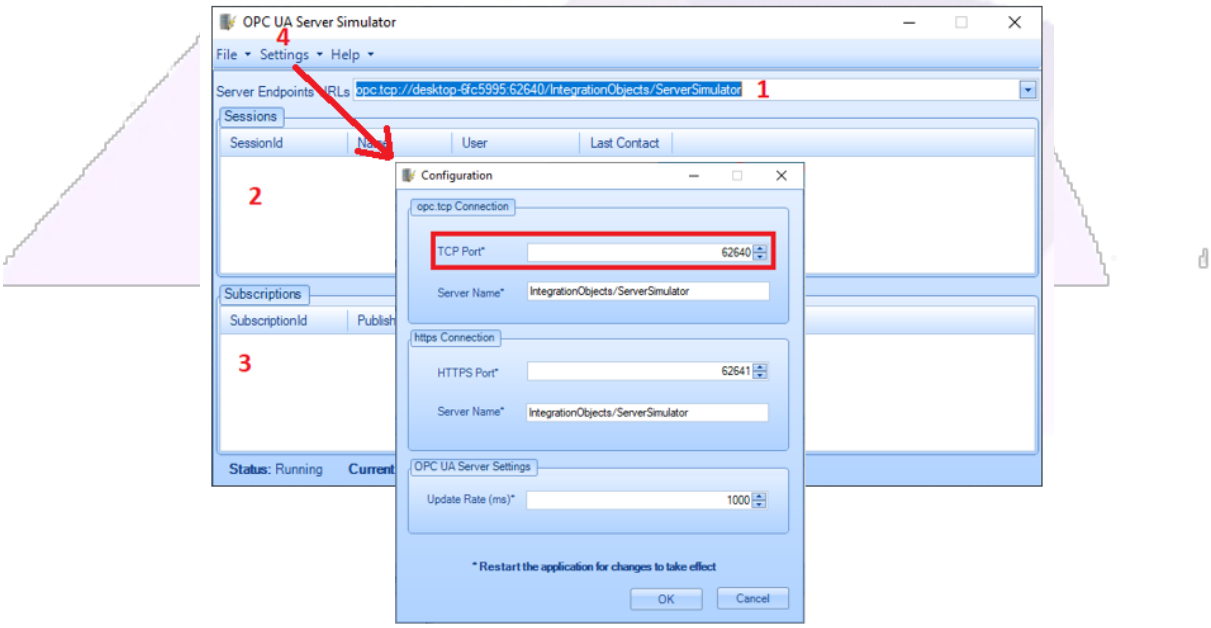


3. Then open the folder and run the simulator through the “OPCUAServerSimulator” file, as shown in the image below.

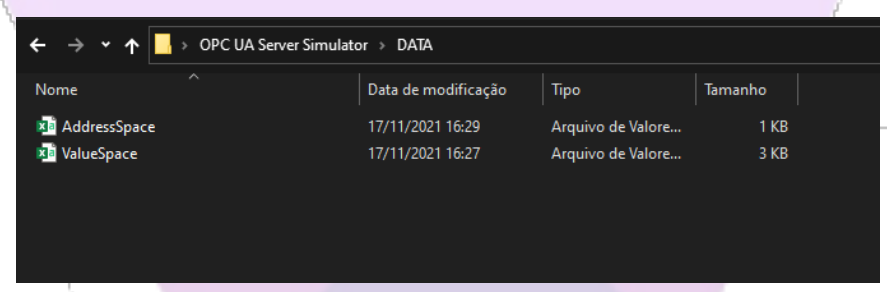


Now, let's get to know the simulator a little.

1- OPC-UA Server communication URL;
2- Shows all active sessions;
3- Shows all active subscriptions;
4- Clicking on “Settings” and then on the “Configuration” option, a window will open showing the TCP Port, HTTP Port, server name and the address update rate.

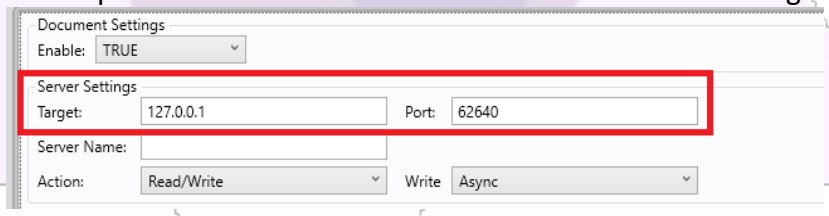


Like the Modbus Driver, we will use this OPC Server simulator to simulate the process variables. Therefore, we will need to have the addresses of the variables in the simulator. This simulator has a limitation. It is not possible to manually change the value of each variable, as we did in the Modbus simulator. Because of this, all variables have already been added to this simulator. To view each variable, open the "OPC UA Server Simulator" simulator folder, then open the "DATA" subfolder. This folder has two documents which can be opened by excel or notepad. The "AddressSpace" document has all the variables created according to the project's tag list. And the document "ValueSpace" has the behavior of each variable every 1000ms. You will not need to change anything in these documents. If you want, click on each document to view the content.



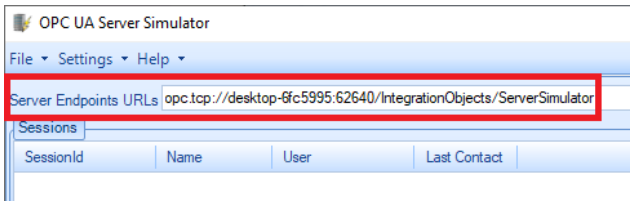
After getting to know the OPC-UA Server Simulator, leave the simulator open and go back to ADISRA SmartView to finish configuring the OPC-UA Client Driver.

1. Let's start configuring the OPC-UA Driver by filling in the "Target" field with the IP "127.0.0.1", which is the IP of the machine where the OPC-UA Server is located. Then change the "Port" field from 9000 to 62640 as shown in the image below. This is the communication port of the OPC-UA Server as seen in the Server configuration window.

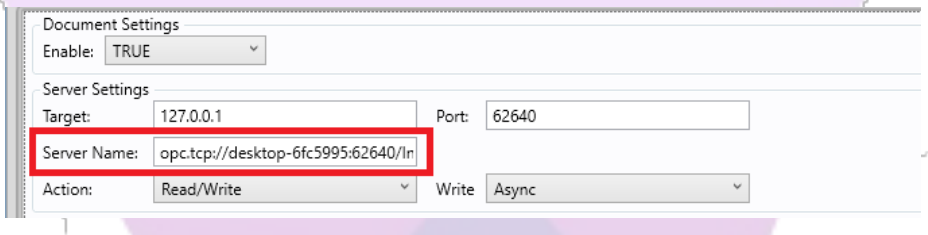


2. Now, let's fill in the "Server Name" field with the communication URL of the OPC-UA Server. To find this URL, go back in the Simulator and copy the "Server Endpoints URLs" as shown below.

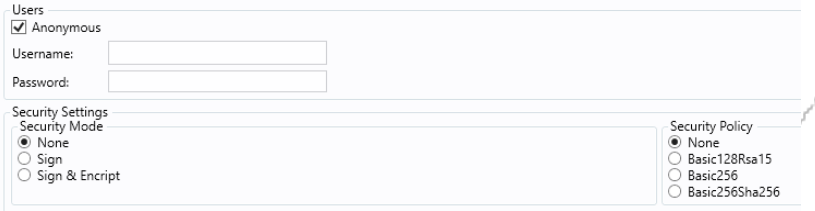
note: *Probably the URL shown below will not be the same as the URL of your simulator, because the URL is composed of the name of the computer on which the simulator is being run and the name of the server.*



3. After copying the URL from the OPC-UA Server, go back to ADISRA SmartView and paste the URL into the "Server Name" field. The Driver configuration should be as shown in the image below.



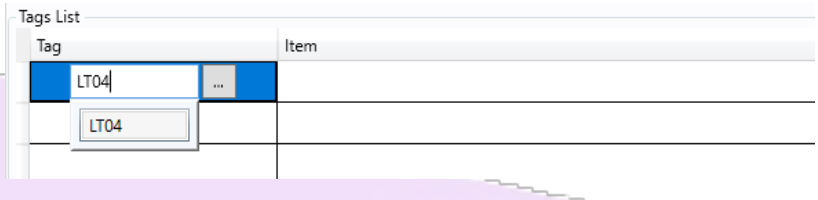
4. The "Users" and "Security Settings" settings should be as shown in the image below.



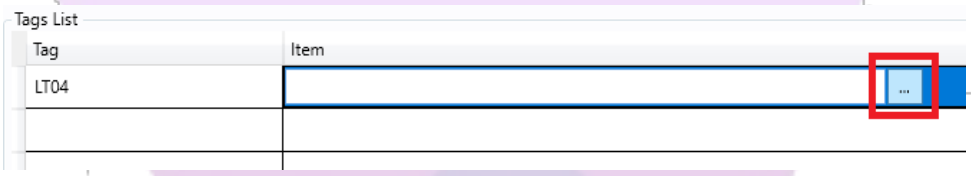
5. Now, let's associate the application's tags with the Simulator's tags. As shown in the image below, the variables highlighted in the red rectangle represent the variables of the PLC02 OPC-UA.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TCP	40001	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TCP	40002	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TCP	40003	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TCP	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TCP	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TCP	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TCP	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TCP	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TCP	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TCP	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TCP	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TCP	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TCP	7	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TCP	8	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TCP	9	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA	-	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_01_FAIL	(Fail) Flow Valve 01 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

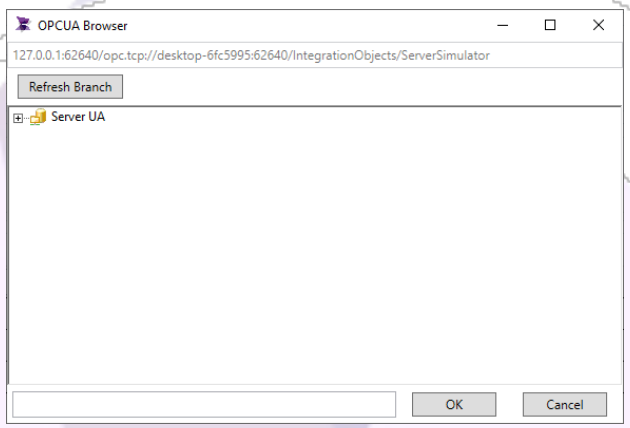
6. Let's start by associating the tag “LT_04”. To do this, click on the first line of the “Tag” column and type “LT04”.



7. Now, let's add the Simulator tag. To do so, double-click the left mouse button on the “Item” column row and then click the “...” button.

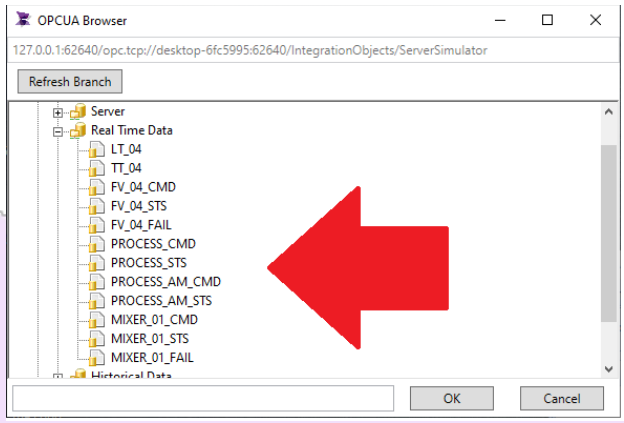


8. If you open the window as shown in the image below, the communication was successful with the OPC-UA Simulator.



note: *If nothing appears, or the error “BadConnectionRejected” appears, the Simulator is probably closed or some communication configuration was not done correctly.*

9. Assuming that the communication was successful, the OPC-UA Simulator tags will be located in the path “Server UA/Objects/Real Time Data”. Then select the tag “LT_04” and click the “OK” button.

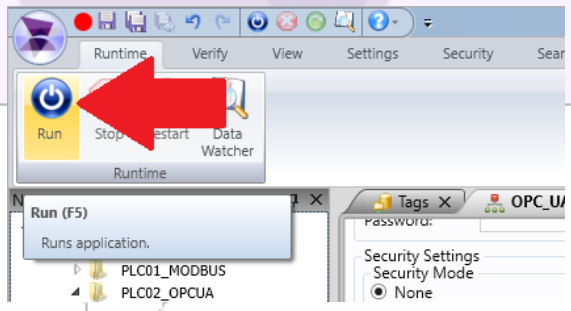


10. Repeat the step by step for all PLC 02 tags. The list of OPC-UA Driver tags should be as shown in the image below.

Tags List

Tag	Item
TT04	ns=2;s=TT_04
LT04	ns=2;s=LT_04
FV04.CMD	ns=2;s=FV_04_CMD
FV04.STS	ns=2;s=FV_04_STS
FV04.FAIL	ns=2;s=FV_04_FAIL
MIXER_01_CMD	ns=2;s=MIXER_01_CMD
MIXER_01_STS	ns=2;s=MIXER_01_STS
MIXER_01_FAIL	ns=2;s=MIXER_01_FAIL
PROCESS_CMD	ns=2;s=PROCESS_CMD
PROCESS_STS	ns=2;s=PROCESS_STS
PROCESS_AM_CMD	ns=2;s=PROCESS_AM_CMD
PROCESS_AM_STS	ns=2;s=PROCESS_AM_STS

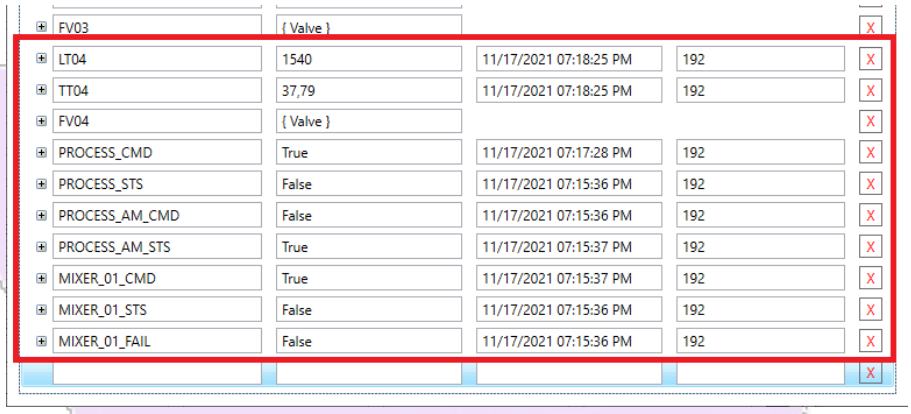
11. After configuring the OPC-UA Driver, save the changes and make sure the OPC-UA Simulator is open. Then start the ADISRA SmartView RunTime.



note[1]: *If the Runtime is already in "Run" mode, restart it by clicking on.*

note[2]: *If the simulator is closed, restart the Runtime.*

12. With the communication established, open the Data Watcher, click on “Load All Tags” and check if the PLC 02 tags have quality “192”.



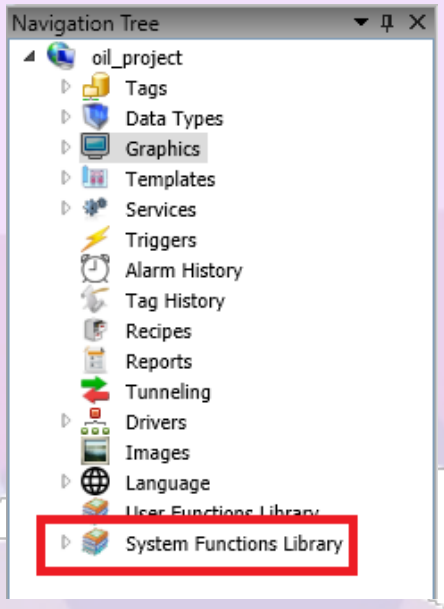
FV03	{ Valve }			X
LT04	1540	11/17/2021 07:18:25 PM	192	X
TT04	37,79	11/17/2021 07:18:25 PM	192	X
FV04	{ Valve }			X
PROCESS_CMD	True	11/17/2021 07:17:28 PM	192	X
PROCESS_STS	False	11/17/2021 07:15:36 PM	192	X
PROCESS_AM_CMD	False	11/17/2021 07:15:36 PM	192	X
PROCESS_AM_STS	True	11/17/2021 07:15:37 PM	192	X
MIXER_01_CMD	True	11/17/2021 07:15:37 PM	192	X
MIXER_01_STS	False	11/17/2021 07:15:36 PM	192	X
MIXER_01_FAIL	False	11/17/2021 07:15:36 PM	192	X

note: *Only the “LT04” and “TT04” tags will be varying their values. The other tags will have a fixed value of “True” or “False”.*

If the entire test was performed as per the above steps, the OPC-UA communication between the Simulator and ADISRA SmartView was successful.

12. System Functions Library Document

The Document System Functions Library aims to provide a collection of system libraries with predetermined functions that the developer can use in the application. These functions cannot be changed or deleted as they are built into ADISRA SmartView. To access Document System Functions Library functions, click on the document located at the bottom of the Navigation Tree.



Next, we will briefly know each function group, because during the development of our application, we will use some functions.

SVAlarms – The functions of this group are intended to handle application alarms, such as creating or deleting an Alarm document, acknowledging a particular alarm, disabling and enabling alarms, etc.

SVApplications – The functions of this group are intended to manipulate the application, such as returning the application directory, disconnecting the viewer, pausing all server modules, restarting a certain module, stopping or starting the viewer, etc.

SVDBConnection - The functions of this group are intended to manipulate the database, such as creating a database, creating a table and using the main functions (delete, insert, select and update).

SVDrivers - The functions in this group are intended to manipulate driver documents, such as creating or deleting a new driver document, adding tags to a specific driver document and configuring the reading and writing of the driver.

SVEvent - The functions of this group are intended to handle application events, such as saving or removing a new event in the system, sending an event, configuring events, etc.
SVFile - The functions of this group are intended to manipulate a text file, such as creating a new ".txt" file in a given directory, copying or deleting a file, writing a text in the file, and so on.

SVGraphics - The functions of this group are intended to manipulate Graphic documents, such as opening or closing a graphic, copying the current mouse position, etc.

SVHistory -The functions of this group are intended to manipulate the Historic Tag document, such as exporting the history to ".CSV", adding or removing tags from the history, updating tags by deadband and so on.

SVMath - The functions of this group are intended to perform mathematical calculations, such as calculating area and volume, log, angle Sin, Cos, Tan, etc.

SVRecipe - The functions of this group are intended to manipulate the Recipe document, such as creating or removing a new recipe document, adding or removing recipe tags, loading and saving the recipe, etc.

13. Graphics Document

The Graphic Document is a user interface object, and can contain various types of other objects, which can be deleted, copied, moved, resized, grouped, or configured. It is also possible to insert script using the C# programming language, for example, executing a script on opening the Graphic object, on closing or while it is open.

Now that we have the application's tags in communication with the PLC's, let's develop the graphic screens of our Oil project and then associate the tags to the screen objects.

14. Developing the Graphic Main

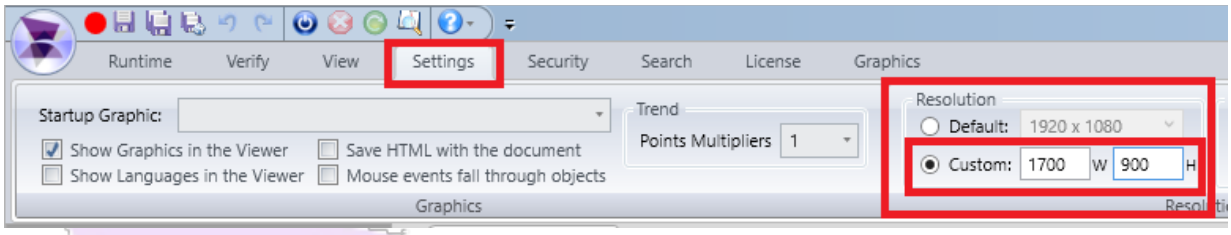
Let's start the development of the screens by the Graphic “Main”. This screen will be the first one to open when RunTime starts. It will be the screen where the alarm summary, navigation buttons, title of the current screen, login button, logoff button, date and time, name of the logged in user and most importantly, the Screen object, which will be responsible for displaying the selected screen on the navigation buttons.

Below is the demonstration of how the Graphic Main should be.

14.1. Setting Screen Resolution

Before we add any Graphics to our application, we must define the resolution of all future screens.

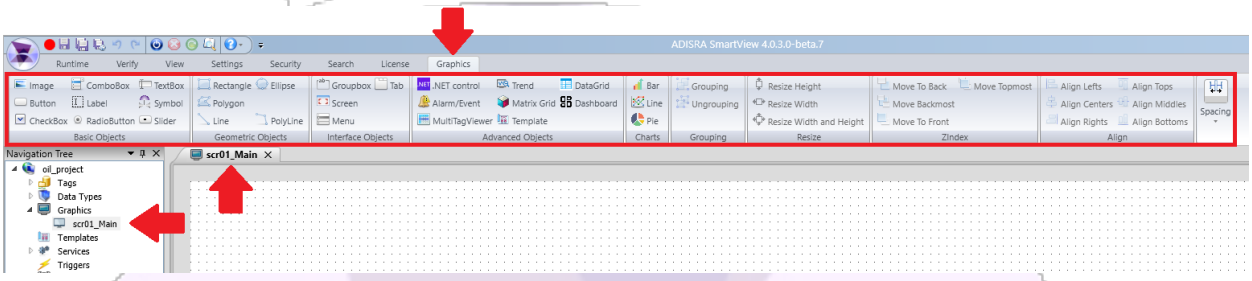
1. With ADISRA SmartView open, click on the “Settings” menu, then click on “Custom” and add the “1700 x 900” resolution. From now on, all new graphics screens will be created with this default resolution. But if you want, you can change the resolution of each screen during development.



note: *If you want, you can choose another screen resolution according to your monitor. But for this training, we recommend that you keep this resolution, as the entire application will be developed in "1700 x 900".*

14.2. Creating the Graphic Main

While the Document of type "Graphic" is open, a new menu called "Graphic" will be available at the top of the ADISRA SmartView, presenting various features such as basic objects, geometric objects, interface objects, advanced objects, charts, resources for grouping, moving and aligning objects. Therefore, if you want to add any object to a screen, click on the type of object to be inserted, for example the "Button" object, and then hold the left mouse button on the screen and drag to the desired size.



note: *If you have added any object on the "scr01_Main" screen, delete it, as we will add the objects to our application later in the training. To delete, select the object and then click on the "delete" button located on the keyboard.*

Below the screen area, we can select the "View Mode". We can view the screen in "Design", "Script" and "Split" mode.

1- Design Mode, graphically presents the screen itself and all added objects;

2- Script mode, presents the selected object's script in textual form;

3- Split Mode, features Design Mode and Script split horizontally;

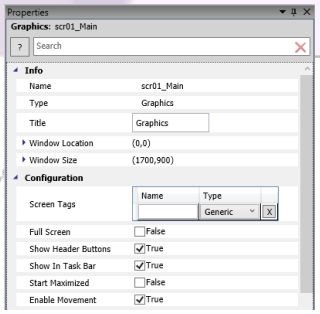
4- Area for script development using the C# language;

5- In the “On Open” event, a developed script will be triggered when the screen opens;

6- In the “On While” event, a developed script will be triggered while the screen is open;

7- In the “On Close” event, a developed script will be triggered when the screen closes;

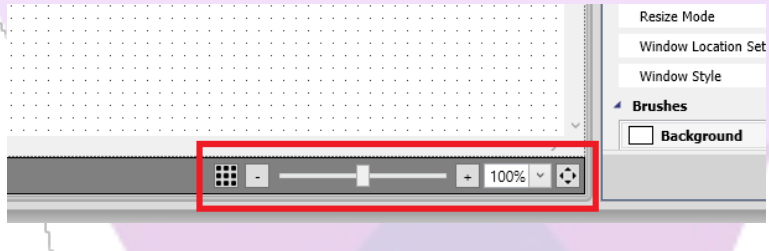
8- List with all screen objects “scr01_Main”.



On the right side of the canvas area, we have the property window for the selected object. In the image obove, for example, the screen “scr01_Main” belongs, containing properties such as object name, object type, object title,

location, screen resolution and several other properties that we will know during the training.

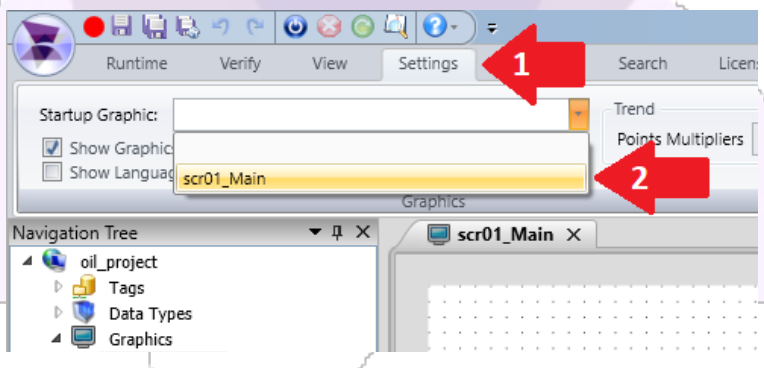
In the lower right corner of the screen area, we can enable or disable the “Grid” property of the screen, and then the “Zoom” tool, used to facilitate the visualization of the screen area. With the screen object selected, it is also possible to change the “Zoom” of the screen according to the keyboard's CTRL button + Mouse Scrolling.



14.3. Defining the Home Screen

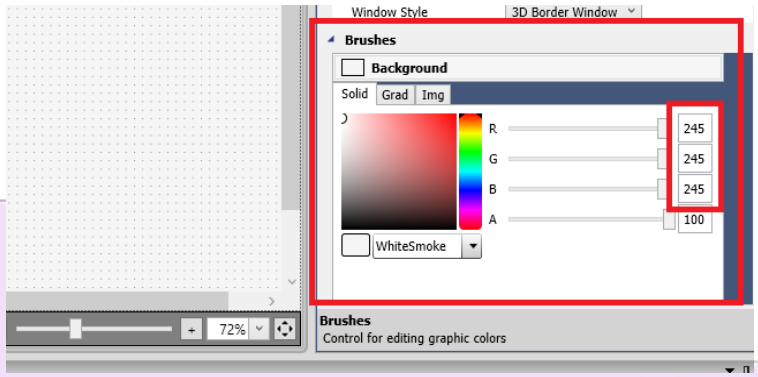
Now, let's configure in ADISRA SmartView the initial screen for the application, that is, every time RunTime is started, a screen will be opened automatically. According to our application, we need to make the screen “scr01_Main”as the home screen.

1. On the top menu, click on the “Settings” option and then select the “scr01_Main” screen.



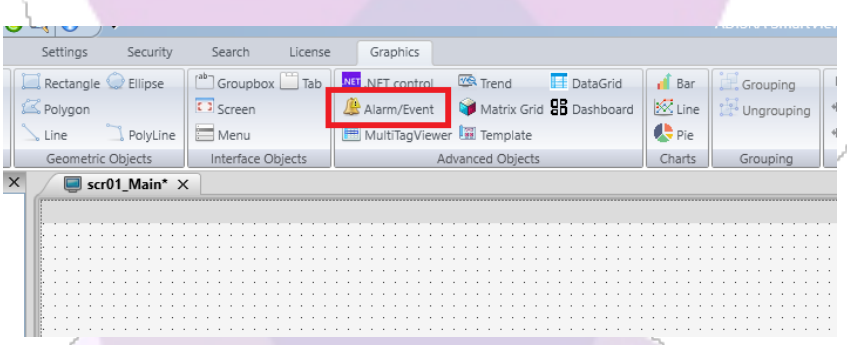
14.4. Configuring Backgrounds

Select the “scr01_Main” screen and in the properties list, find the “Brushes” property, select the “Solid” tab and then change the colors of the RGB palette to (245, 245, 245) as shown in the image below. This will be the default background color for our application.

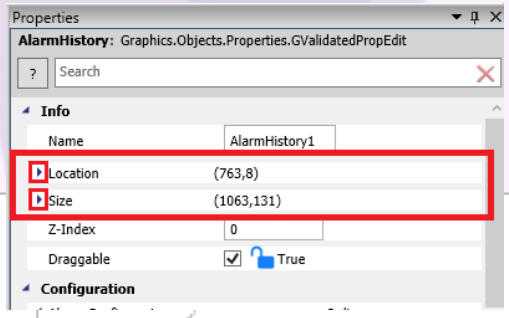


14.5. Inserting Alarm/Events

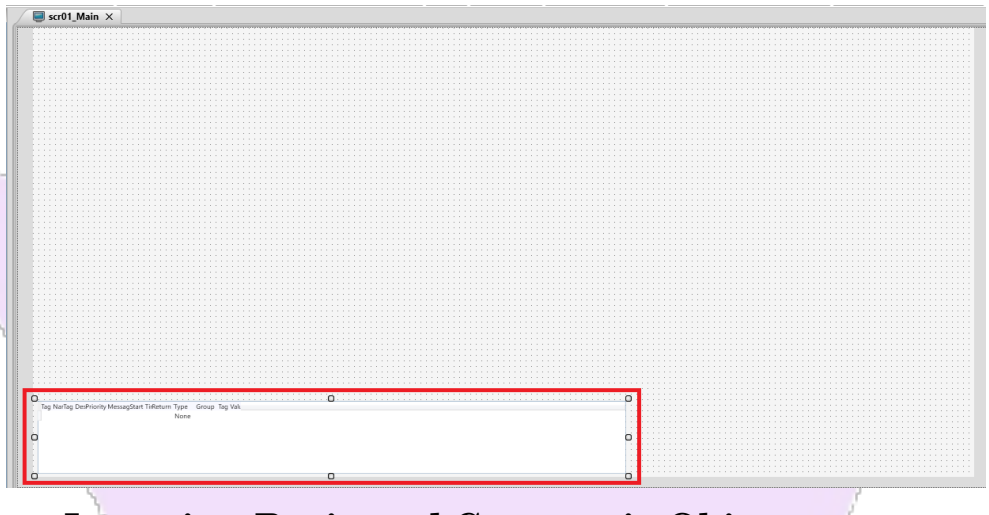
1- In the “Graphics” menu, click on the “Alarm/Event” option and then add “scr01_Main” on the screen.



2- Select Alarm/Event and change the following properties: Location (763, 8) and Size (1063, 131).

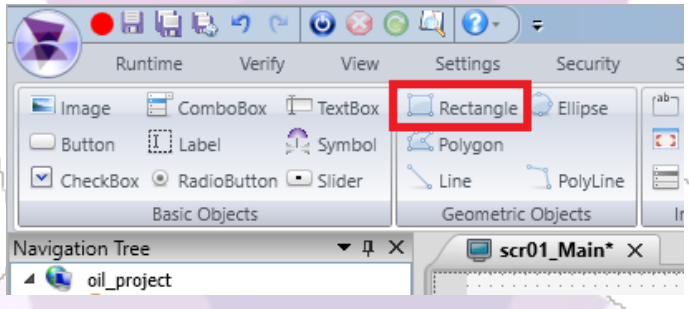


3- Below is the expected result of the Alarm/Event settings.



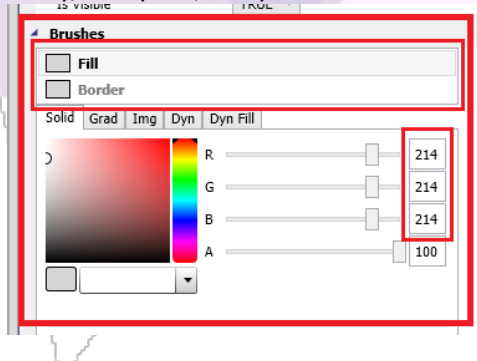
14.6. Inserting Basic and Geometric Objects

1- In the “Graphics” menu, click on the “Rectangle” option and then add “scr01_Main” to the screen.



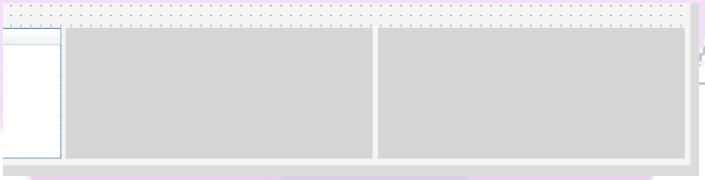
2- Select the “scr01_Main” screen and in the properties list, find the “Brushes” property, select the “Solid” tab and then change the colors of the RGB palette to (214,214,214). Also change the following properties:

Location(763, 1075), Size(307, 131).

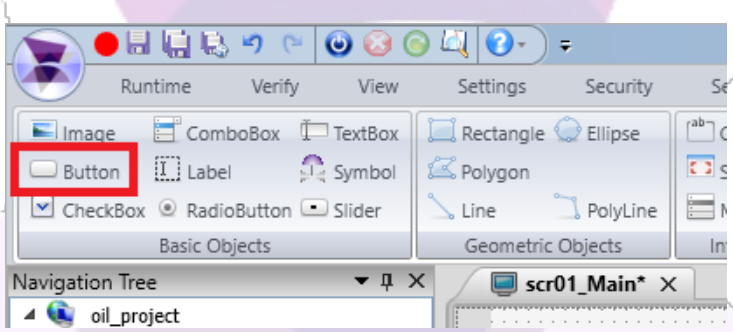


3- Below is the expected result of the Rectangle settings.

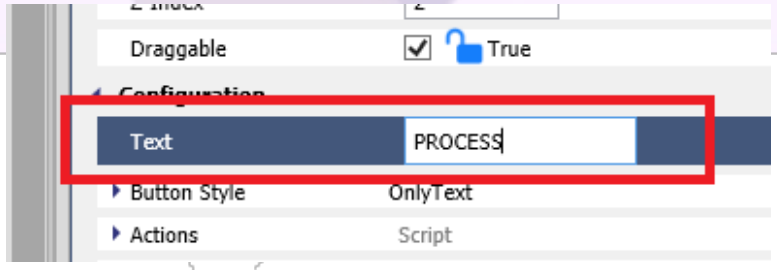
4- Add one more rectangle with the following settings:
Fill/Solid (214, 214, 214), Location (763, 1388), Size (307,131).
Below is the expected result of the rectangles settings.

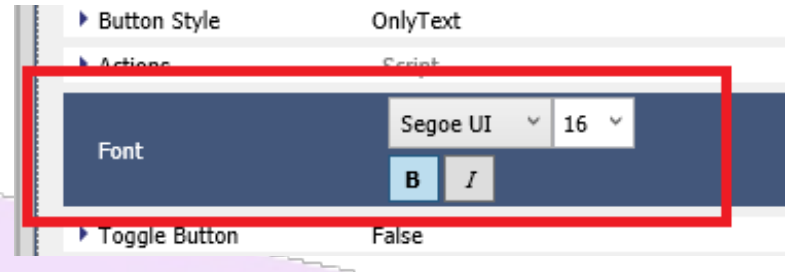


5- Next, we'll add the screen navigation buttons. For that, in the “Graphics” menu, click on the “Button” option and then add “scr01_Main” on the screen.

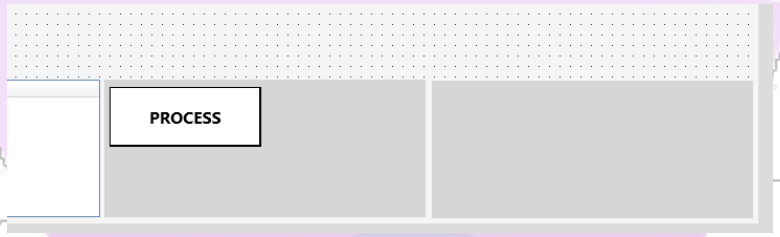


6- Select the button added on the screen and change the following properties:
Location (770, 1080), Size (145, 57), Text (PROCESS) and Font (Segoe UI, 16, Bold).





7- Below is the expected result of the “Process” button settings.



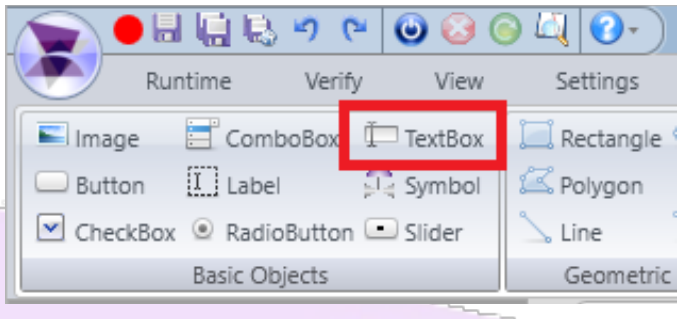
8- Now create five more buttons according to the settings below:

History button: Location (770, 1232), Size (145, 57), Text (TREND) and Font (Segoe UI, 16, Bold).
Recipe button: Location (831, 1080), Size (145, 57), Text (RECIPE) and Font (Segoe UI, 16, Bold).
Alarm button: Location (831, 1232), Size (145, 57), Text (ALARM & EVENT) and Font (Segoe UI, 16, Bold).
Login button: Location (770, 1393), Size (145, 32), Text (LOGIN) and Font (Segoe UI, 16, Bold).
Logoff button: Location (807, 1393), Size (145, 32), Text (LOGOFF) and Font (Segoe UI, 16, Bold).

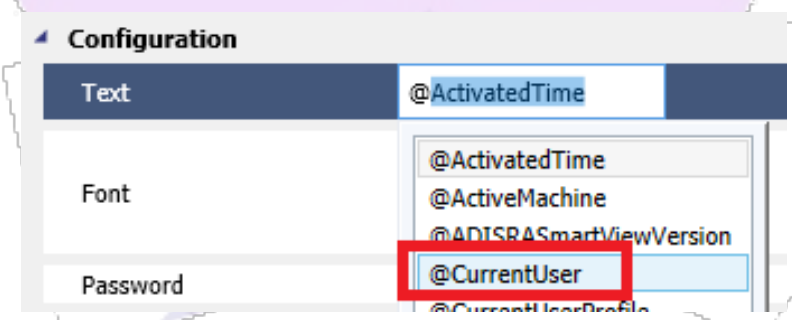
9- Below is the expected result of the navigation buttons settings.

note: *The “Login” and “Logoff” buttons will be configured in the “Security and User” chapter.*

10- In the “Graphics” menu, click on the “TextBox” option and then add “scr01_Main” on the screen.



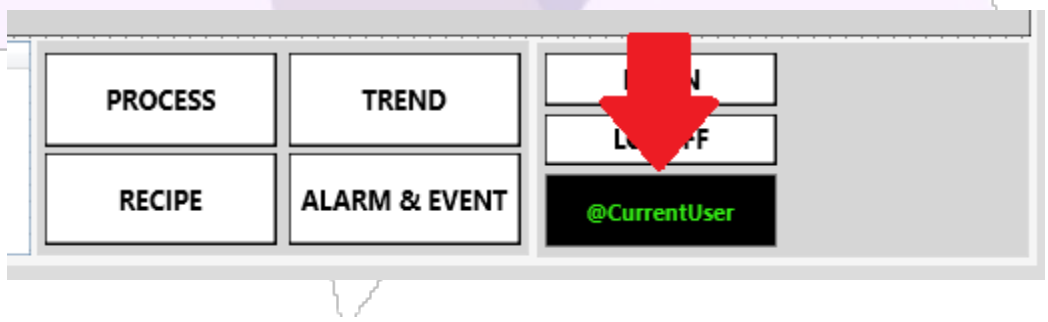
11- Select the TextBox and change the following properties:
Location (844,1392), Size (145,45) and Text (@CurrentUser, Segoe UI, 14, Alignment Center), (backgroung/Solid 0, 0, 0), (foreground /Solid 52, 237, 21).



note[1]: *Note that this is the first time we associate a tag with an object. Every tag can be associated with a property of an object using the "@" character before any tag. In the TextBox Object above, we associated it with a system tag. This tag can be found in the document "System Tags", located in the navigation tree.*

note[2]: *The tag "@CurrentUser" shows the current user in the application. By default, the initial user is the "Guest" user.*

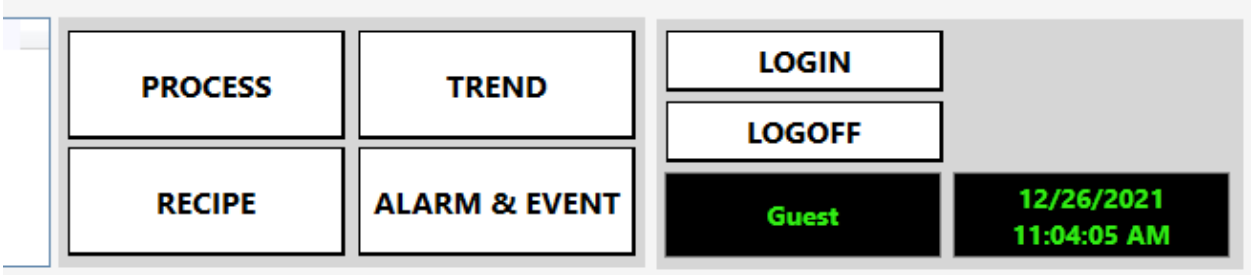
12- Below is the expected result of the TextBox CurrentUser settings.



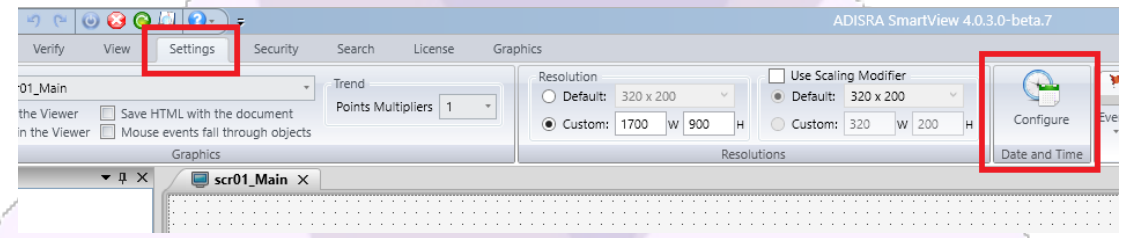
13- Add one more TextBox with the following settings:
Location (844,1543), Size (145,45) and Text (@DateTime, Segoe UI, 14, Alignment Center), (background/Solid 0, 0, 0), (foreground/Solid 52, 237, 21).

Below is the expected result of the TextBox DateTime settings.

After adding and configuring the objects above on the “scr01_Main” screen, let's save the application and start RunTime.



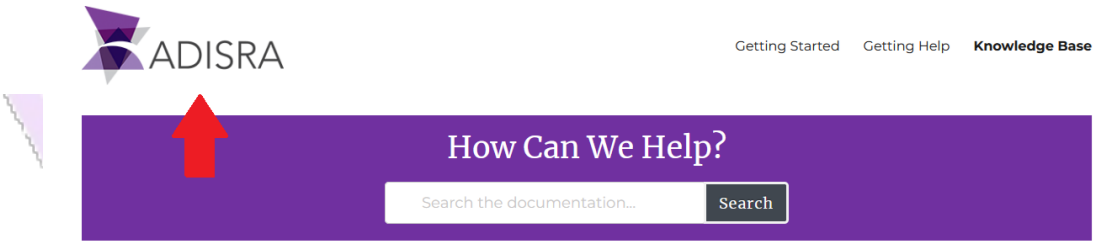
Note that the TextBox @DateTime displays the Windows date and time. ADISRA SmartView allows you to change the displayed format. To do this, go back to the development environment, in the top menu, select the “Settings” option and then click on the “Configure Date and Time” option.



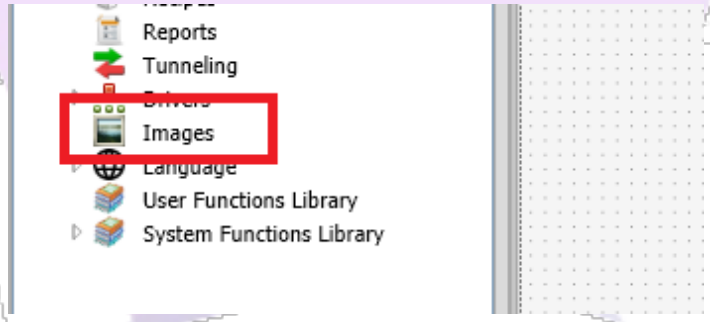
14.7. Inserting Images

ADISRA SmartView allows you to add images in the application. The allowed formats are: .bmp, .ico, .jpg, .png, .jpeg, .tif, .gif, .jpe, .jfif, and .tiff. In the hypothetical project, we will add the ADISRA logo at the bottom of the “scr01_Main” screen.

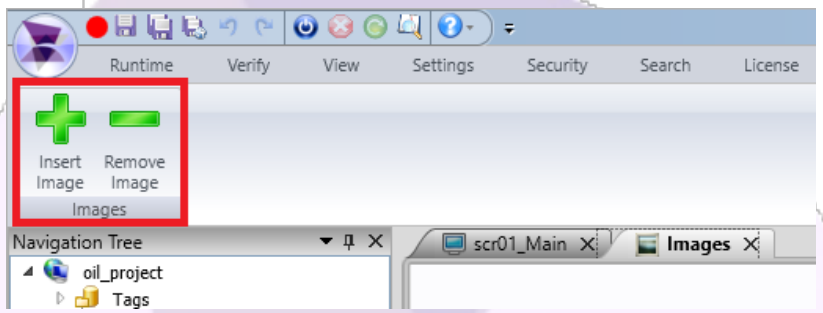
1- Open your internet browser and type in the browser the website “https://guides.adisra.com “. Once this is done, right-click on the ADISRA logo and save it to your desktop.



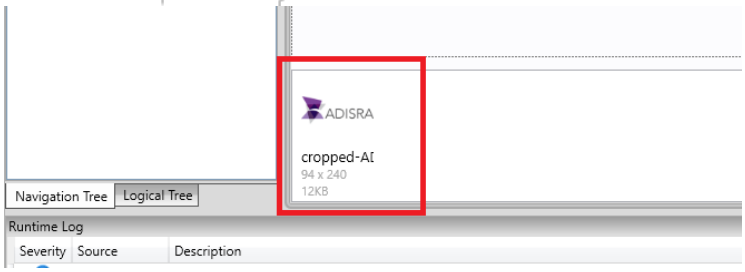
2- Return to ADISRA SmartView. Now, let's add the logo to the application. To do this, in the navigation tree click on the document “Images”.



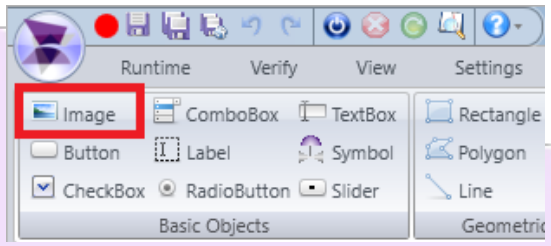
3- Then, at the top, click on “Insert Image”. Look for the logo and click “Open”.



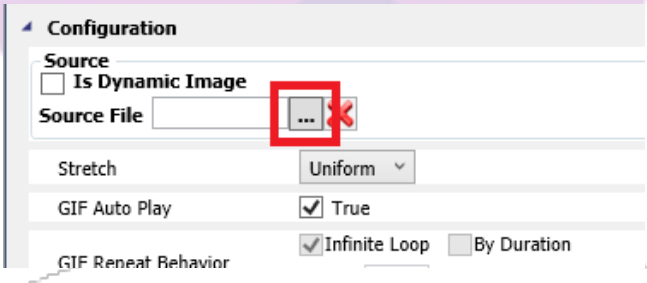
4- Note that the logo was added to the application's image bank. Therefore, the added image has no link to the desktop image. If desired, delete the image saved on the desktop.



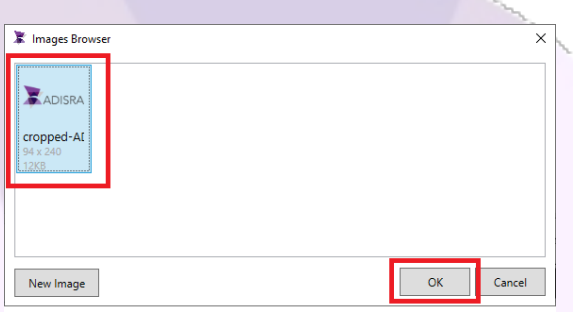
5- Now, let's add the ADISRA logo to the “scr01_Main” screen. To do this, go back to the screen and click on the “Graphics” menu and then on the “Image” option. Then add the image to the canvas.



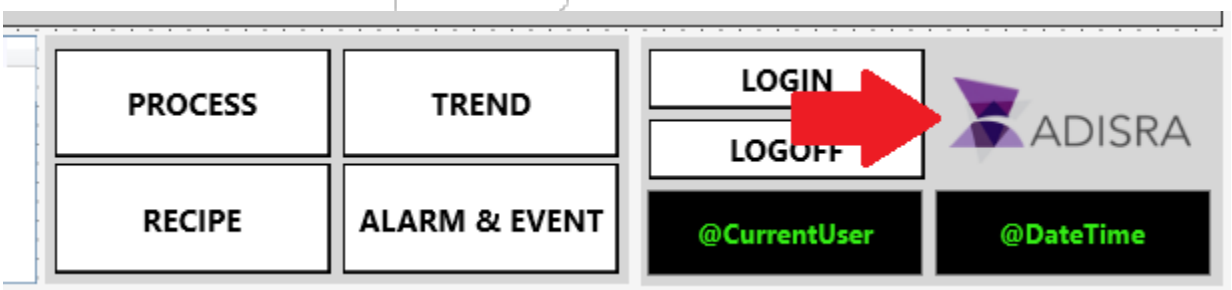
6- Select the default image, in the properties list, click on the Source File property, as shown in the image below.



7- Select the logo and then click “OK”.

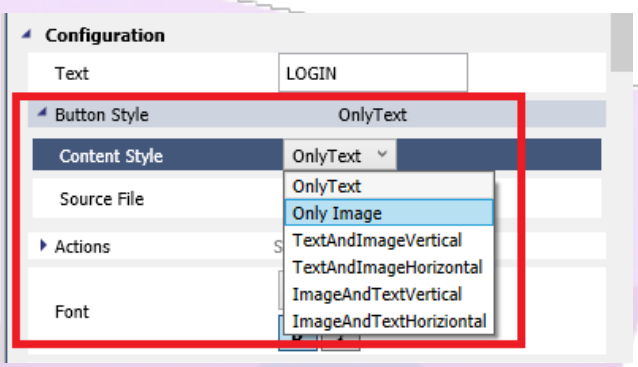


8- Select the logo and change the following properties:
Location (783, 1554), Size (127, 46) and Strech (Fill). Below is the expected result of the image settings.

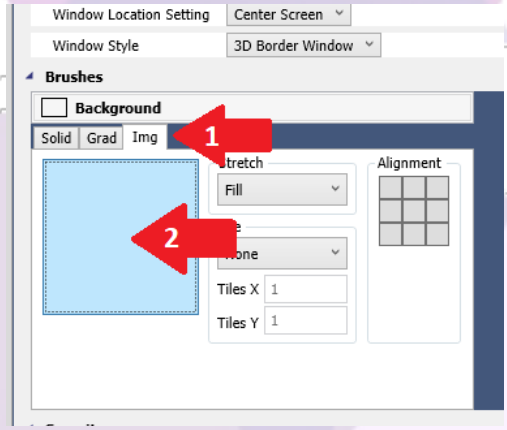


ADISRA SmartView also allows you to add images to buttons and screen backgrounds.

If you want to add an image to the background of a button, select the specific button, in the "Button Style" property, select the desired style and then point to the added image in the image bank.



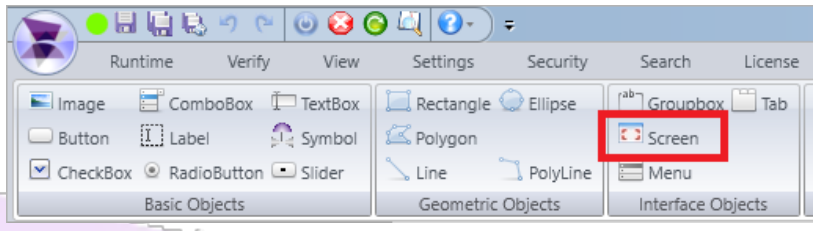
If you want to add an image to the background of a screen, select the screen, in the "Brushes" property, click on the "Img" tab and then click on the blue frame as shown in the image below.



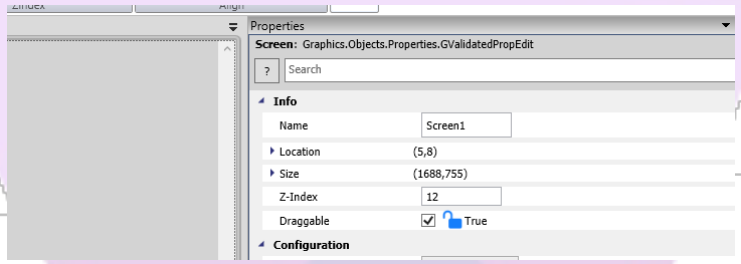
14.8. Inserting Screen Object

A Screen object lets you open different Graphs within its area one at a time. This allows the user to switch between different graphs during RunTime accordingly, for example, with the click of a button. As the "scr01_Main" screen is our main screen, we must add a Screen object to it to allow navigation between the application screens.

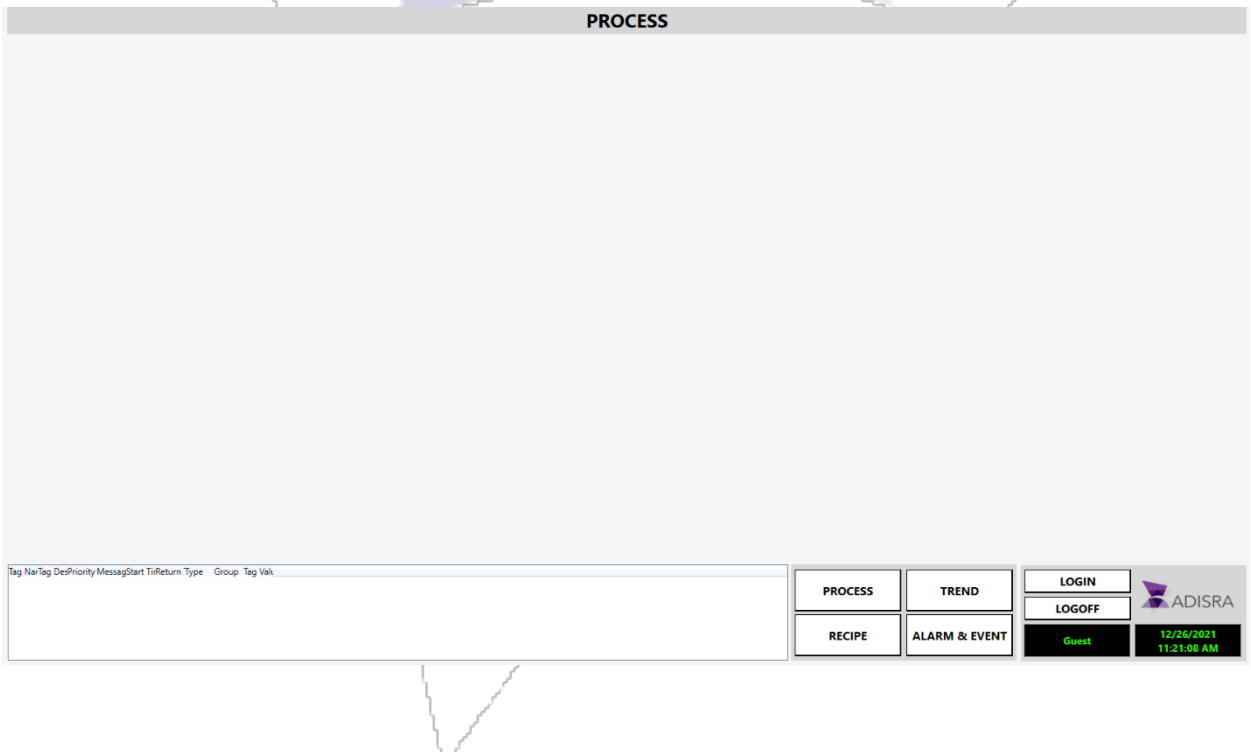
1- To add a Screen object, select the "scr01_Main" screen, click on the "Graphics" menu, then click on the "Screen" option and then add "scr01_Main" to the screen.



2- Select the Screen Object and change the following properties: Location (5, 7) and Size (1688,725).



3- Below is the expected result of the Screen Object settings.

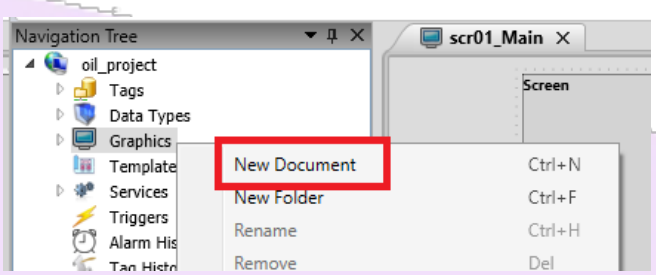


14.9. Inserting Screen Navigation

Now that we've added the Screen object to the “scr01_Main” screen, we need to create the other navigation screens and configure the navigation buttons we added earlier. First, we will

create the “Process”, “Trend”, “Recipe” and “AlarmEvent” screens and then we will configure the screen navigation settings.

1- Let's create the “Process” screen. To do this, click on the “Graphics” document, then click on the “New Document” option. Save the screen with the name “scr02_Process”.

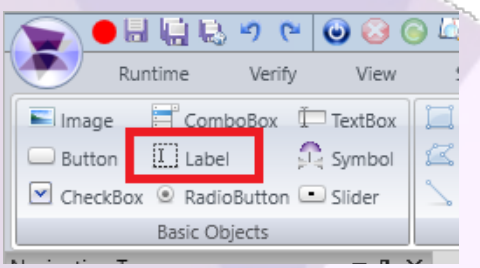


2- Select the “scr02_Process” screen and change the following properties: Size (1688, 755), background (245, 245, 245).

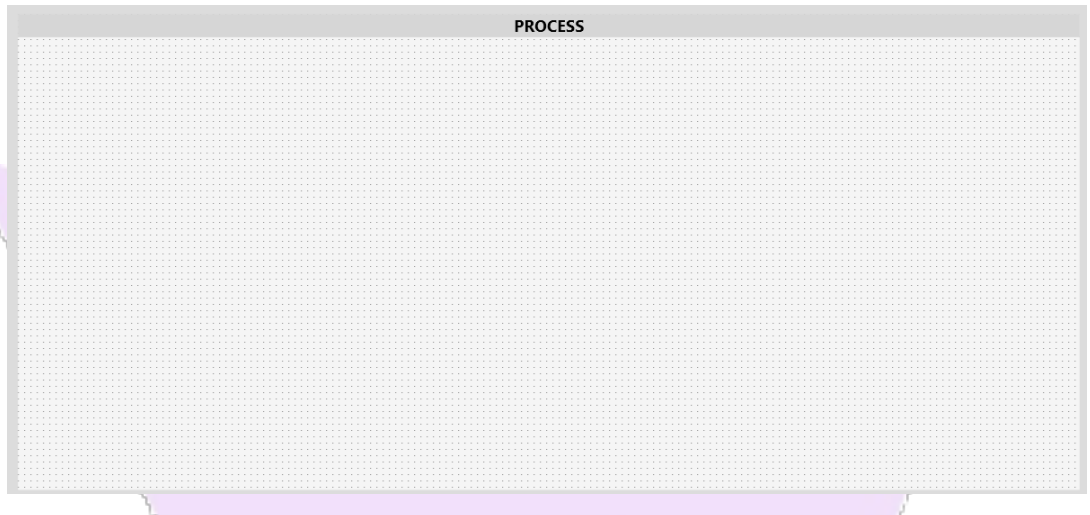
3- Then add a rectangle with the following properties:

Location (0, 0), Size (1688, 37), Brushes/Fill (214, 214, 214) and Brushes/Border (214, 214, 214).

4- Finally, add a Label object with the following properties: Location (0, 705), Size (279, 37) and Text (PROCESS, 26, Bold).



5- Below is the expected result of the settings on the “scr02_Process” screen.



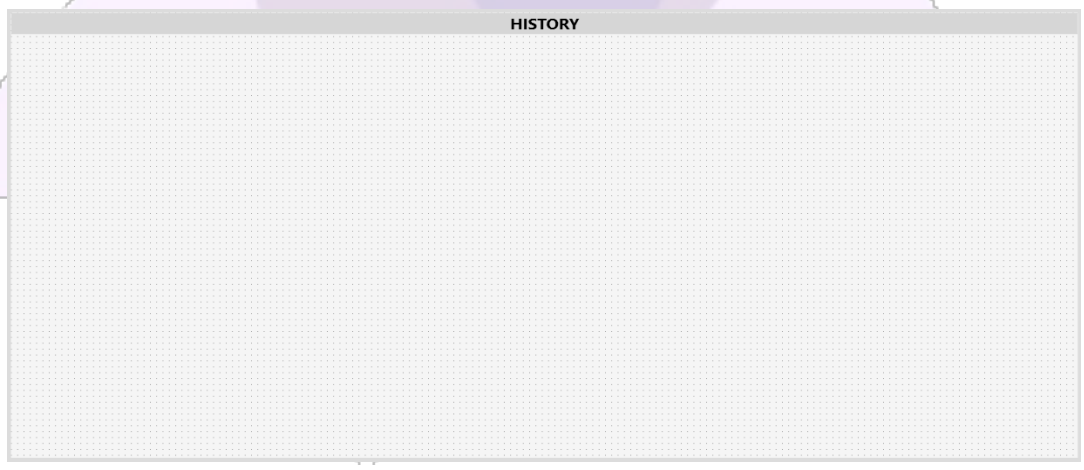
6- Repeat settings for the “Trend” screen. To do this, click on the “Graphics” document, then click on the “New Document” option. Save the screen with the name “scr03_Trend”.

7- Select the “scr03_Trend” screen and change the following properties: Size (1688, 755), background (245, 245, 245).

8- Then add a rectangle with the following properties: Location (0, 0), Size (1688, 37), Brushes/Fill (214, 214, 214) and Brushes/Border (214, 214, 214).

9- Finally, add a Label object with the following properties: Location (0, 705), Size (279, 37) and Text (TREND, 26, Bold).

10- Below is the expected result of the “scr03_Trend” screen settings.



11- Repeat the settings for the “Recipe” screen. To do this, click on the “Graphics” document, then click on the “New Document” option. Save the screen with the name “scr04_Recipe”.

12- Select the “scr04_Recipe” screen and change the following properties: Size (1688, 755), background (245, 245, 245).

13- Then add a rectangle with the following properties: Location (0, 0), Size (1688, 37), Brushes/Fill (214, 214, 214) and Brushes/Border (214, 214, 214).

14- Finally, add a Label object with the following properties:

Location (0, 705), Size (279, 37) and Text (RECIPE, 26, Bold).

15- Below is the expected result of the “scr04_Recipe” screen settings.

16- Repeat the settings for the “AlarmEvent” screen. To do this, click on the “Graphics” document, then click on the “New Document” option. Save the screen with the name “scr05_AlarmEvent”.

17- Select the “scr05_AlarmEvent” screen and change the following properties: Size (1688, 755), background (245, 245, 245).

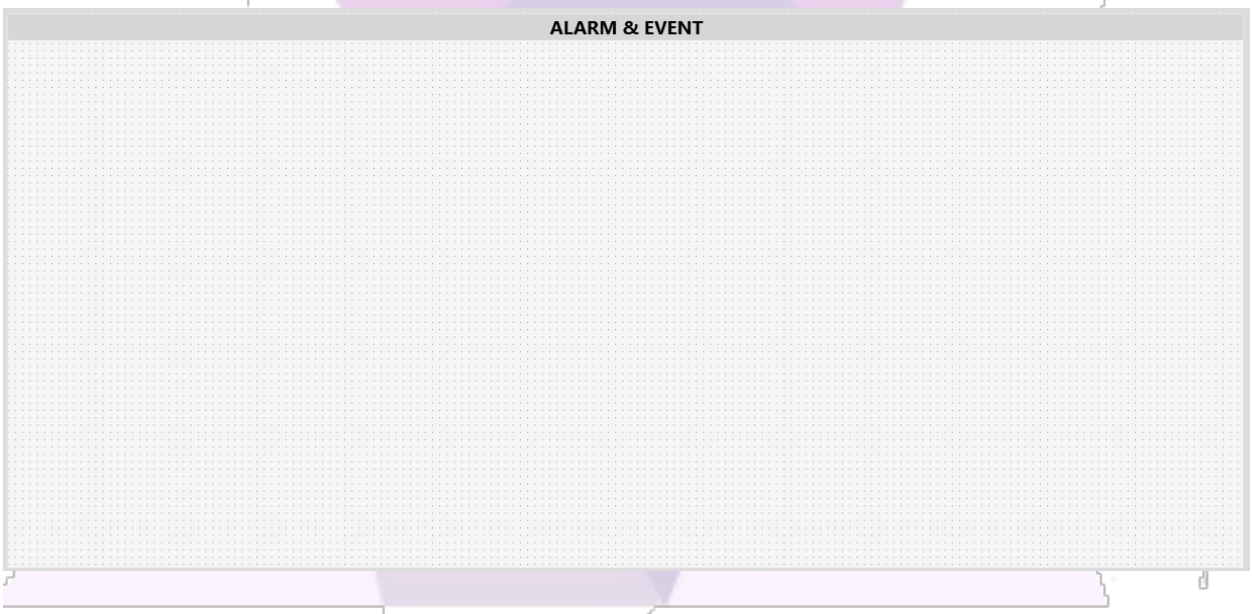
18- Then add a rectangle with the following properties:

Location (0, 0), Size (1688, 37), Brushes/Fill (214, 214, 214) and Brushes/Border (214, 214, 214).

19- Finally, add a Label object with the following properties:

Location (0, 705), Size (279, 37) and Text (ALARM & EVENT, 26, Bold).

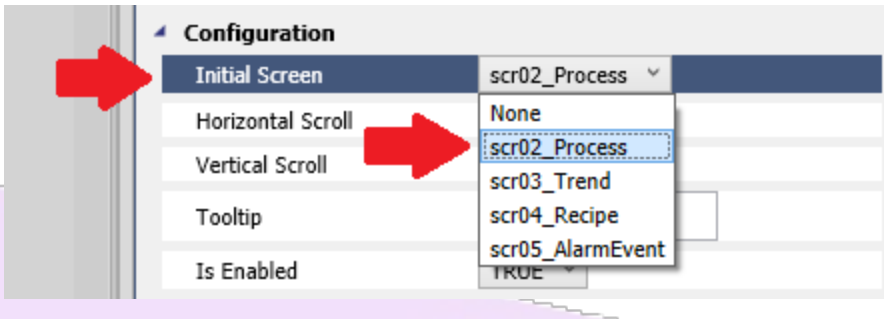
20- Below is the expected result of the “scr05_AlarmEvent” screen settings.



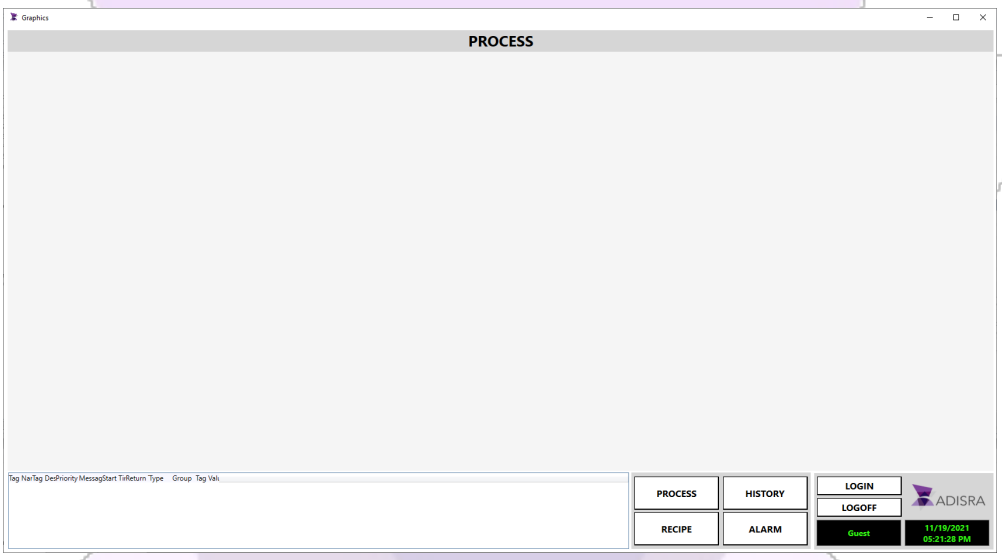
After creating all the screens of our application, let's define the “scr02_Process” screen as the initial screen of the Screen Object and the configuration of each navigation button.

note: *The content itself of each screen will be left to develop in the following chapters.*

1- To define an initial screen in the Screen object, open the “scr01_Main” screen and then choose the “scr02_Main” screen in the “Initial Screen” property. From now on, when we start RunTime, the “Scr01_Main” screen will open automatically, and within this screen, the Screen object will start the “scr02_Process” screen.

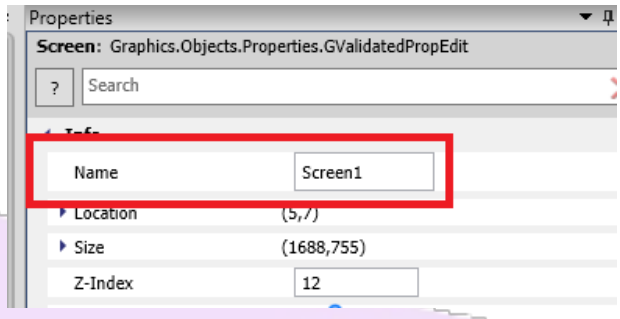


2- So, let's test it! Save the application and then start RunTime. The image below shows the expected result. Then close Viewer and stop RunTime.

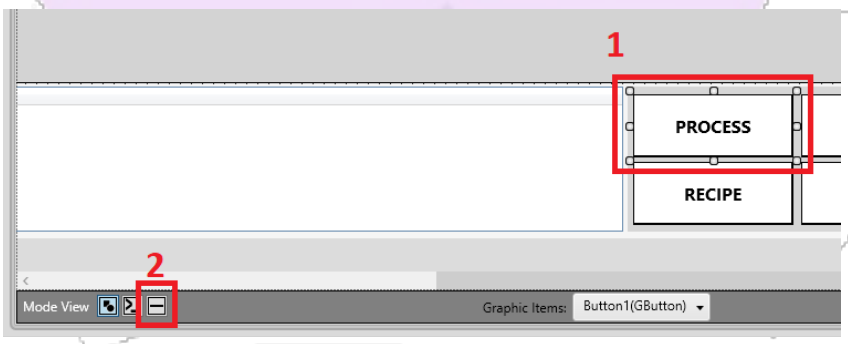


Now, we need to configure the navigation buttons located on the “scr01_Main” screen. Basically, we will add scripts in the “Mouse Down” event of each navigation button to change the current screen of the Screen Object.

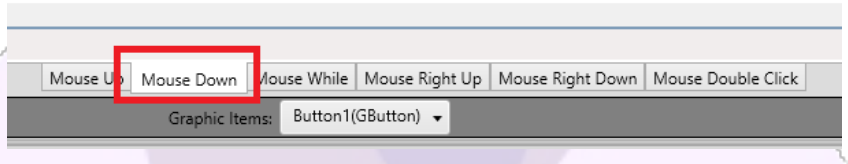
1- Before adding the first script, open the “scr01_Main” screen and then click on the Screen object on which the screens will be displayed. In the list of properties, check the name of the Screen Object. According to the image below, and probably from your application, the name is “screen1”. Make a note of this name, as we will use it in the button script.



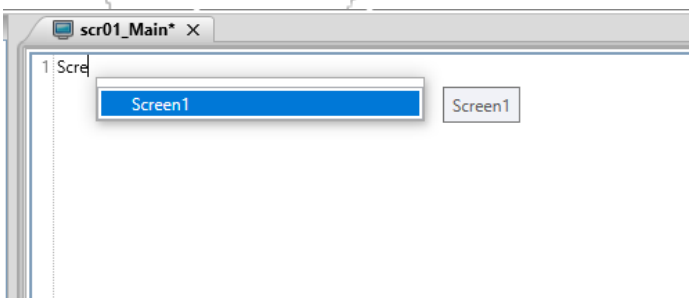
2- Now, let's configure the first button. Click on the “Process” button and then open the “Script” view, as shown in the image below.



3- Click on the “Mouse Down” event.

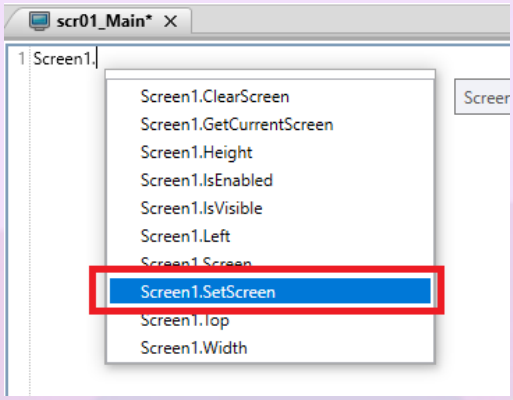


4- On line 01, add the script to pass the name of the screen to be opened in the Screen object. Therefore, start typing the object's name and then the object's name will appear in a list. Click on the “Screen1” object.

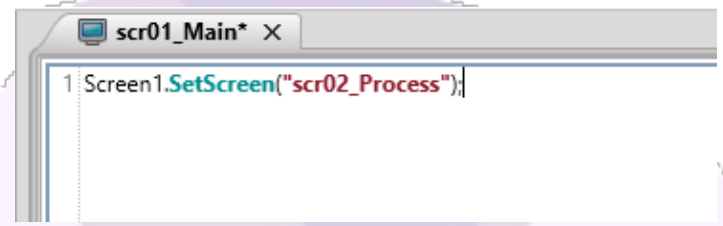


note: *For the above list to appear, the first letter of the word "Screen" must be capitalized.*

5- After adding the object name, we need to access the "SetScreen" property of the Screen object. To do this, type "." after "Screen1" and a list will open with the properties on which we can interact. Click on the "SetScreen" property.

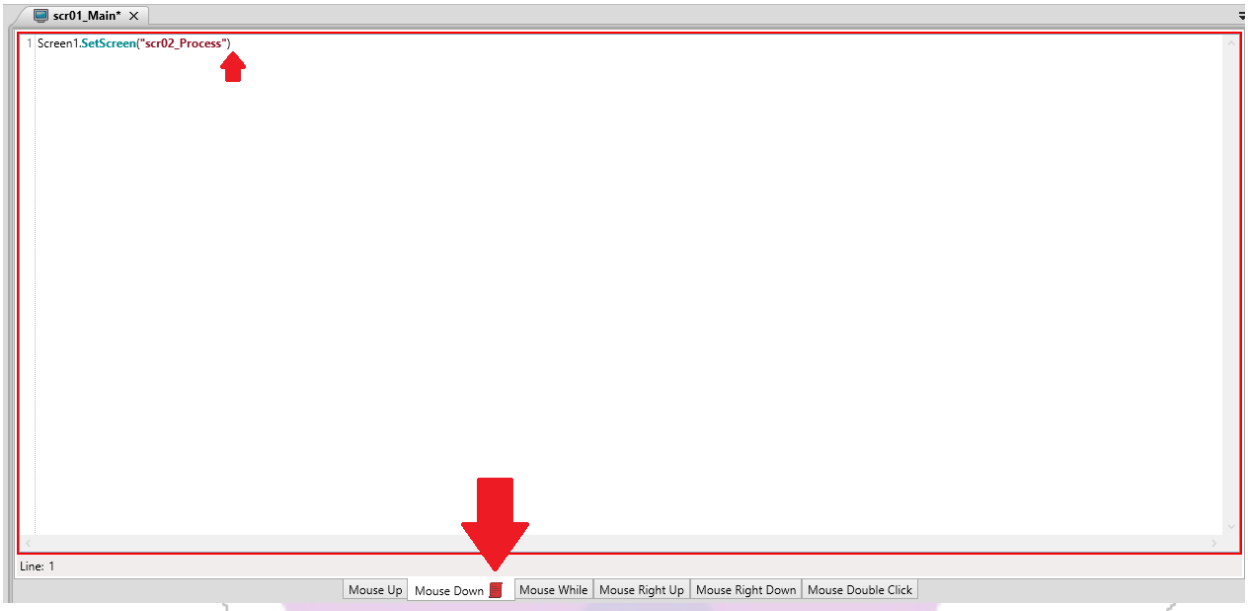


6- Now we need to pass the name of the screen on which it will be loaded in the Screen object. Then, after the "SetScreen" property, type: ("scr02_Process");

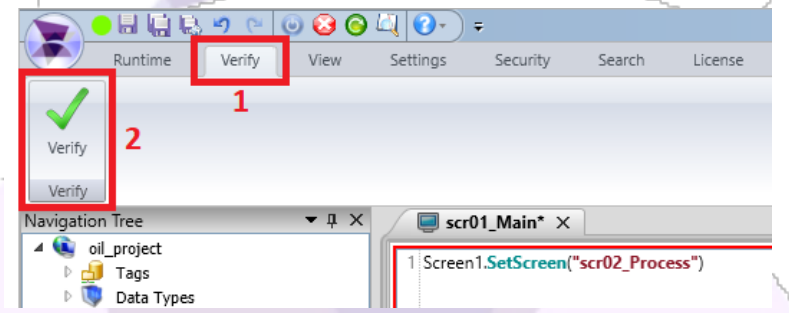


Heads up: *Don't forget to put the character ";" at the end of the script line.*

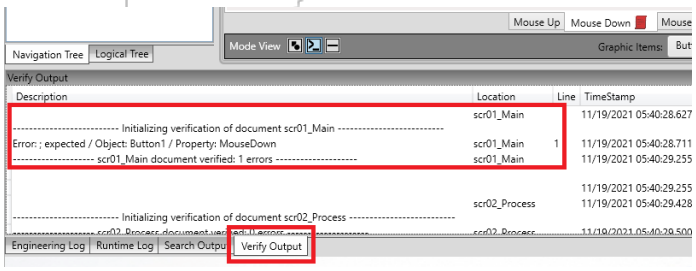
For the sake of knowledge, if the ";" is not added at the end of the script line, the event script will be in error, making it impossible to execute the script when the button is clicked. Below is an example if the character ";" is not added. Note that the script development area is highlighted in red and the "Mouse Down" tab has the script symbol in red, because the script is in error. To check if the script is really in error, as in the image below, navigate between the button events, for example, click on the "Mouse Up" event and then click on the "Mouse Down" event. If the "Mouse Down" tab and the script development area are highlighted in red, the script is in error.



The verification method presented above only verifies the “Mouse Down” event script of the “Process” button. However, ADISRA SmartView has a feature called “Verify”, which checks the entire application for errors. To use this feature, click on the “Verify” menu and then on the “Verify” option, as shown in the image below.

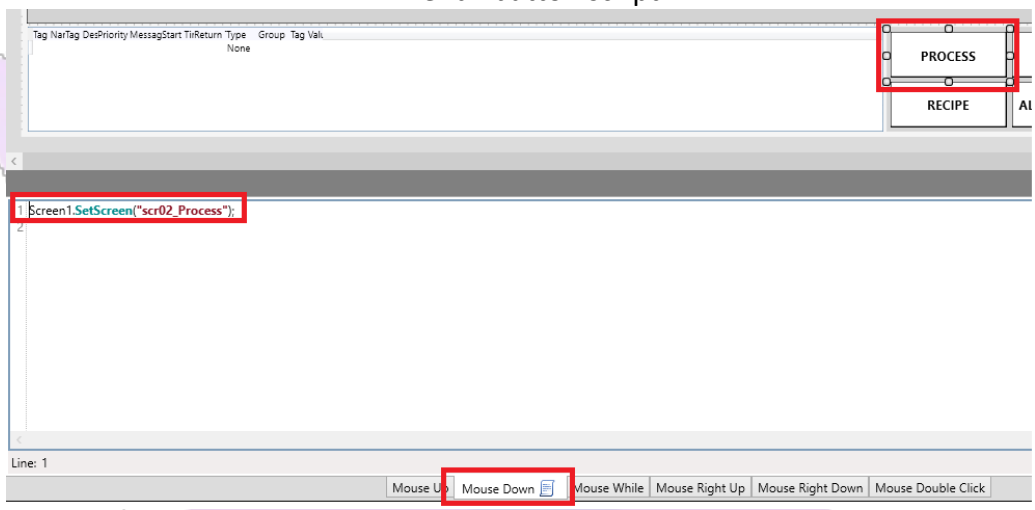


After clicking on “Verify”, a log will be generated and displayed on the “Verify Output” tab, containing a summary of all application verification. Note that the image below shows exactly one error in the “Mouse Down” event of the “Button1” object located on the “scr01_Main” screen. Go back to the script development area and add the “;” character again and check if the script is still in error.

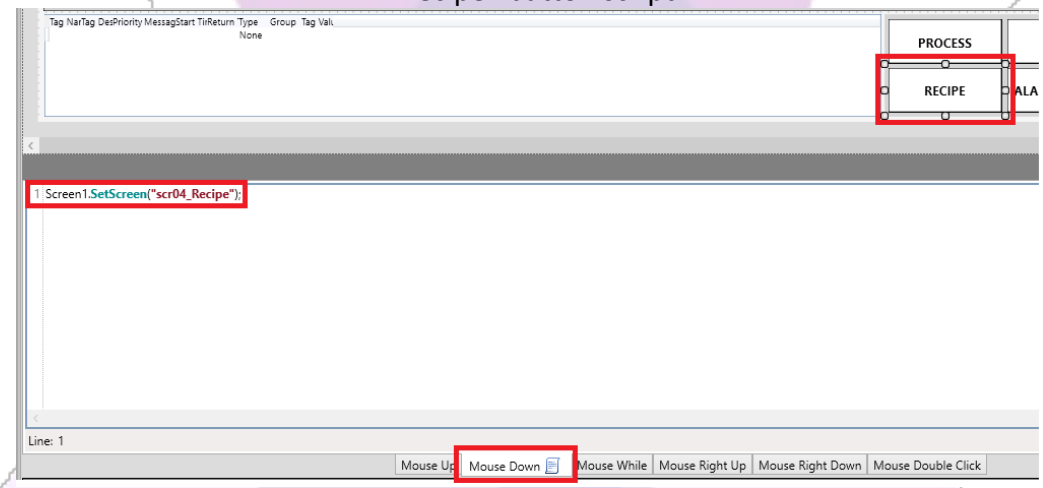


7- Now, let's add the same script to the other buttons, only changing the screen call according to the button. The image below shows the expected result.

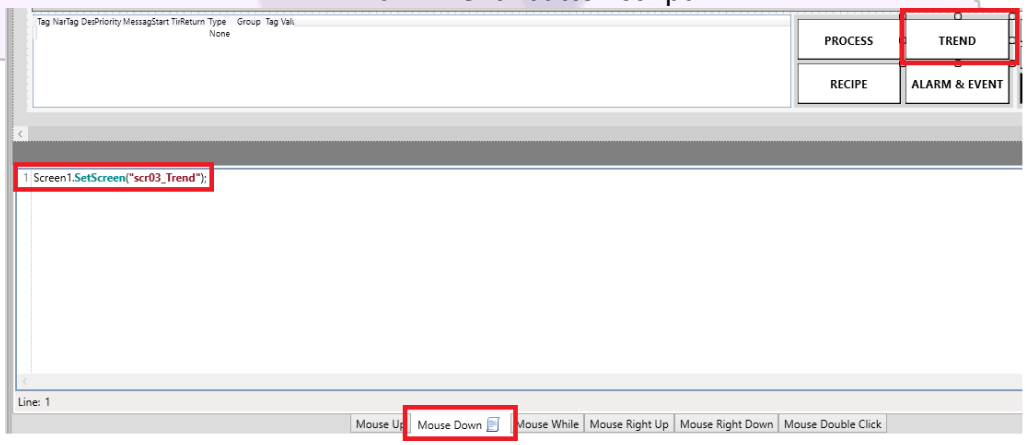
"Trend" button script



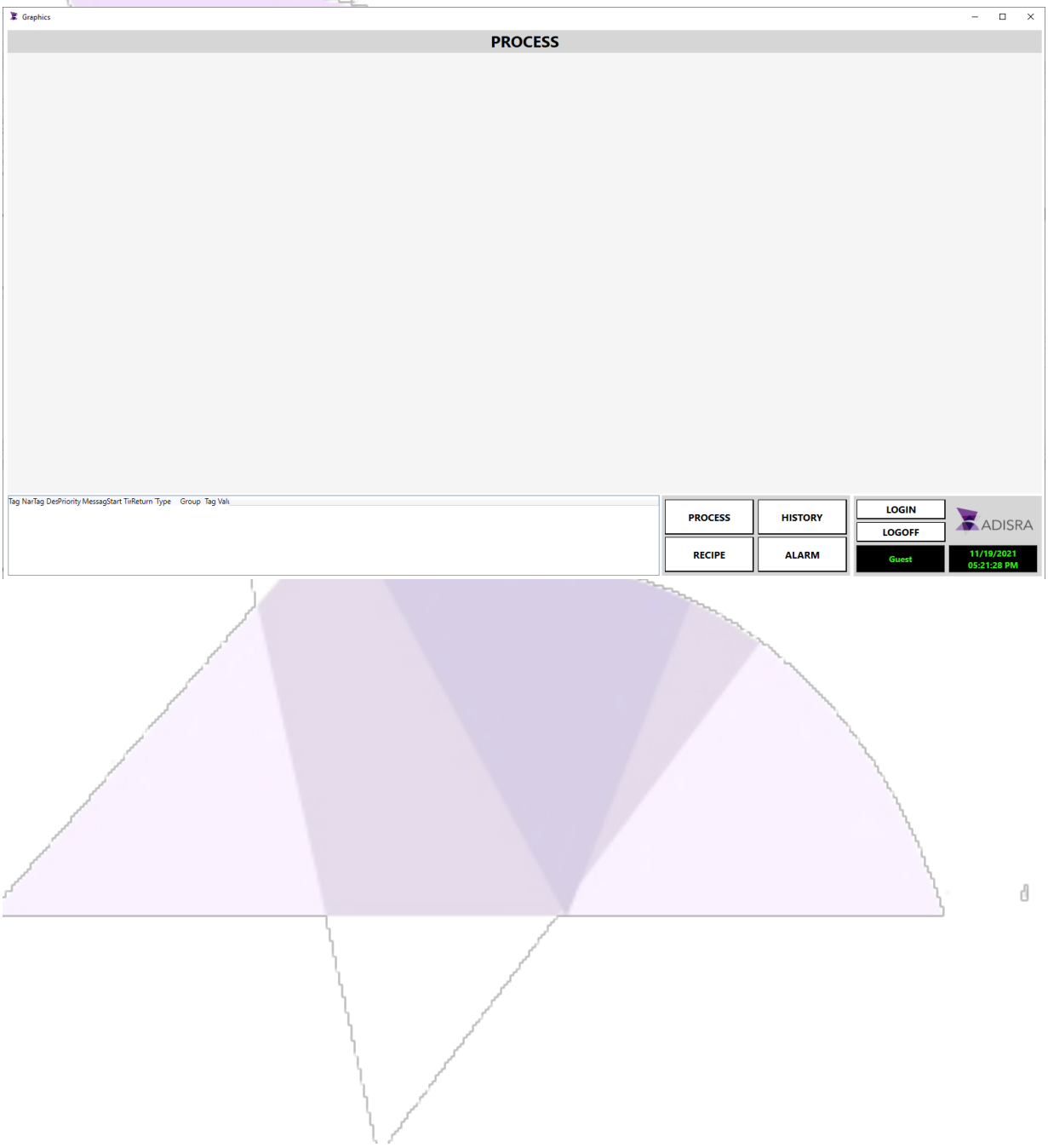
"Recipe" button script



"AlarmEvent" button script



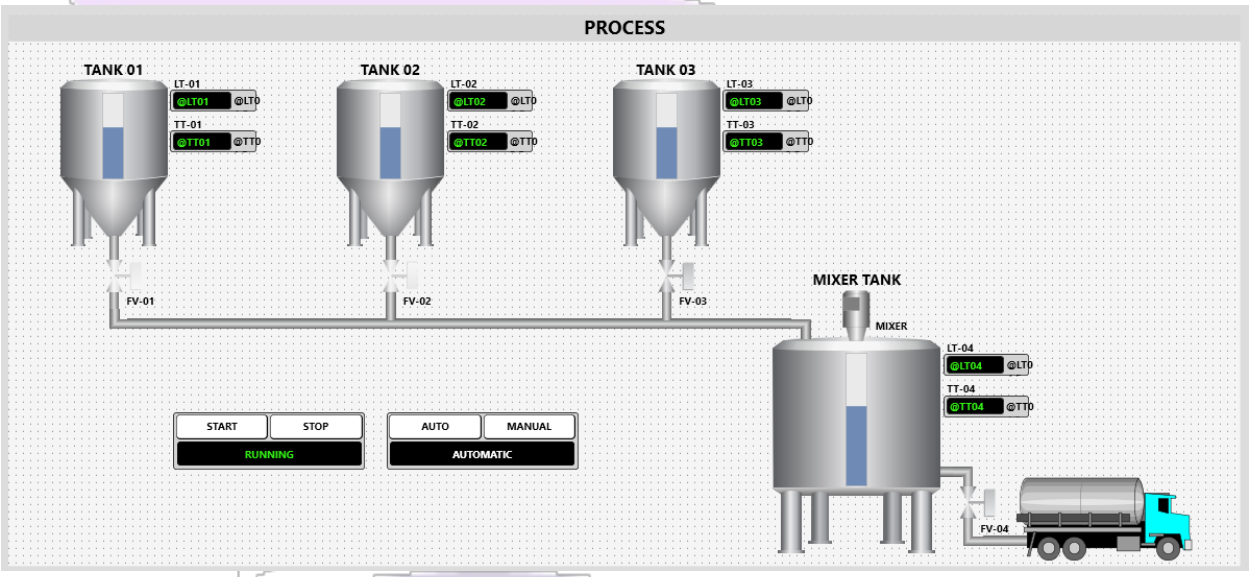
After adding and verifying that there are no errors in the scripts, let's test it! Save the application and then start RunTime. When the RunTime starts, the screen “scr01_Main” will be opened in the viewer, and the Screen object will show the screen “scr02_Process” in which the initial screen was previously configured. Then test the navigation of the buttons. Below is the expected result.



15. Developing the Graphic Process

Now, let's develop the Graphic “Process”. This Graphic will consist of tanks, level and temperature displays, pipes, valves, engine, etc.

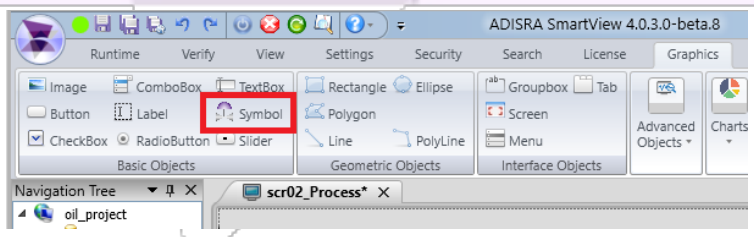
Below is how the Graphic Process should look.



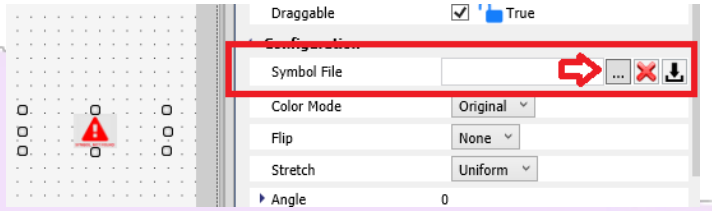
15.1. Inserting the Symbols Object

To add images of tanks, valves and other equipment in the industrial environment, we will use a resource called Symbol. This resource has a library with several images that can be used in application screens.

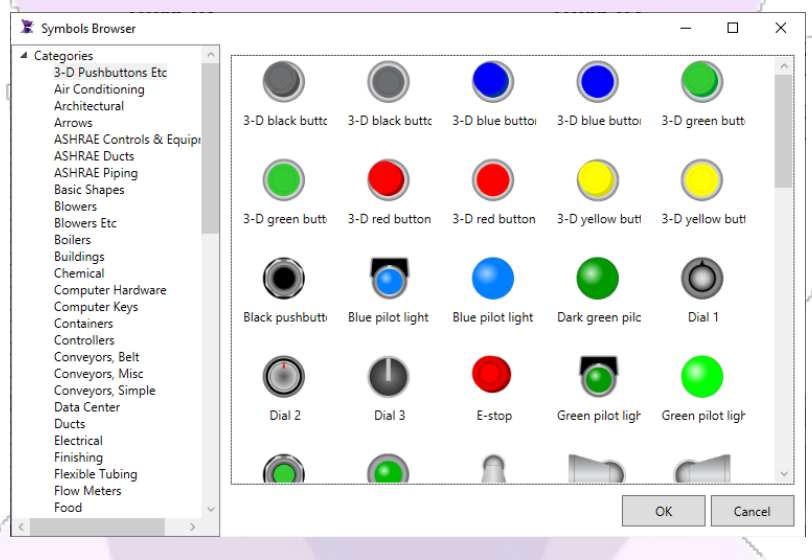
1. To add a Symbol type object, open the Graphic “scr02_Process”, then click on the “Graphics” menu and then on the “Symbol” option. Then click on the screen to add it.



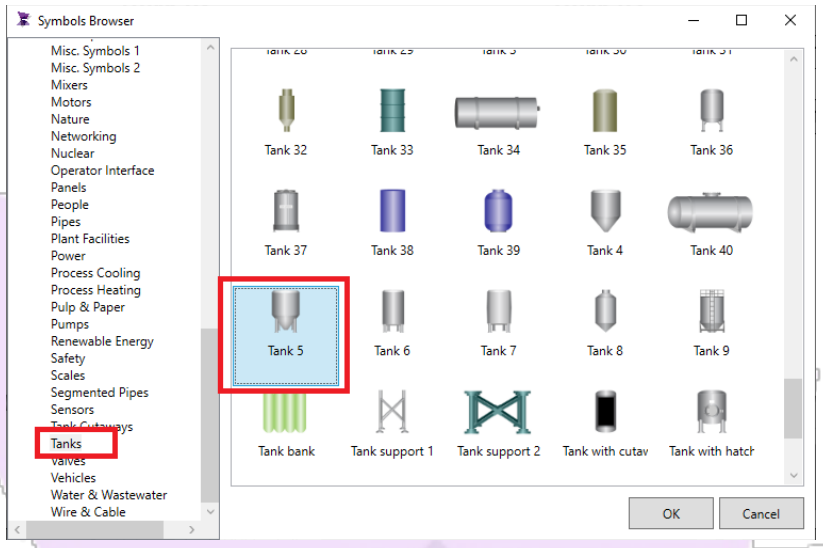
2. With the Symbol object selected, click on the “Symbol File” property and a window will open.



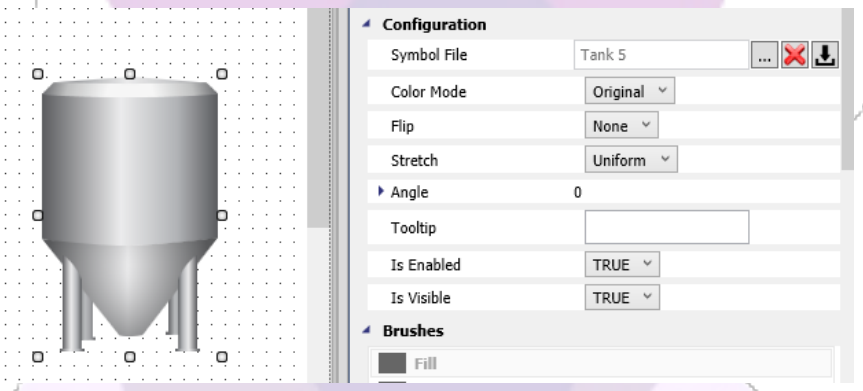
3. Symbols are organized by categories. The left of the window has a list of categories and the content of the selected category on the right.



4. Let's add our first object of type “Symbol”, which will be a tank. In the category list, look for the “Tanks” category and then click on the “Tank 5” symbol. Then click OK.



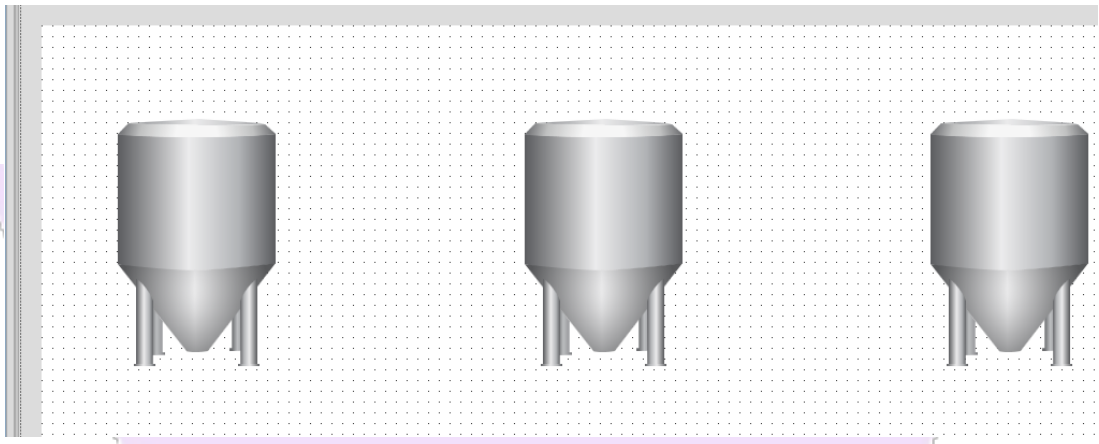
5. Below is the expected result.



6. Select the tank and change the following properties:
   Location (88, 71), Size (147, 230) and Strech (Uniform). This tank will represent the "TANK 01".

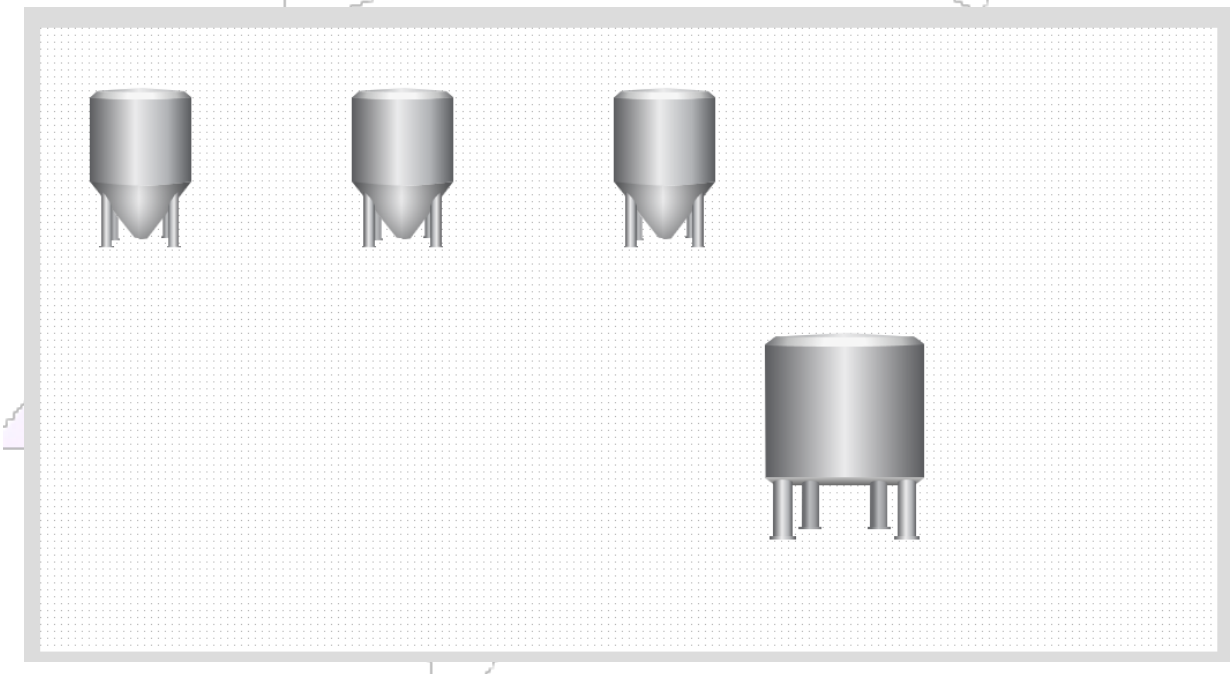
7. Now, add 02 more tanks and change the following properties:
   Tank 02: Location (88, 450), Size (147, 230) and Strech (Uniform).
   Tank 03: Location (88, 828), Size (147, 230) and Strech (Uniform).

8. Below is the expected result.

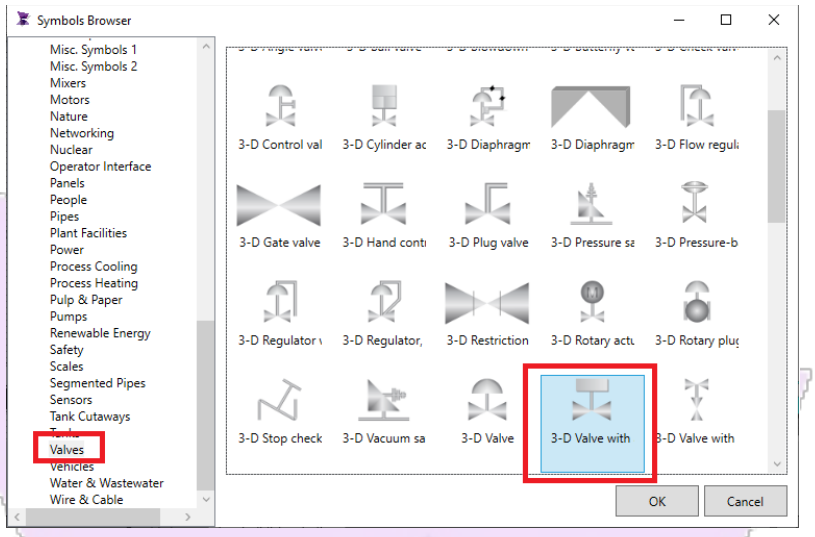


note: *It is possible to add a new Symbol by selecting a Symbol object and then using the keyboard command CTRL + C and then CTRL + V.*

9. Now, let's add the Symbol to the “Mixer Tank”. The Symbol will be different. Then, add a new Symbol on the “scr02_Process” screen. In the Symbols window, click on the “Tank” category, then on the “Tank 3” Symbol. Select the new object and then change the following properties: Location(442, 1047), Size(230, 297) and Strech(Uniform). Below is the expected result.



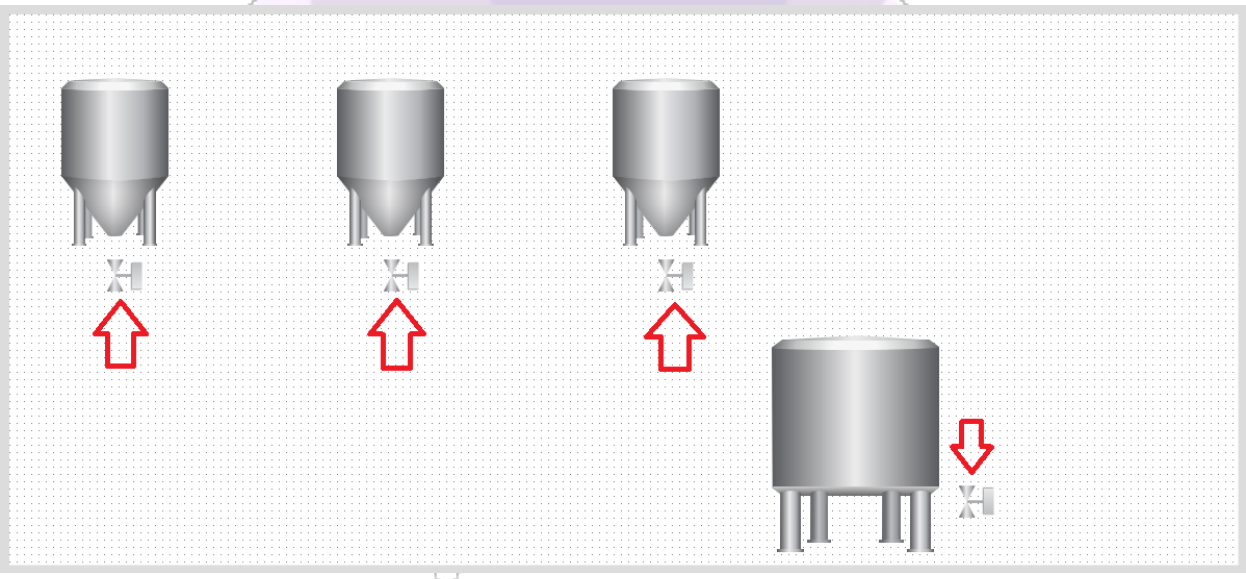
10. After adding the tanks, let's add the valves. Add one more Symbol object on the screen and then, in the Symbols window, click on the “Valve” category and select the “3-D Valve with” Symbol.



11. Select the valve and change the following properties:
    Location (334, 136) and Size (44, 48). This valve will represent the FV-01. Add 03 more valves and change the following properties:

    FV02:Location (334, 515), Size (44,48) and Angle/Fixed Angle (90)
    FV03:Location (334, 893), Size (44,48) and Angle/Fixed Angle (90)
    FV04:Location (644, 1306), Size (44.48) and Angle/Fixed Angle (90)

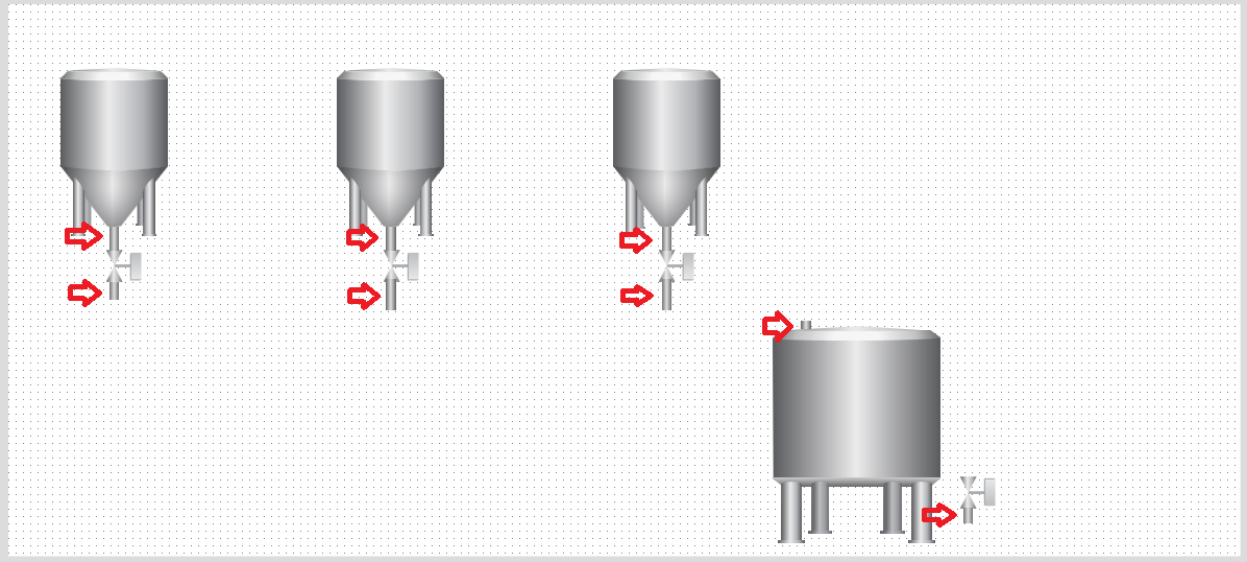
12. Below is the expected result of the valve settings.



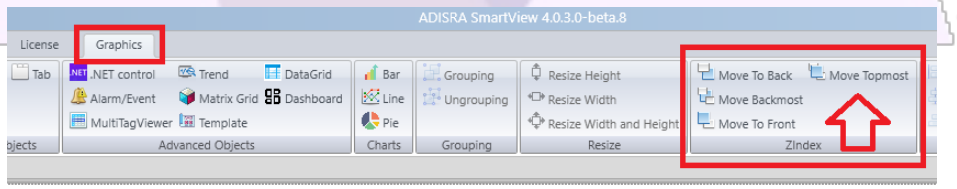
13. After adding the valves, let's add the pipes in a vertical position. Add one more Symbol and in the Symbols window, click on the “Pipe” category and select the “Short Vertical Pipe” symbol. Add 7 more pipes vertically. Change the following properties:

Vertical Piping 01: Location (304, 138), Size (14.34), Stretch (Fill)
Vertical Piping 02: Location (304, 517), Size (14.34), Strech (Fill)
Vertical Piping 03: Location (304, 895), Size (14.34), Stretch (Fill)
Vertical Piping 04: Location (380, 138), Size (14.24), Stretch (Fill)
Vertical Piping 05: Location (376, 517), Size (14.43), Strech (Fill)
Vertical Piping 06: Location (376, 895), Size (14.43), Stretch (Fill)
Vertical Piping 07: Location (433, 1086), Size (14.11), Stretch (Fill)
Vertical Piping 08: Location (689, 1308), Size (14.21), Strech (Fill)

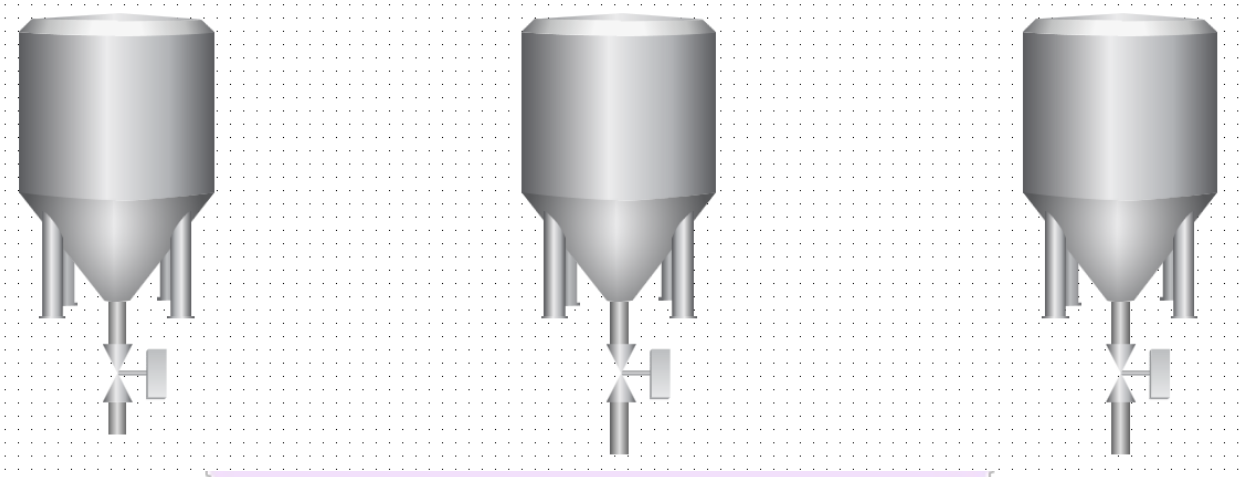
14. Below is the expected result of the vertical piping configurations.



15. The valves must be overlapping the pipes. If not, select all valves and use the “Zindex” feature, located in the “Graphic” menu. This feature allows you to move objects on the “Z” axis, that is, the depth of the objects. Click on the “Move Topmost” option. Below is the expected result.



16. Below is the expected result of the vertical piping configurations.

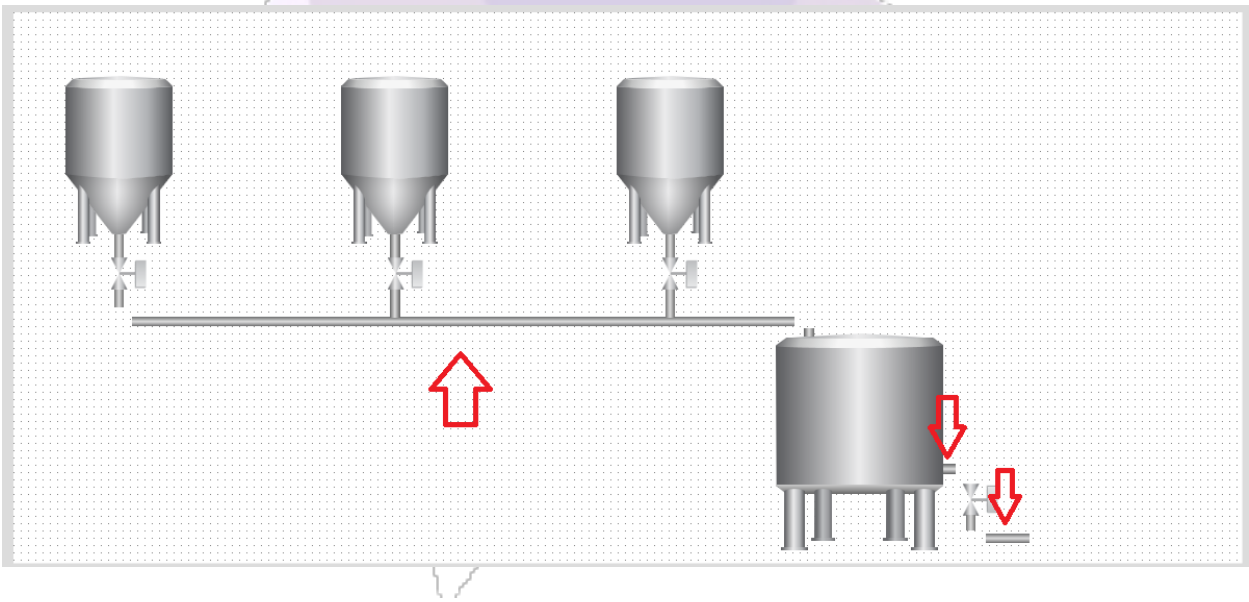


17. After adding the pipes in the vertical position, let's add the pipes in the horizontal position. Add one more Symbol and in the Symbols window, click on the “Pipe” category and select the “Short Horizontal Pipe” symbol. Add 2 more horizontal pipes. Change the following properties:

    Horizontal Piping 01: Location (417, 163), Size (910.14), Stretch (Fill)
    Horizontal Piping 02: Location (619, 1277), Size (17.14), Stretch (Fill)
    Horizontal Piping 03: Location (714, 1335), Size (6114.14), Strech (Fill)

    Select all horizontal pipes and in the “Zindex” feature, click on “Move Backmost”.

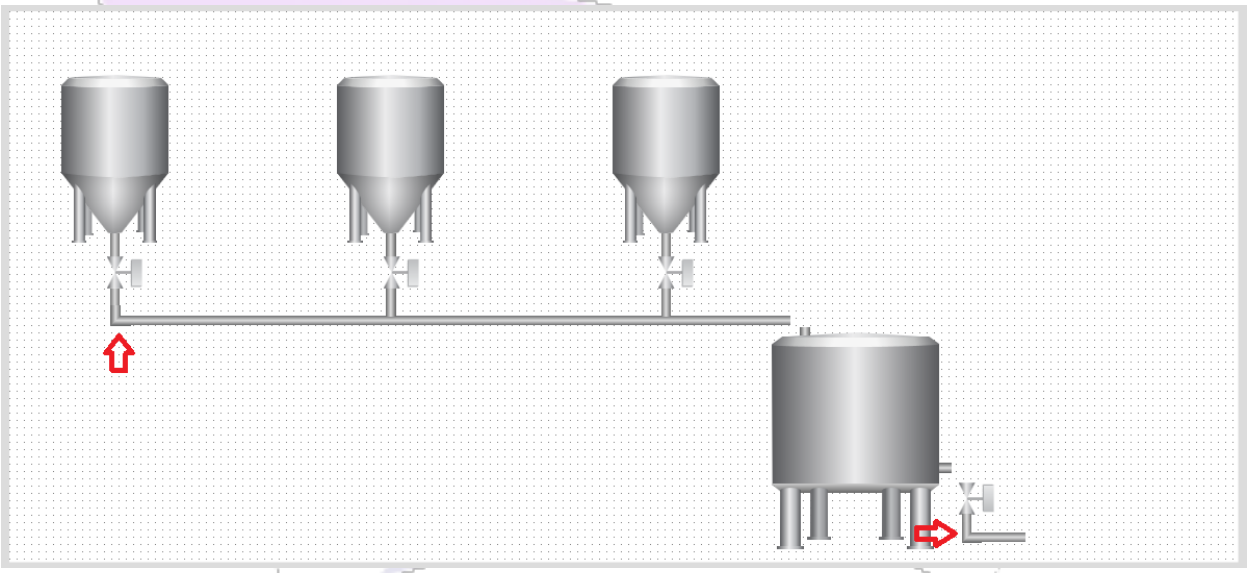
18. Below is the expected result of the horizontal piping configurations.



19. Now let's add the pipes on the right curve. Add a Symbol and in the Symbols window, click on the “Pipe” category and select the “Right Angle 3” symbol. Add one more curve. Change the following properties:

Right turn 01: Location (403, 138), Size (28,28), Strech (Uniform)
Right turn 02: Location (700, 1308), Size (28.28), Strech (Uniform)

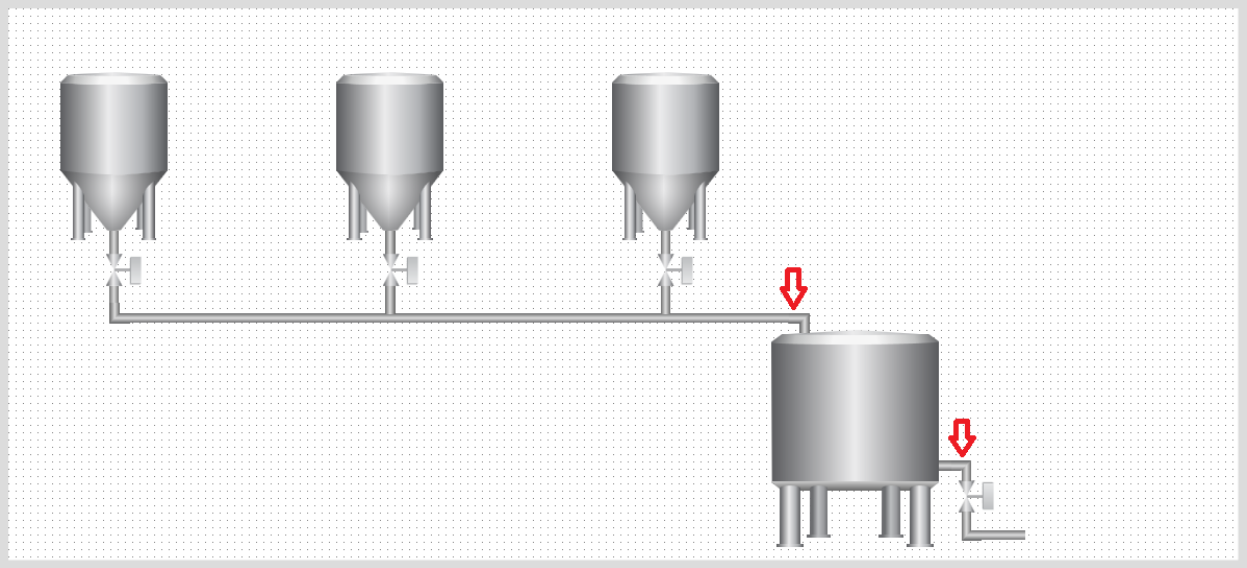
20. Below is the expected result of the configurations of the curves on the right.



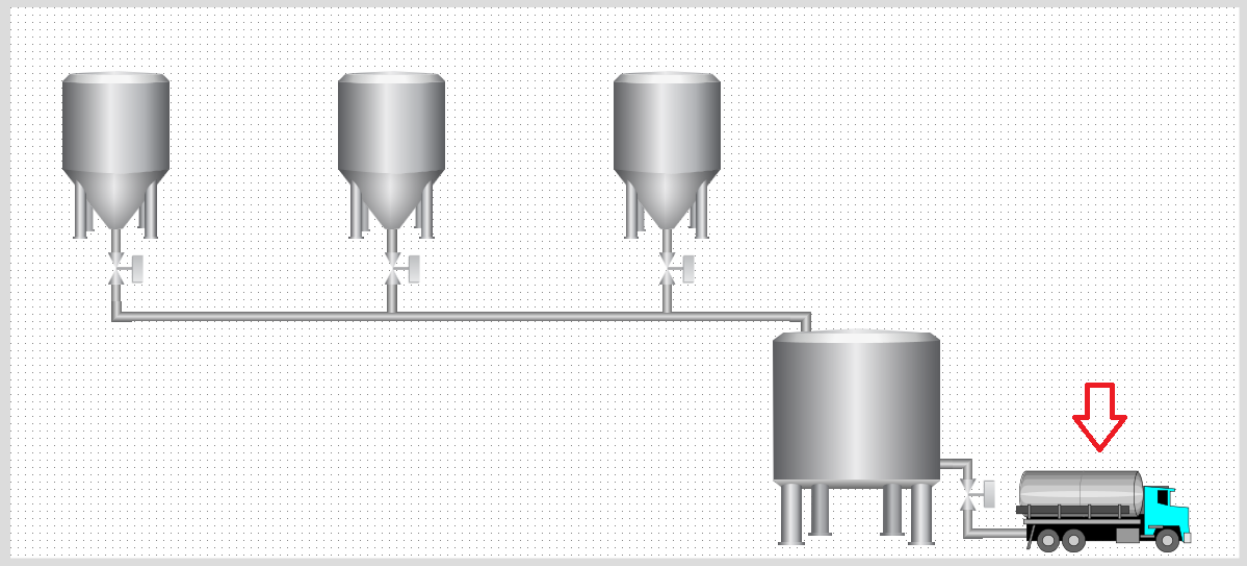
21. Now let's add the left bend pipes. Add a Symbol and in the Symbols window, click on the "Pipe" category and select the "Right Angle 2" symbol. Add one more curve. Change the following properties:

    Left turn 01: Location (417, 1072), Size (28.28), Strech (Uniform)
    Left turn 02: Location (619, 1294), Size (28.28), Strech (Uniform)

22. Below is the expected result of the configurations of the curves on the left.



23. After adding the curves, let's add a truck symbol. Add a Symbol and in the Symbols window, click on the “Vehicles” category and select the “Tanker Truck 2” symbol. Change the following properties:
Location (635, 1382), Size (239,112), Strech (Uniform), and Flip (Horizontal). Below is the expected result.

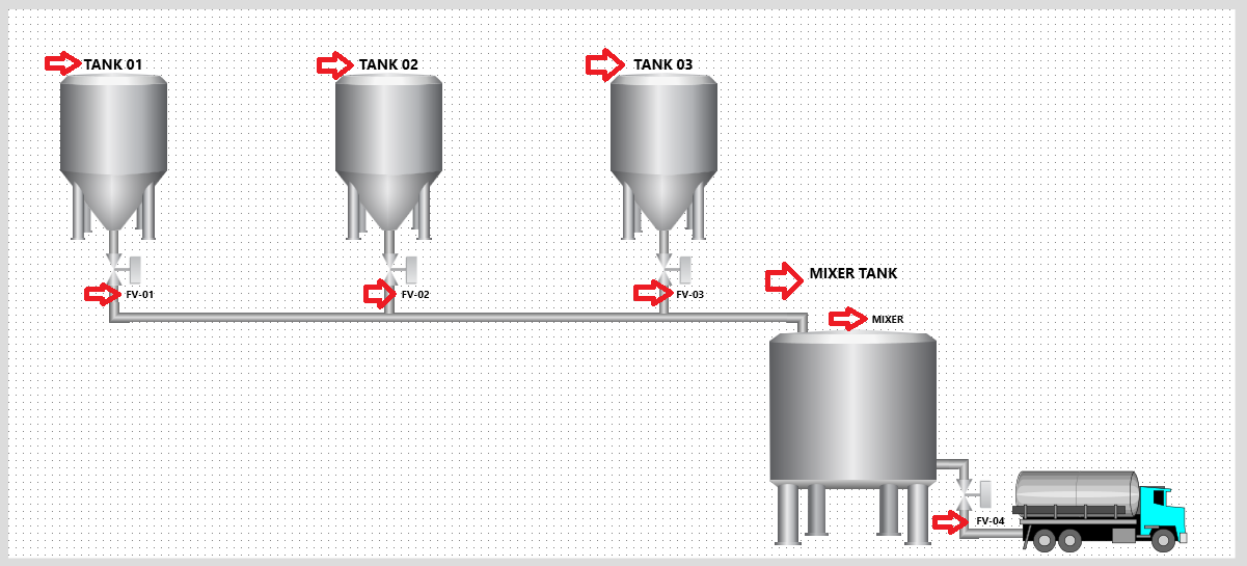


24. Now let's add the identification of each Tank and Valve. In the “Graphics” menu, add 08 “Label” objects and change the following properties:

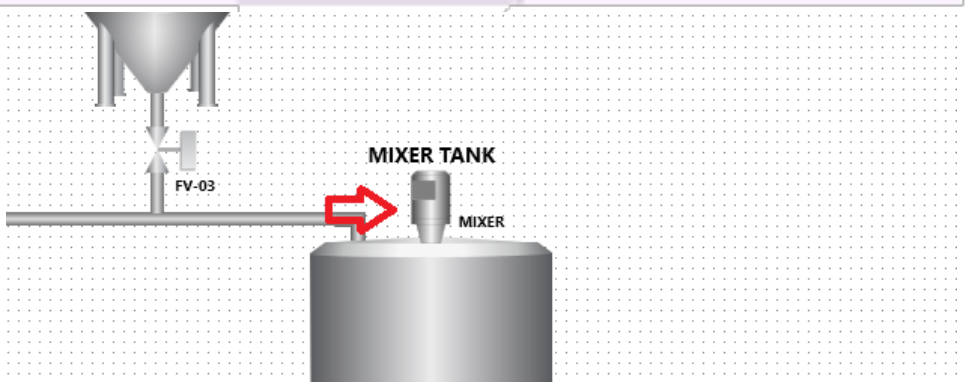
Label 01: Location (61, 96), Size (97, 24), Text (TANK 01, Segoe UI, Bold, 20, Center)
Label 02: Location (61, 475), Size (97, 24), Text (TANK 02, Segoe UI, Bold, 20, Center)

Label 03: Location (61, 853), Size (97, 24), Text (TANK 03, Segoe UI, Bold, 20, Center)
Label 04: Location (349, 1092), Size (141, 24), Text (MIXER TANK, Segoe UI, Bold, 20, Center)
Label 05: Location (382, 161), Size (47, 16), Text (FV-01, Segoe UI, Bold, 14, Center)
Label 06: Location (382, 540), Size (47, 16), Text (FV-02, Segoe UI, Bold, 14, Center)
Label 07: Location (382, 918), Size (47, 16), Text (FV-03, Segoe UI, Bold, 14, Center)
Label 08: Location (694, 1331), Size (47, 16), Text (FV-04, Segoe UI, Bold, 14, Center)
Label 09: Location (417, 1187), Size (47, 16), Text (MIXER, Segoe UI, Bold, 14, Center)

25. Below is the expected result of the label settings.



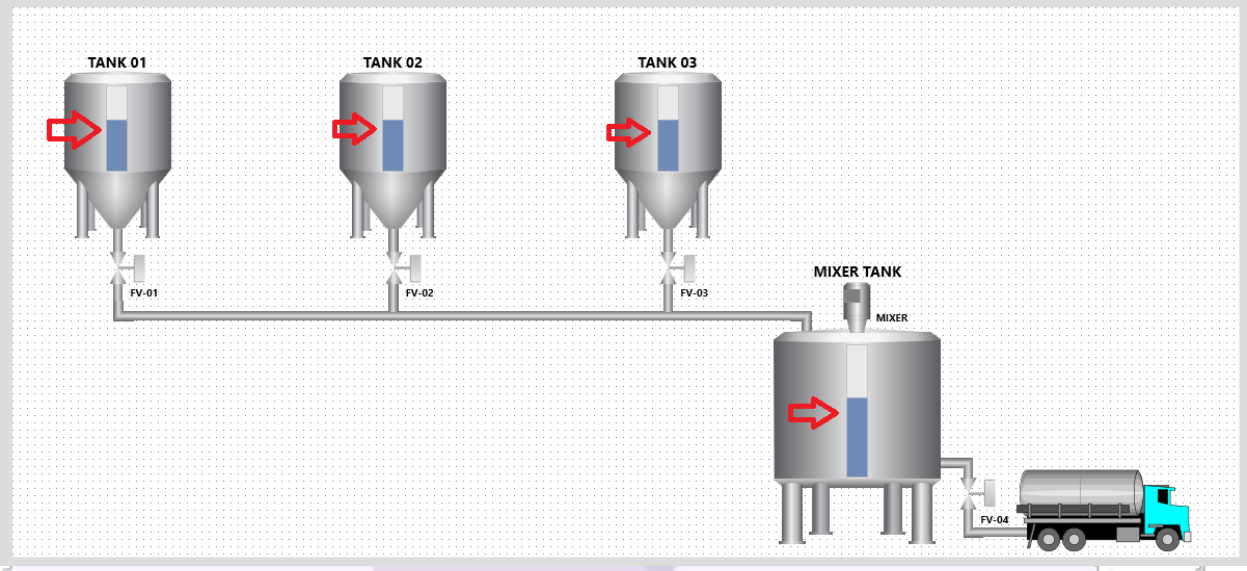
26. Add one more symbol and in the symbol window, click on the “Motors” category and select the “Simple Motor 1” symbol. Change the following properties: Location (393, 1126), Size (71,39) and Angle/Fixed Angle (90).
Below is the expected result.



27. After adding the mixer, let's create a level gauge for each tank. For this, add 04 rectangles and change the following properties:

    Dial 01: Location (107, 129), Size (30, 119), Brushes/Fill/Dyn Fill/Tag(@LT01), Orientation(Vertical), BackGround (234, 234, 234), Fill (110, 138, 181), Brushes/Border/Solid (183, 183, 183)
    Dial 02: Location (107, 508), Size (30, 119), Brushes/Fill/Dyn Fill/Tag(@LT02), Orientation(Vertical), BackGround (234, 234, 234), Fill (110, 138, 181), Brushes/Border/Solid (183, 183, 183)
    Dial 03: Location (107, 887), Size (30, 119), Brushes/Fill/Dyn Fill/Tag(@LT03), Orientation(Vertical), BackGround (234, 234, 234), Fill (110, 138, 181), Brushes/Border/Solid (183, 183, 183)
    Dial 04: Location (463, 1147), Size (30, 183), Brushes/Fill/Dyn Fill/Tag(@LT04), Orientation(Vertical), BackGround (234, 234, 234), Fill (110, 138, 181), Brushes/Border/Solid (183, 183, 183)

28. Below is the expected result.

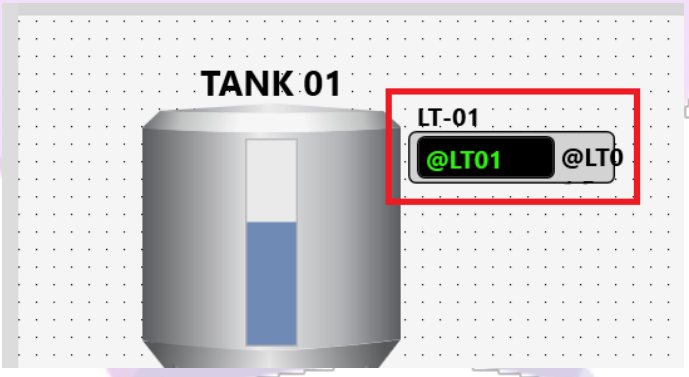


note: *Note that we associate the "LT" tags of each tank to animate the "Foreground" property of the respective displays. Therefore, the filling of the blue color in the rectangle will be proportional to the value of the "LT" tags.*

29. Now let's add the Level and Temperature analog displays for each Tank. First we will make the level gauge for Tank 01, that is, the level transmitter LT-01. Then add 01 rectangle, 02 labels and 01 textbox. Change the following properties:
    Rectangle 01: Location (103, 222), Size (118, 30), Corner Radius (4), Brushes/Fill/Solid (211,211,211)

Label 01: Location (85, 226), Size (80, 16), Text (LT-01, Segoe UI, Bold, 14, Alignment Left)
Label 02: Location (108, 308), Size (45, 25), Text (@LT01.EngUnits, Segoe UI, Bold, 14, Alignment Left)
TextBox 01: Location (106, 227), Size (79, 24), Text (@LT01, Segoe UI, Bold, 14, Alignment Left), Brushes/Foreground/Solid (52, 237,21) and Brushes/Background/ Solid (0, 0, 0)

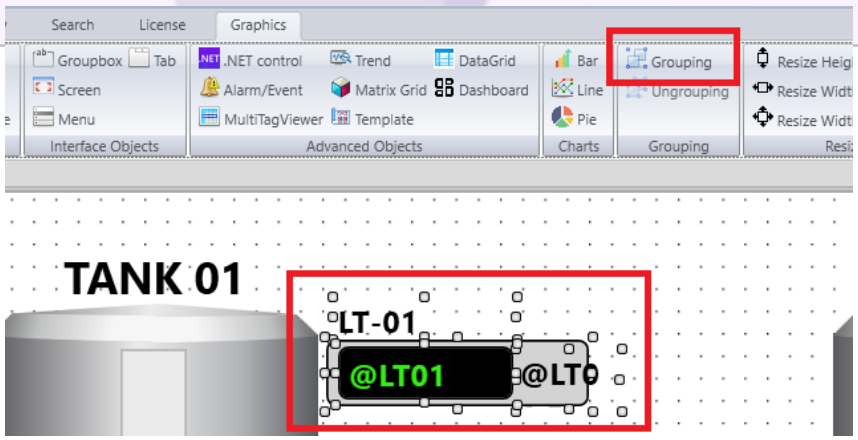
30. Below is the expected result.



note[1]: *The set of objects previously added is intended to show the current value of the tag @LT01, the name of the display, that is, "LT-01" and the engineering unit to which the tag @LT01.EngUnits was associated.*

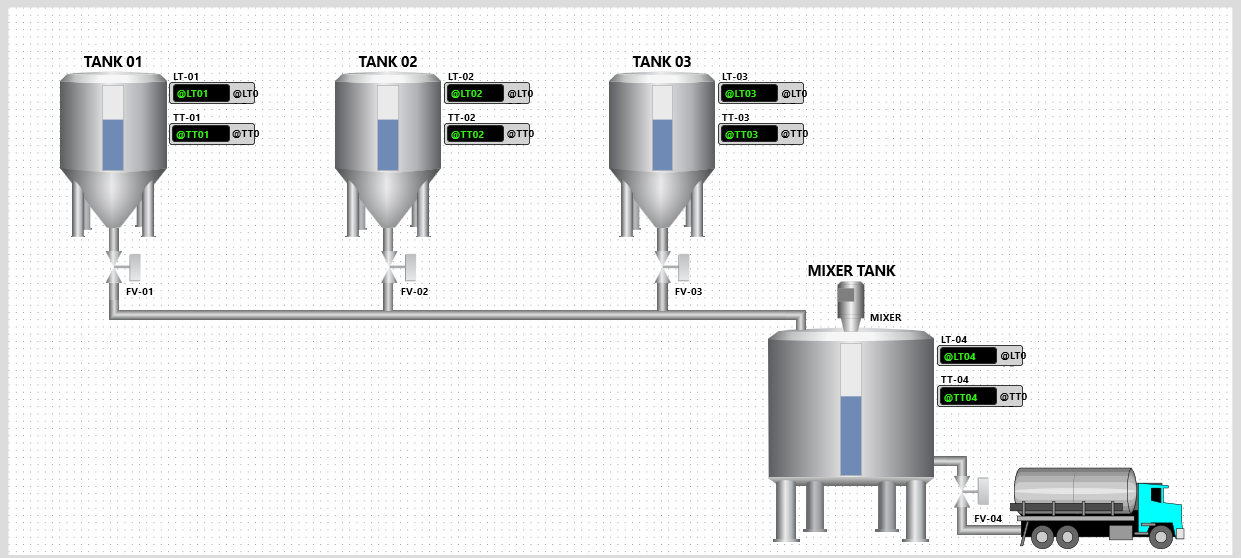
note[2]: *We will use this same pattern to create the analog display for the TT-01, which will be located below the FT-01, and then we will make the displays for each tank.*

31. To facilitate the development, select all the objects on the "FT-01" display and then, in the "Graphics" menu, click on the "Grouping" option. This feature groups all selected objects and makes them as a single object.



32. Select the new group of objects and then use the commands CTRL + C and CTRL + V to copy. Make 07 new copies of the group. Then change the following properties:
    Group 01: Location (141, 222) Undo the group and change all associations to "TT01".
    Group 02: Location (85, 601) Undo the group and change all associations to "LT02".
    Group 03: Location (141, 601) Undo the group and change all associations to "TT02". Group 02: Location (85, 979) Undo the group and change all associations to "LT03".
    Group 03: Location (141, 979) Undo the group and change all associations to "TT03".
    Group 02: Location (447, 1281) Undo the group and change all associations to "LT04".
    Group 03: Location (503, 1281) Undo the group and change all associations to "TT04".

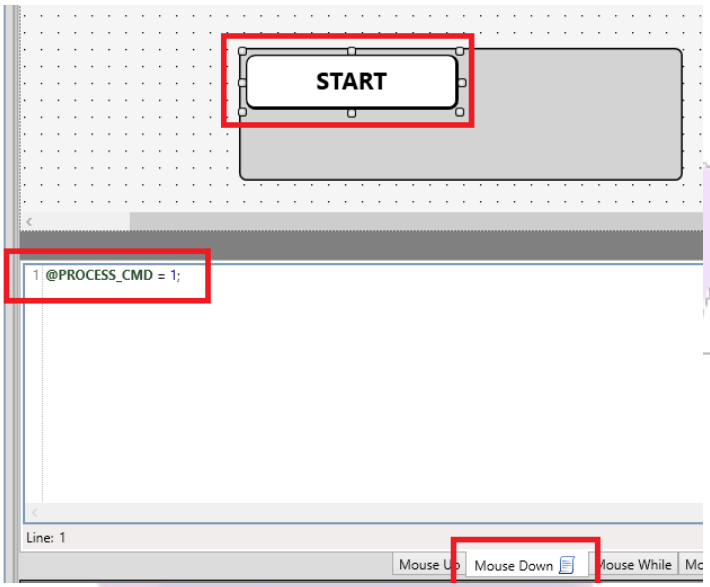
33. Below is the expected result of the analog display settings.



34. Now let's add the command and system status buttons (RUNNING/STOP). Add 1 rectangle, 2 buttons and 2 textboxes. For the rectangle object, change the following properties:
    Rectangle 01: Location (539, 227), Size (263, 79), Corner Radius (4), Brushes/Fill/Solid (211,211,211)

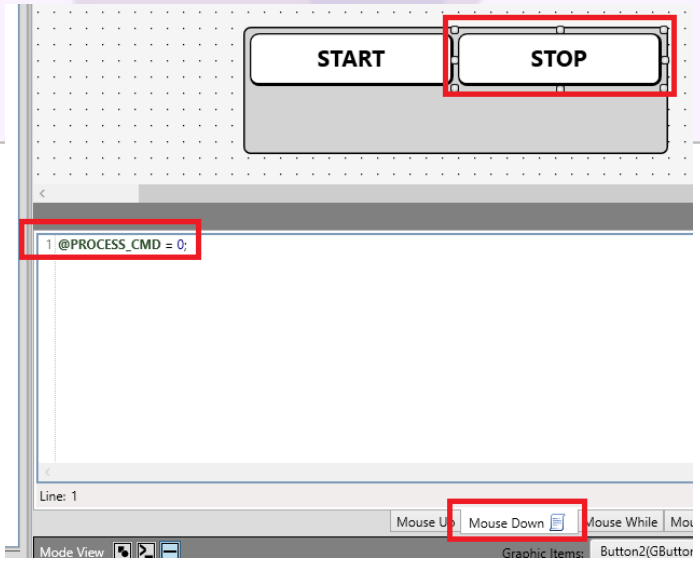
35. On the first button, change the following properties: Location (543, 231), Size (126, 33), Text (START, Segoe UI, Bold, 14, Alignment Center). As this button will be responsible for starting the system, let's add a script in the "Mouse Down" event. Add the following script: @PROCESS_CMD = 1;

When the user clicks on the “START” button, the value “1” will be assigned to the “@PROCESS_CMD” tag, which represents the “Turn on the System” according to the PLC tag list. Below is the expected result.

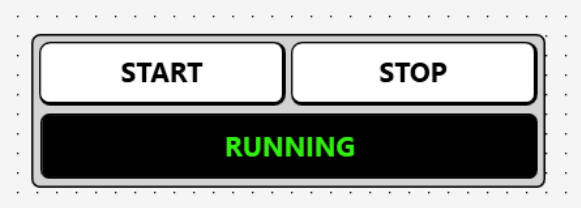


36. In the second button, change the following properties: Location (543, 360), Size (126, 33), Text (STOP, Segoe UI, Bold, 14, Alignment Center). As this button will be responsible for stopping the system, let's add a script in the “Mouse Down” event. Add the following script: @PROCESS_CMD = 0;

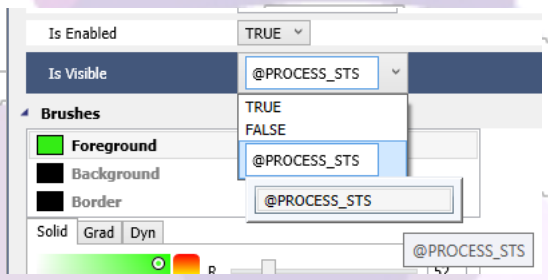
    When the user clicks on the “STOP” button, the value “0” will be assigned to the “@PROCESS_CMD” tag, which represents the “Shut down the System” according to the PLC tag list. Below is the expected result.



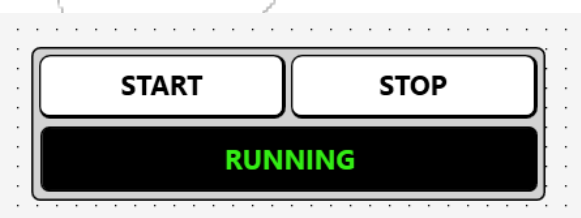
37. Now let's add the textbox, which will represent the Running and Stop status of the system. The first textbox will represent the “RUNNING” status. So change the following properties:
Location(580, 232), Size(253, 33), Text(RUNNING, Segoe UI, Bold, 14, Alignment Center), Brushes/Foreground/Solid(52,237,21), Brushes/Background /Solid(0, 0, 0) and Brushes/Border/Solid(0, 0, 0). Below is the expected result.



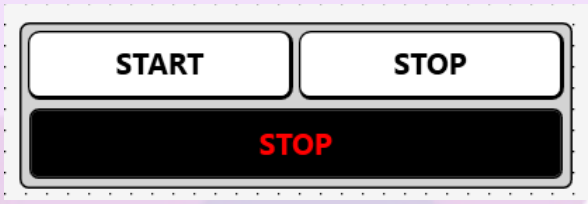
38. After the initial settings of the “RUNNING” textbox, let's configure the status animation. Select the “RUNNING” textbox and add the tag “@PROCESS_STS” in the “Is Visible” property. From now on, when the “@PROCESS_STS” tag is “TRUE”, the “RUNNING” textbox will be visible, showing that the system status is in “RUNNING”. Therefore, the visibility of this object will depend on the condition.



39. The second textbox will represent the “STOP” status. So, change the following properties:
Location (580, 232), Size (253, 33), Text (STOP, Segoe UI, Bold, 14, Alignment Center), Brushes/Foreground/Solid (255,0,0), Brushes /Background/Solid (0, 0, 0) and Brushes/Border/Solid (0, 0, 0). The Textbox “STOP” should be behind the textbox “RUNNING”. Below is the expected result.



note: *It will not be necessary to configure the animation of the "STOP" textbox, as it must be between the rectangle objects and the "RUNNING" textbox. When the "@PROCESS_STS" tag is "TRUE", the "RUNNING" textbox will be visible, hiding the "STOP" textbox. And when the tag "@PROCESS_STS" is "FALSE", the textbox "RUNNING" will be invisible. Soon, the textbox "STOP" will be visible, informing that the status of the system is "STOP".*



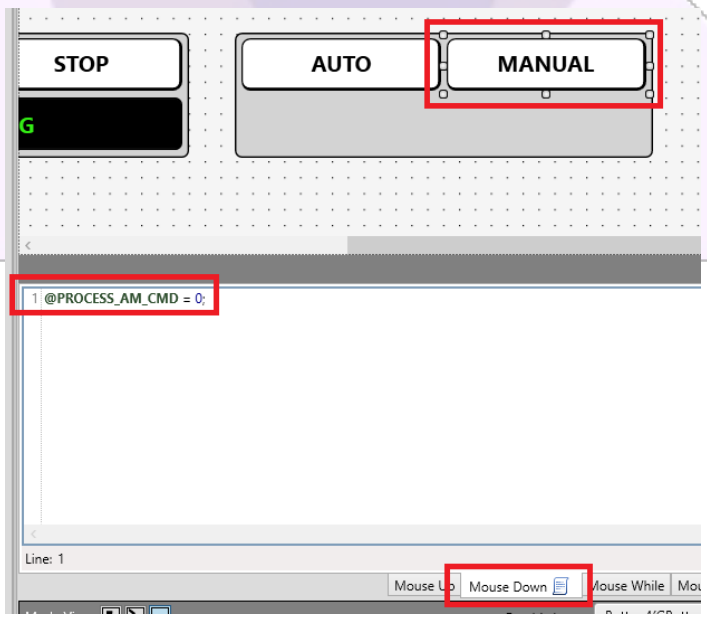
40. Now let's add the command and system status buttons (AUTOMATIC/MANUAL). Add 1 rectangle, 2 buttons and 2 textboxes. For the rectangle object, change the following properties:
Rectangle 01: Location (539, 519), Size (263, 79), Corner Radius (4), Brushes/Fill/Solid (211,211,211)

41. On the first button, change the following properties:
Location (543, 523), Size (126, 33), Text (AUTO, Segoe UI, Bold, 14, Alignment Center).
As this button will be responsible for putting the system in automatic mode, let's add a script in the "Mouse Down" event. Add the following script: @PROCESS_AM_CMD = 1;

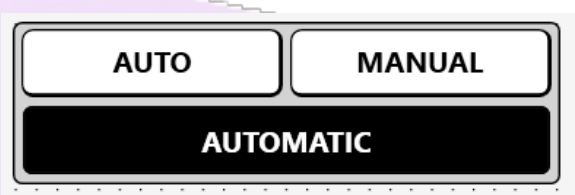
When the user clicks on the "AUTO" button, the value "1" will be assigned to the "@PROCESS_AM_CMD" tag, which represents the "Put the System in automatic mode" according to the PLC tag list. Below is the expected result.

42. In the second button, change the following properties: Location (543, 652), Size (126, 33), Text (MANUAL, Segoe UI, Bold, 14, Alignment Center). As this button will be responsible for putting the system in manual mode, let's add a script in the “Mouse Down” event. Add the following script: @PROCESS_AM_CMD = 0;

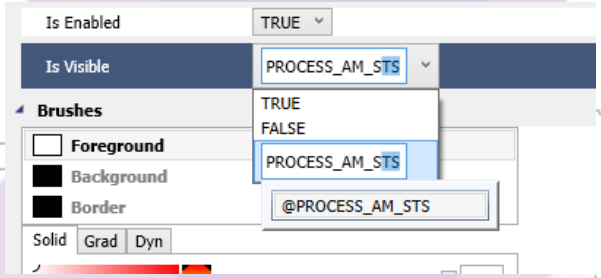
    When the user clicks on the “MANUAL” button, the value “0” will be assigned to the “@PROCESS_AM_CMD” tag, which represents the “Put the System in manual mode” according to the PLC tag list. Below is the expected result.



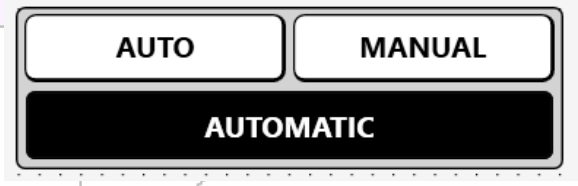
43. Now let's add the textboxes, which will represent the system's automatic and manual status. The first textbox will represent the “AUTOMATIC” status. So change the following properties: Location (580, 523), Size (253, 33), Text (AUTOMATIC, Segoe UI, Bold, 14, Alignment Center), Brushes/Foreground/Solid (255,255,255), Brushes/Background/Solid (0, 0, 0) and Brushes/Border/Solid (0, 0, 0). Below is the expected result.



44. After making the initial settings of the “AUTOMATIC” textbox, let's configure the status animation. Select the “AUTOMATIC” textbox and add the tag “@PROCESS_AM_STS” in the “Is Visible” property. From now on, when the “@PROCESS_AM_STS” tag is “TRUE”, the “AUTOMATIC” textbox will be visible, showing that the system status is “Automatic”. Therefore, the visibility of this object will depend on the condition.

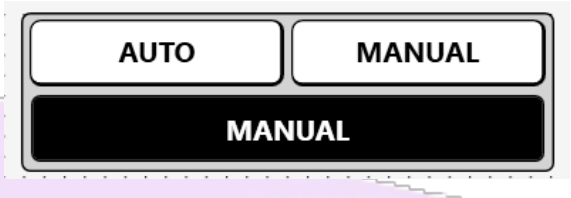


45. The second textbox will represent the “MANUAL” status. So, change the following properties: Location (580, 523), Size (253, 33), Text (MANUAL, Segoe UI, Bold, 14, Alignment Center), Brushes/Foreground/Solid (255,255,255), Brushes/Background/Solid (0, 0, 0) and Brushes/Border/Solid (0, 0, 0). The Textbox “MANUAL” should be behind the textbox “AUTOMATIC”. Below is the expected result.

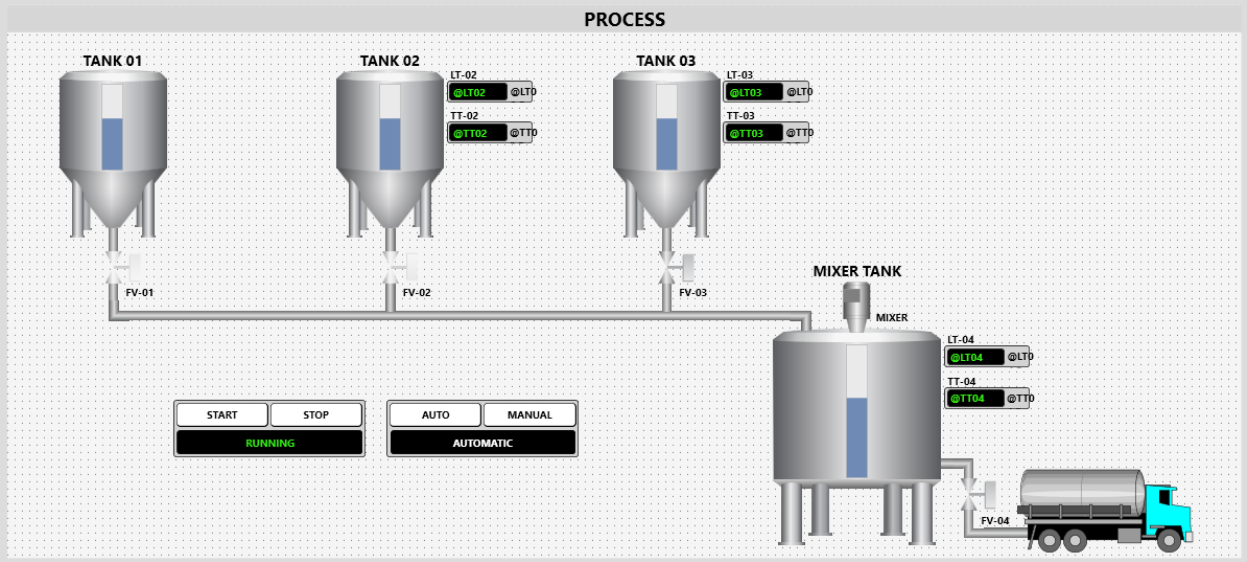


note: *It will not be necessary to configure the animation of the “MANUAL” textbox, as it must be between the rectangle objects and the “AUTOMATIC” textbox. When the tag “@PROCESS_AM_STS” is “TRUE”, the textbox “AUTOMATIC” will be visible, hiding the textbox “MANUAL”. And when the tag “@PROCESS_AM_STS” is “FALSE”, the textbox*

"AUTOMATIC" will be invisible. Soon, the textbox "MANUAL" will be visible, informing that the system mode is "MANUAL".



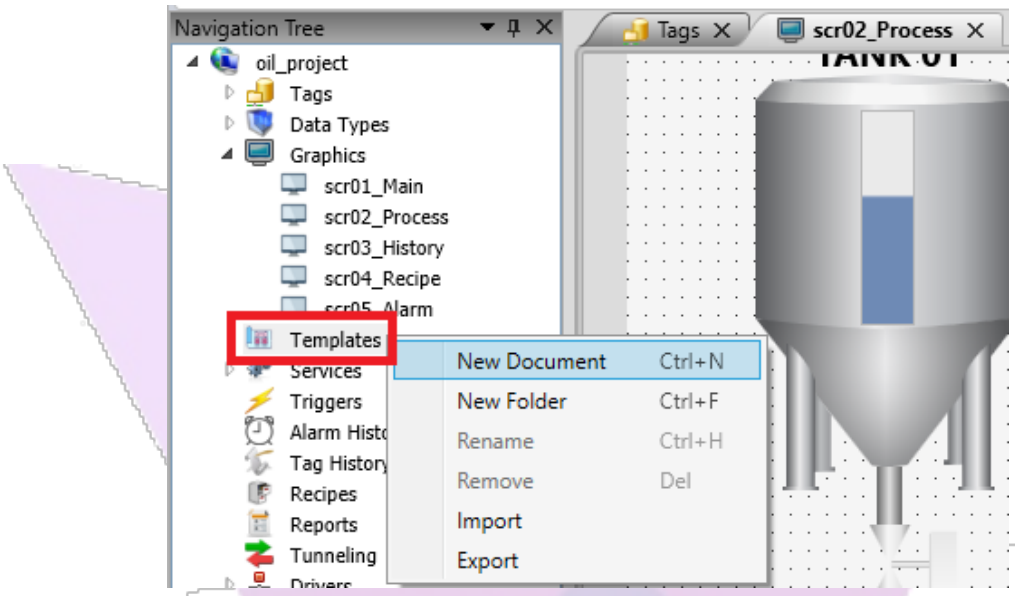
46. Below is the expected result of the "scr02_Process" screen.



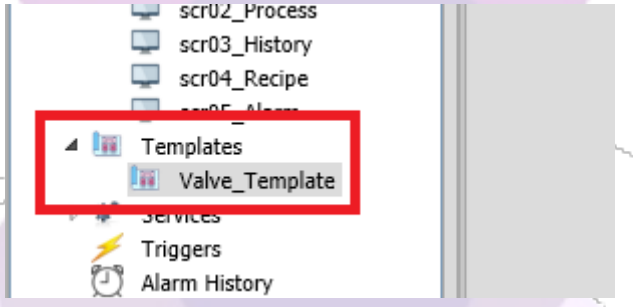
15.2. Inserting Templates

ADISRA SmartView has a document called Templates, which allows the user to create an object or a set of objects from an application in a library, in which they can be shared between projects without having to re-develop each object, thus facilitating the development of repetitive objects. Next, we will create a Template which will represent the valves.

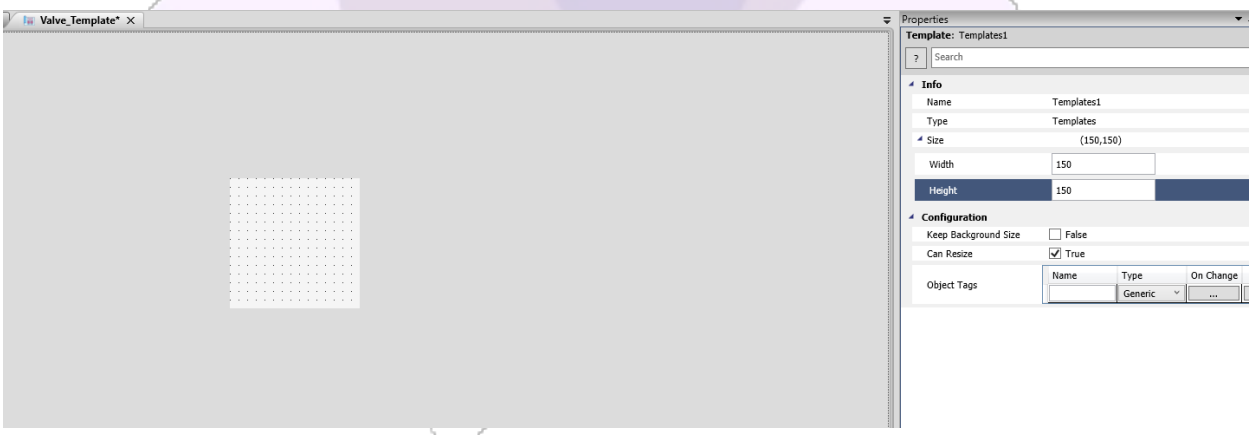
1. Right-click on "Templates" and then on the "New Document" option.



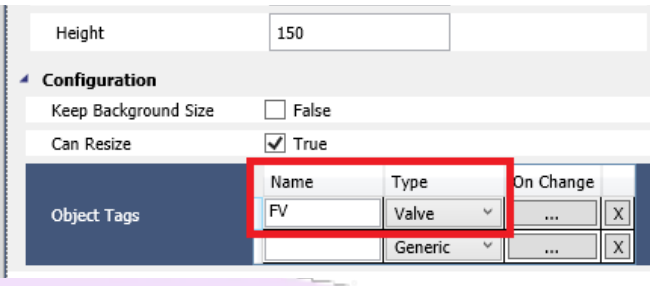
2. Save the new document as “Valve_Template”.



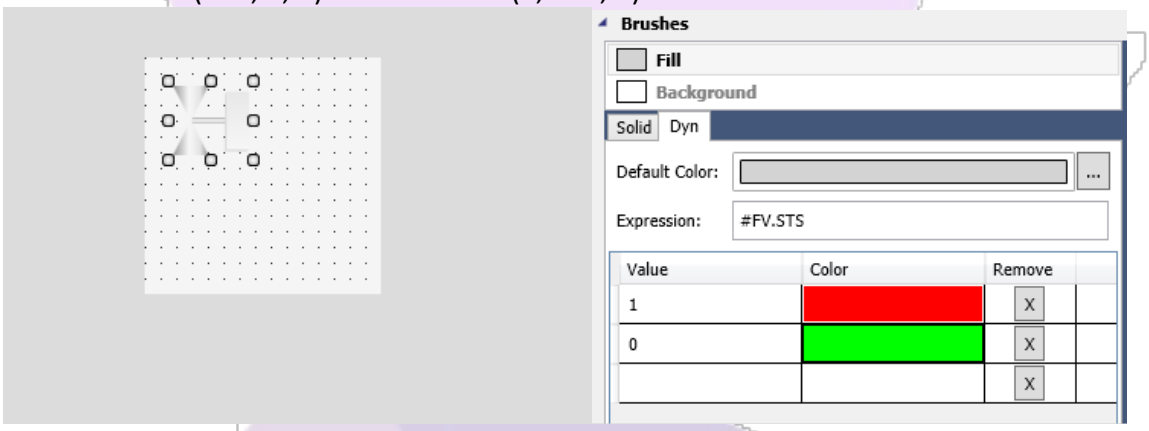
3. Click on the template canvas and change the following properties: Size (150, 150).



4. In the list of template properties, add an “Object Tags” called “FV” and datatype “Valve”. This datatype was previously created in the DataType chapter, in which it was created to facilitate the structuring of tags.

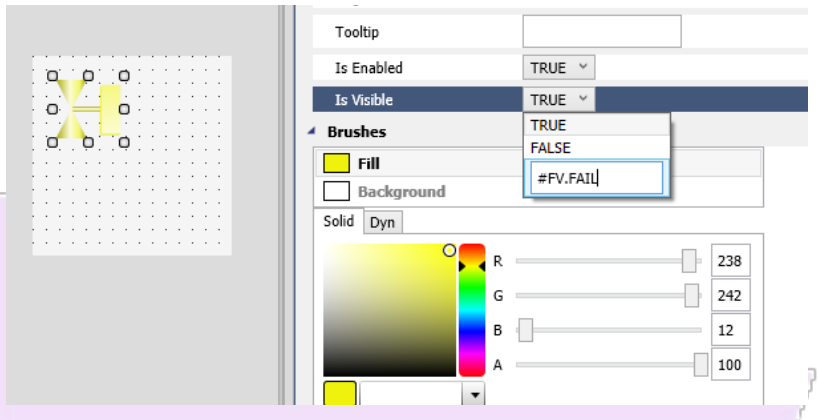


5. Now, let's add the Symbol of a valve which will represent the status of valve open and closed. In the Symbols window, click on the "Valves" category, then on the "3-D Valve with" Symbol. Change the following properties: Location (16, 20), Size (44, 48), Angle/Fixed Angle (90), Color Mode (Shade), Brushes/Fill/Dyn/Expression: (#FV.STS) - Value: True (255, 0, 0) - Value: False (0, 255, 0).

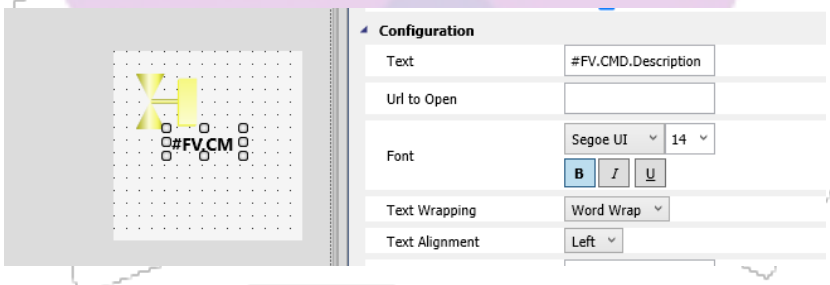


6. After adding the symbol which will represent the valve's open and closed status, let's add one more valve symbol to represent the valve's failure status. If the valve is faulty, this symbol will be visible above the previous symbol. Add one more "3-D Valve with" symbol.
   Change the following properties:

   Location (16, 20), Size (44, 48), Angle/Fixed Angle (90), Color Mode (Shade), Brushes/Fill (238, 242, 12) and Is Visible (#FV.FAIL). For this animation to work correctly, the yellow valve symbol must be on top of the previous symbol. If the valve is faulty, the yellow valve symbol will override the previous symbol. Below is the expected result.

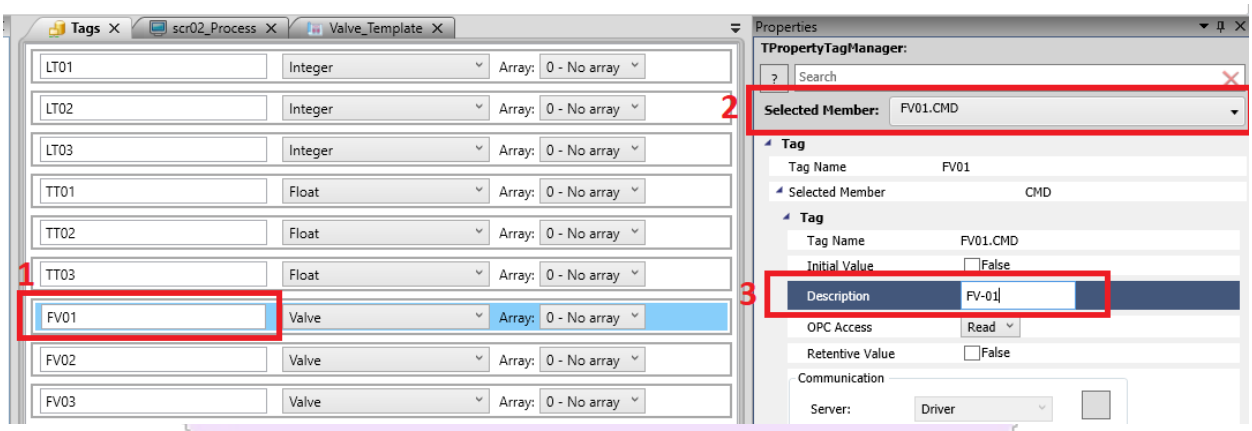


7. Now, let's add a label object which will show the name of the valve. Change the following properties: Location (64, 45), Size (55, 16) and Text (#FV.CMD.Description, Segoe UI, Bold, 14). Below is the expected result.

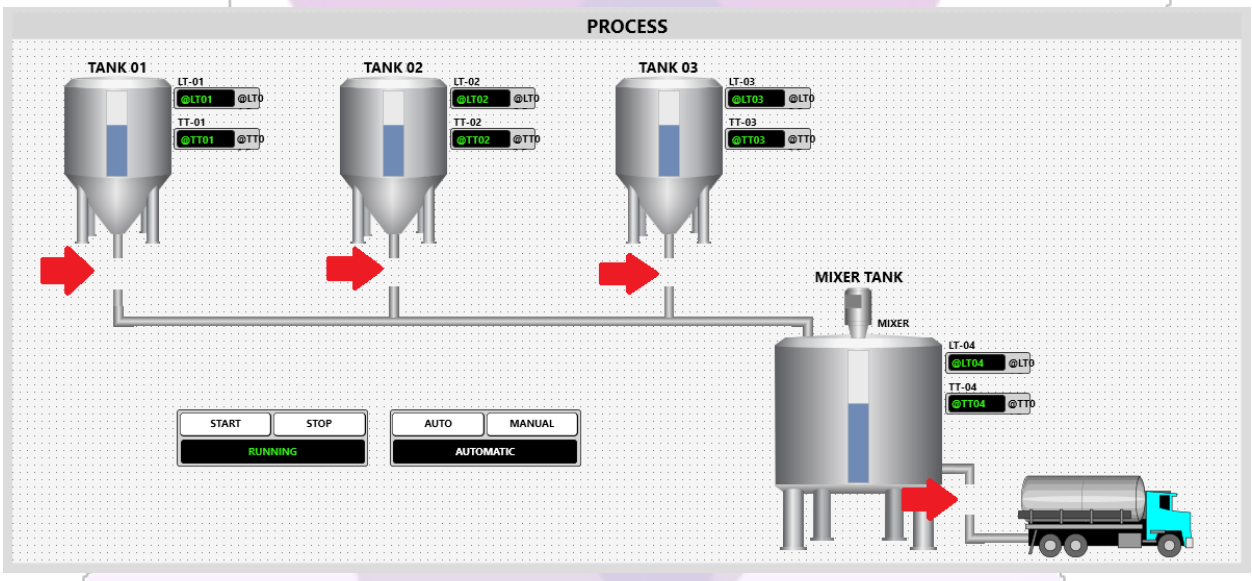


note: *The "Text" property of the label object was associated with the "Description" property of the valve's CMD tag. Probably, when we start RunTime to visualize the result, the label will have the description "(Command) Flow Valve 01 - Oil Tank 01". But since we want to show the name of each valve, for example "FV-01", we will need to change the "Description" property of each CMD tag.*

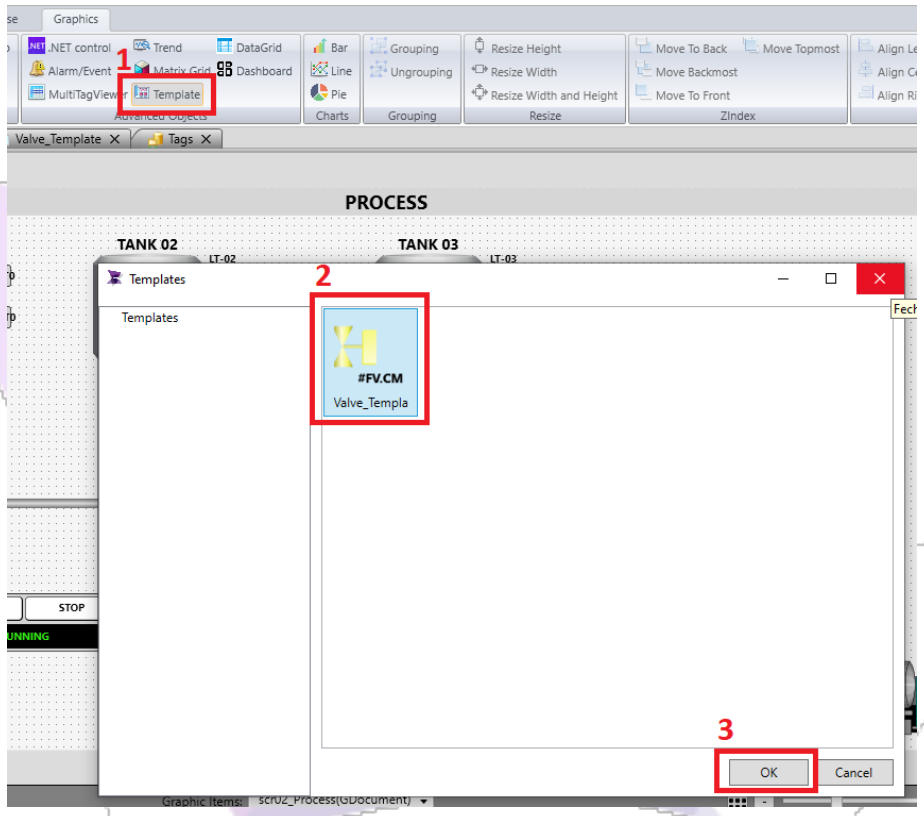
We will do it this way because we will not use the FVXX.CMD tag description in our application. Therefore, open the tags document, and change the description of the CMD tags of valves FV01, FV02, FV03 and FV04. After making the changes, save the application. Below is the expected result.



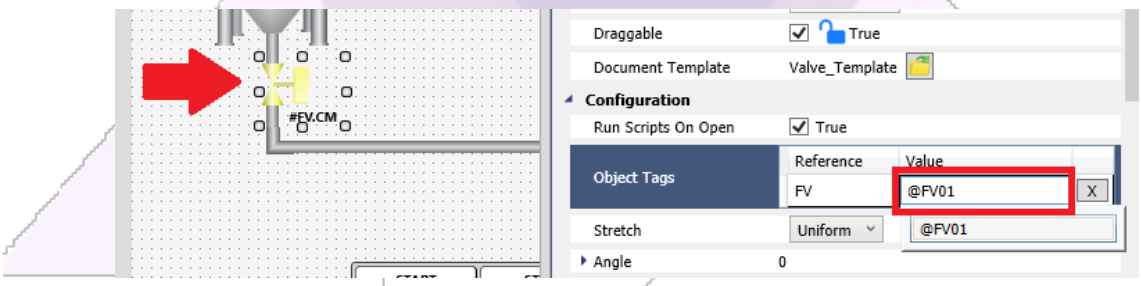
8. After making the above changes, save the application and return to the Graphic "scr02_Process". Now that we have the valve template created, let's add it to the Graphic. But first, let's delete the symbols and labels of the valves added earlier.



9. Finally, let's add a template by clicking on the template option and then clicking on the "scr02_Process" graphic.



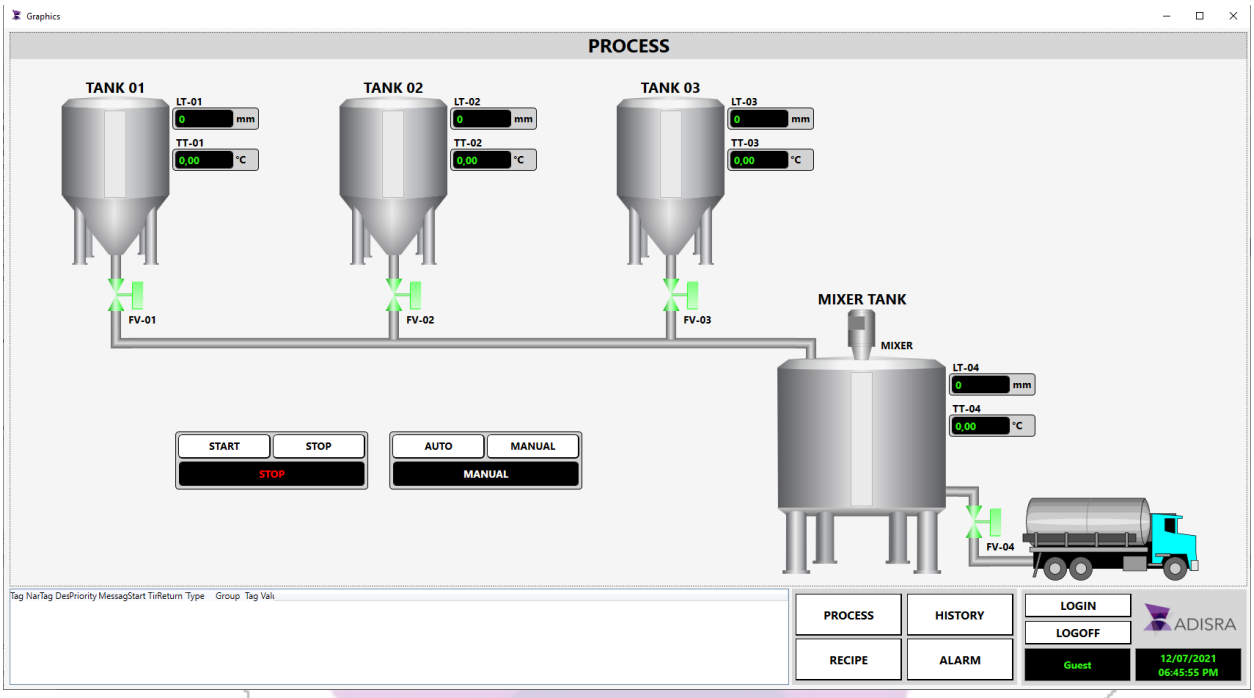
10. After adding it, click on the template and change the following property: Object Tags (@FV01). From now on, when the Graphic "scr02_Process" is started, the selected template will show the status of FV01 and the description of the tag FV01.CMD, as the reference "FV" used in the template will be replaced by FV01.



11. Now, let's add a template for each valve. Copy and Paste the previously added template for each remaining valve. Change the following properties:

    Template FV02: Location (334, 515) and Object Tags (@FV02)
    Template FV03: Location (334, 893) and Object Tags (@FV03)
    Template FV04: Location (643, 1306) and Object Tags (@FV04)

    Once this is done, save the application and start RunTime. Below is the expected result.

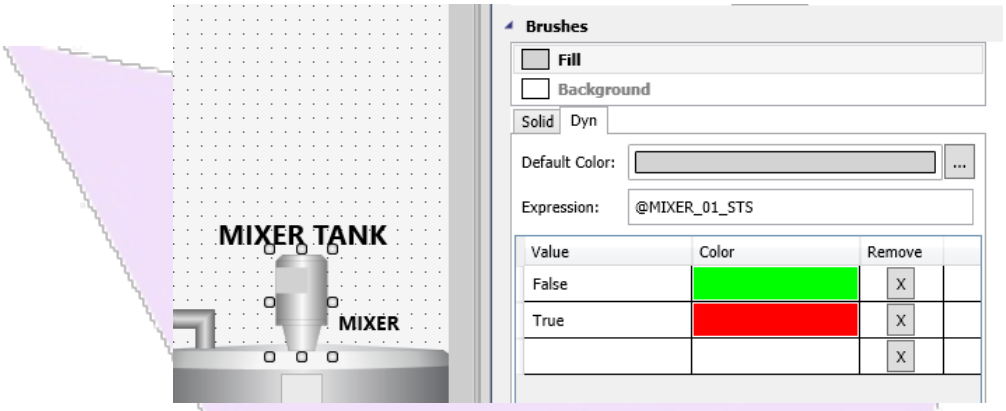


note: *After starting RunTime, open Data Watcher and test all gauges, valve status and process status. If you want to test without using the simulators, disable each communication driver and then change the state of the tags directly in the Data Watcher.*

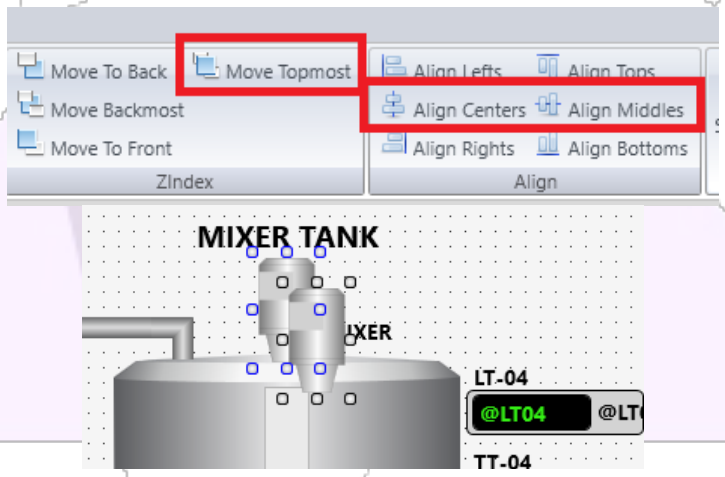
12. Now, let's configure the animation of the last industrial equipment. The Mixer, located above the Mixer Tank.

13. As this equipment is being used only in this one point of the application, we will not need to create a Generic Graphic. So, click on Symbol Mixer, and then change the

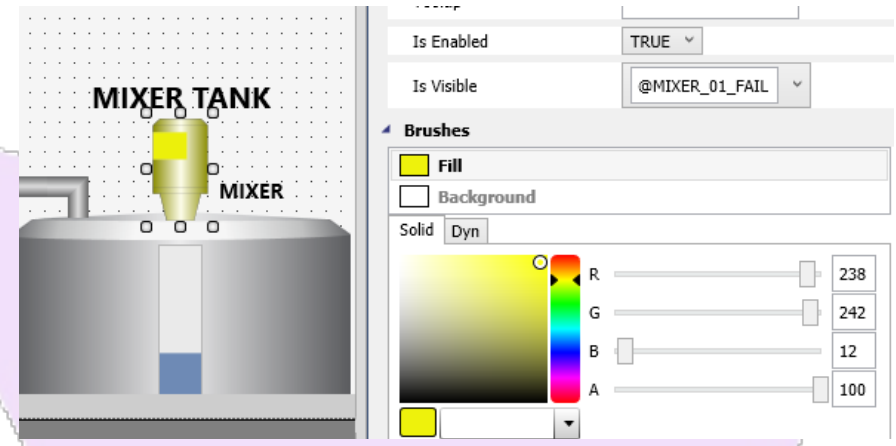
following properties: Brushes/Fill/Expression: @MIXER_01_STS - Value: False (Color 0, 255, 0) - Value: True (Color 255, 0, 0). Below is the expected result.



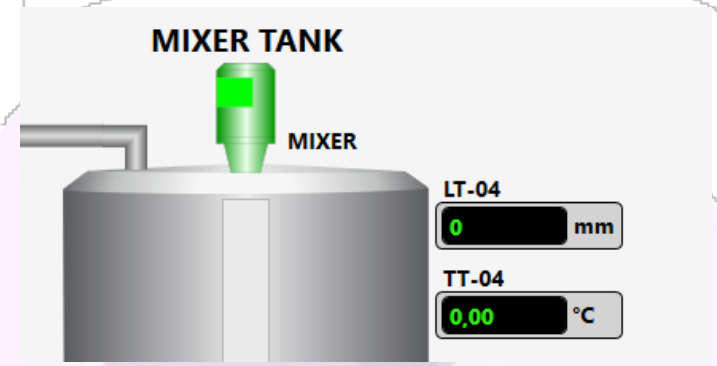
14. Now, let's add one more Symbol to the “Failed Mixer” animation. Copy and paste the current Symbol from the Mixer and then align the new Symbol with the previous symbol, so that it overlaps the previous symbol. To do this, first select the “Symbol Mixer in failure”, then hold down the “SHIFT” button on the keyboard and then select the previous Symbol, which will be the symbol reference for the alignment. Also use the “Move Topmost” feature to place the “Failed Mixer” overlaying the previous Symbol.



15. Now, select only the “Failed Mixer” and change the following properties:
Is Visible: @MIXER_01_FAIL and Brushes/Fill/Solid (Color 238, 242, 12) Below is the expected result.



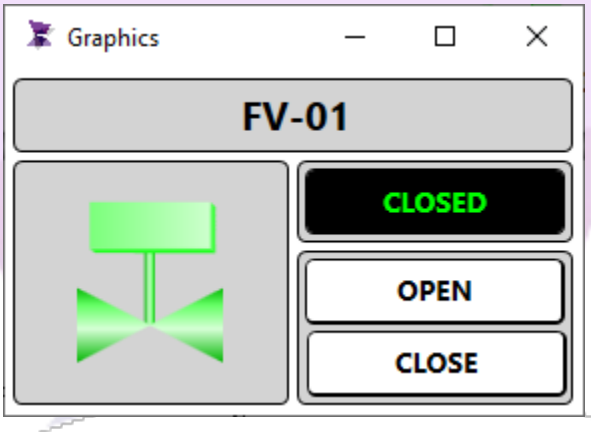
16. Once the above settings are done, start RunTime and test the Mixer animation by changing the @MIXER_01_STS tag values between “False” and “True” and it will also change the Failed Mixer visibility animation by changing the tag@MIXER_01_FAILin Data Watcher.



16. Developing Popups

Now let's develop the valve and mixer popups. Just like the application's Graphics, popups must be created in the “Graphics” document. As we will use the “Graphic Generic” feature, it will not be necessary to create a popup for each valve. Basically, the popups will consist of: Valve Identification, Valve Status and command buttons for opening and closing the valve.

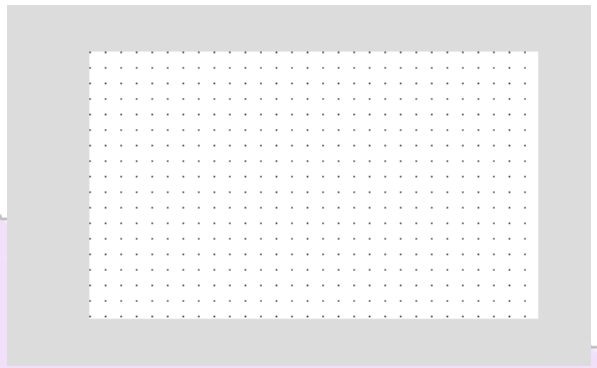
Below is a demonstration of the Popup Valves interface.



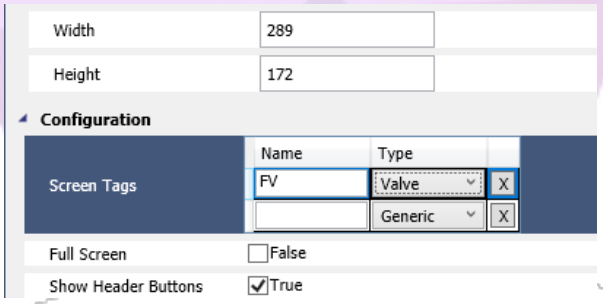
16.1. Inserting Screen Object

ADISRA SmartView allows us to configure a generic Graphic for different devices, bringing several benefits, such as: Decreased development time, creation of a standard interface and code reuse. Therefore, from now on we will create the popups interface and at the same time we will configure the graphic as generic. For this we will use the “Screen Tags” property.

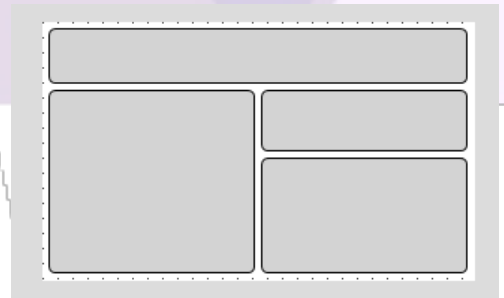
1. Let's start by creating the valves Popup. Right-click on the “Graphics” Document and then click on the “New Folder” option. Rename the folder to “Popups”.
2. Right-click on the new “Popup” folder and click on the “New Document” option.
3. Save the new Graphic as “pop01_Valve”.
4. With the Graphic “pop01_Valve” open, change the following property: Size (289, 172).



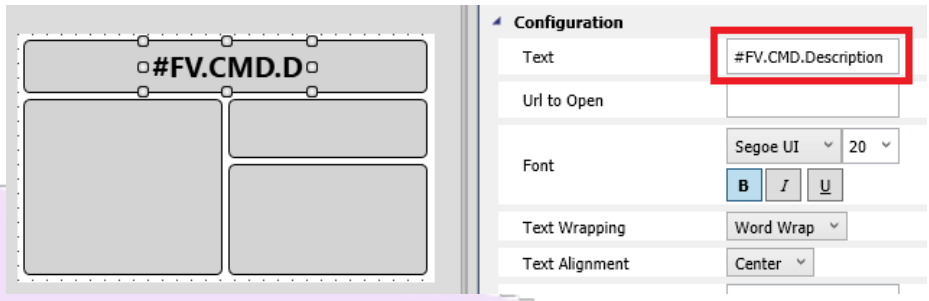
5. Now let's add a generic tag called “FV” in the Screen Tag property and select the Data Type “Valve”. This datatype was previously created in the DataType chapter, in which it was created to facilitate the structuring of tags.



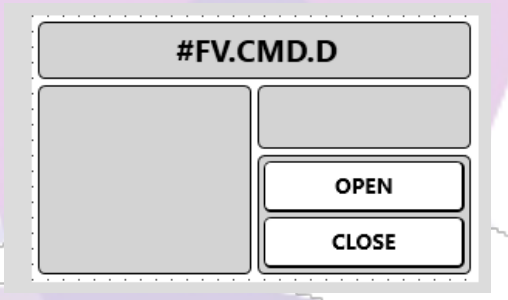
6. Now let's develop the graphical interface of the popup. Add four (4) rectangles and change the following properties:
Rectangle 01: Location (4, 4), Size (280, 37) and Corner Radius (4);
Rectangle 02: Location (45, 4), Size (138, 122) and Corner Radius (4);
Rectangle 03: Location (45, 146), Size (138, 41) and Corner Radius (4);
Rectangle 04: Location (90, 146), Size (138, 77) and Corner Radius (4);
Below is the expected result.



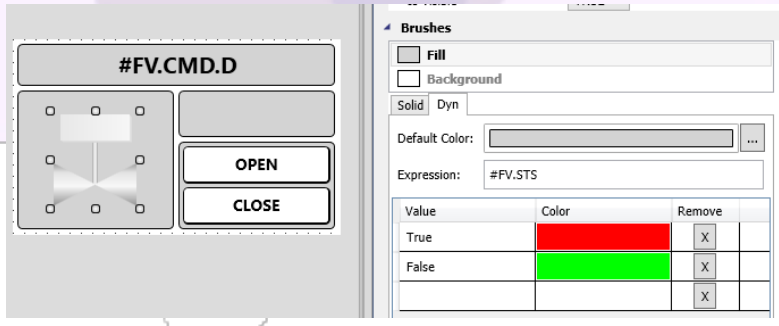
7. Add a Label and change the following properties: Location (8, 90), Size (111, 29) and Text (#FV.CMD.Descrption, Segoe UI, Bold, 20).



8. Add two (2) buttons and change the following properties:
   Button 01: Location (94, 150), Size (130, 33), Text (OPEN, Selgoe UI, Bold, 14) and add a script in the "Mouse Down" event (#FV.CMD = 1;).
   Button 02: Location (130, 150), Size (130, 33), Text (CLOSE, Selgoe UI, Bold, 14) and add a script in the "Mouse Down" event (#FV.CMD = 0;).
   Below is the expected result.

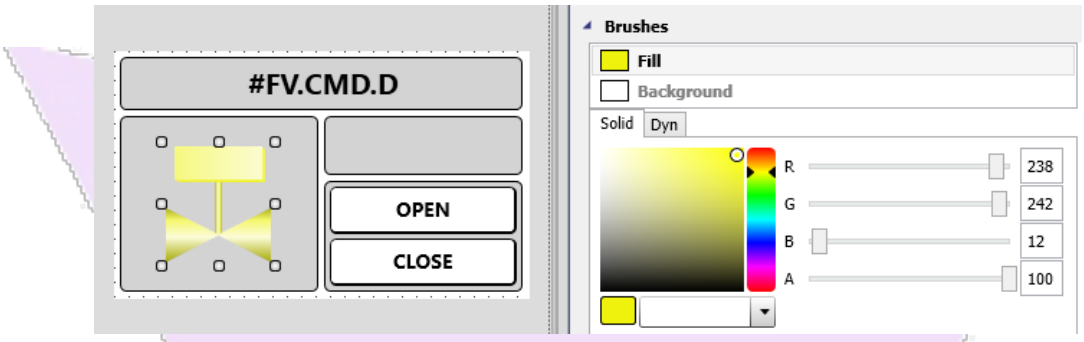


9. Now, let's add the Symbol of a valve which will represent the status of valve open and closed. In the Symbols window, click on the "Valves" category, then on the "3-D Valve with" Symbol. Change the following properties: Location (66, 36), Size (73, 80), Color Mode (Shade), Brushes/Fill/Dyn/Expression: (#FV.STS) - Value: True (255, 0, 0) - Value: False (0, 255, 0).

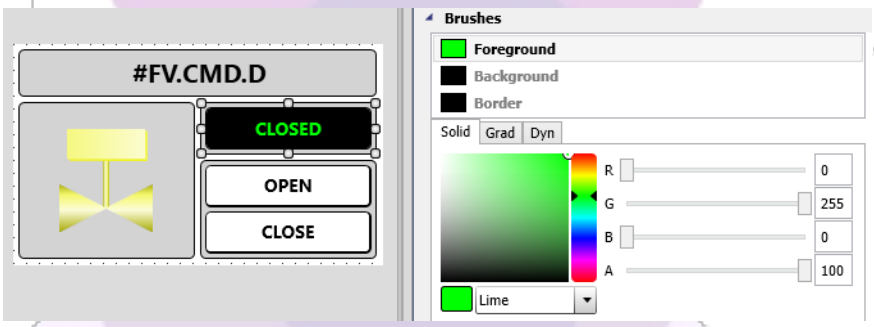


10. After adding the symbol which will represent the valve's open and closed status, let's add one more valve symbol to represent the valve's failure status. If the valve is faulty, this symbol will be visible above the previous symbol. Add one more "3-D Valve with" symbol. Change the following properties: Location (66, 36), Size (73, 80), Color Mode (Shade), Brushes/Fill (238, 242, 12) and Is Visible (#FV.FAIL). For this animation to work correctly, the

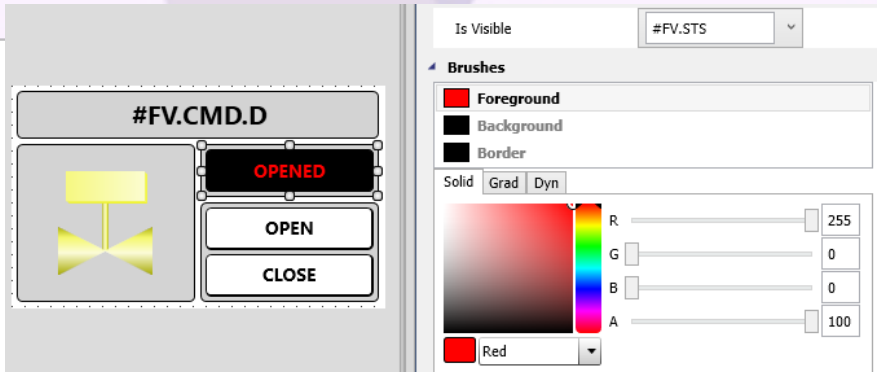
yellow valve symbol must be on top of the previous symbol. If the valve is faulty, the yellow valve symbol will override the previous symbol. Below is the expected result.



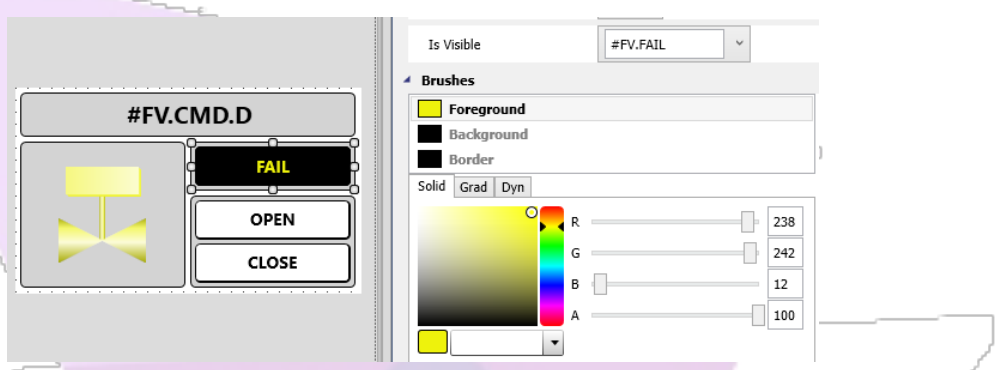
11. Now we will add another way to show the valve status. Add 1 Textbox and change the following Properties:
Location (49, 150), Size (130, 33), Corner Radius (4), Textbox (CLOSED, Selgoe UI, Bold, 14), Brushes/Foreground (Color 0, 255, 0) and Brushes/Background (Color 0, 0, 0).



12. Add 1 more Textbox which will represent the “Opened” status and change the following Properties:
Location (49, 150), Size (130, 33), Corner Radius (4), Textbox (OPENED, Selgoe UI, Bold, 14), Brushes/Foreground (Color 255, 0, 0), Brushes/Background (Color 0, 0, 0) and IsVisible (#FV.STS).

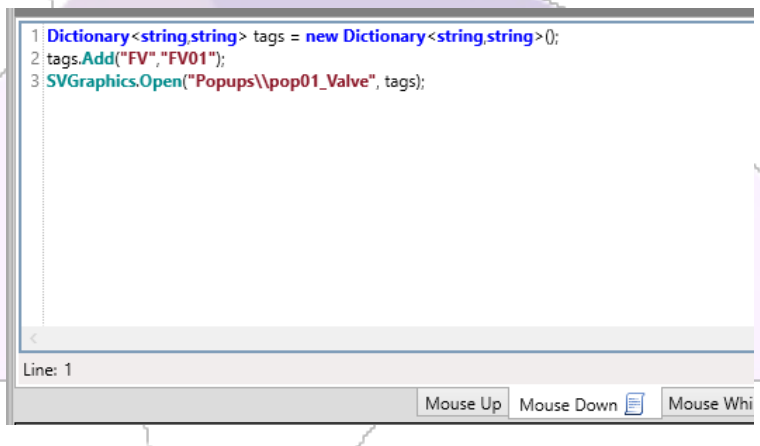


13. Add 1 more Textbox which will represent the “Fail” status and change the following Properties:
    Location (49, 150), Size (130, 33), Corner Radius (4), Textbox (FAIL, Selgoe UI, Bold, 14), Brushes/Foreground (Color 238, 242, 12), Brushes/Background (Color 0, 0, 0) and IsVisible (#FV.FAIL).



14. After creating the graphical interface of the valve popups, let's open the Graphic “scr02_Process” and configure the opening of the popups in the valve symbols. After opening the Graphic “scr02_Process”, click on the symbol and add a script in the “Mouse Down” event.

```
Dictionary<string,string>tags =new Dictionary<string,string>();
tags.add("FV","FV01");
SVGraphics.open("Popups\\pop01_Valve",tags);
```



The first line of the script declares a variable of type string in the dictionary. The second adds the contents of the variable, where FV01 is the tag and FV is the datatype. And the third line executes an opening function of Graphic and the path of the Graphic is passed and then the tag added above which will be passed to the Screen Tag of the Graphic “pop01_Valve”.

15. Once the settings for the first FV-01 are done, repeat the configuration for the other FV's and change only the name of each FV. Then start RunTime and test the valves Popup opening, and then open Data Watcher and test the valve animation by changing the

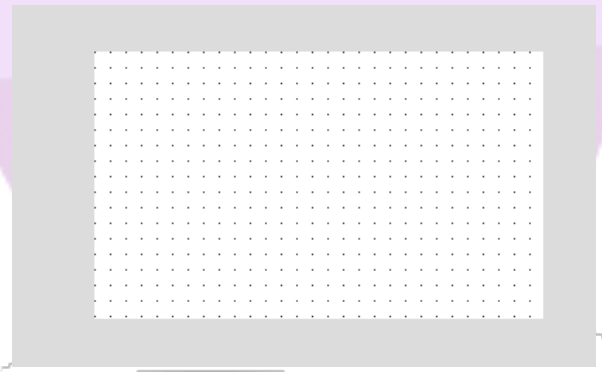
@FV01.STS tag values between “False” and “True”, the visibility animation of the failed valve by changing the tag@FV01FAIL and the command buttons.

Now, let's create the popup for the Mixer. However, it will not be necessary to configure it as Graphic Generic. Therefore, we will not use Screen Tags. The association will be direct with Mixer tags.

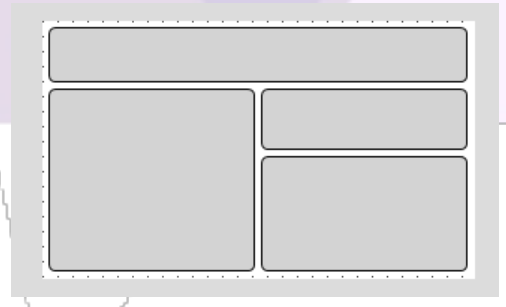
16. Right-click on the new “Popup” folder and click on the “New Document” option.

17. Save the new Graphic as “pop02_Mixer”.

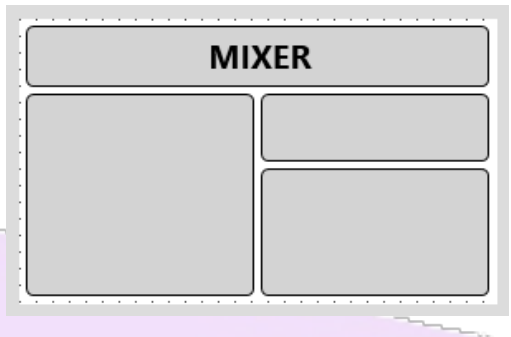
18. With the Graphic “pop02_Mixer” open, change the following property: Size (289, 172).



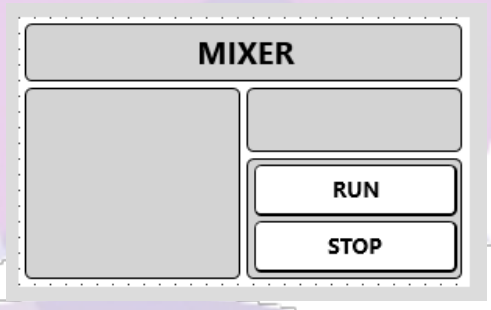
19. Add four (4) rectangles and change the following properties:
    Rectangle 01: Location (4, 4), Size (280, 37) and Corner Radius (4);
    Rectangle 02: Location (45, 4), Size (138, 122) and Corner Radius (4);
    Rectangle 03: Location (45, 146), Size (138, 41) and Corner Radius (4);
    Rectangle 04: Location (90, 146), Size (138, 77) and Corner Radius (4);
    Below is the expected result.



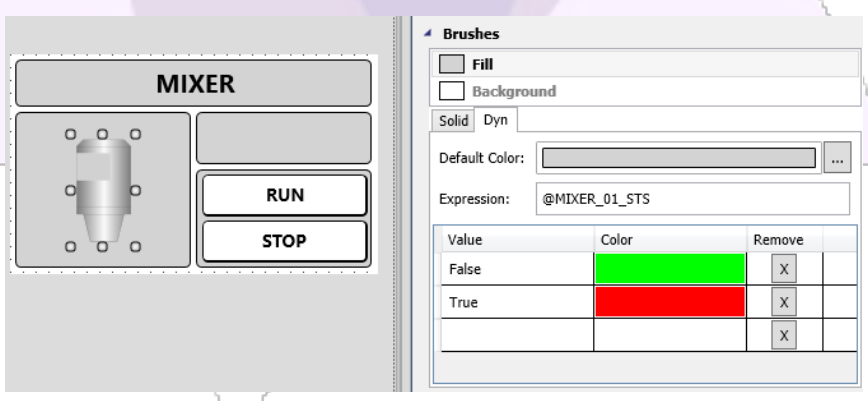
20. Add a Label and change the following properties: Location (8, 90), Size (111, 29) and Text (MIXER, Segoe UI, Bold, 20).



21. Add two (2) buttons and change the following properties:
    Button 01: Location (94, 150), Size (130, 33), Text (RUN, Selgoe UI, Bold, 14) and add a script in the “Mouse Down” event (@MIXER_01_CMD = 1;).
    Button 02: Location (130, 150), Size (130, 33), Text (STOP, Selgoe UI, Bold, 14) and add a script in the “Mouse Down” event (@MIXER_01_CMD = 0;).

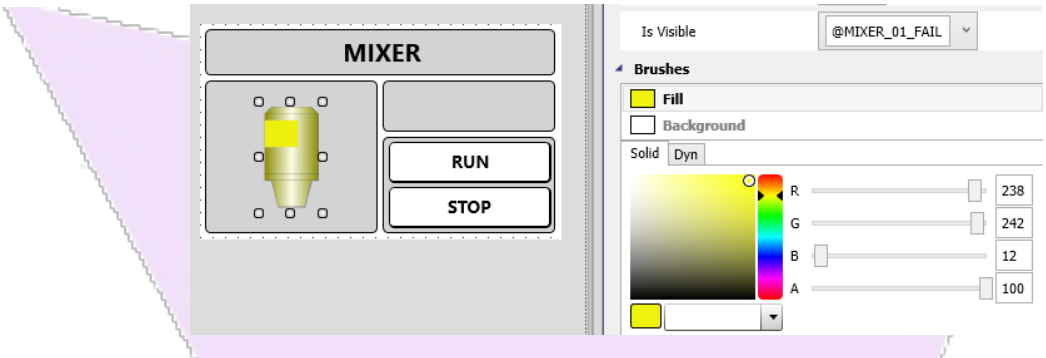


22. Now, let's add the Symbol of a Mixer which will represent the Mixer Run or Stop status. In the Symbols window, click on the “Motors” category, then on the “Simple Motor 01” Symbol. Change the following properties: Location (84, 32), Size (83, 45), Color Mode (Shade), Brushes/Fill/Dyn/Expression: (@MIXER_01_STS) - Value: True (255, 0, 0) - Value : False (0, 255, 0).

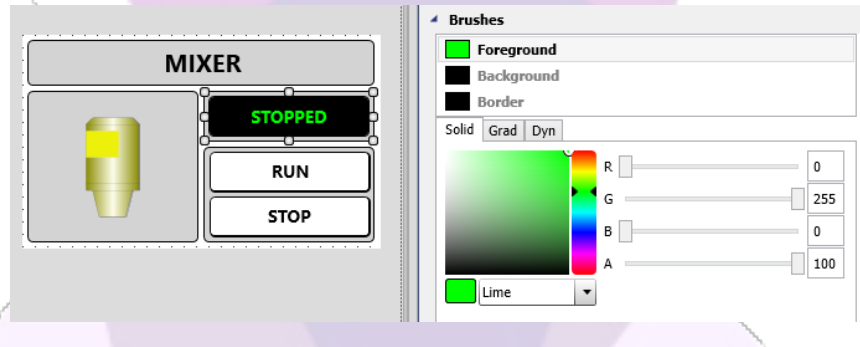


23. After adding the symbol that will represent the Run and Stop status of the Mixer, let's add one more Mixer symbol to represent the failure status. If the Mixer is faulty, this symbol will be visible above the previous symbol. Add one more symbol “Simple Motor 01”. Change the following properties: Location (84, 32), Size (83, 45), Color Mode (Shade), Brushes/Fill (238,

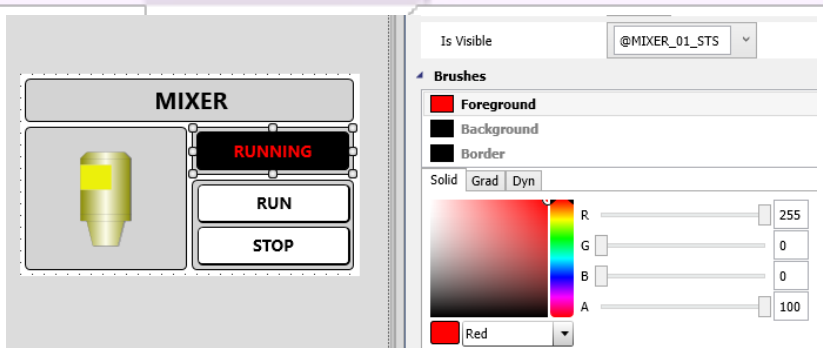
242, 12) and Is Visible (@MIXER_01_FAIL). For this animation to work correctly, the Mixer symbol in yellow must be on top of the previous symbol. If the valve is faulty, the yellow Mixer symbol will overwrite the previous symbol. Below is the expected result.



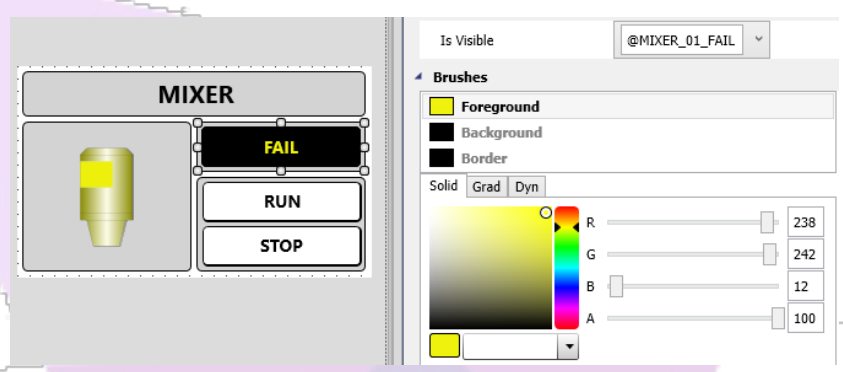
24. Now we'll add another way to show the Mixer status. Add 1 Textbox and change the following Properties:
Location (49, 150), Size (130, 33), Corner Radius (4), Textbox (STOPED, Selgoe UI, Bold, 14), Brushes/Foreground (Color 0, 255, 0) and Brushes/Background (Color 0 , 0, 0).



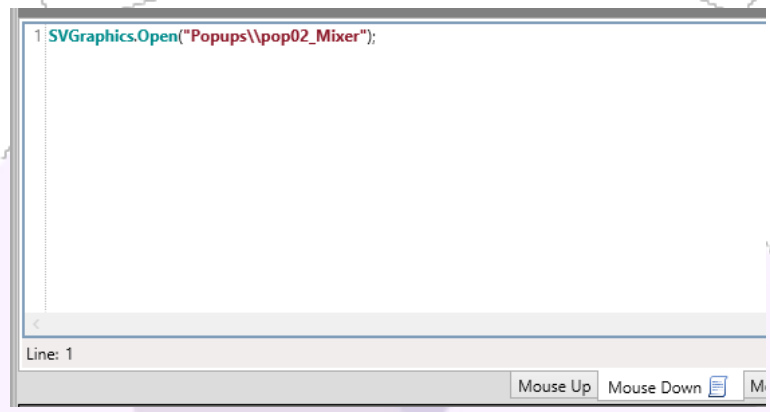
25. Add 1 more Textbox which will represent the “Running” status and change the following Properties:
Location (49, 150), Size (130, 33), Corner Radius (4), Textbox (RUNNING, Selgoe UI, Bold, 14), Brushes/Foreground (Color 255, 0, 0), Brushes/Background (Color 0 , 0, 0) and IsVisible (@MIXER_01_STS).



26. Add one more (1) Textbox which will represent the “Fail” status and change the following Properties:
    Location (49, 150), Size (130, 33), Corner Radius (4), Textbox (FAIL, Selgoe UI, Bold, 14), Brushes/Foreground (Color 238, 242, 12), Brushes/Background (Color 0 , 0, 0) and IsVisible (@MIXER_01_FAIL).



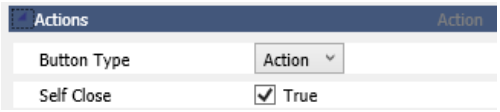
27. Now, let's configure the Graphic opening script “pop02_Mixer”. Open the Graphic “scr02_Process” and in the two Symbols that represent the status of the Mixer, add the script below in the “Mouse Down” event.Script:
    “SVGraphics.Open("Popups\\pop02_Mixer");”



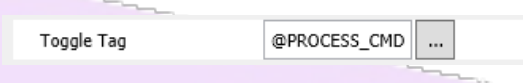
note: *Add the above script in Symbol representing “Running” and “Stop” status as in Symbol representing “Fail” status.*

For knowledge, it is also possible to open a graphic by passing a tag. For example, if you want to create the same Graphic call “pop02_Mixer” without script, add a button, and then click on the “Actions” property group. Then click on the “Button Type” property and enable the “Action” option. From now on, some properties are enabled, such as:

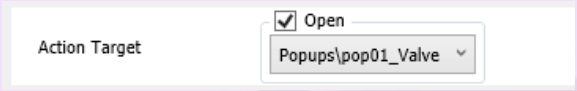
Self Close: If enabled, when clicking on the button, the Graphic where the button is located will be closed. This enabled property can, for example, be used as a “Close Popup” button.



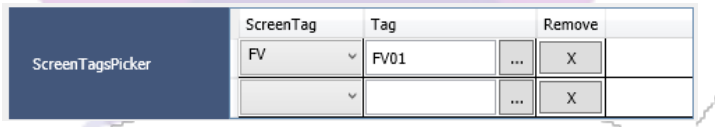
Toggle Tag: If enabled, when clicking on the button, a certain tag has its current value changed. This enabled property can be used, for example, to open or close a valve, switching from "False" to "True".



Action Target: If enabled, select a Graphic in the list and when clicking the button, the selected Graphic will open in a new window.

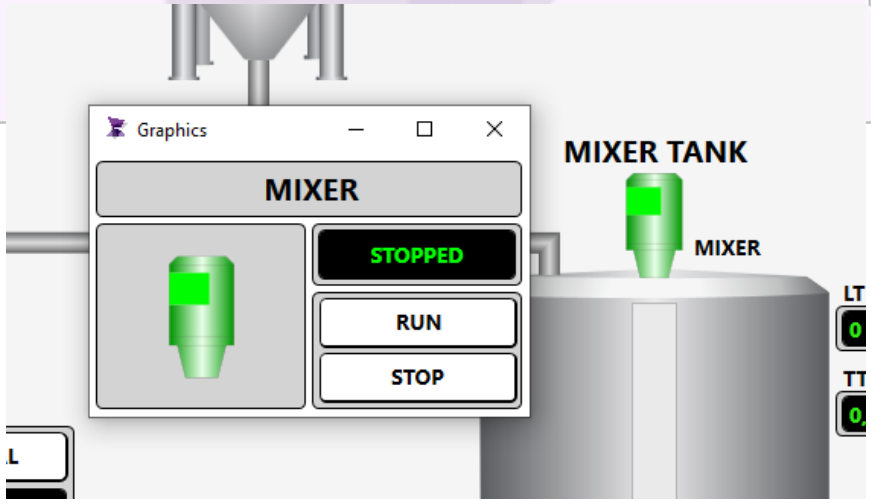


ScreenTagsPicker: If the Action Target is enabled, after selecting which Graphic will be opened, identify the Data Type and Tag name. For example, this property can be used to replace the script in which the Graphic "pop01_Valve" opens.



28. After knowing the "Actions" properties of a button and having made the opening Graphic settings of the Symbol Mixer, start RunTime and test the Mixer Popup opening, and then open the Data Watcher and test the Mixer animation by changing the @MIXER01_STS tag values between "False" and "True", the Mixer visibility animation fails by changing the tag@MIXER_01_FAIL and the command buttons.

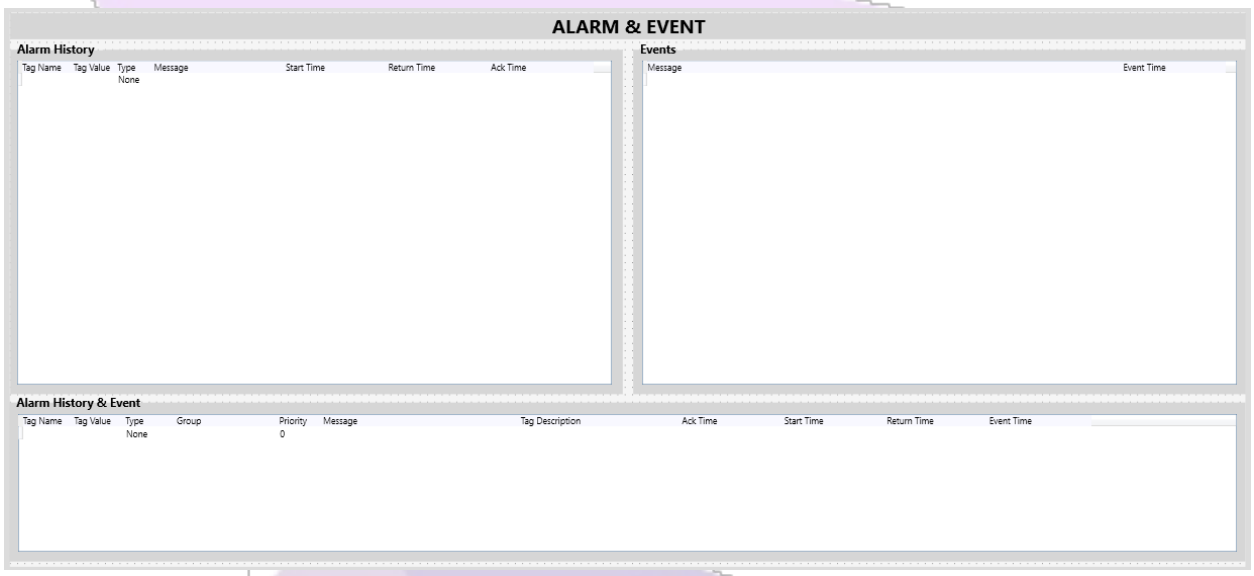
    Below is the final expected result of the Mixer Popup.



17. Developing Graphic Alarm & Event

Graphic Alarm & Event will be responsible for displaying all alarms and events in the application. In this chapter we will learn what are alarms & events are and how to configure their main functionalities and objects.

Below is the Graphic Alarm & Event demo.



17.1. What is an Event?

Event is the occurrence of an activity. It is a situation that occurs in the industrial process and has relevance for the operator as information but does not require any corrective action.

To configure the application's events, it is necessary to follow these steps:

1. Enable a proprietary or external database to store events;
2. Insert the “SVEvent” function in the objects in which the events will be generated;
3. Insert an “Alarm/Event” or “Datagrid” object to show the stored events.

We will know each step in the next topics.

17.2. What is an Alarm?

An alarm is a type of event indicated by auditory or visual means that indicates an unexpected condition in the process, equipment, system or instrument that requires corrective action in a restricted time, that is, it is a situation that occurs in the industrial process and requires that the operator takes action in relation to it under penalty of putting the plant in a situation of disturbance or even danger. Therefore, we can say that every alarm is an event, but not every

event is an alarm.

Alarms are commonly used in SCADA software and add important functionality for monitoring tag values. If, for example, a process tag has a value that could be dangerous, say, outside an acceptable range, the application must notify the operator of this danger. In ADISRA SmartView, the user can create alarms for the tags and save the alarms in the database, display them on a monitoring screen, allowing the operator to view the alarms and acknowledge them, and take action to normalize the alarm. In addition to real-time monitoring of alarms, ADSIRA SmartView provides a document called “Alarm History” in which all configured alarms can be stored and consulted in a proprietary or external database,

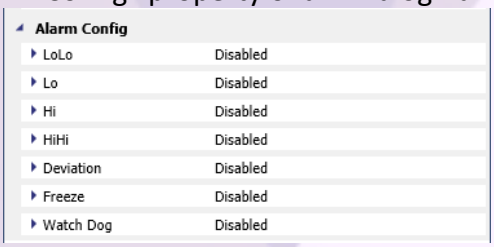
To configure a tag alarm, it is necessary to follow these steps:

1. Enable and configure the alarms for each tag to be monitored;
2. Insert an “Alarm/Event” object in the Graphic;
3. Insert and configure the tags in the “Alarm/Event” object, in which their alarms will be displayed;
4. Create an “Alarm History” document and insert the tags in which your alarms will be stored in the database.

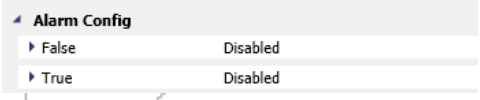
17.2.1. Types of Alarm

Alarms are divided into types, and each type triggers the alarm based on some conditions. Below is a description of each alarm type available in ADISRA SmartView.

“Alarm Config” property of an Analog Variable



“Alarm Config” property of a Digital Variable



- **Analog Alarm Type (LoLo, Lo, Hi and HiHi)**

This alarm type allows you to monitor an analog variable based on four alarm conditions.

LoLo condition

The alarm is triggered if the value of the monitored tag is lower than the configured value.

Lo condition

The alarm is triggered if the value of the monitored tag is lower than the configured value.

Hi condition

The alarm is triggered if the value of the monitored tag is greater than the configured value.

HiHi Condition

The alarm is triggered if the value of the monitored tag is greater than the configured value.

- **Analog Alarm Type (Deviation)**

  This type of alarm allows monitoring an analog variable in case the value of the monitored tag varies very quickly.

- **Analog Alarm Type (Freeze)**

  This type of alarm allows monitoring an analog variable by defining a deadband value, that is, a maximum limit in relation to a reference value (SetPoint). In this way, it is possible to avoid an unnecessary amount of alarms for small oscillations of the monitored variable.

- **Analog Alarm Type (Watch Dog)**

  This type of alarm allows an analog variable to be monitored for a tag variation that has been interrupted. Some process variables are expected to change their value within a time interval and the watchdog will be triggered if it does not change its value. It is similar to the Freeze alarm, but in this case, there is no need to configure Deadband. It can be any scale of change in value.

- **Digital Alarm Type (True)**

  This type of alarm allows monitoring a digital variable on the rising edge, that is, when the variable's value changes from False to True.

- **Digital Alarm Type (False)**

  This type of alarm allows monitoring a digital variable on the falling edge, that is, when the variable's value changes from True to False.

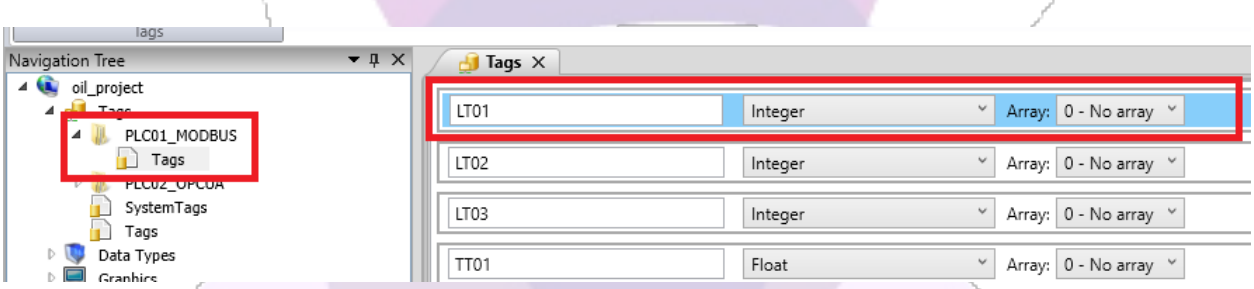
17.2.2. Configuring Alarms

Now that we know what alarms and their types are, let's start configuring our application's alarm system.

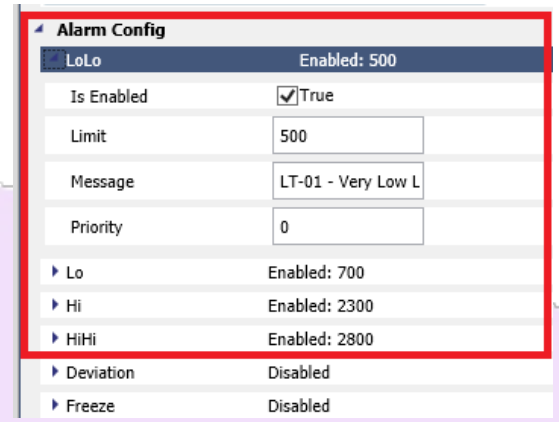
1. Initially, we will configure the “Alarm Config” property of each tag in our project based on the tag list provided at the beginning of the training. So, according to the tag list, the first tag “LT-01” needs to be configured as follows: Lolo = 500, Lo = 700, Hi = 2300 and HiHi = 2800.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TCP	40001	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TCP	40002	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TCP	40003	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TCP	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TCP	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TCP	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TCP	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TCP	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TCP	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TCP	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TCP	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TCP	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TCP	7	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TCP	8	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TCP	9	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA	-	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_04_FAIL	(Fail) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

2. Then, to configure the alarm properties of the LT-01 tag, open the “Tags” document in the “PLC01_MODBUS” folder and click on the “LT-01” tag.



3. After selecting the tag “LT-01”, change the following properties:
   LoLo Condition - IsEnabled (True), Limit(500), Priority(0) and Message(LT-01 - Very Low Level);
   Lo Condition - IsEnabled (True), Limit(700), Priority(0) and Message(LT-01 - Low Level);
   Condition Hi - IsEnabled (True), Limit(2300), Priority(0) and Message(LT-01 - High Level);
   HiHi Condition - IsEnabled (True), Limit(2800), Priority(0) and Message(LT-01 - Very High Level);



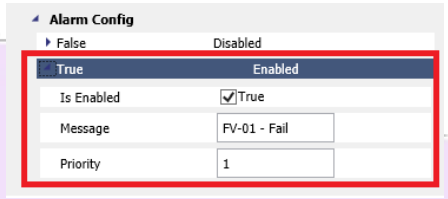
4. Repeat the alarm configuration for all analog variables in the project according to the tag list. Below, all analog variables are highlighted in red. For the Temperature variables, replace the word “Level” with “Temperature” in the message field. To facilitate the individual configuration of each tag, it is possible to export the list of tags to a “.CSV” format file and change the alarm settings respecting the file structure.

   After editing, save and import the “.CSV” into ADISRA SmartView. The Import and Export Buttons are located in the top left menu of ADISRA SmartView.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TCP	40001	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TCP	40002	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TCP	40003	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TCP	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TCP	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TCP	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TCP	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TCP	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TCP	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TCP	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TCP	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TCP	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TCP	7	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TCP	8	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TCP	9	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA	-	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_04_FAIL	(Fail) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

5. After configuring the analog tags, let's configure the digital tags. Regarding the list of tags, we have 05 digital tags which represent valve failure. Therefore, access the first digital tag called “FV_01_FAIL and change the following properties:

   True Condition - IsEnalbed (True), Message(FV-01 - Fail) and Priority(1).



   note: *We only enable the “True” condition, because we want the alarm to be triggered when the FV_01_FAIL tag goes from “False” to “True”.*

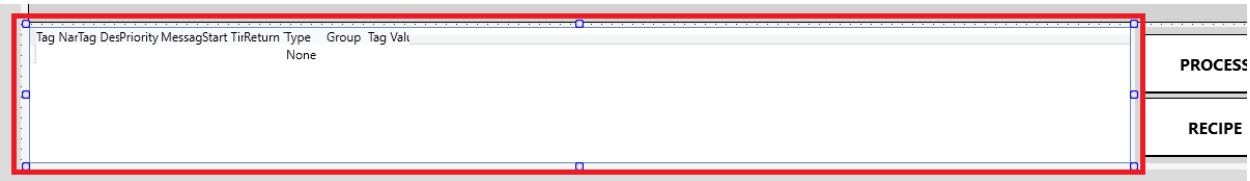
6. Repeat the alarm configuration for the project's digital variables according to the tag list. Below, all digital variables are highlighted in red. To facilitate the individual configuration of each tag, it is possible to export the list of tags to a “.CSV” format file and change the alarm settings respecting the file structure. After editing, save and import the “.CSV” into ADISRA SmartView. The Import and Export Buttons are located in the top left menu of ADISRA SmartView.

PLC	TAG	DESCRIPTION	PROTOCOL	MODBUS ADDRESS	TYPE	RANGE	ENG. UN.	ALARMS			
								LL	L	H	HH
PLC_01	LT_01	Level Transmitter - Oil Tank 01	Modbus-TCP	40001	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_02	Level Transmitter - Oil Tank 02	Modbus-TCP	40002	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	LT_03	Level Transmitter - Oil Tank 03	Modbus-TCP	40003	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_01	TT_01	Temperature Transmitter - Oil Tank 01	Modbus-TCP	40011	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_02	Temperature Transmitter - Oil Tank 02	Modbus-TCP	40015	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	TT_03	Temperature Transmitter - Oil Tank 03	Modbus-TCP	40019	FLOAT	0 - 100	°C	15	30	70	85
PLC_01	FV_01_CMD	(Command) Flow Valve 01 - Oil Tank 01	Modbus-TCP	1	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_02_CMD	(Command) Flow Valve 02 - Oil Tank 02	Modbus-TCP	2	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_03_CMD	(Command) Flow Valve 03 - Oil Tank 03	Modbus-TCP	3	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_01	FV_01_STS	(Status) Flow Valve 01 - Oil Tank 01	Modbus-TCP	4	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_02_STS	(Status) Flow Valve 02 - Oil Tank 02	Modbus-TCP	5	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_03_STS	(Status) Flow Valve 03 - Oil Tank 03	Modbus-TCP	6	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_01	FV_01_FAIL	(Fail) Flow Valve 01 - Oil Tank 01	Modbus-TCP	7	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_02_FAIL	(Fail) Flow Valve 02 - Oil Tank 02	Modbus-TCP	8	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_01	FV_03_FAIL	(Fail) Flow Valve 03 - Oil Tank 03	Modbus-TCP	9	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	LT_04	Level Transmitter - Mixer Tank	OPC-UA	-	INTEGER	0 - 3000	mm	500	700	2300	2800
PLC_02	TT_04	Temperature Transmitter - Mixer Tank	OPC-UA	-	FLOAT	0 - 100	°C	15	30	70	85
PLC_02	FV_04_CMD	(Command) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Open = 1 / Close = 0)	-	-	-	-	-
PLC_02	FV_04_STS	(Status) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Opened = 1 / Closed = 0)	-	-	-	-	-
PLC_02	FV_04_FAIL	(Fail) Flow Valve 04 - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	MIXER_01_CMD	(Command) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	MIXER_01_STS	(Status) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	MIXER_01_FAIL	(Fail) Mixer - Mixer Tank	OPC-UA	-	BOOL	(Fail = 1 / Ok = 0)	-	Active Alarm			
PLC_02	PROCESS_CMD	(Command) Process Run / Stop	OPC-UA	-	BOOL	(Run = 1 / Stop = 0)	-	-	-	-	-
PLC_02	PROCESS_STS	(Status) Process Running / Stopped	OPC-UA	-	BOOL	(Running = 1 / Stopped = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_CMD	(Command) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-
PLC_02	PROCESS_AM_STS	(Status) Process Automatic/Manual	OPC-UA	-	BOOL	(Auto = 1 / Man = 0)	-	-	-	-	-

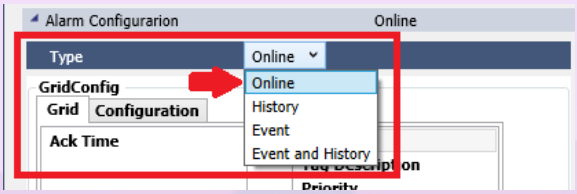
17.2.3. Configuring the Alarm/Event Object

After configuring the alarm properties of each tag, the next step will be to configure the Alarm/Event object, which will be responsible for displaying the application's active alarms.

1. Open the Graphic “scr01_Main” and click on the summary object located at the bottom of the Graphic.



2. Then configure the following properties: Type (Online)



As mentioned earlier, the summary object is intended to show active alarms. However, in addition to showing active alarms, that is, currently triggered alarms, it is also possible to configure it to show alarm history, online events and the combination of events and alarm history.

The Alarm/Event object can be configured in 04 ways through the Type property, such as Online, History, Event and Event and History.

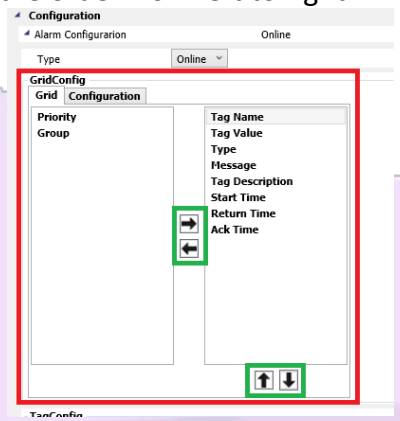
Online: Shows only currently triggered alarms. As long as the alarm is active, it will remain on the display. If the alarm is normalized, the alarm will change its color and will only disappear after the operator acknowledges it. If the operator recognizes the alarm still active, the alarm will have its color changed and when the alarm is normalized, it will automatically disappear from the summary, as it has already been acknowledged previously.

History: It only shows the history of alarms triggered over time. The alarm history of a particular tag will only appear in the summary if the “AlarmHistory” document is configured. We will find out about this document soon.

Event: Show only events triggered by the application. Events are configured and triggered by the “SVEvent” function, which we will learn about later.

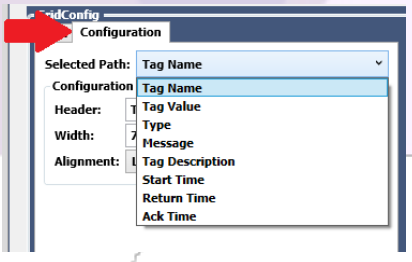
Event and History: Shows the events generated by the “SVEvent” function and the alarm history.
Next, let's continue with the configuration of the Alarm/Event object of the “scr01_Main” graphic.

3. After selecting the “Alarm/Event” object and changing the “Type” property to online, change the “GridConfig” property as shown in the image below. This configuration allows us to define which fields will be displayed in the alarm summary. Use the arrows to select which fields will be displayed and set the order from left to right.



4. **Definition of each field:**
   Tag Name: Name of the tag on which the alarm is being monitored;
   Tag Value: Tag value at the time of the triggered alarm;
   Type: Alarm Condition (Lolo, Lo, Hi, HiHi);
   Message: Alarm Message;
   Tag Description: Description of the tag on which the alarm is being monitored;
   Start Time: Date and time the alarm was triggered;
   Return Time: Date and time the alarm was normalized;
   Ack Time: Date and time the alarm was acknowledged;
   Priority: Alarm priority level;
   Group: Group that the alarm represents.

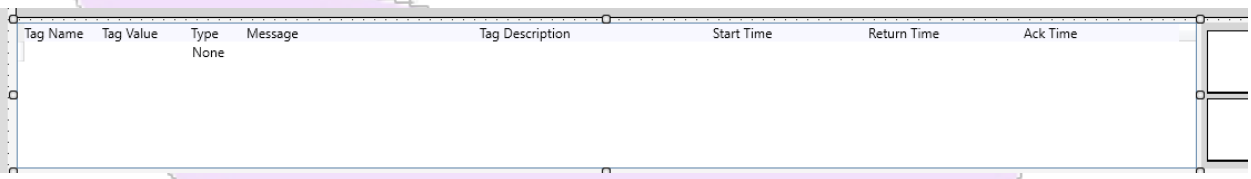
5. In the “GridConfig” property, click on the “Configuration” tab.



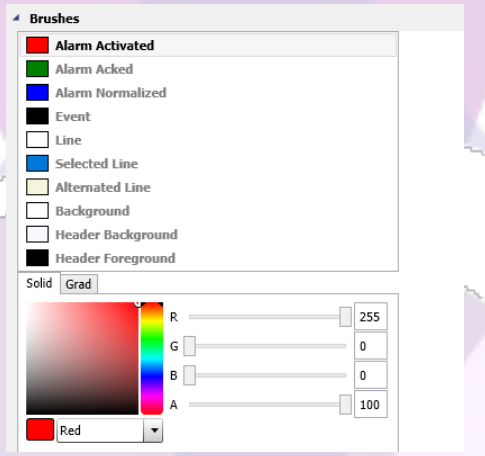
6. This tab allows us to configure the name, width and alignment of each field selected in the “Grid” tab. So change the following settings:
   Tag Name: Header(Tag Name), Witdth(70) and Alignment(Left);
   Tag Value: Header(Tag Value), Witdth(80) and Alignment(Left);
   Type: Header(Type), Witdth(50) and Alignment(Left);
   Message: Header(Message), Witdth(210) and Alignment(Left);
   Tag Description: Header(Tag Description), Witdth(210) and Alignment(Left);

Start Time : Header(Start Time), Witdth(140) and Alignment(Left);
Return Time: Header(Return Time), Witdth(140) and Alignment(Left);
Ack Time: Header(Ack Time), Witdth(140) and Alignment(Left);

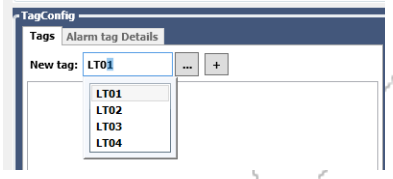
7. Then below is the expected result.

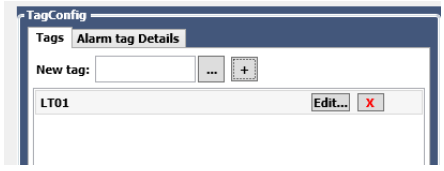


8. Browsing the properties of the Alarm/Event object, there is a property called “Brushes”. This property allows us to change the text color for each alarm situation. For example, when an alarm is active, the alarm text will be in red, if the alarm normalizes, the alarm text will be in blue, if the alarm is acknowledged, the alarm text will be in green and so on. It will not be necessary to change this property as we will keep the current configuration.

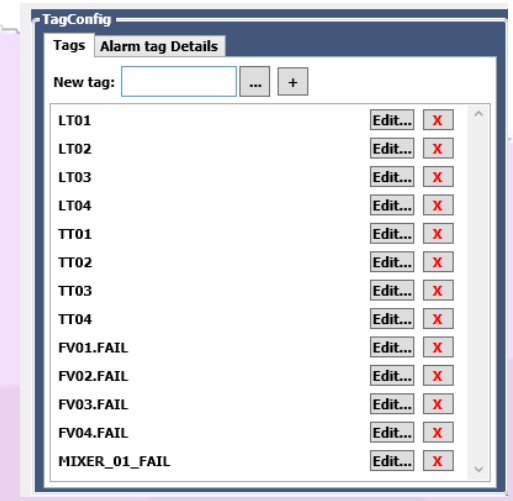


9. After configuring the alarm summary interface, let's add the tags that will have their alarms shown in this summary. To do so, with the object selected, find the “TagConfig” property and type in the “New Tag” the tag “LT01” and then click the “+” button. It is also possible by clicking the “...” button.

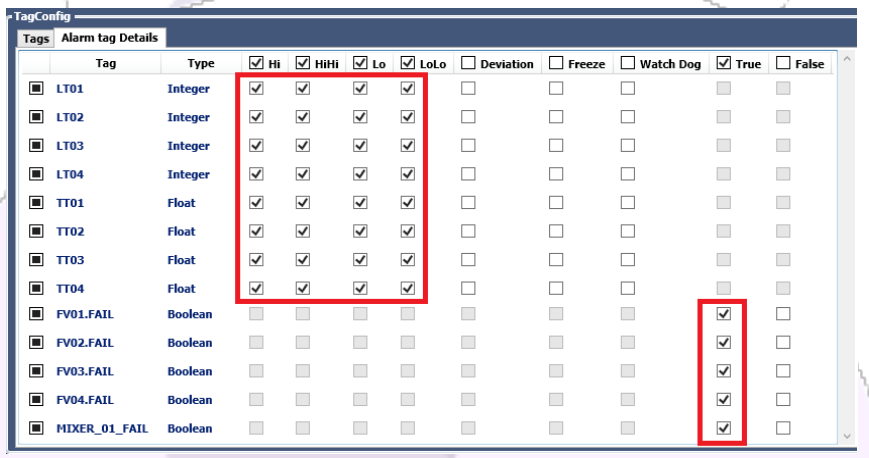




10. Now add any remaining tags that will be monitored by the alarm summary. Below is the expected result.



11. After adding the tags, click on the "Alarm tag Details" tab to enable which alarm conditions for each tag will be shown in the summary.

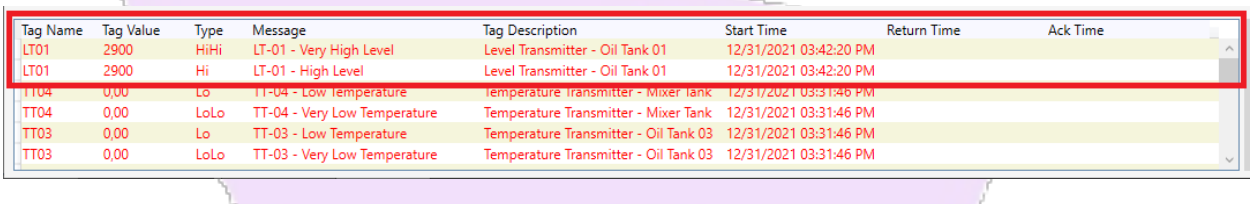


After knowing the alarm settings, what is alarm and event, the configuration of the Alarm/Event object, save the application and start RunTime. Once the viewer is open, the alarm summary should look like this:

Tag Name	Tag Value	Type	Message	Tag Description	Start Time	Return Time	Ack Time
TT04	0,00	Lo	TT-04 - Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:31:46 PM		
TT04	0,00	LoLo	TT-04 - Very Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:31:46 PM		
TT03	0,00	Lo	TT-03 - Low Temperature	Temperature Transmitter - Oil Tank 03	12/31/2021 03:31:46 PM		
TT03	0,00	LoLo	TT-03 - Very Low Temperature	Temperature Transmitter - Oil Tank 03	12/31/2021 03:31:46 PM		
TT02	0,00	Lo	TT-02 - Low Temperature	Temperature Transmitter - Oil Tank 02	12/31/2021 03:31:46 PM		
TT02	0,00	LoLo	TT-02 - Very Low Temperature	Temperature Transmitter - Oil Tank 02	12/31/2021 03:31:46 PM		

All analog alarms have Lo and LoLo conditions active because the current value of all tags is zero. The digital alarms will not be active because the “True” condition has not been triggered, since the digital alarms start with the value “False”.

12. Open Data Watcher and test the triggering of each alarm. For example, change the value of tag “LT-01” to 2900, and note that the HiHi alarm will be triggered as shown in the image below.

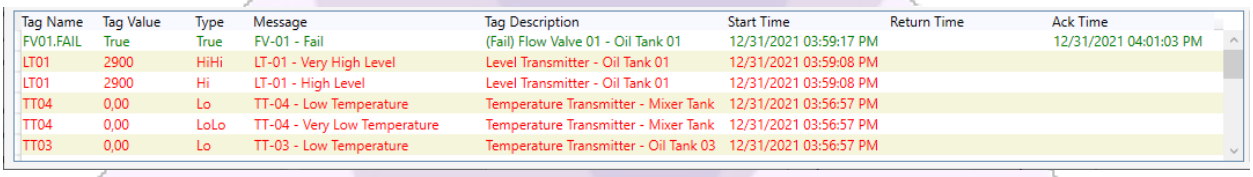


Tag Name	Tag Value	Type	Message	Tag Description	Start Time	Return Time	Ack Time
LT01	2900	HiHi	LT-01 - Very High Level	Level Transmitter - Oil Tank 01	12/31/2021 03:42:20 PM		
LT01	2900	Hi	LT-01 - High Level	Level Transmitter - Oil Tank 01	12/31/2021 03:42:20 PM		
TT04	0,00	Lo	TT-04 - Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:31:46 PM		
TT04	0,00	LoLo	TT-04 - Very Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:31:46 PM		
TT03	0,00	Lo	TT-03 - Low Temperature	Temperature Transmitter - Oil Tank 03	12/31/2021 03:31:46 PM		
TT03	0,00	LoLo	TT-03 - Very Low Temperature	Temperature Transmitter - Oil Tank 03	12/31/2021 03:31:46 PM		

13. Now, change the value of the tag “FV01.FAIL” to “True” and note that the alarm will be triggered as shown in the image below.

Tag Name	Tag Value	Type	Message	Tag Description	Start Time	Return Time	Ack Time
FV01.FAIL	True	True	FV-01 - Fail	(Fail) Flow Valve 01 - Oil Tank 01	12/31/2021 03:59:17 PM		
LT01	2900	HiHi	LT-01 - Very High Level	Level Transmitter - Oil Tank 01	12/31/2021 03:59:08 PM		
LT01	2900	Hi	LT-01 - High Level	Level Transmitter - Oil Tank 01	12/31/2021 03:59:08 PM		
TT04	0,00	Lo	TT-04 - Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:56:57 PM		
TT04	0,00	LoLo	TT-04 - Very Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:56:57 PM		
TT03	0,00	Lo	TT-03 - Low Temperature	Temperature Transmitter - Oil Tank 03	12/31/2021 03:56:57 PM		

14. Then, double-click with the left mouse button on the “FV01.FAIL” tag alarm and note the color change from red to green, which represents the acknowledged and non-normalized alarm.

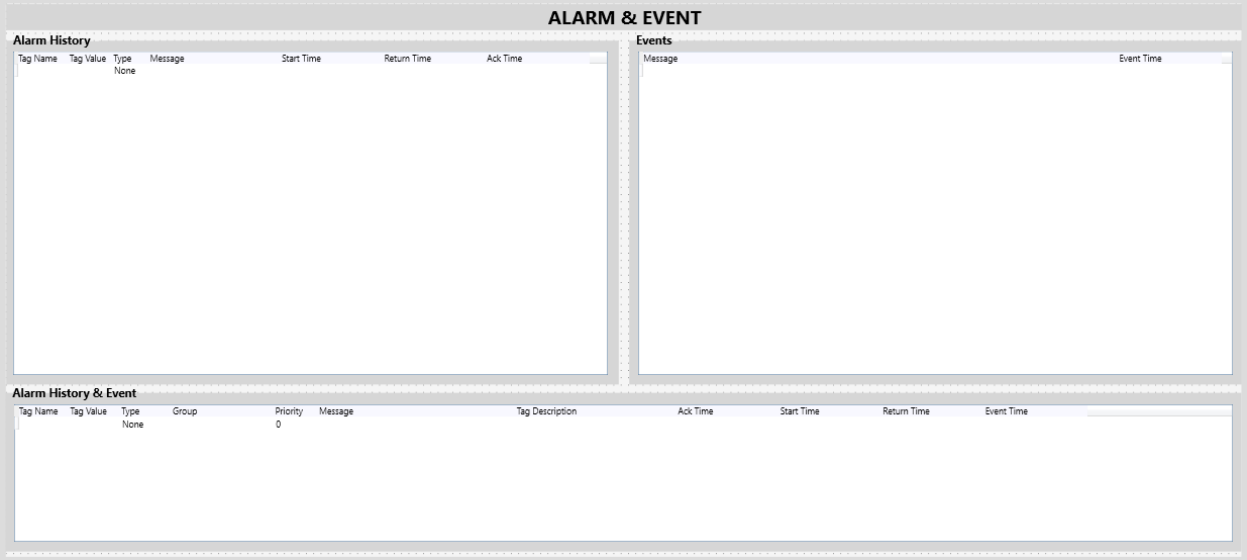


Tag Name	Tag Value	Type	Message	Tag Description	Start Time	Return Time	Ack Time
FV01.FAIL	True	True	FV-01 - Fail	(Fail) Flow Valve 01 - Oil Tank 01	12/31/2021 03:59:17 PM		12/31/2021 04:01:03 PM
LT01	2900	HiHi	LT-01 - Very High Level	Level Transmitter - Oil Tank 01	12/31/2021 03:59:08 PM		
LT01	2900	Hi	LT-01 - High Level	Level Transmitter - Oil Tank 01	12/31/2021 03:59:08 PM		
TT04	0,00	Lo	TT-04 - Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:56:57 PM		
TT04	0,00	LoLo	TT-04 - Very Low Temperature	Temperature Transmitter - Mixer Tank	12/31/2021 03:56:57 PM		
TT03	0,00	Lo	TT-03 - Low Temperature	Temperature Transmitter - Oil Tank 03	12/31/2021 03:56:57 PM		

15. Now, normalize the value of the tag “FV01.FAIL” changing it to “False”. Note that the alarm automatically disappeared from the alarm summary, as it has been acknowledged before. If it had not been acknowledged, the alarm would remain blue in the summary, requesting recognition by the operator.

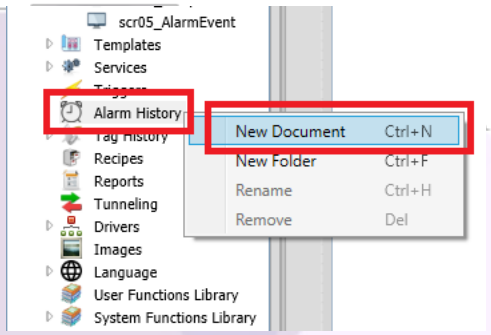
17.2.4. Configuring the Alarm History Document

After configuring the “Alarm/Event” object of the “scr01_Main” graphic, which will be responsible for displaying online alarms, that is, the current alarms, we will start to develop and configure the objects of the “scr05_AlarmEvent” graphic. This graphic will consist of 03 “Alarm/Event” objects, which will show the history of alarms, events and a combined history of alarms and events. Below is the final expected result of the graphic “scr05_AlarmEvent”.

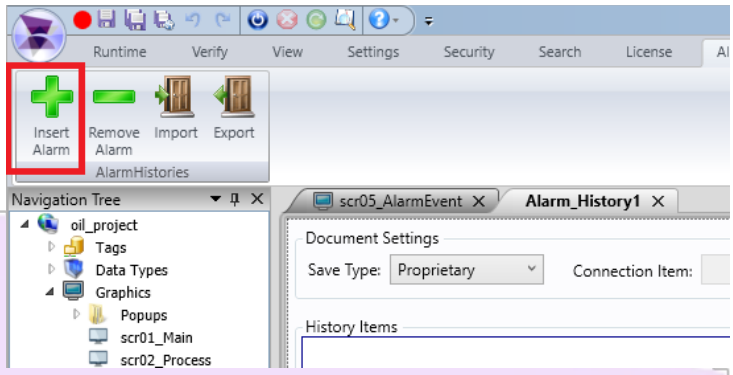


Well, before we add the objects to the “scr05_AlarmEvent” graphic, let's start by configuring the alarm history.

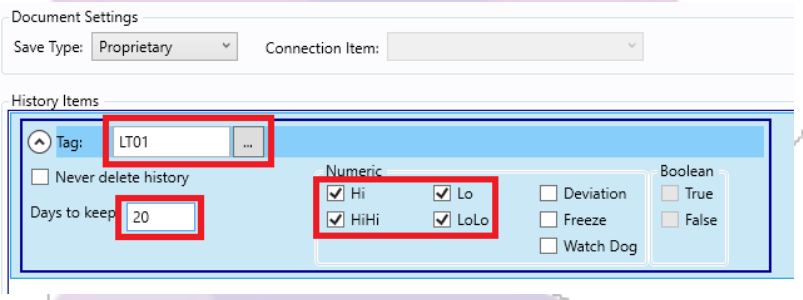
1. In the navigation tree, right-click on the “Alarm History” document and then click on “New Document”. Save the document as the suggested name “Alarm_History1”.



2. Now, click on the “Insert Alarm” button to insert the first tag in which the alarms will be storied.



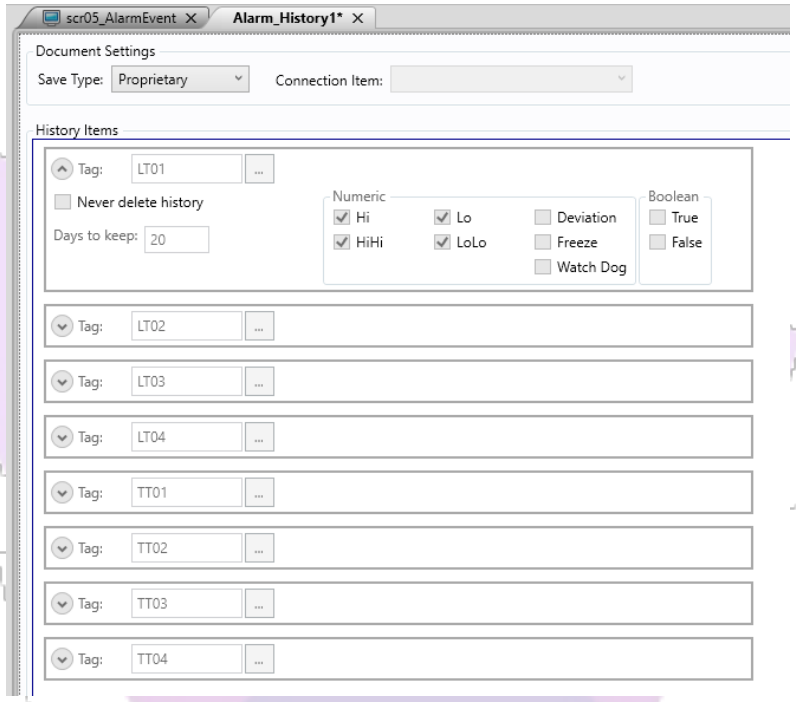
3. In the Tag field, enter tag "LT01". Then set this first history item as follows: Days to keep: 20 and enable the conditions HiHi, Hi, Lo, Lolo. We can say that the first item will record the analog alarms (HiHi, Hi, Lo and Lolo) of the LT01 tag and will store it for 20 days in the proprietary ADISRA SmartView database. From now on, when starting RunTime, LT01 tag alarms are being stored.



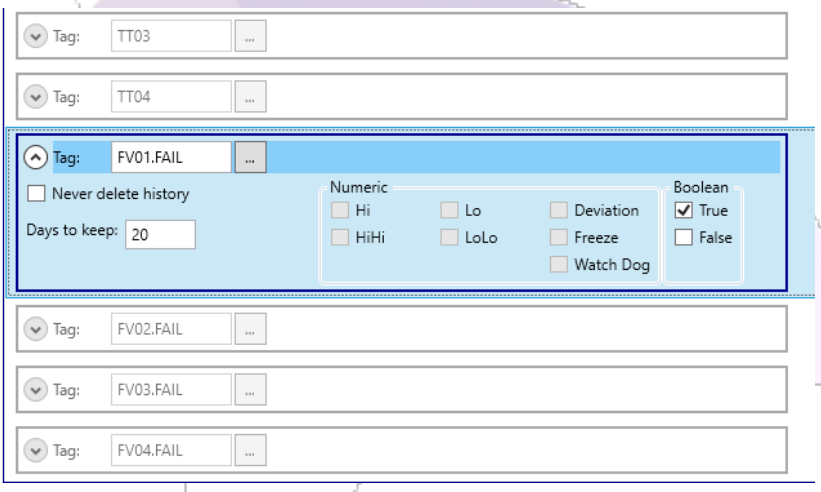
note[1]: *By default, the database in which you will store the alarm history will be the proprietary database.* It is possible to store alarm history in an external database, such as SQL Server, but these settings will not be presented in this training. For any historical data, we will use the proprietary ADISRA SmartView database.

note[2]: *To facilitate the configuration of the "Alarm History" document, it is possible to import and export the document but always respecting the structure of the ".CSV" file.*

4. Now, let's add all the tags that will have their alarms storied. So let's add analog tags first, like LT02, LT03, LT04, TT01, TT02, TT03 and TT04. Change the following properties: Days to keep: 20 and enable the conditions HiHi, Hi, Lo, Lolo. Below is the expected result,

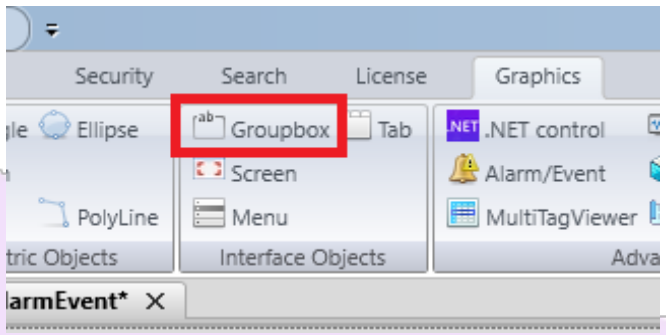


5. Now let's add the digital tags, like FV01.FAIL, FV02.FAIL, FV03.FAIL and FV04.FAIL. Change the following properties: Days to keep: 20 and enable the “True” condition. Below is the expected result.

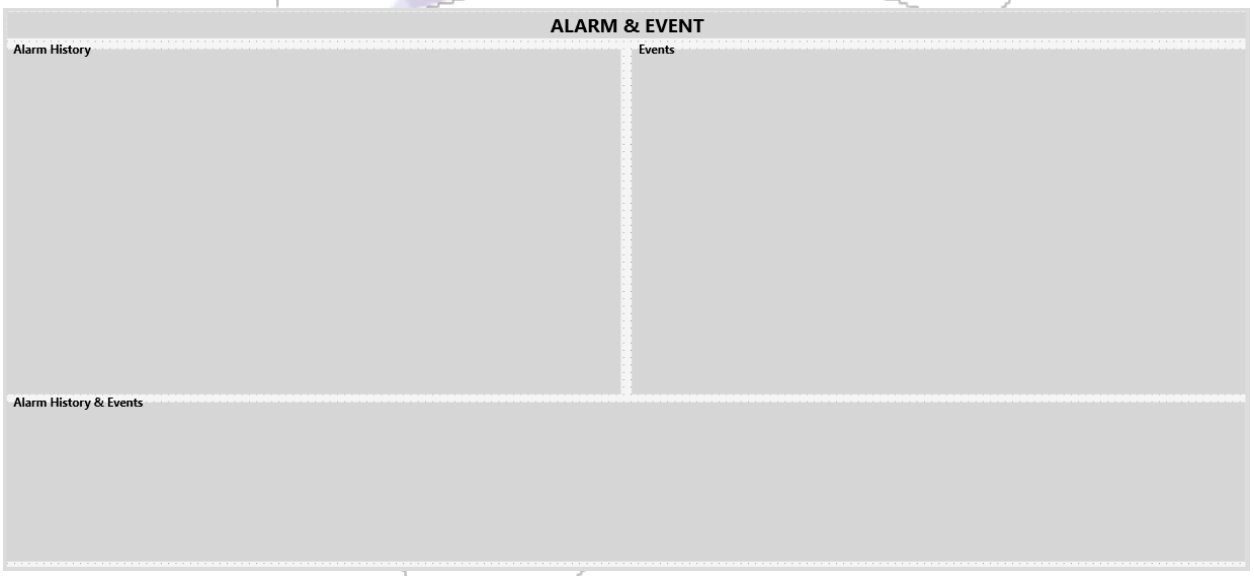


Ready! Now all alarms in our application will be logged while RunTime is running.

6. Now, save the document “Alarm History” and go back to the graphic “scr05_AlarmEvent” to insert the objects. In the top menu, add 03 (three) “GroupBox” objects.



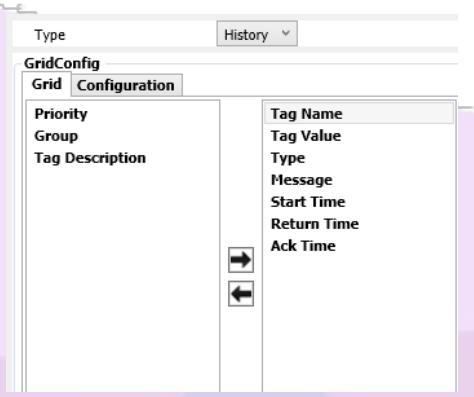
7. Change the following properties:
   GroupBox1: Location(40, -1), Size(838, 481),Text (Alarm History, Selgoe UI, Bold, 16), Brushes/Foreground (0, 0, 0), Brushes/Background (214, 214, 214), Brushes/Border (214, 214, 214), .
   GroupBox2: Location(40, 851), Size(838, 481),Text (Events, Selgoe UI, Bold, 16), Brushes/Foreground (0, 0, 0), Brushes/Background (214, 214, 214), Brushes/Border (214, 214, 214).
   GroupBox3: Location(520, -1), Size(1689, 228),Text (Alarm History & Events, Selgoe UI, Bold, 16), Brushes/Foreground (0, 0, 0), Brushes/Background (214, 214, 214), Brushes/Border (214, 214, 214).



8. Now, let's add the first object “Alarm/Event”, which will only show the alarm history. To facilitate the development, let's copy the object “Alarm/Event” added in the graphic “scr01_Main” and place it in the graphic “scr05_AlarmEvent”. Heads up! After placing the object in the graphic, it is possible that it will be placed outside the graphic. So use Zoom to look for the pasted object.

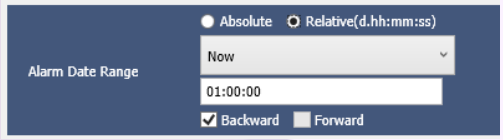
9. For the new object “Alarm/Event”, which will only show the alarm history, change the following properties: Location (64,10), Size (812, 442) and Type (History).

10. In the “Grid” tab of the “GridConfig” property, remove the priority, group and tag description fields. Below is the expected result.

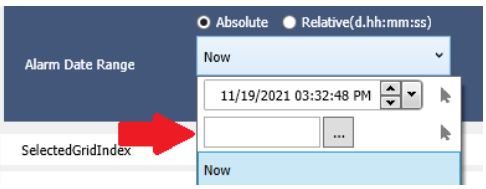


11. In the “Configuration” tab of the “GridConfig” property, configure the fields as follows:
    Tag Name: Header(Tag Name), Witdth(70) and Alignment(Left);
    Tag Value: Header(Tag Value), Witdth(60) and Alignment(Left);
    Type: Header(Type), Witdth(50) and Alignment(Left);
    Message: Header(Message), Witdth(180) and Alignment(Left);
    Start Time: Header(Start Time), Witdth (140) and Alignment(Left);
    Return Time: Header(Return Time), Witdth(140) and Alignment(Left);
    Ack Time: Header(Ack Time), Witdth(140) and Alignment(Left).

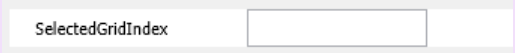
12. After configuring the “GridConfig” property, find the “Alarm Data Range” property and keep it configured as shown in the image below.



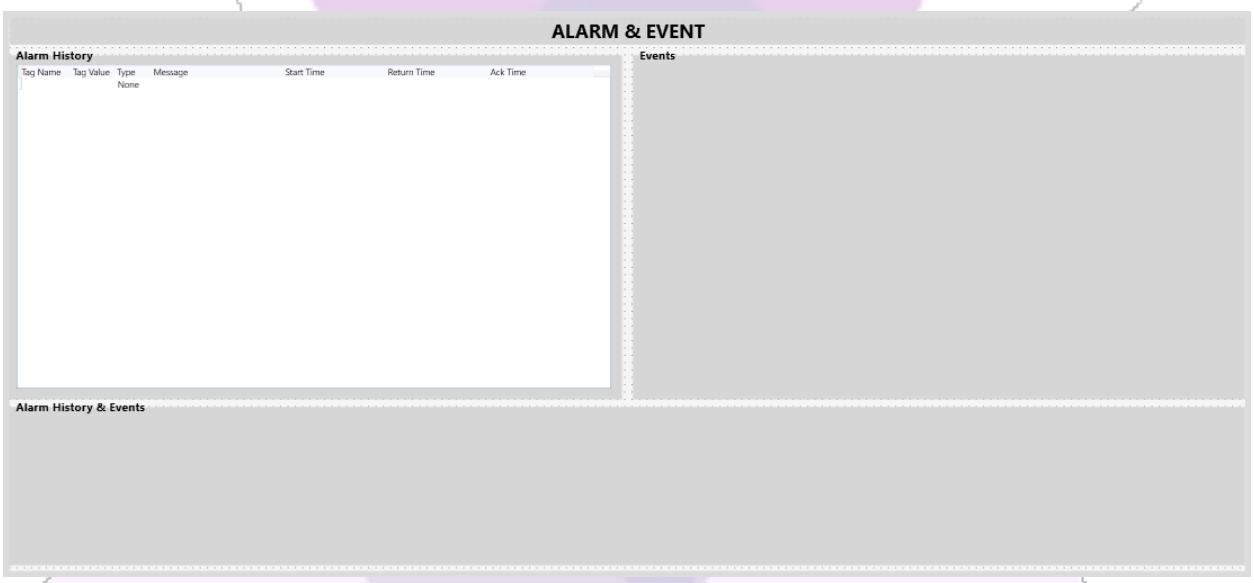
This property has the objective of defining the time interval that the alarms will be consulted and shown in the “Alarm/Event” object. Therefore, according to the image above, the alarms shown in the object will be consulted and shown from “Now” up to one hour (01:00:00) backwards. Any alarm that was triggered in this time interval will be shown on the object's display. If it is necessary to consult and show the alarm history in a certain time interval specified, for example by a user, select the “absolute” option and associate it to a tag which will be the start date and another tag informing the end date of the query. For our application, we will keep, according to the image above, the “Relative” option, Start date (01:00:00) and end date (Now).



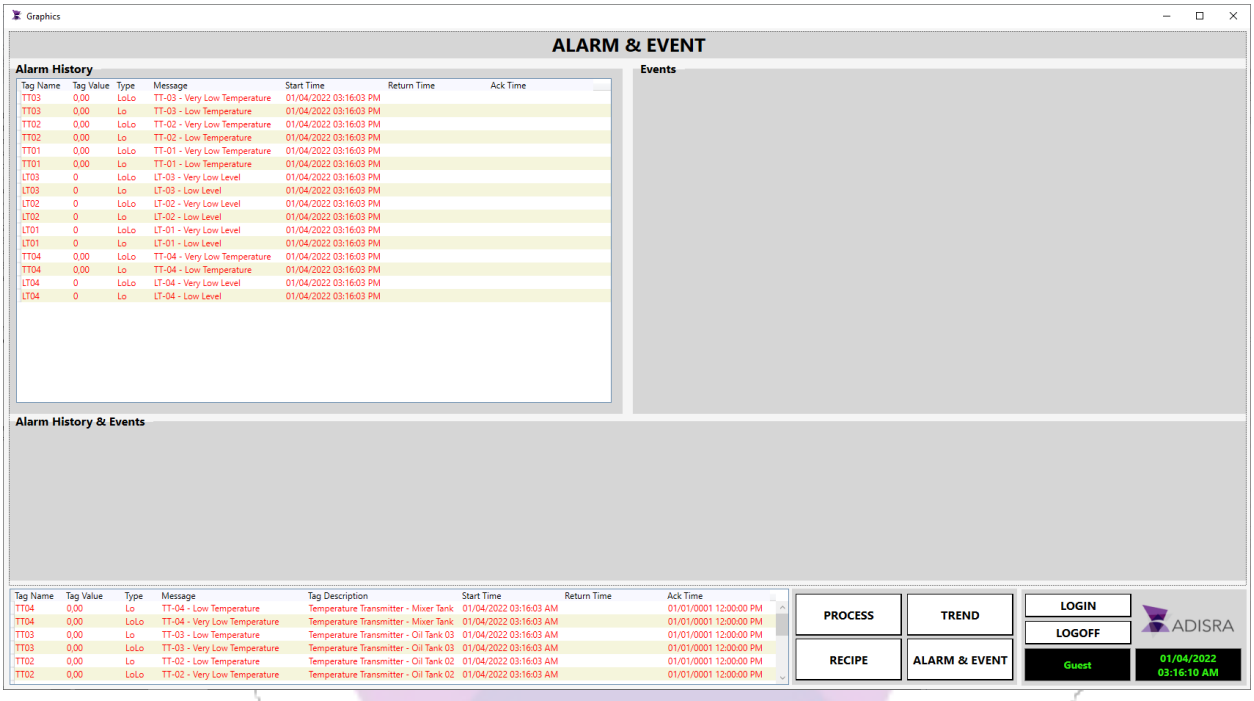
The Alarm/Event object also has a very interesting property called “SelectedGridIndex”. This property returns the index value of the grid selected by the user in Runtime mode. Therefore, a tag can be associated with this property to receive the selected index. For our application, we will not use this property.



13. Below is the expected result after adding the first Alarm/Event object.



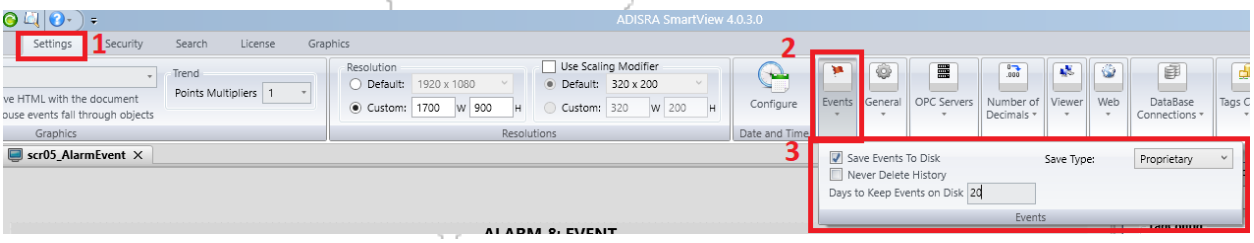
14. Once you have configured the Alarm History document and the Alarm/Event object, which will show the alarms triggered in the configured time interval, start the Runtime and check the triggered alarms. The alarms shown will be the alarms triggered during the training. In the image below it is possible to notice the alarms triggered during the tests.



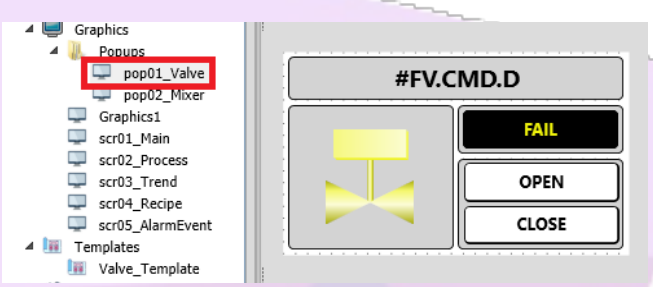
17.2.5. Configuring Events

Now, let's configure an Alarm/Event object in which it will be responsible for presenting the application's events. As mentioned before, to configure the application's events, it is necessary to follow the following steps: Enable a proprietary or external database to store the events, insert the "SVEvent" function in the objects in which the events will be generated and insert an object " Alarm/Event" or "Datagrid" to show stored events. So, let's configure!

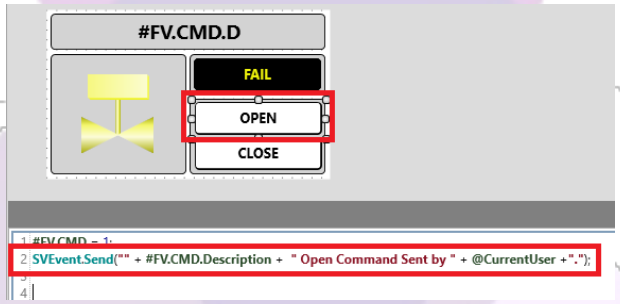
1. To enable the database in which you will be responsible for storing the application's events, click on the "Settings" menu and then on the "Events" option. Change the following properties: Enable the option "Save Events To Disk", Days to Keep on Disk (20) and Save Type (Proprietary). Ready! Now any event generated by the "SVEvent" function will be automatically stored in the ADISRA SmartView proprietary bank.



2. After enabling the database, we will insert the “SVEvent” function in each screen object on which a certain event will be generated. For our application, we will insert this function in the “OPEN” and “CLOSE” buttons of the valves and in the “RUN” and “STOP” buttons of the Mixer. Therefore, every time the user clicks on these buttons, an event must be generated, informing the type of event and the name of the current user logged into the system. So, navigate to the Navigation Tree and click on the Graphic “pop01_Valve”.



3. Select the “Open” button, open the “Mouse Down” event and add the following script:
SVEvent.send("" + #FV.CMD.Description +"Open Command Sent by " + @CurrentUser +".");



note[1]: *According to the ADISRA SmartView Help, this function is intended to trigger an event in the database. So, whatever is inside the parentheses will be the event message. The tag “@CurrentUser” has the current user logged into the application. By default, the current user is “Guest”.*

note[2]: *Note that it will not be necessary to add this script for each valve, as we are using a generic graphic.*

4. Add the same function for the “Close” button. Select the button, open the “Mouse Down” event and add the following script:
SVEvent.send("" + #FV.CMD.Description +"Close Command Sent by " + @CurrentUser +".");

```
#FV.CMD = 0;
SVEvent.Send("" + #FV.CMD.Description +  " Close Command Sent by " + @CurrentUser +".");
```

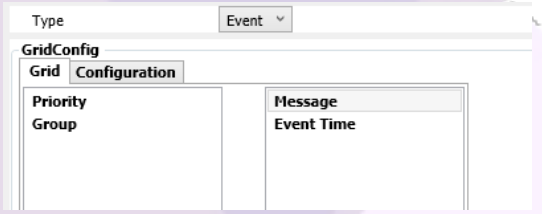
5. Now, save the graphic “pop01_Valve” and open the graphic “pop02_Mixer”. Repeat the same settings for the Mixer's “Run” and “Stop” buttons.

6. For the “Run” button, add the following script:
   SVEvent.send("Mixer Run Command Sent by " + @CurrentUser +".");

7. For the “Stop” button, add the following script:
   SVEvent.send("Mixer Stop Command Sent by " + @CurrentUser +".");

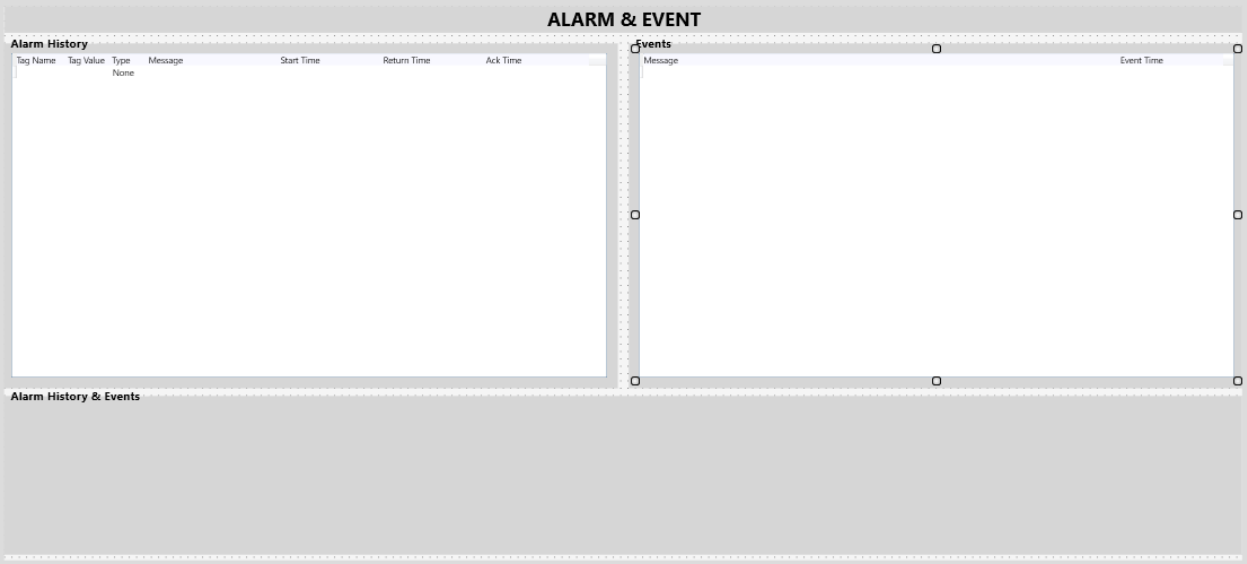
8. After configuring the event system, let's add the remaining 02 (two) Alarm/Event objects. This time, it will not be necessary to copy and paste an existing Alarm/Event object. Let's add it manually by the icon located in the top menu. After adding the second Alarm/Event, in which you will be responsible for querying and displaying the generated events, change the following properties: Location(64, 865), Size(812, 442) and Type(Event).

9. On the “Grid” tab of the “GridConfig” property, remove the priority and group fields. Below is the expected result.



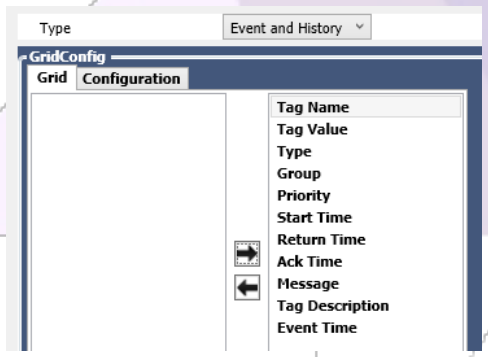
10. In the “Configuration” tab of the “GridConfig” property, configure the fields as follows:
    Message: Header(Message), Witdth(650) and Alignment(Left);
    EventTime: Header(Event Time), Witdth(140) and Alignment(Left).

Below is the expected result.



11. After configuring the Alarm/Event object, which will only show the application's events, we will add the last Alarm/Event object, which will show the combined alarm history and the application's events.
12. As this Alarm/History object will also show the alarm history, for ease of configuration, let's copy and paste the first Alarm/Event object added in which it only shows the alarm history. After that, change the following properties:
    Location(547, 11), Size(1665, 187) and Type(Event and History).

13. On the “Grid” tab of the “GridConfig” property, keep all the fields. Below is the expected result.

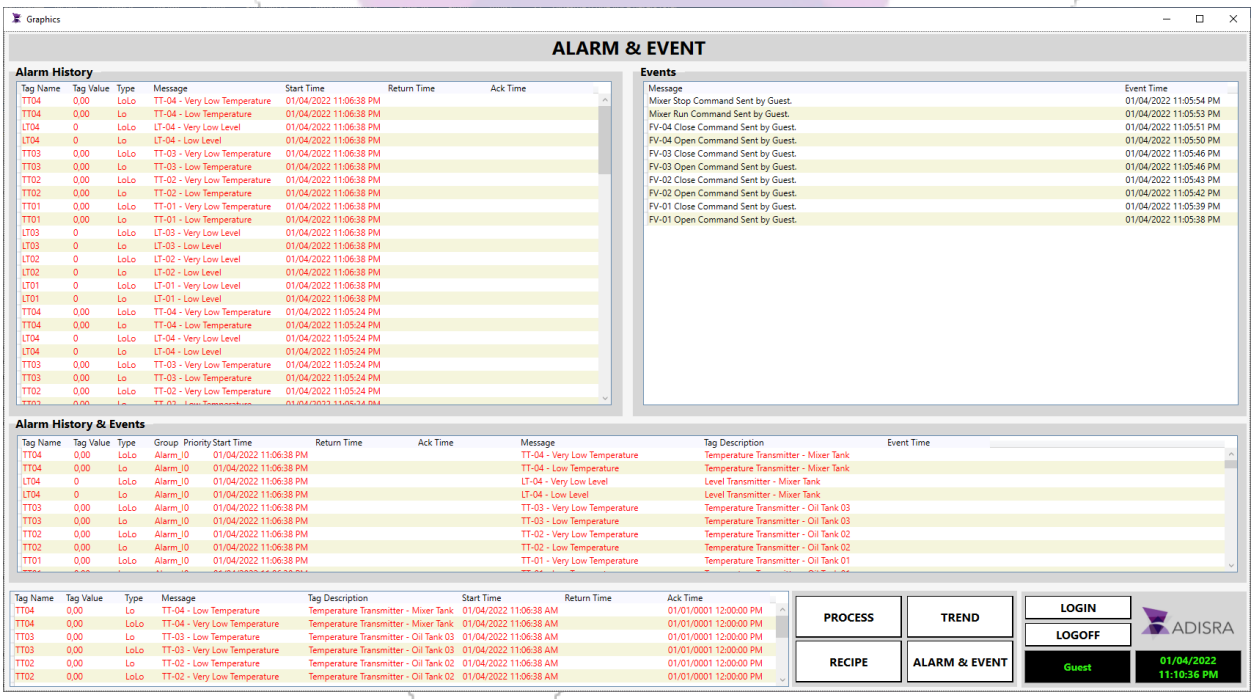


note: *Notice that the Alarm/Event object now has alarm and event fields.*

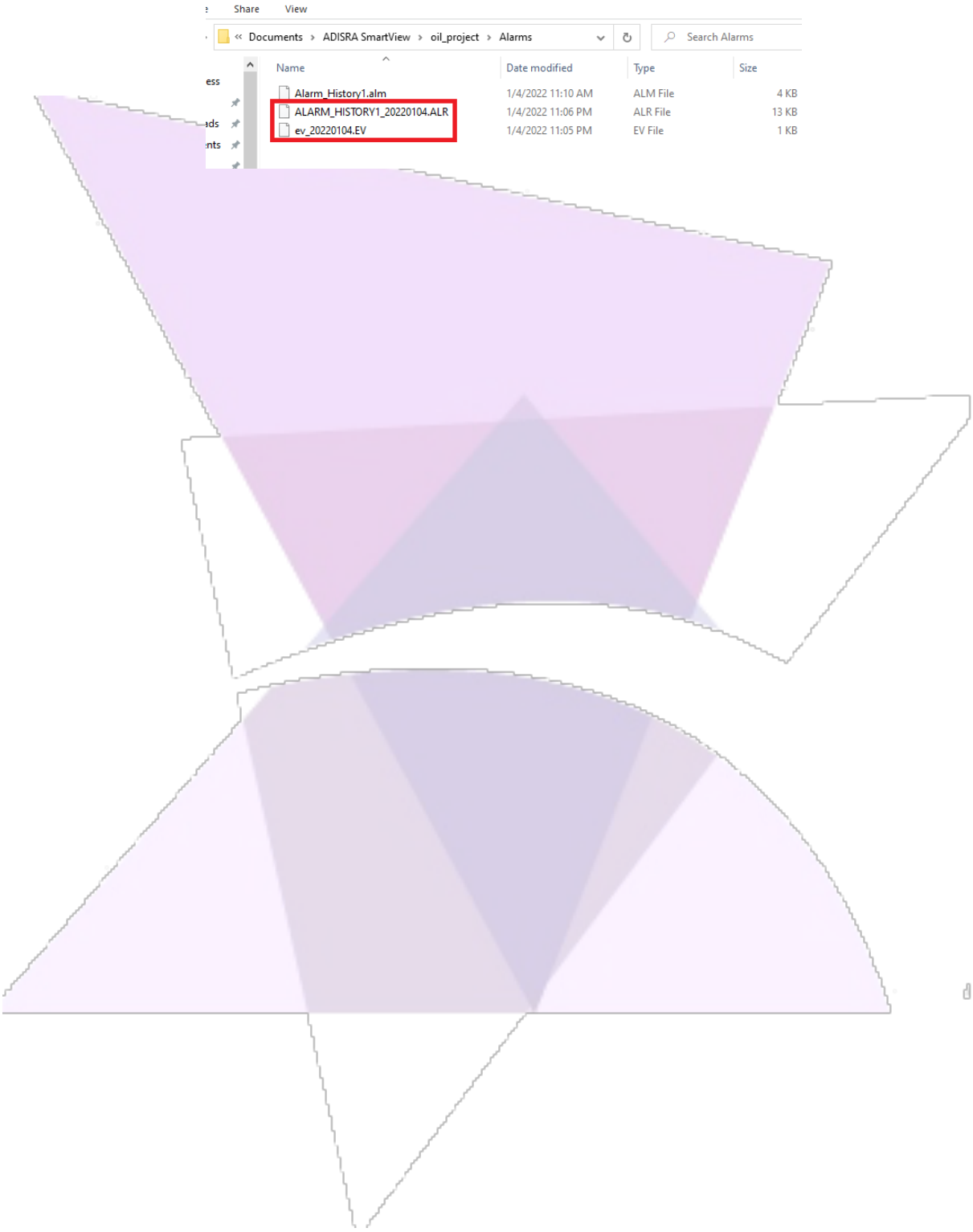
14. On the “Configuration” tab of the “GridConfig” property, configure the fields as follows:

Tag Name: Header(Tag Name), Witdth(70) and Alignment(Left);
Tag Value: Header(Tag Value), Witdth(60) and Alignment(Left);
Type: Header(Type), Witdth(50) and Alignment(Left);
Group: Header(Group), Witdth(50) and Alignment(Left);
Priority: Header(Priority), Witdth(50) and Alignment(Left);
Start Time: Header(Start Time), Witdth(140) and Alignment(Left);
Return Time: Header(Return Time), Witdth(140) and Alignment(Left);
Ack Time: Header(Ack Time), Witdth(140) and Alignment(Left).
Message: Header(Message), Witdth(250) and Alignment(Left);
Tag Description: Header(Tag Description), Witdth(250) and Alignment(Left);
Event Time: Header(Event Time), Witdth(140) and Alignment(Left);

15. Ready! We have finished configuring the graphic “scr05_AlarmEvent”. Save the application and start the Runtime. After starting, open each valve and Mixer popup and click on the command buttons in which we configure the event functions. Then open the “Alarm & Event” graphic and check if the events have been generated. Below is the expected result.



note: *As we use the proprietary database to store the alarm history and application events, the alarm history is stored in a file with the extension “.ALR” and the events in a file with the extension “.ER”, located in the folder “Alarms” of the application. You can open both files in “Notepad”, “Microsoft Excel” or “Notepad++”.*


Share
View
Documents
ADISRA SmartView
oil_project
Alarms
Search Alarms
Name
Date modified
Type
Size
Alarm_History1.alm
1/4/2022 11:10 AM
ALM File
4 KB
ALARM_HISTORY1_20220104.ALR
1/4/2022 11:06 PM
ALR File
13 KB
ev_20220104.EV
1/4/2022 11:05 PM
EV File
1 KB

18. Developing Graphic Trend

Graphic Trend will be responsible for presenting the behavior of variables as a function of time through the “Trend” object, with a button to generate a report, a button to visualize the chart objects, and a “ComboBox” object for selecting the pen.

A “Trend” object can show the user the behavior of one or more variables as a function of the current time or historical time, that is, the time interval configured in the object. However, for the history functionality to become available, we must configure the “Tag History” document. This document is responsible for configuring the tag history. If the “Tag History” document is not configured, the “Trend” object will only show the behavior of the variables as a function of the current time.

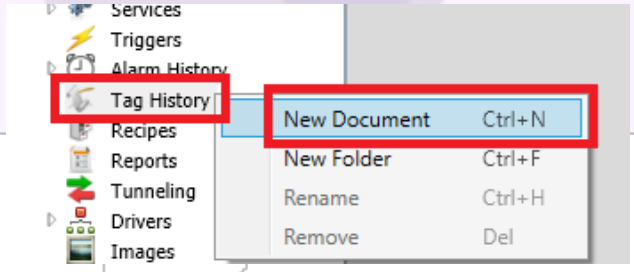
Thus, to configure the tag history, it is necessary to follow these steps:

1. Insert the document “Tag History”;
2. Configure the “Tag History” document;
3. In the “Tag History” document, insert the tags in which they will be storied in the proprietary database;
4. Insert the “Trend” object in the Trend graphic;
5. Configure the “Trend” object.

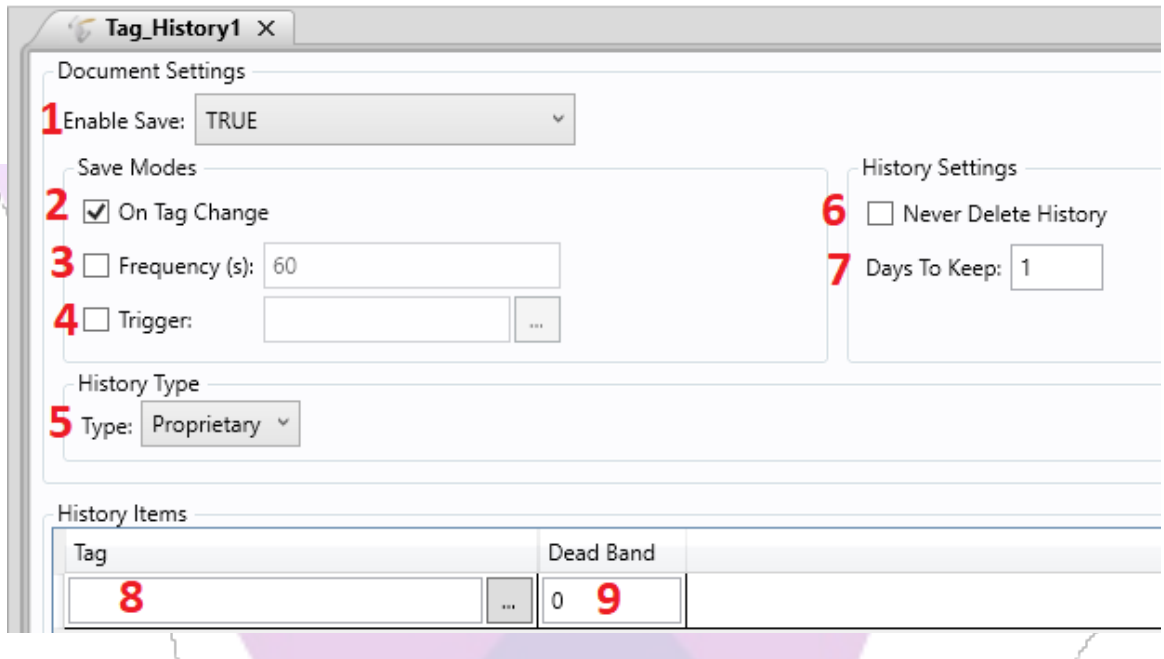
18.1. Configuring the Tag History Document

As per the step-by-step above, let's start by inserting and configuring the “Tag History” document.

1. In the navigation tree, right-click on the “Tag History” document and then click on “New Document”. Save the document as the suggested name “Tag_History1”.



Before continuing the step-by-step, let's get to know each configuration field of the “Tag History” document.



1- Enables or disables the Document “Tag History”;
2- The tag's value will be recorded when its value is changed;
3- The tag value is recorded every X time defined by the developer;
4- The tag value is recorded if the tag value associated with the Trigger field is “True”;
5- Configures in which database the tags will be stored;
6- If enabled, recorded data will never be deleted;
7- The number of days the recorded data will be stored;
8- Field to associate the tag in which it will be recorded;
9- Field to identify the deadband value, that is, if the tag value exceeds the deadband value, the tag will be recorded.

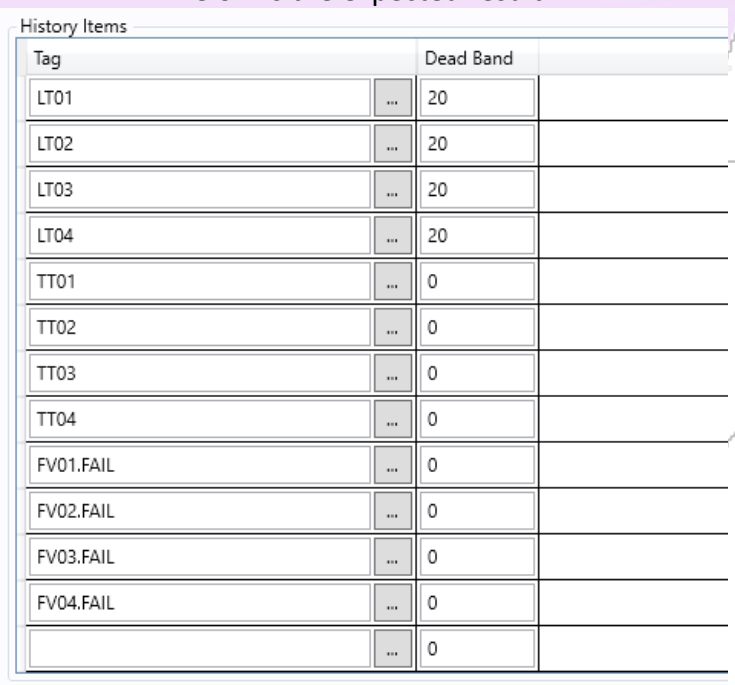
Now that we already know each field of the Document “Tag History”, let's configure and associate the tags in which they will be recorded in the proprietary database.

2. In the “Tag History” document, change the following properties:
   Enable Save: Enable;
   Save Modes: On Tag Change
   History Type: Proprietary
   Never Delete History: False
   Days to Keep: 20

3. Now, let's add the tags to be recorded with the following deadband:
   LT01 – Deadband(20)
   LT02 – DeadBand(20)
   LT03 – Deadband(20)
   LT04 – DeadBand(20)

TT01 – DeadBand(0)
TT02 – DeadBand(0)
TT03 – DeadBand(0)
TT04 – DeadBand(0)
FV01.FAIL – DeadBand(0)
FV02.FAIL – DeadBand(0)
FV03.FAIL – DeadBand(0)
FV04. FAIL - DeadBand(0)

Below is the expected result.

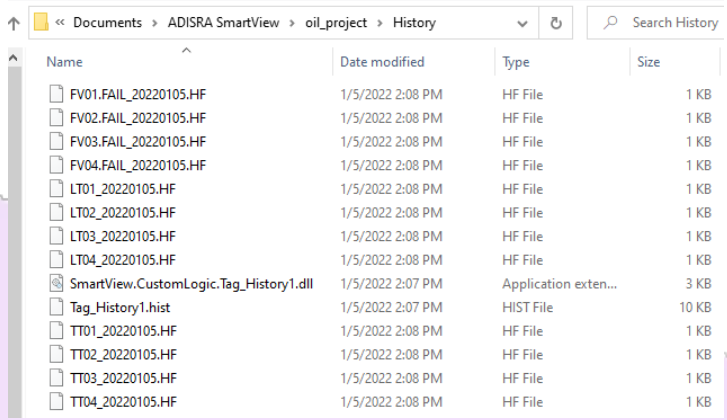

History Items

Tag	Dead Band
LT01	20
LT02	20
LT03	20
LT04	20
TT01	0
TT02	0
TT03	0
TT04	0
FV01.FAIL	0
FV02.FAIL	0
FV03.FAIL	0
FV04.FAIL	0
	0

note[1]: *By default, the database in which to store the tag history will be the owner database.* It is possible to store tag history in an external database, such as SQL Server, but these settings will not be presented in this training. For any historical data, we will use the proprietary ADISRA SmartView database.

note[2]: *To facilitate the configuration of the "Tag History" document, it is possible to import and export the document but always respecting the structure of the ".CSV" file.*

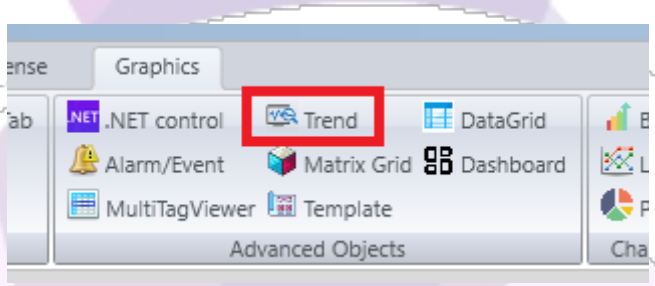
note[3]: *As we use the proprietary database to store the history of the application's tags, the history of each tag will be stored in a ".HF" file, located in the "History" folder of the application. You can open the files in "Notepad", "Microsoft Excel" or "Notepad++".*



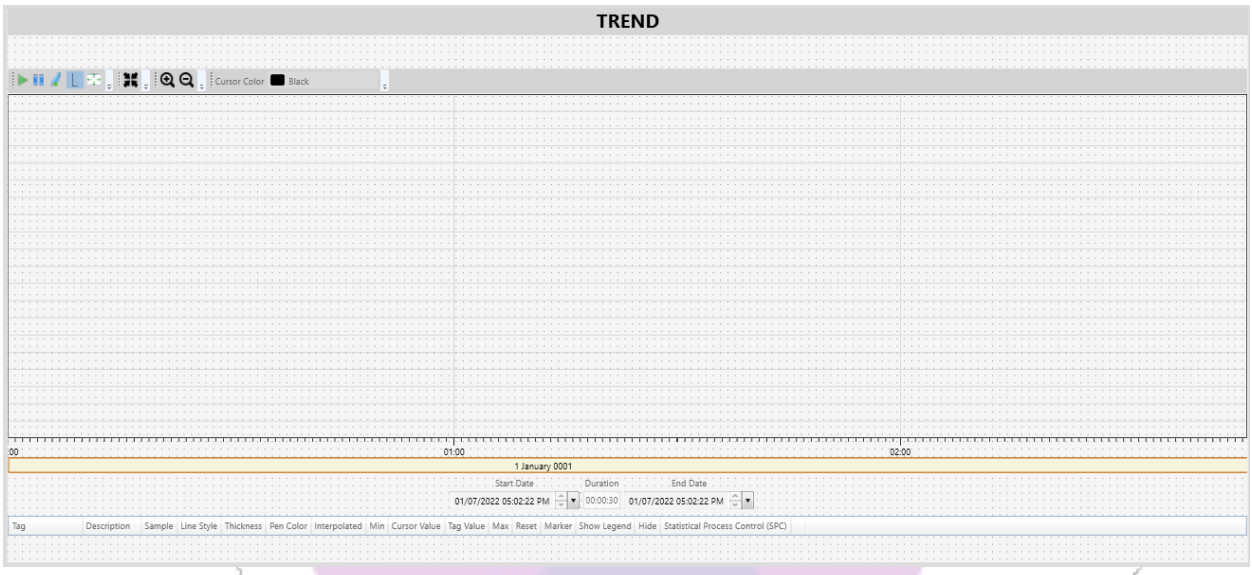
18.2. Configuring the Trend Object

1. After creating and configuring the “Tag History” document, we will develop the content of the “scr03_Trend” graphic. Save the application and then open the graphic “scr03_trend”.

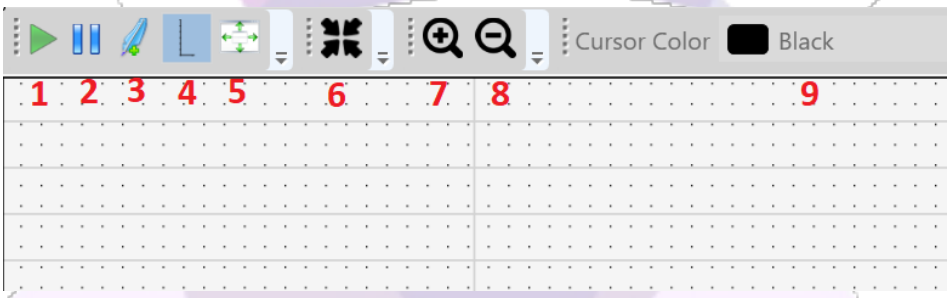
2. On the top menu, add the “Trend” object to the “scr03_Trend” graphic;



3. Select the “Trend” object and change the following properties: Location(83, 1), Size(1686, 665). Below is the expected result.

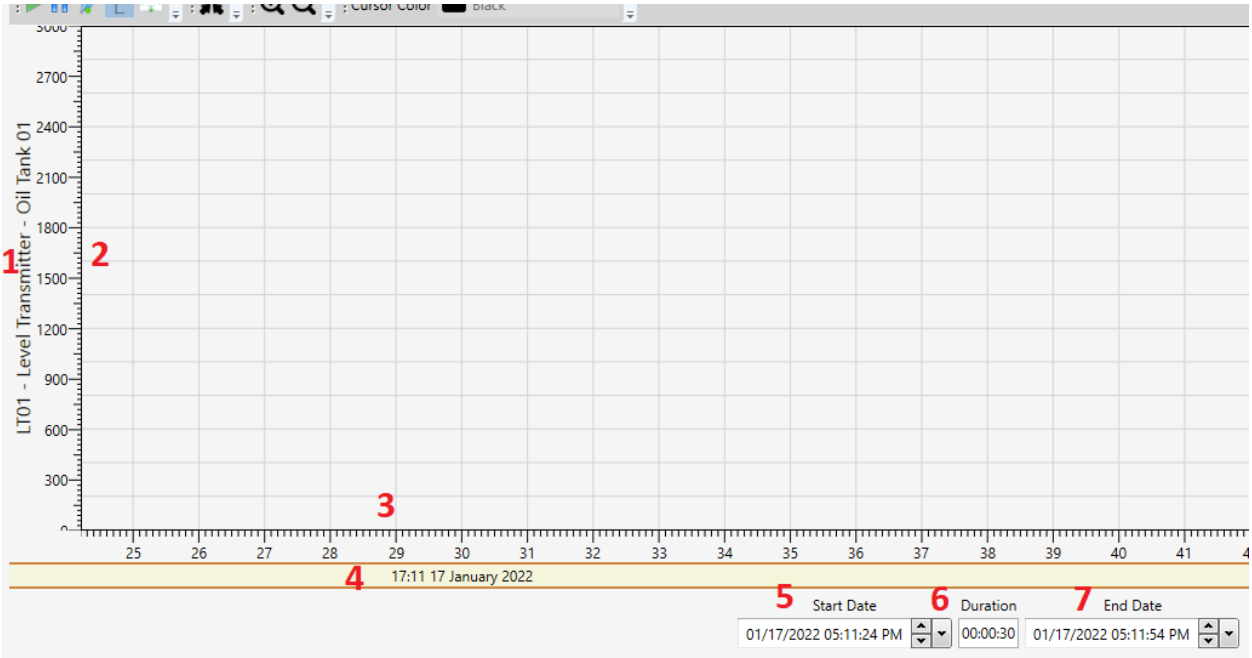


Before proceeding with the configurations, let's get to know the Trend object's interface. In this way, save the application and start the RunTime mode and then open the “Trend” graph.



1- Starts the Trend object by drawing the behavior of the selected “Pen”.
2- For the Trend Object, that is, for the drawing of the selected “Pen”.
3- Adds a “Pen” in the Trend object.
4- Makes the scale of the X and Y axes.
5- Adjusts the scale of the X and Y axes according to the minimum and maximum values of the “Pen” on which it is being plotted;
6- Makes the DataGrid Pen visible or not, located at the bottom of the graphic.
7- Zoom in on the plot area;
8- Zoom out in the plot area;
10- Change the color of all “Pens” inserted in the Trend object

Now, to better understand the other features of the Trend object, insert a “Pen” through the button shown above and associate the tag @LT01.



Note that after adding a “Pen” to the tag @LT01, we had the following changes:

1- The description of the tag @LT01 was inserted in the left corner and vertically;

2- The Y Axis scale was inserted according to the property minimum and maximum of the tag @LT01;

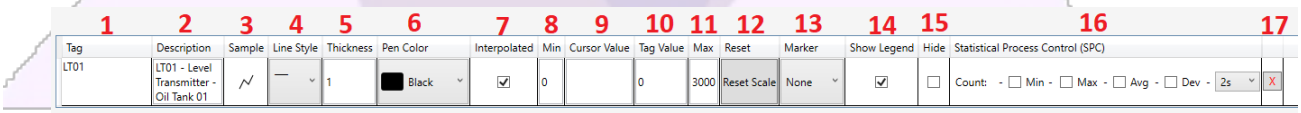
3- The Y Axis scale is showing the date and time;

4- Below the Y Axis time scale, the TimeStamp of the tag associated with “Pen” is displayed;

5- Allows viewing and change the plot's start date;

6- Allows you to view and change the time difference between the start date and the end date of the "Pen" plot. In this case, 30 seconds of plot will be shown in the Trend object;

7- Allows viewing and changing the final date of the plot.



1- Tag name associated with a black “Pen”;

2- Tag Description;

3- Defines the format in which the “Pen” will be plotted in the plotting area;

4- Defines the style of the plotted line;

5- Defines the thickness of the plotted line;

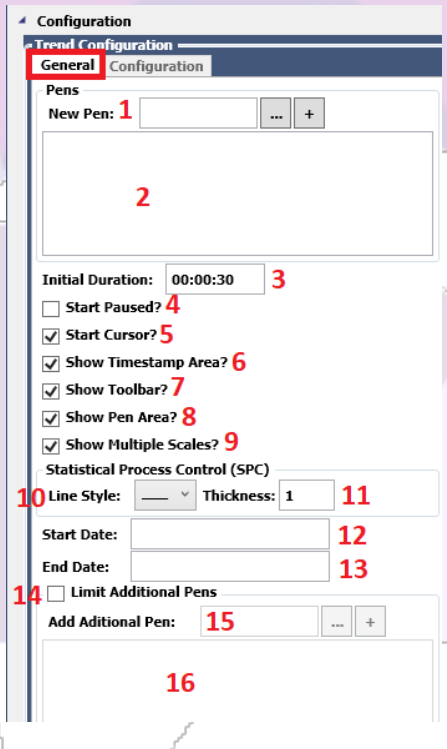
6- Defines the color of the “Pen”;

7- Interpolate the values of the “Pen”, making straight lines between the points instead of steps.

8- Minimum value of the Y Axis;

9- Current value of the Cursor. (We will see what a course is later on in this training);
11- Current value of the Tag associated with “Pen”;
11- Maximum Y-Axis value;
12- Resets the Y-Axis scale to the original values of the Trend object;
13- Allows you to choose between marking Tag values with circles or triangles, making it easier to view, or leaving it unchecked;
14- When enabled, it shows the tag description in the upper right corner of the plotting area;
15- Hides the associated “Pen” the tag @LT01;
16- When enabled, shows the mean, higher or lower value and average deviation of the tag value;
17- Removes the "Pen" from the Trend object.

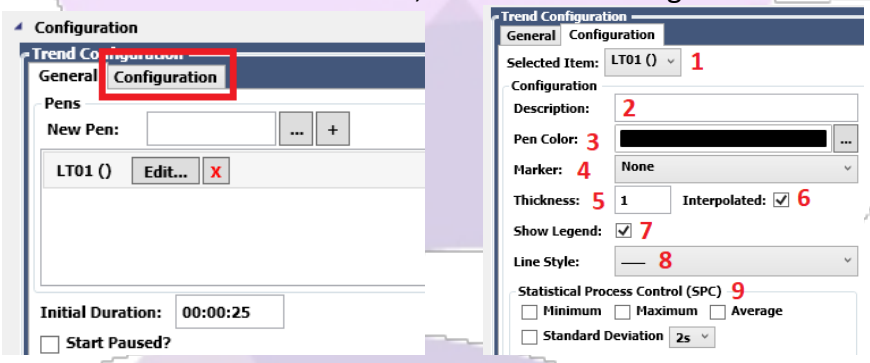
Now that we know the graphical interface of the trend object in Runtime mode, let's get to know its properties. Go back to the Graphic “Trend” and select the Trend object.



1- Insert “Pens” to be plotted as soon as the RunTime is started;
2- List of inserted “Pens”;
3- Allows you to view and change the time difference between the start date and the end date of the “Pen” plotting. In this case, 30 seconds of plotting will be shown in the Trend object;
4- When enabled, the Trend Object starts paused;
5- When enabled, the Cursor starts with the Trend Object;

6- When enabled, the TimeStamp area will be visible;
7- When enabled, the toolbar will be visible;
8- When enabled, the “Pens” Datagrid will be visible;
9- When enabled, it allows multiple scales;
10- Defines the style of “Pens”;
11- Defines the thickness das “Pens”;
12- Allows you to change the plot's start date;
13- Allows you to change the plot's end date;
14- When enabled, it will enable the “Limit Additional Pens” field.
15- Allows you to limit the "Pens" that can be added to be plotted by RunTime mode;
16- List of inserted "Pens".

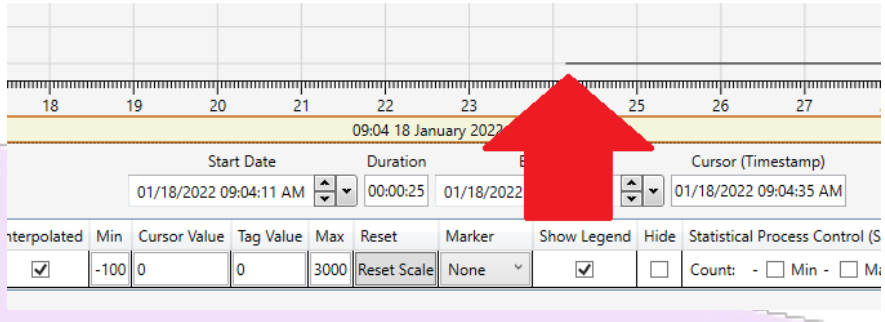
Now, to get to know the “Configuration” tab, let's insert a new “Pen” associated with the tag @LT01 in item 01 mentioned above. After that, click on the “Configuration” tab.



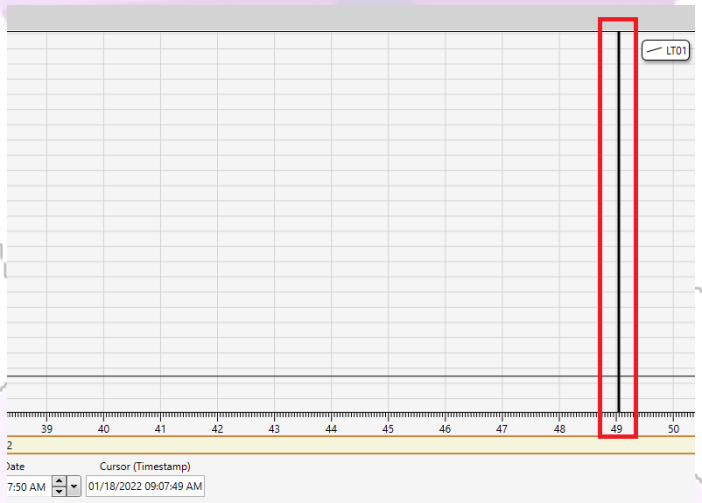
1- Allows you to select the “Pen” to be configured individually;
2- Allows you to insert a caption for the “Pen”;
3- Allows you to change the color of the “Pen”;
4- Allows you to choose between marking the Tag values with circles or triangles, facilitating its visualization, or leave it unchecked;
5- Defines the thickness of the “Pen” line;
6- When enabled, it allows to Interpolate the “Pen” values, making straight lines between the points instead of steps.
7- When enabled, it shows the tag description in the upper right corner of the plot area;
8-Defines the style of the “Pen” line;
9- When enabled, it shows the average, higher or lower value and average deviation of the tag value;

note: *Do not forget to remove the “Pen” associated with the tag @LT01 from the “Pens” list, as we will add the “Pens” to the Trend object in RunTime mode.*

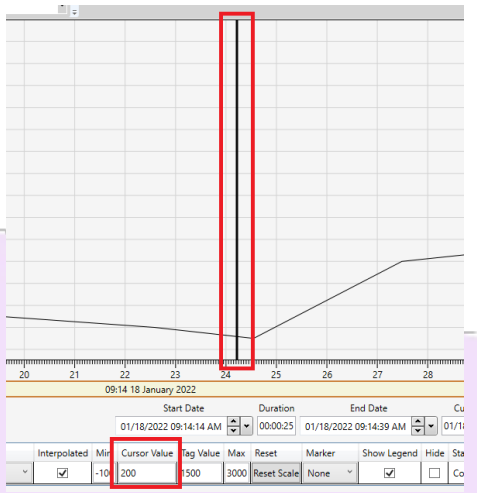
Now, start RunTime to get to know the Cursor features. With Graphic Trend open, notice that after adding the tag @LT01, a line started to be plotted in the graphics area.



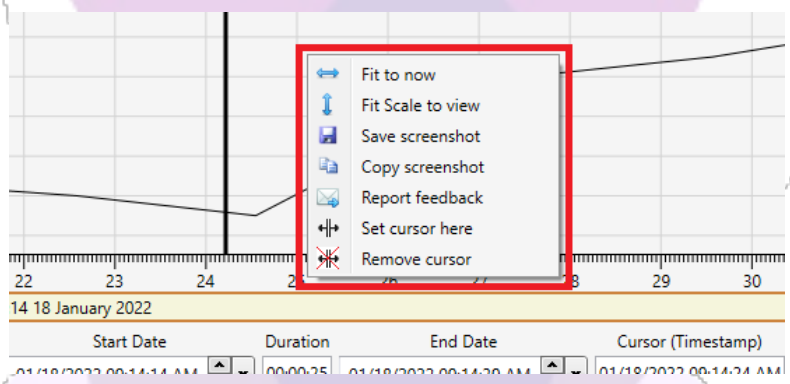
The “Cursor” we talked about earlier is this line highlighted in red in the image below. This Cursor can be used to show the current value of a “Pen” at the point where the Cursor is. It is possible to move the position of the selected Cursors with the mouse and by moving from left to right.



After moving, note that the “Cursor Value” field shows the value 200 and the “Tag Value” field shows the current value of the tag, in which it has 1500. This tool allows us to discover the value of the “Pen” at a given moment.



During RunTime Mode, you can open some Cursor settings by right-clicking anywhere in the plot area.

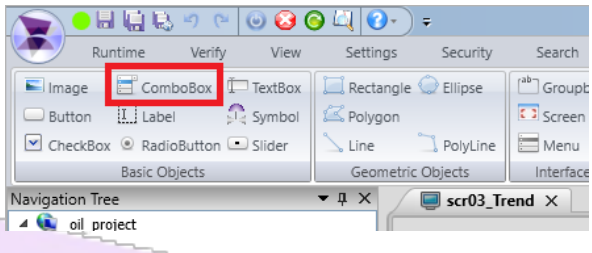


Fit to now – Adjusts the X-Axis to the current time;
Fit Scale to view – Adjusts the Y-Axis to the Pen's minimum and maximum scale;
Save ScreenShot – Allows you to save a ScreenShot of the plot area;
Copy ScreenShot – Makes a ScreenShot and copy to Windows clipboard;
Report feedback – Allows you to send a Suggestion or help message to ADISRA;
Set Cursor here – Sets a new Cursor position;
Remove Cursor – Removes the Cursor from the plot area.

18.3. Configuring the ComboBox Object

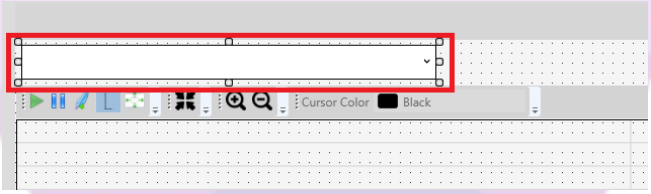
Now that we have the Trend object configured, for didactic purposes, we are going to insert a “ComboBox” object and two buttons that will be responsible for inserting and removing “Pens” in the trend object.

1. Go back to the development environment and insert a ComboBox object in the graphic “scr03_Trend”.

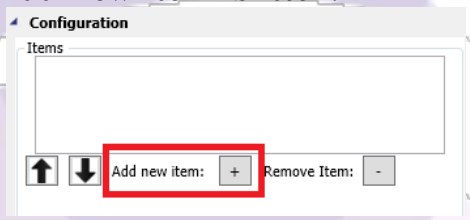


2. Go back to the development environment and insert a ComboBox object in the graphic "scr03_Trend". Then change the following properties: Location(42, 6), Size(410, 35).

Below is the expected result.

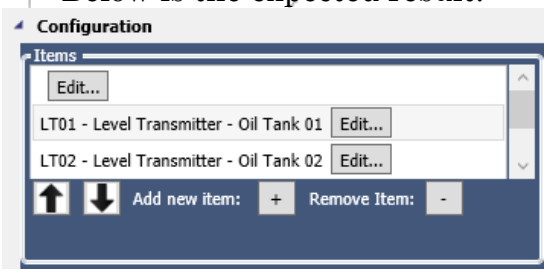


3. Select the ComboBox and in the properties list, insert 4 items in the items list by clicking on the "Add New Item" button.



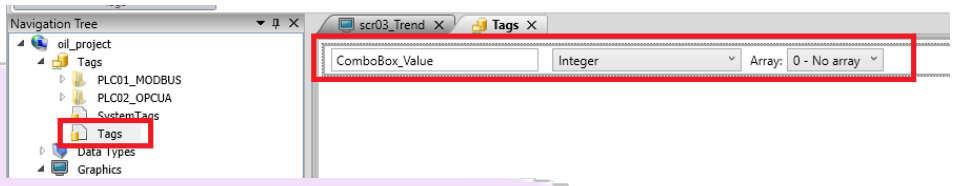
Item 01: LT01 - Level Transmitter - Oil Tank 01
Item 02: LT01 - Level Transmitter - Oil Tank 02
Item 03: LT01 - Level Transmitter - Oil Tank 03
Item 04: LT01 - Level Transmitter - Oil Tank 04

Below is the expected result.

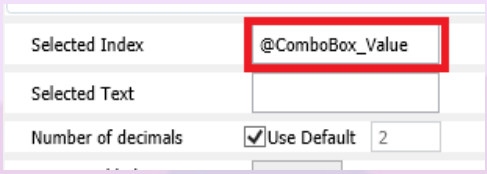


4. Now, let's create a tag of type integer to receive the values of the item selected by the user in RunTime mode. In the navigation tree, click on the

document “Tags” and insert a tag called “ComboBox_Value” and datatype “Integer”.

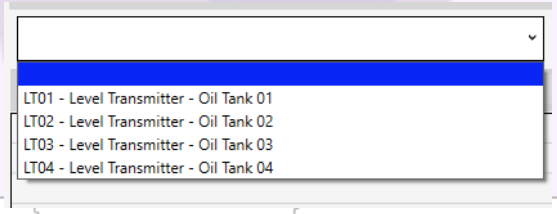


5. Then, save the application, open the graphic “scr03_Trend” again and open the property list of the ComboBox object. In the “Selected Index” property, insert the “ComboBox_Value” tag.



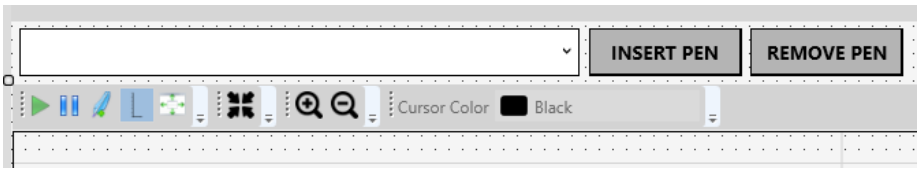
From now on, when the user selects, for example, the first item called “LT01 - Level Transmitter - Oil Tank 01”, the “ComboBox_Value” tag will receive the value of 0 (zero). If the user selects the second item, the tag will receive the value of 1, if the user selects the third item, the tag will receive the value of 2 and so on. With this information, we will know which item the user selected and use it to make a script. Note that we can also receive the text of the item selected by the “Selected Text” property.

Save the application and start RunTime to view the created items. So far, selecting items will have no action as we haven't added any scripts.



6. Now, go back to the development environment and insert two buttons and change the following properties:

   Button 01: Location (42, 423), Size (112, 35), Textbox (INSERT PEN, Selgoe UI, Bold, 14) and Brushes/ Background (Color 180, 180, 180).
   Button 02: Location (42, 541), Size (112, 35), Textbox (REMOVE PEN, Selgoe UI, Bold, 14) and Brushes/Background (Color 180, 180, 180).

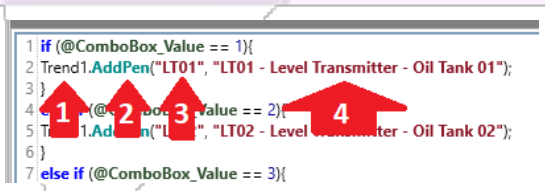


7. Select the "INSERT PEN" button, open the script mode and in the "Mouse Down" event, insert the script below:

```
if (@ComboBox_Value==1) {
Trend1.AddPen("LT01", "LT01 - Level Transmitter - Oil Tank 01");
}
else if (@ComboBox_Value==two) {
Trend1.AddPen("LT02", "LT02 - Level Transmitter - Oil Tank 02");
}
else if (@ComboBox_Value==3) {
Trend1.AddPen("LT03", "LT03 - Level Transmitter - Oil Tank 03");
}
else if (@ComboBox_Value==4) {
Trend1.AddPen("LT04", "LT04 - Level Transmitter - Oil Tank 04");
}
else
{
stringmessage ="Please, select the pen.";
MessageBox.Show(message);
}
```

Basically, the script will add a "Pen" depending on the value of the "ComboBox_Value" tag. If the tag has a value of 1, the first condition will be executed, if the tag has a value of 2, the second condition will be executed and so on. If the user does not select any item in the Combobox, a message box will be displayed asking to select a "Pen".

Let's understand the above script line a little bit.



1- Trend object named "Trend1";
2- Property of the "Trend1" object to which a "Pen" will be added;
3- Tag name will be added to new "Pen";
4- Description of the new "Pen".

Now let's configure the “Remove Pen” button. Select the “REMOVE PEN” button, open the script mode and in the “Mouse Down” event, enter the script below:

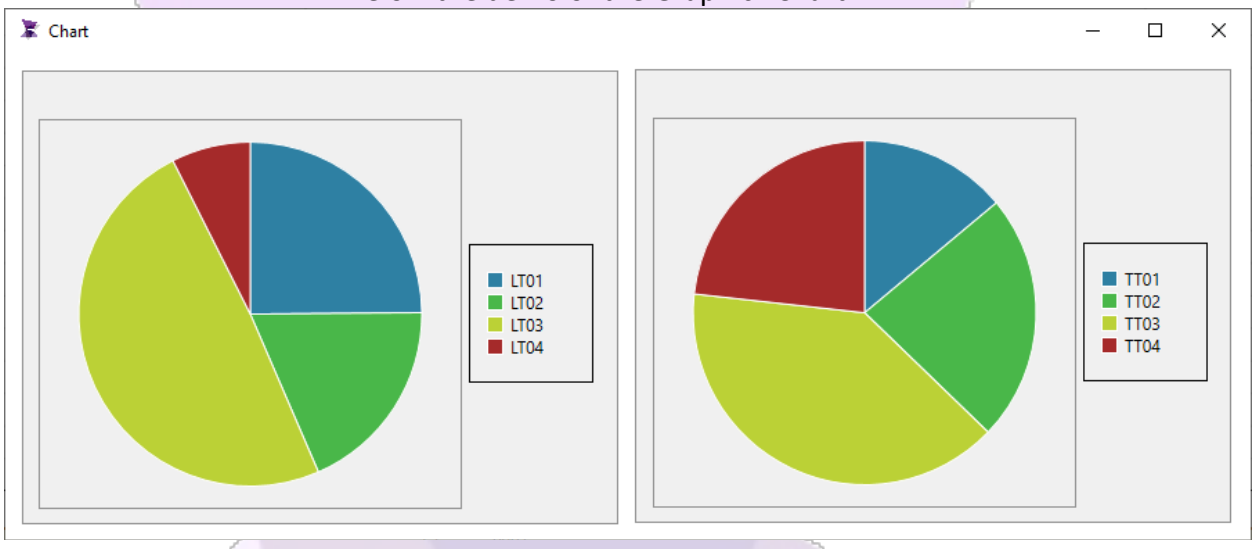
```
if (@ComboBox_Value==1) {
Trend1.removePen("LT01");
}
else if (@ComboBox_Value==two) {
Trend1.removePen("LT02");
}
else if (@ComboBox_Value==3) {
Trend1.removePen("LT03");
}
else if (@ComboBox_Value==4) {
Trend1.removePen("LT04");
}
else
{
stringmessage ="Please, select the pen.";
MessageBox.Show(message);
}
```

Basically, the script will remove a “Pen” depending on the value of the “ComboBox_Value” tag. If the tag has a value of 1, the first condition will be executed, if the tag has a value of 2, the second condition will be executed and so on. If the user does not select any item in the Combobox, a message box will be displayed asking to select a “Pen”.

19. Developing a Graphic Chart

The Graphic Chart will be responsible for presenting a graph divided into slices, in which each slice displays the proportion related to the sum of all slices. Therefore, we will add two "PIE" Charts, to show the proportions of the Level Transmitters (LT-XX) and the Temperature Transmitters (TT-XX).

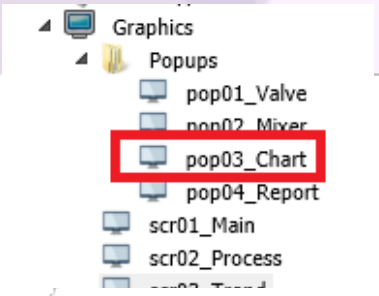
Below the demo of the Graphic "Chart"



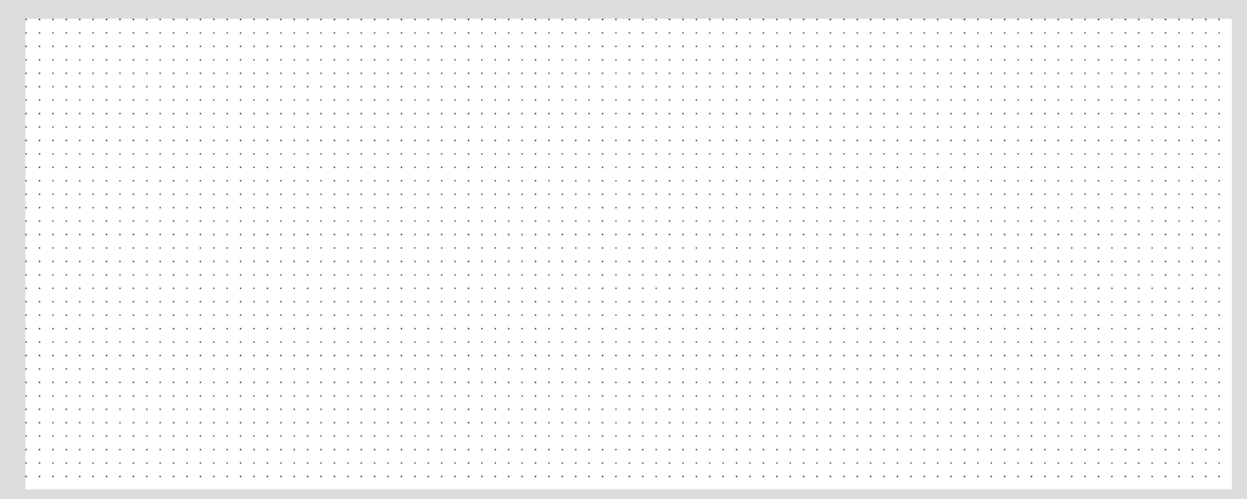
.

19.1. Creating the Graphic Chart

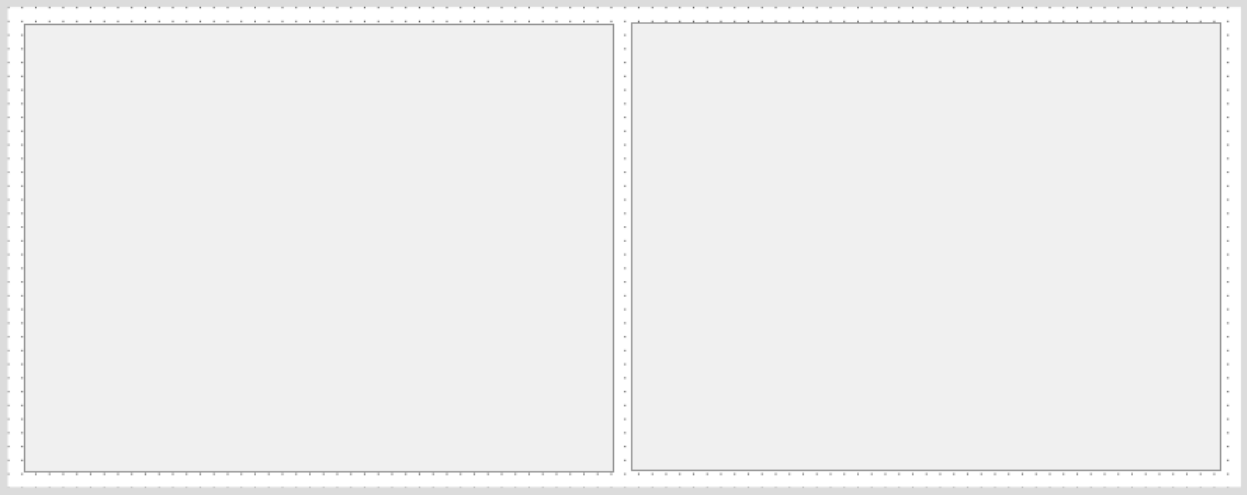
1. In the navigation tree, create a new Graphic called "pop03_Chart" inside the "Popups" folder and then save the application.



2. In the Graphic “pop03_Chart”, change the following properties: Size (900, 350), Brushes/Background Color (255, 255, 255).



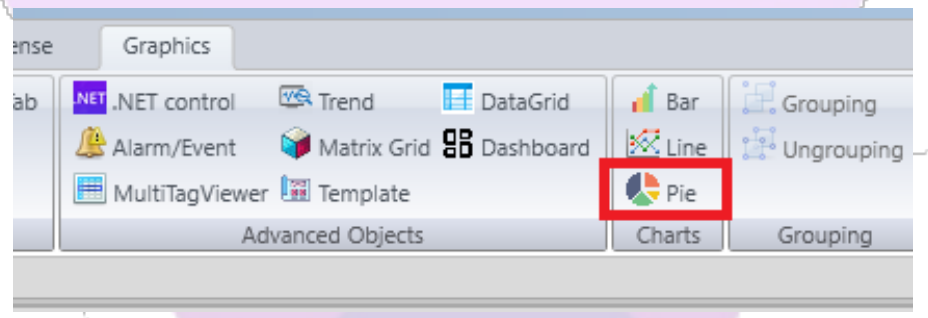
3. Insert two “Rectangle” objects and change the following properties:
   Rectangle 01:
   Location(12, 12), Size(431, 327), Brushes/Fill (240, 240, 240) and Brushes/Border (145, 145, 145) .
   Rectangle 02:
   Location(12, 455), Size(431, 327), Brushes/Fill (240, 240, 240) and Brushes/Border (145, 145, 145).



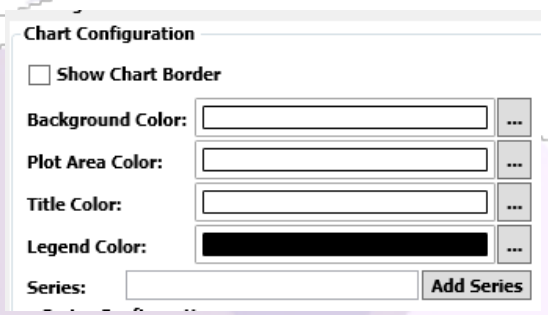
19.2. Configuring the Chart Object

Now let's add the two Chart objects of type “PIE”.

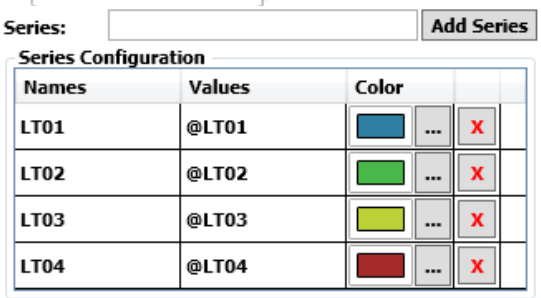
1. Click on the “Graphics” menu and then click on the “PIE” button as shown in the image below. Add two “PIE” type Chart objects to the Graphic Chart.



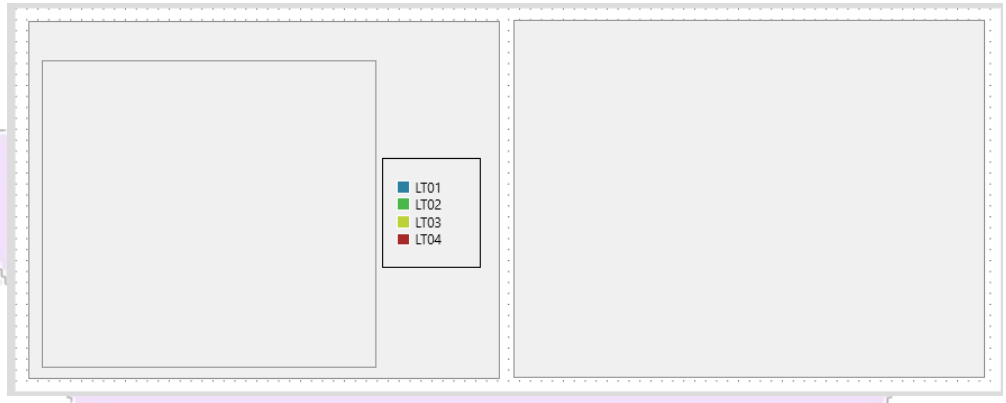
2. Select the first Chart object “PIE” and change the following properties:
   Location(13, 14), Size(426, 325), Title Color(255, 255, 255).



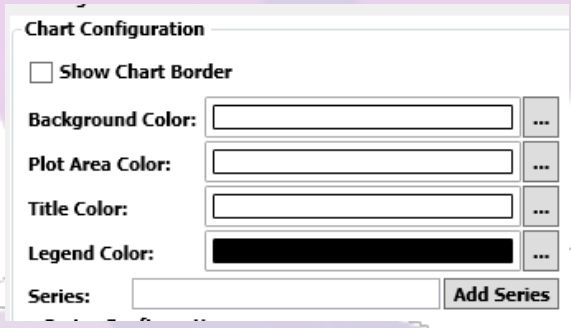
3. In the “Series” property, type the name of the tags (@LT01, @LT02, @LT03 and @LT04) in which they will be presented in the chart. In the Name column, enter the name of the tag to appear in the legend. Below is the expected result.



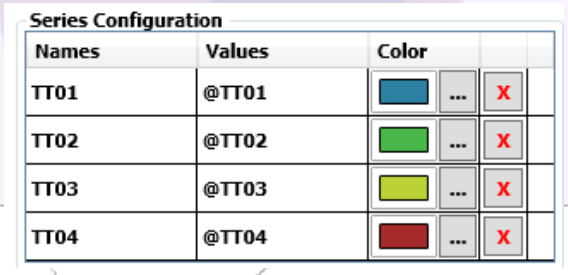
Below is the expected result.



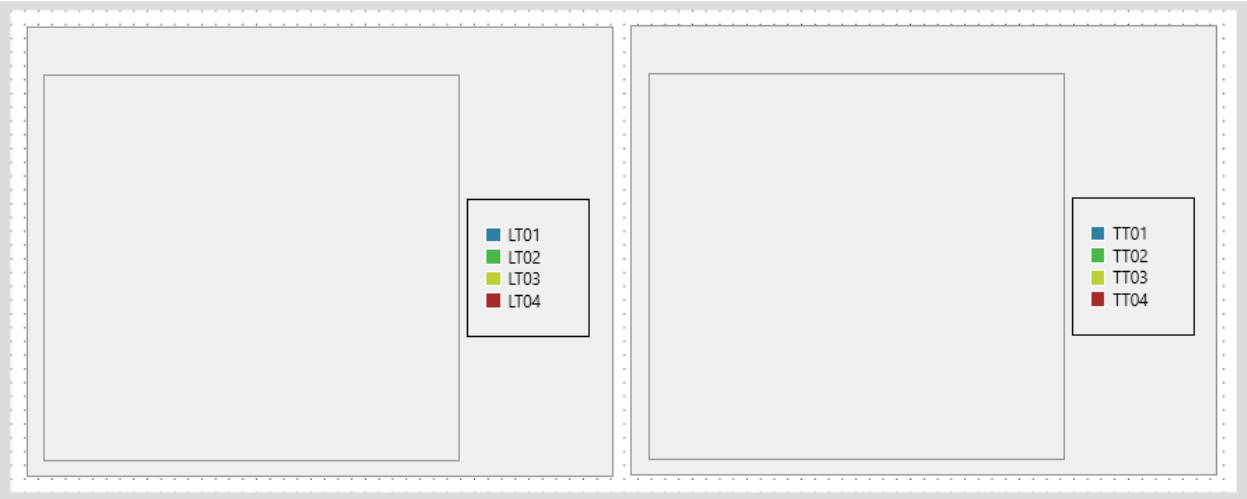
4. Now select the second Chart object “PIE” and change the following properties: Location(12, 458), Size(426, 325), Title Color(255, 255, 255).



5. In the “Series” property, type the name of the tags (@TT01, @TT02, @TT03 and @TT04) in which they will be displayed on the chart. In the Name column, enter the name of the tag to appear in the legend. Below is the expected result.

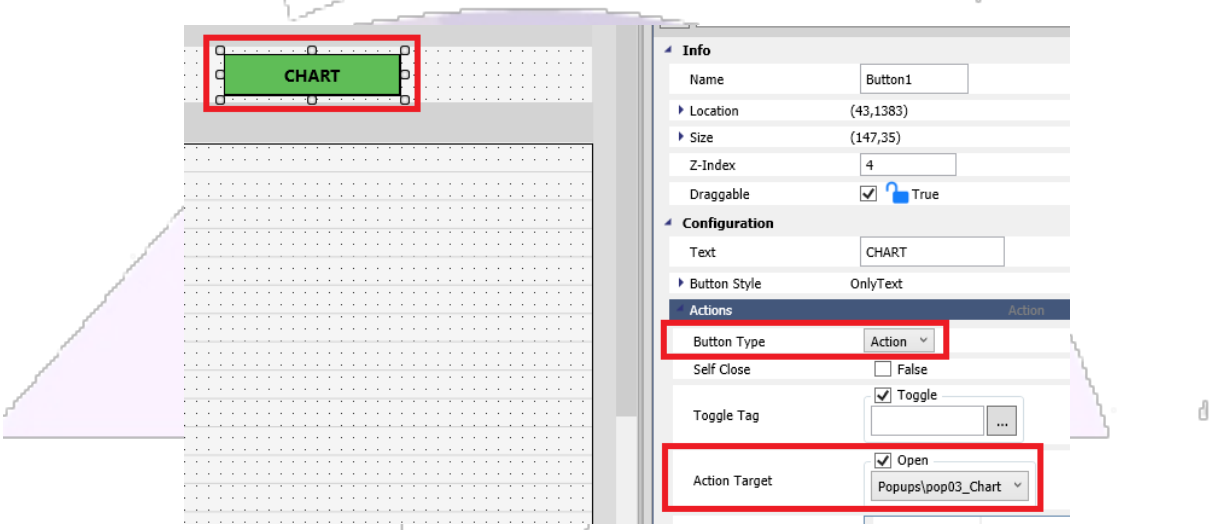


Below is the expected result.

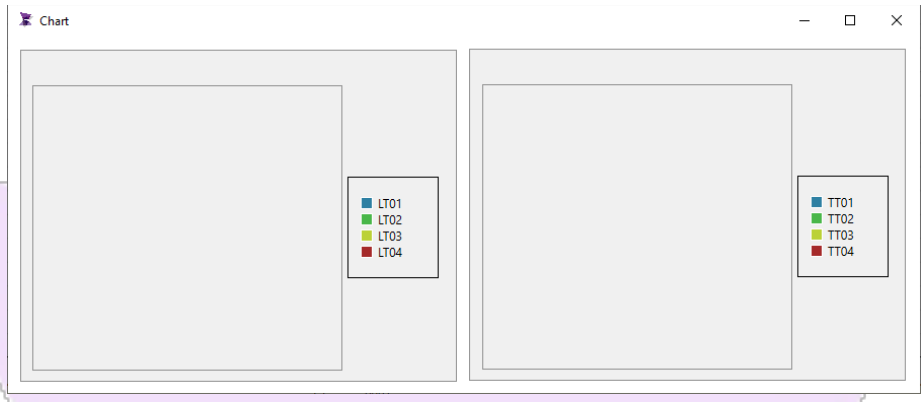


6. Now, let's insert a “Button” object in the “scr03_Trend” Graphic which will be responsible for opening the “pop03_Chart” graphic. In this way, save the graphic “pop03_Chart” and then open the graphic “scr03_Trend”. Insert a button object and change the following properties:

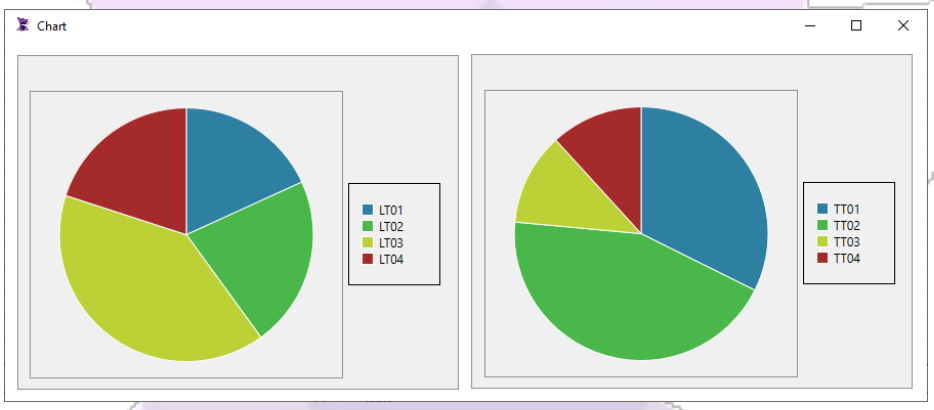
   Location(43, 1383), Size(147, 35), Text(CHART, Segoe UI, Bold, 14) , Background Color (95, 189, 87), Action/Button Type (Action) and Action Targe (Popups\pop03_Chart).



7. Save the graphic, start RunTime and navigate to the Graphic “Chart”. Probably the two “PIE” Charts will not be showing any information as shown in the image below.



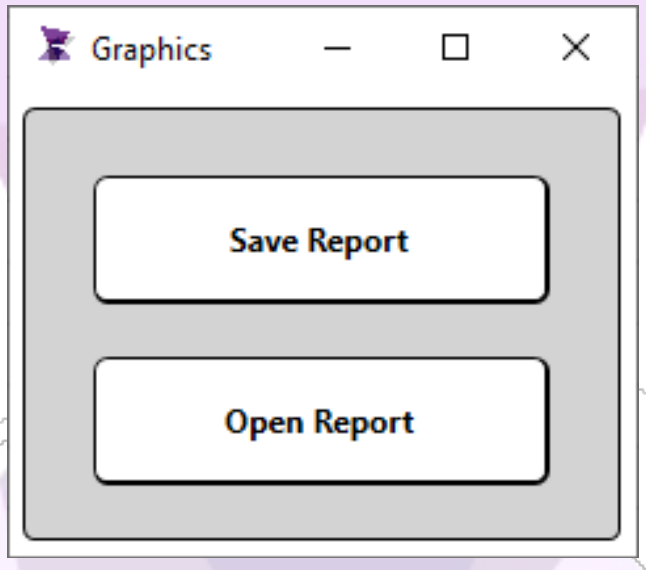
8. To view Chart “Pie” objects displaying the information correctly, open Data Watcher and put any value in tags LT-01 to LT-04 and tags TT-01 to TT-04. Below is the expected result.



20. Developing a Graphic Report

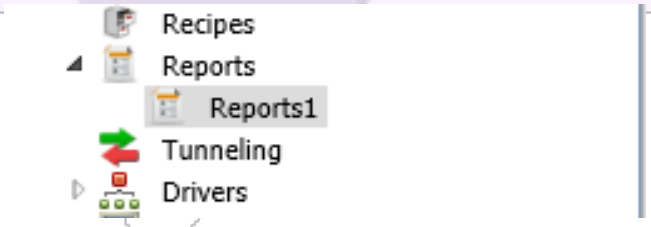
The Graphic Report will allow the user to save and open an application report. For Graphic to work, we must create the report using the Report document, located in the navigation tree. Therefore, we will first develop the report document and then the Graphic Report.

The Graphic Report will consist of a “Save Report” button and another button called “Open Report”. Each button will have a script using functions to save and open a report. First the user must save the report to open the report. Below is the demonstration of what the Graphic Report will look like.

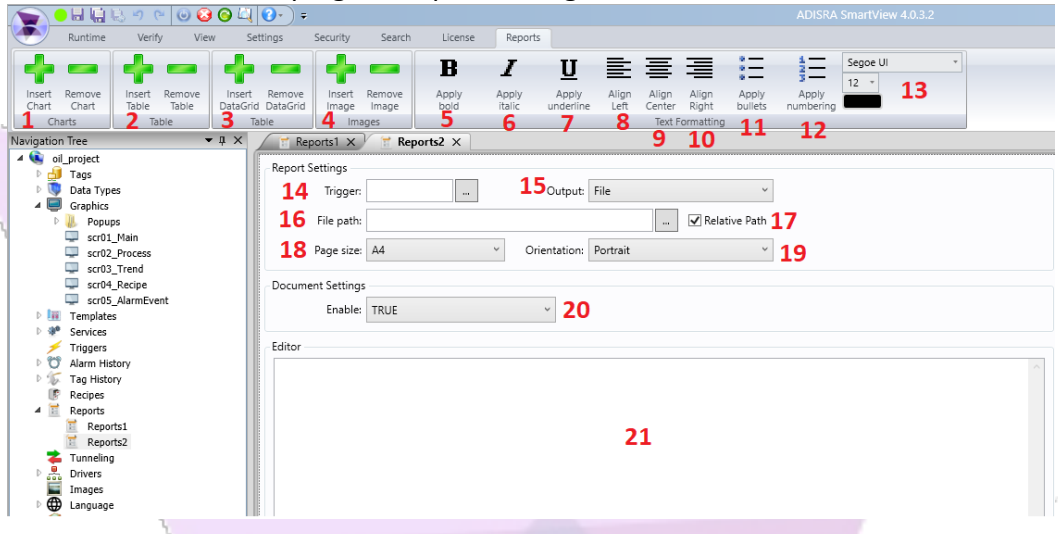


20.1. Creating the Report Document

1. In the navigation tree, right-click on the “Reports” document and then on “New Document”. Save the new document as “Reports1”.



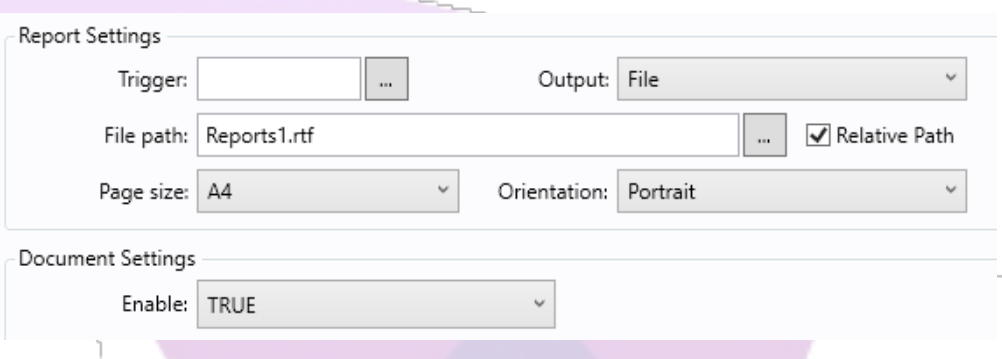
Before developing the report, let's get to know its main features.



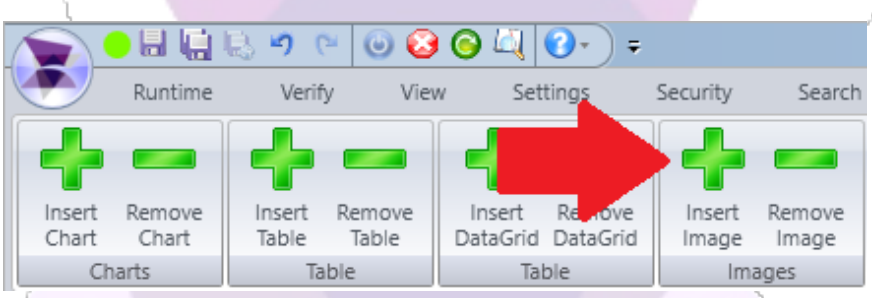
1- Insert and remove a Chart object;
2- Insert or remove a Table object;
3- Insert or remove a Datagrid object;
4- Insert or remove an Image object;
5- Bold text;
6- Put a text in Italics;
7- Place an underlined text;
8- Align text to the left;
9- Align text to the center;
10- Align text to the right;
11- Insert bullets before a text;
12- Insert numeric markers before text;
13- Format the size, font and color of a text;
14- Insert trigger tag. When the tag inserted in the field is “True”, the “Output” field will be executed;
15- Allows saving the report as a file (File), or sending the report for printing;
16- If the item above is selected (File), it is necessary to enter the directory where the report will be saved;
17- If selected, the file will be saved in the “Reports” folder of the project;
18- Sets the page size;
19- Sets the page orientation;
20- Allows you to enable or disable the document;
21- Report development area.

Now that we know each feature of the report document, let's configure and create our report.

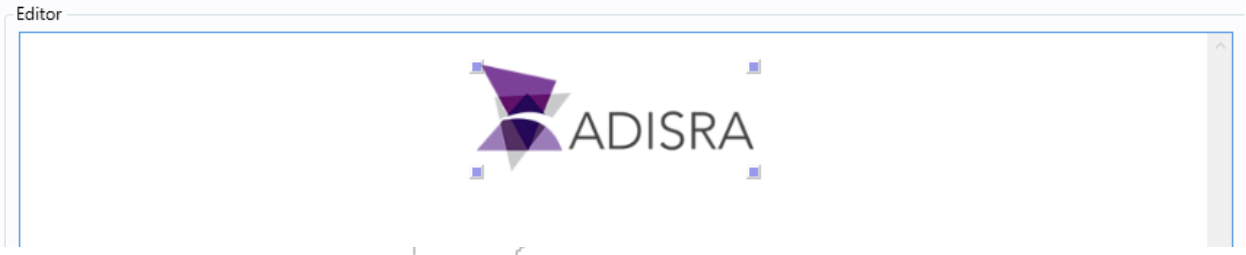
2. Let's start by configuring our report. To save and send the report for printing, we can use the configuration fields presented above or use the SVReports.SaveFile functions to save a report and the SVApplications.Run function to open a file. For our application, we will use the functions. Therefore, it will not be necessary to worry about the “Trigger”, “Output”, “File path” and “Relative Path” fields. Keep these fields as follows:



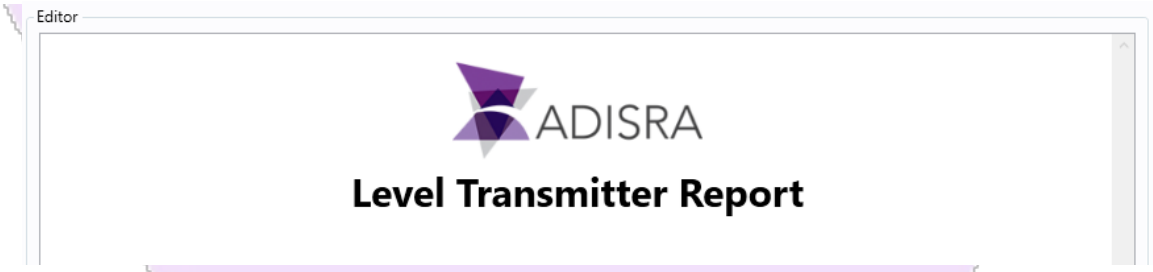
3. Now let's configure the interface of our report. Click on the “Insert Image” button and then click on the ADISRA logo.



4. Select the image and change the following properties: Size (190, 73) and Stretch (Fill). Then place the logo on the second line of the report and align it to the center. Below is the expected result.



5. On the bottom line, write the following text: “Level Transmitter Report”. Select the text and change the following properties: Size (22), Format (Bold), Font (Segoe UI), Color (Black) and Alignment (Align Center). Below is the expected result.



6. Skip three lines and write the following text: “Report generated in @DateTime”. Select the text and change the following properties: Size (12), Format (Not applicable), Font (Segoe UI), Color (Black) and Alignment (Align Center). Below is the expected result.

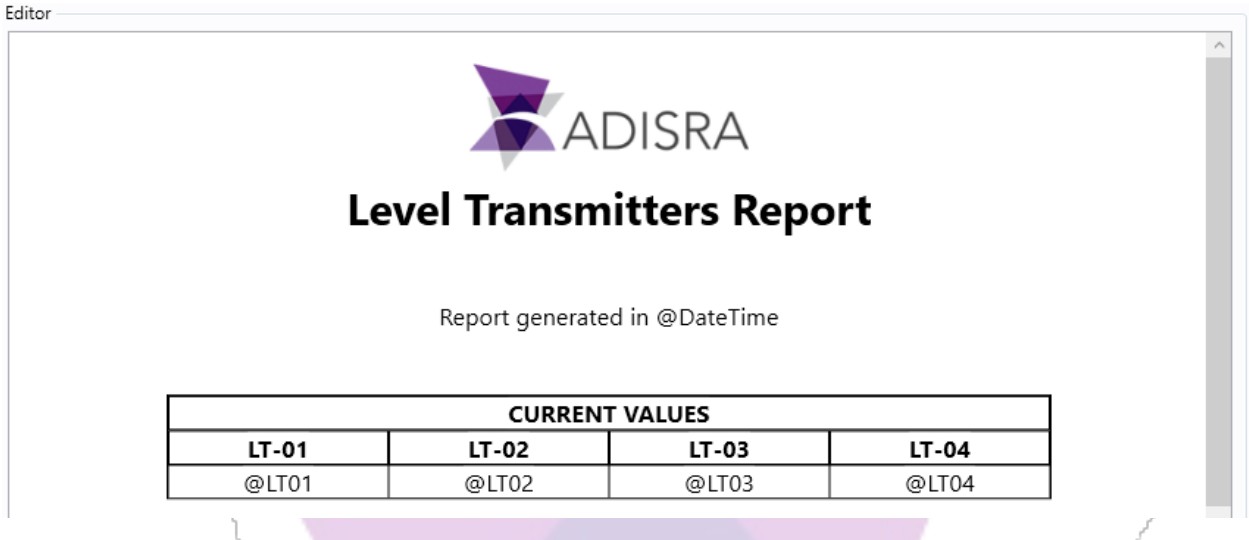
7. Skip two lines and then click on “Insert Table”. Select the table and change the following properties: Table Alignment (Center), Show Grid Lines (True), Number of Columns (4), Column Width (150) and Number of Rows (3). Below is the expected result.

TABLE TITLE			
Column Name 1	Column Name 2	Column Name 3	Column Name 4
Data11	Data12	Data13	Data14

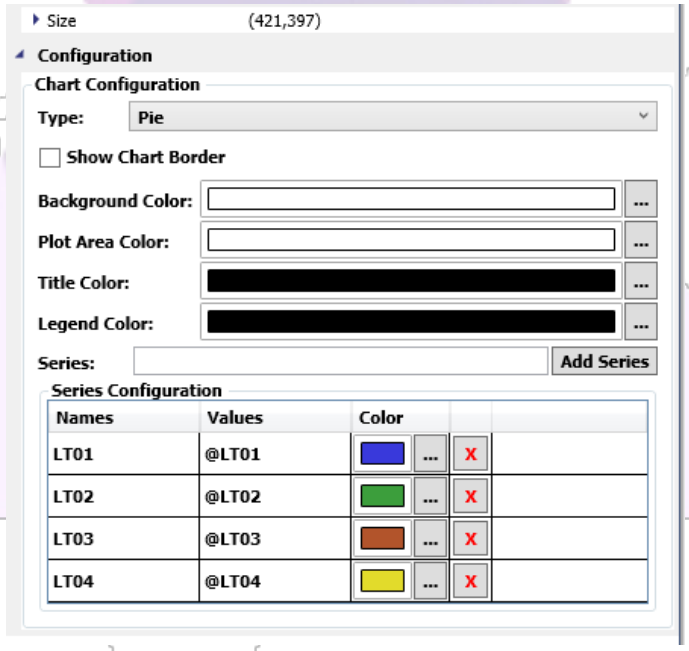
8. In the first row of the table, change the text “Table Title” to “CURRENT VALUES” and change the following properties: Size (12), Format (Bold), Font (Segoe UI) and alignment (Center).

9. In the second line, put the name of the transmitters in each column with the same format as the top line.

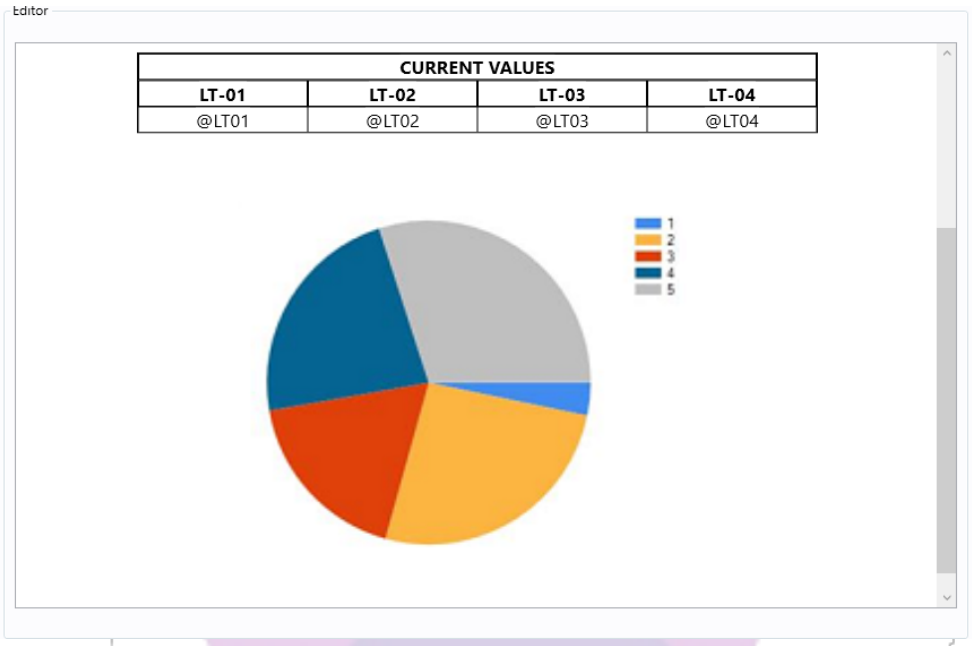
10. In the third line, place in each column the tag of each transmitter with the same formatting, but do not apply “Bold”. Below is the expected result:



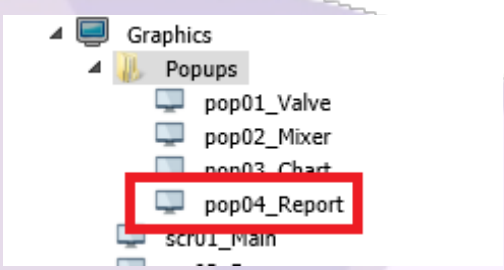
11. Below the table, insert a Chart object. To do this, click on the “Insert Chart” button. Select the object and change the properties according to the image below.



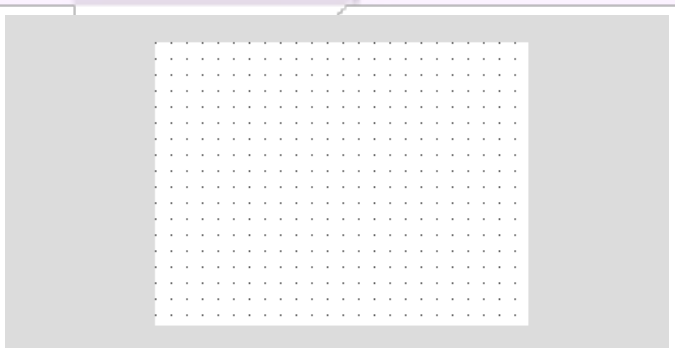
12. Below is the expected result.



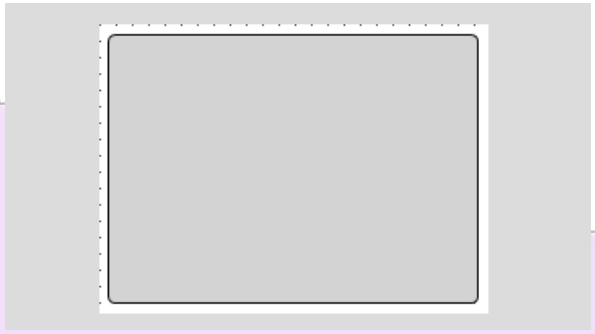
13. After configuring the Document report, let's create and develop the Graphic Report. So, open the Graphics Document, and inside the Popups folder, create the graphic "pop04_Report".



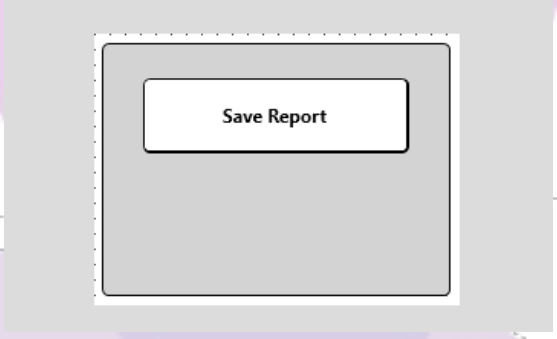
14. With the graphic "pop04_Report" open, change the following properties: Window Size (239, 177), Brushes/Background (255, 255, 255).



15. Now, insert a “Rectangle” object and change the following properties: Location (6, 5), Size (228, 165), Corner Radius (4) and Brushes/Fill (211, 211, 211). Below is the expected result.



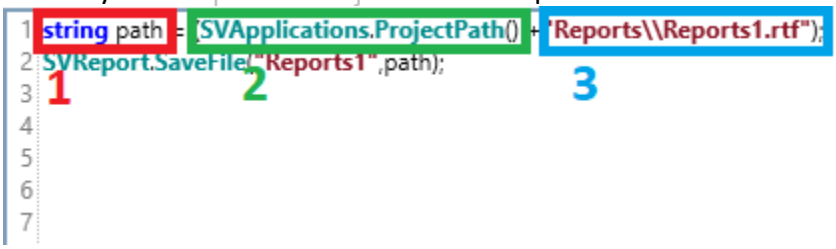
16. Insert a “Button” object and change the following properties: Location (29, 32), Size (174, 49), Text (Save Report, Selgoe UI, Bold, 12) and Corner Radius (4). Below is the expected result.



17. Now, select the “Save Report” button and we will insert the script to save the “Reports1” document. The file will be generated in the “Reports” folder of the application with the “.rtf” extension, which can be opened by Microsoft Word. So, add the script below in the “Mouse Down” event.

```
stringpath =(SVApplications.ProjectPath( ) +"Reports\\Reports1.rtf");
SVReport.SaveFile("Reports1",path);
```

Basically, the script will create a file called “Reports1.rtf” in the Reports folder located in the application directory. Let's understand the above script line a little bit.



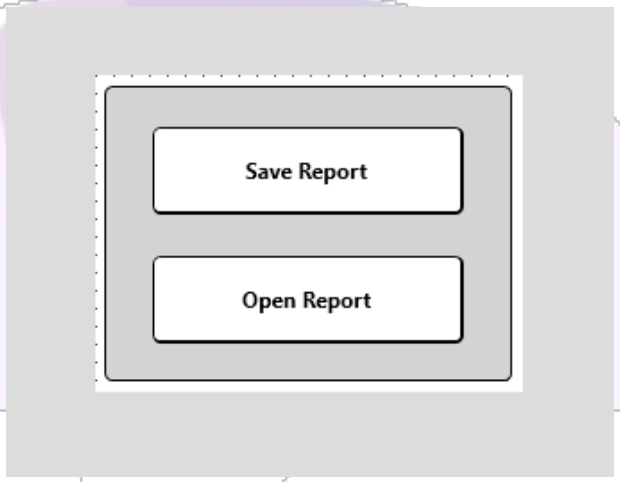
1- Declaration of the variable “path” of type string;
2- The SVApplications.ProjectPath() function returns the application directory;
3- The continuation of the directory and report format, i.e. in .pdf. Therefore, the “path” variable will receive the junction of the “ProjectPath” function plus the continuation of the directory.
In the line below we have the following:

```
1 string path = (SVApplications.ProjectPath() +"Reports\\Reports1.rtf");
2 SVReport.SaveFile("Reports1", path);
3
4
5
6
7
```

1 2 3

1- The SVReport.SaveFile function is intended to save the report;
2- The name of the Report document in which it will be saved;
3- The variable containing the complete directory, which was created in the previous line.

18. Now insert another “Button” object and change the following properties: Location(101, 32), Size(174, 49),Text (Open Report, Selgoe UI, Bold, 12) and Corner Radius (4). Below is the expected result.



19. Then, select the “Open Report” button and enter the script below.

```
stringpath =(SVApplications.ProjectPath( ) +"Reports\\Reports1.rtf");
SVApplications.Run(path);
```

The above script will open the file created by the report in the specified path.

```
1 string path = (SVApplications.ProjectPath() +"Reports\\Reports1.rtf");
2 SVApplications.Run(path);
3
4
5
6
```

The first line will do the same as the first line configured in the “Save Report” button.

```
1 string path = (SVApplications.ProjectPath() +"Reports\\Reports1.rtf");
2 SVApplications.Run path);
3
4
5
6
```

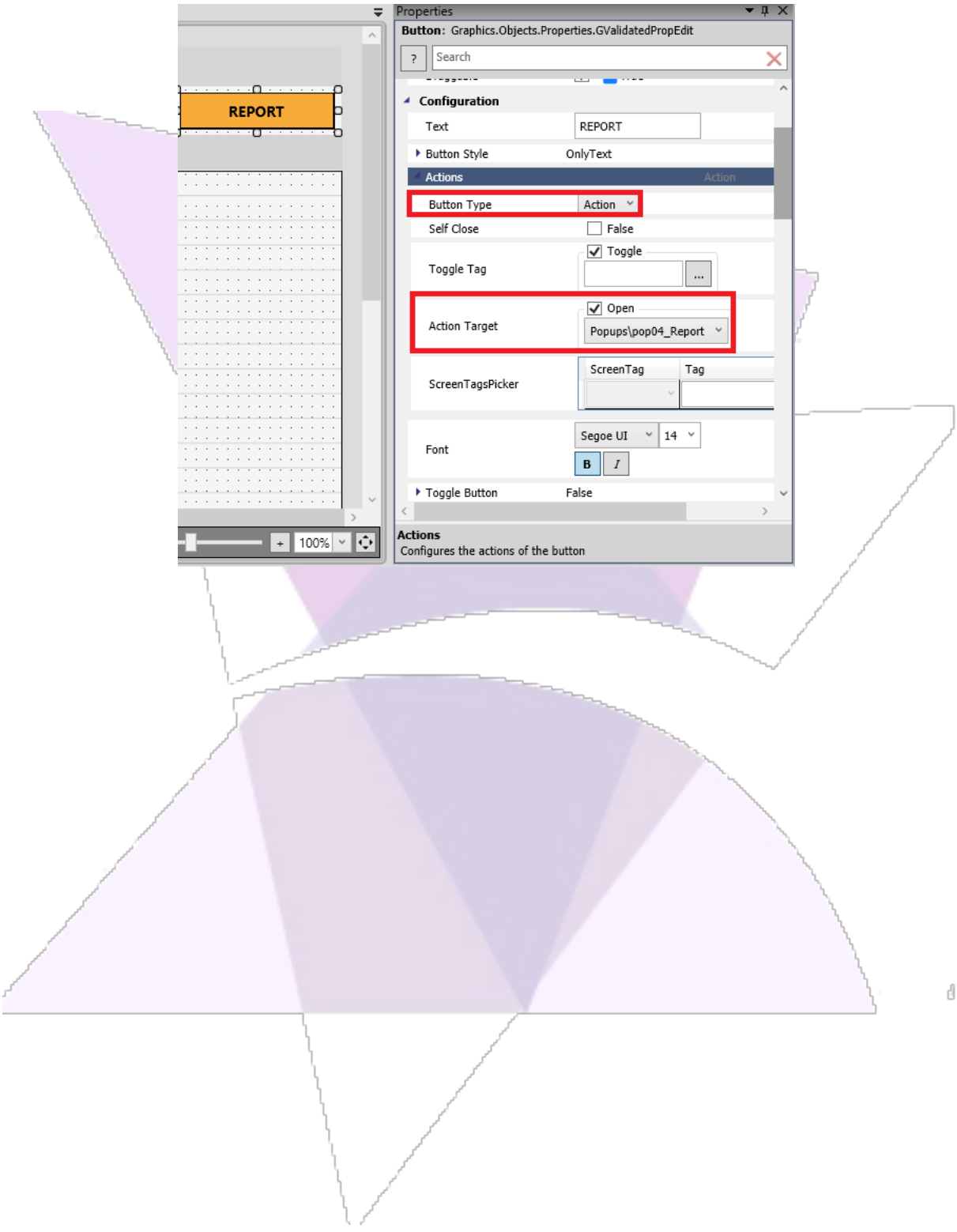
1 2

1- The SVApplications.Run function executes the file identified in the “path”.
2- The “path” variable containing the complete directory, which was created in the previous line.

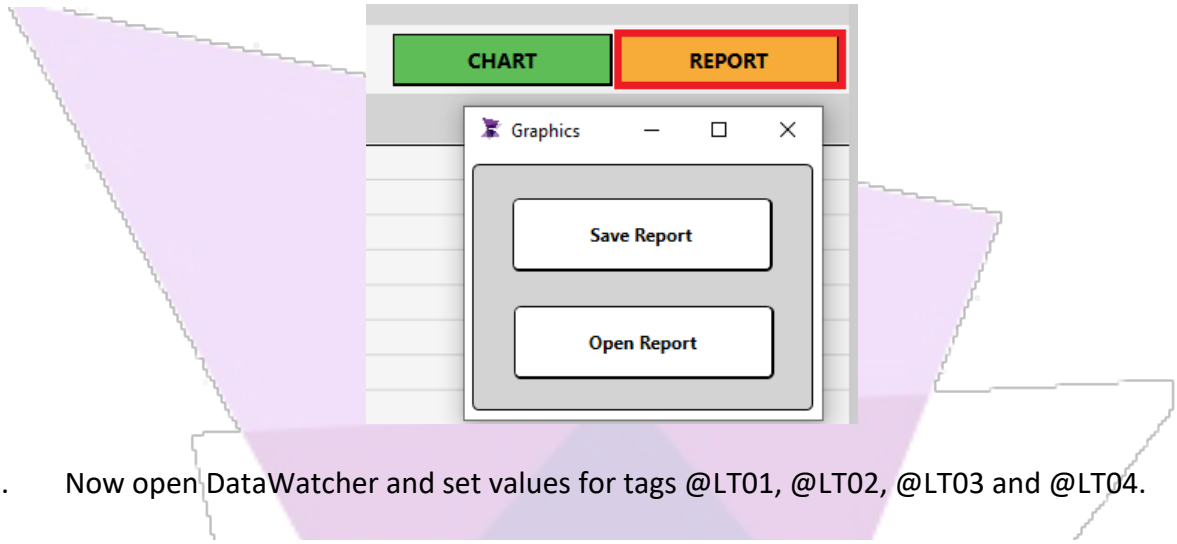
20. Save the changes made to “pop04_Report”.

21. Now, let's insert a “Button” object in the Graphic “scr03_Trend” which will be responsible for opening the graphic “pop04_Report”. Open the graphic “scr03_Trend”, insert a button object and change the following properties:

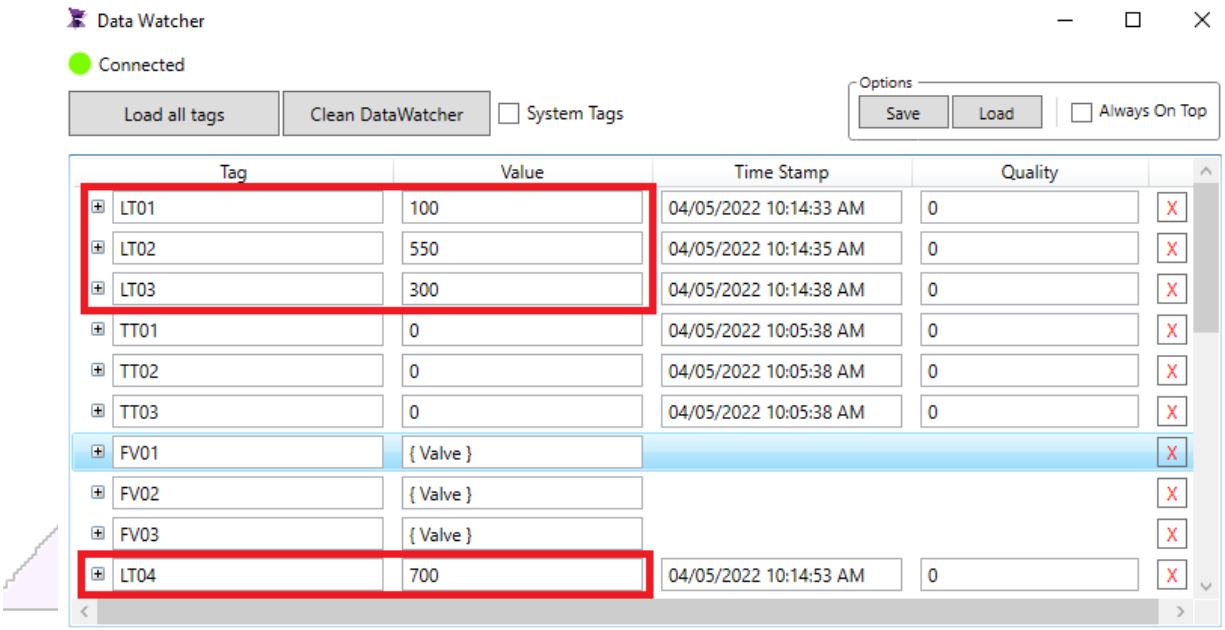
Location(43, 1535), Size(147, 35), Text(REPORT, Segoe UI, Bold, 14) , Background Color (247, 171, 57), Action/Button Type(Action) and Action Target (Popups\pop04_Report).


REPORT
100%
Properties
Button: Graphics.Objects.Properties.GValidatedPropEdit
Search
Configuration
Text
REPORT
Button Style
OnlyText
Actions
Button Type
Action
Self Close
False
Toggle
Toggle Tag
Action Target
Open
Popups\pop04_Report
ScreenTagsPicker
ScreenTag
Tag
Font
Segoe UI
14
B
I
Toggle Button
False
Actions
Configures the actions of the button

22. Save the application and start the Runtime.Open the “Trend” graphic and then click the “report” button.

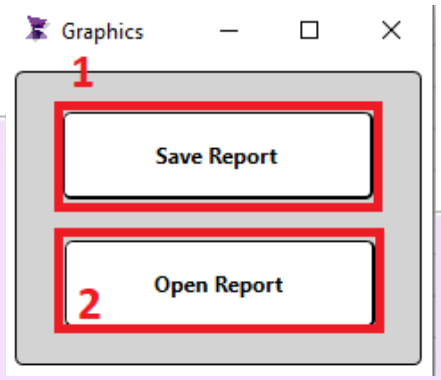


23. Now open DataWatcher and set values for tags @LT01, @LT02, @LT03 and @LT04.

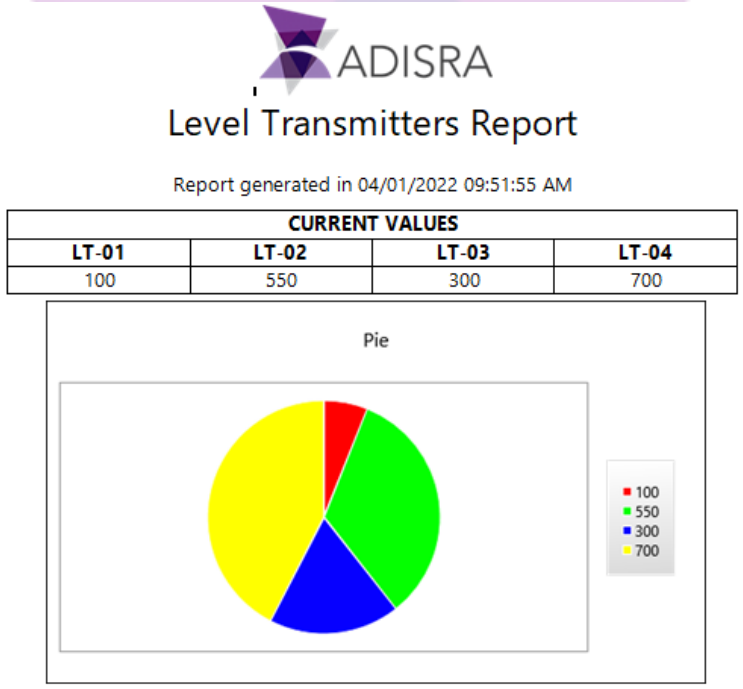


note: *If you save the report without putting any value in the tags, the report will be created, but the graph will not show any information.*

24. After placing the values in the “LT's” tags, click on the “Save Report” button and then click on the “Open Report” button.



25. Below is the final expected result of the configured report.

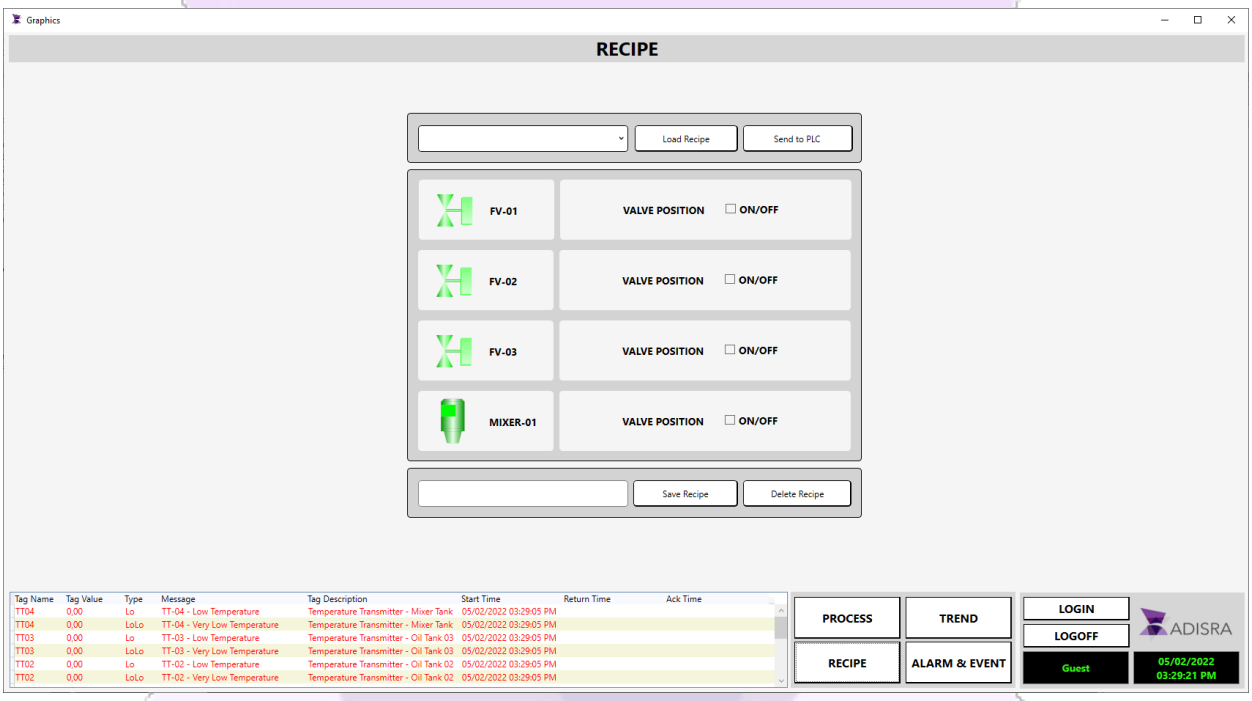


21. Developing a Graphic Recipe

Graphic Recipe will be responsible for allowing the user to create, delete and send a recipe to the controller. In this chapter we will know what revenue is and how to configure its main features.

Below is the graphic “Recipe” demo.

=



21.1. What is Recipe?

The Document Recipe is a resource that allows storing a set of values to be sent to the PLC in order to create pre-defined configurations.

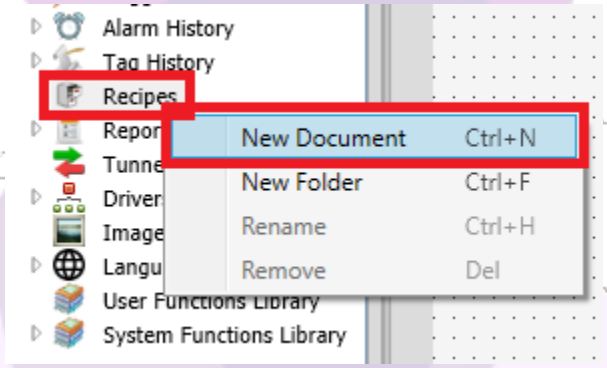
This document is used to facilitate the pre-defined configurations of an industrial process or machine. For example, a soda factory produces several types of soda, each with a different recipe. The user will be able to create and send these recipes, in which they define (the amount of sugar, water, etc.). Each recipe will have a set of tags representing each soda recipe, making it easy to fill in the configuration of machines and processes.

In our application, we will make a recipe for our hypothetical oil project. The user will be able to create, delete and send recipes for the oil process to the PLC. For example, if you create a recipe called “Oil Type A”, the FV-01 must be set to “True”, FV-02 to “False”, FV-03 to “False” and the Mixer-01 to “True”. For each type of oil, the user can create a recipe, save it and then send it to the PLC. Thus, for the process to create the “Oil Type A”, the user will only need to select the type of recipe in the ComboBox.

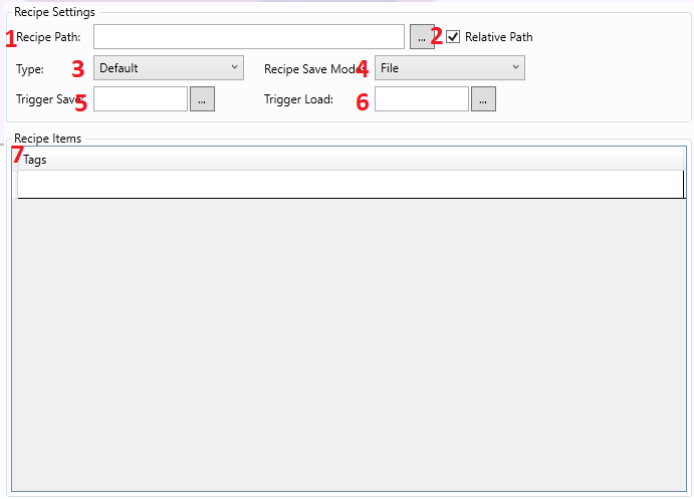
21.2. Configuring a Recipe Document

Now that we know what Recipes are, let's start configuring the Document Recipe.

1. In the navigation tree, right-click on the “Recipes” document and then on “New Document”. Save the new document as “Recipes1”.

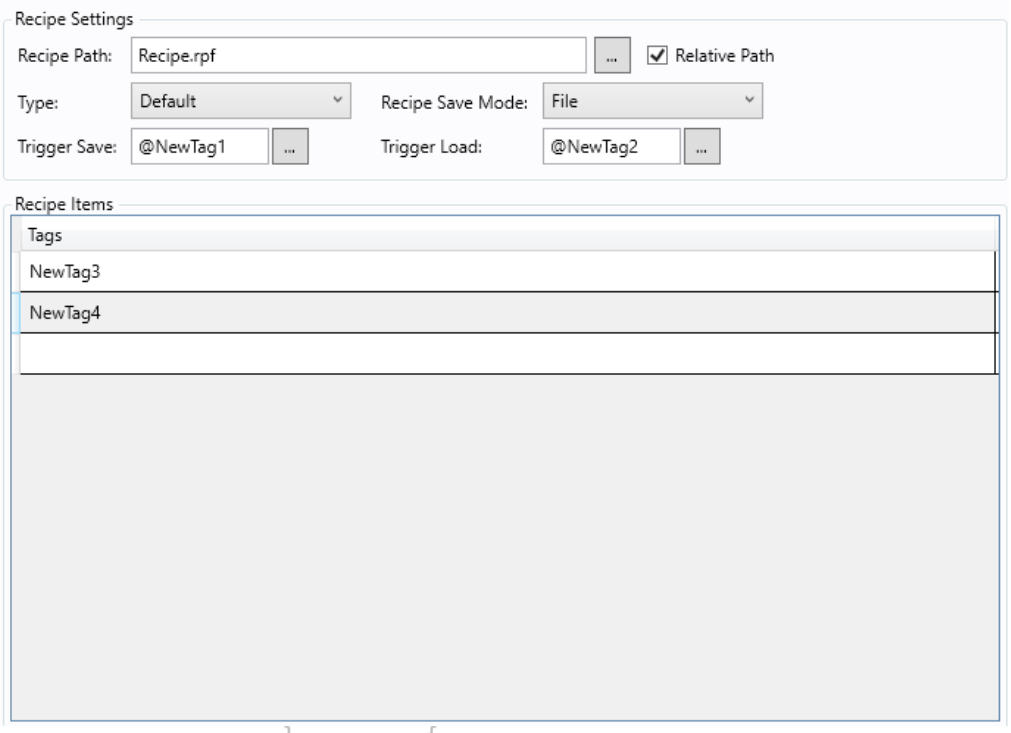


Before developing the recipe, let's get to know its main features.

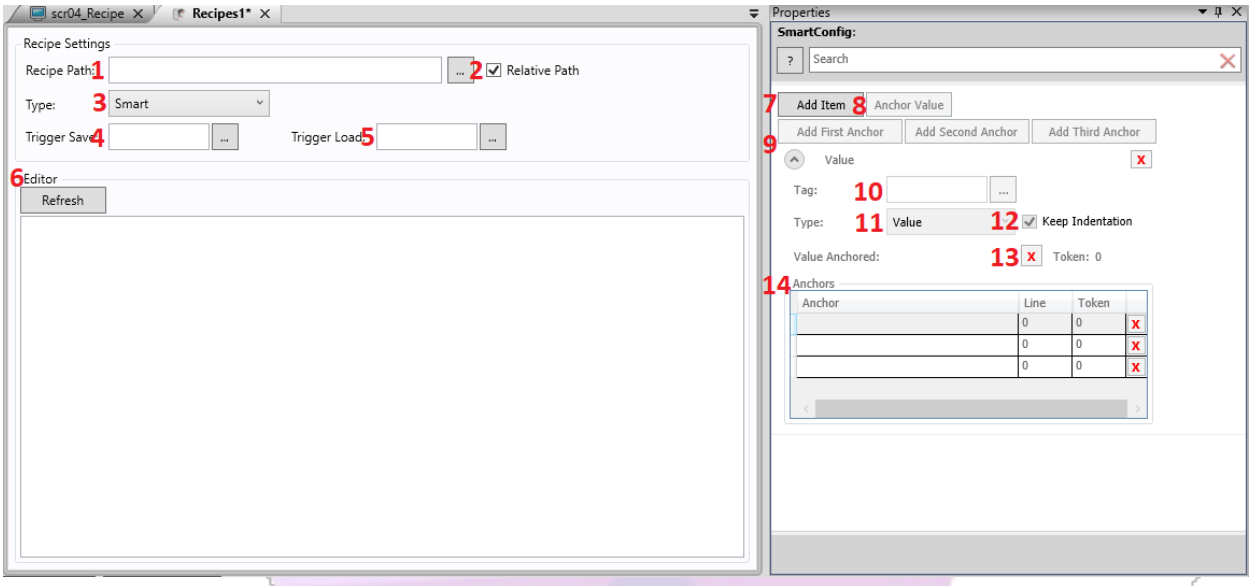


1- Recipe Path: Defines where the recipe reader will be saved in the default format.
2- Relative Path: Defines if the path to the recipe reader will be relative or not.
3- Type: Defines if the recipe is default (as shown in the image above) or smart.
4- Recipe Save Mode: Defines whether the saved recipes will be .rcp files or .xml files.
5- Trigger Save: Defines a condition or Boolean tag so that whenever its value is equal to “True”, the recipe will be created or updated.
6- Trigger Load: Defines a condition or Boolean tag so that whenever its value is equal to ”True”, the reader will load and define all information in the document based on the recipe in the Recipe Path.
7- Recipe Items: Defines which tags the recipe will change.

The following image represents an example of a configured recipe document.



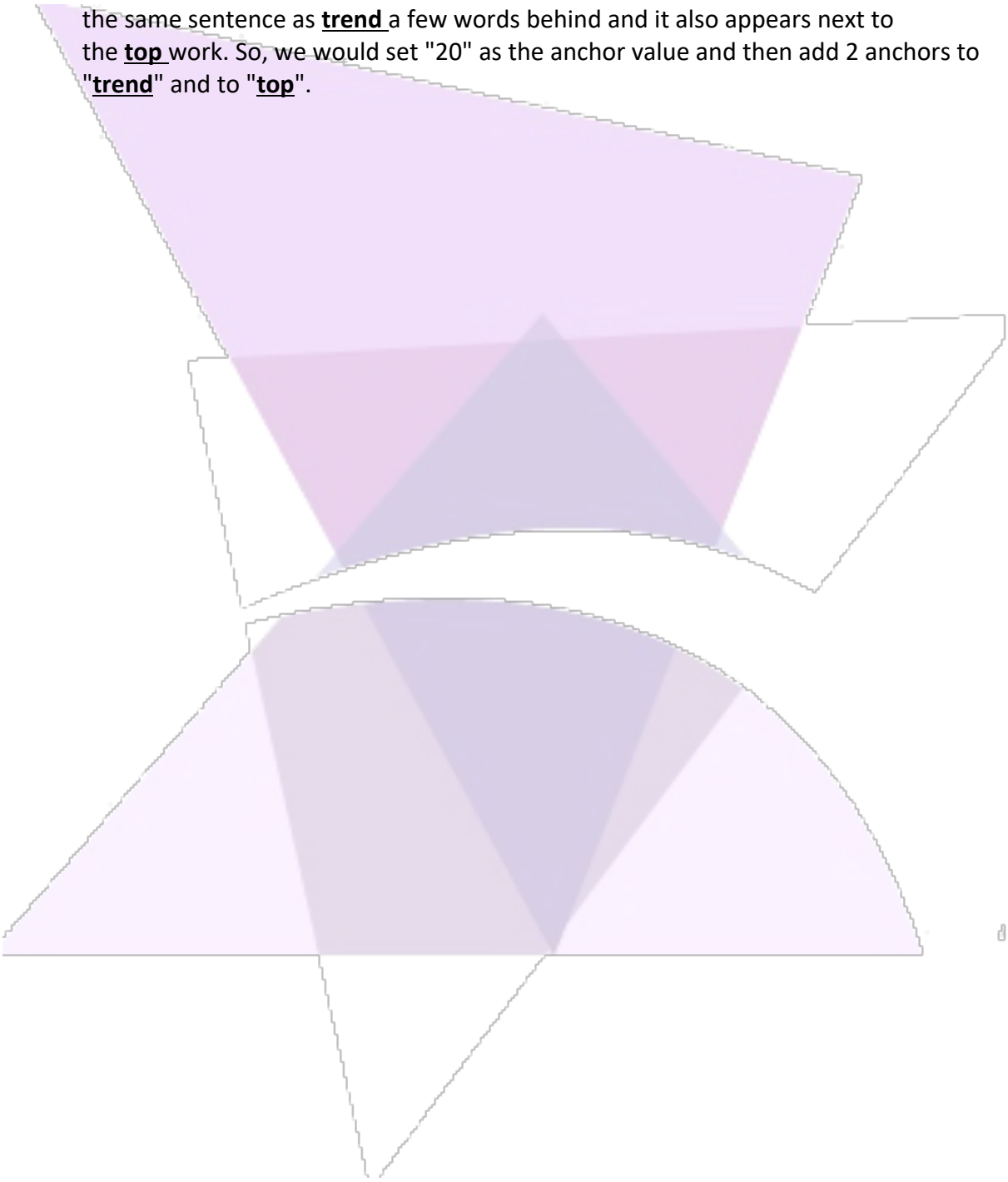
Below is represented how the document and its settings are when the recipe type is smart.



The smart recipe uses a text file so it can read and anchor selected tags to values in the text.

1- **Recipe Path:** Defines which text file the recipe will be based on.
2- **Relative Path:** Defines whether the path to the file will be relative or not.
3- **Type:** Defines the document type.
4- **Trigger Save:** Defines a condition or Boolean tag so that whenever its value is equal to "True", the recipe will be created or updated.
5- **Trigger Load:** Defines a condition or Boolean tag so that whenever its value is equal to "True", the reader will load and define all the information in the document based on the recipe in the Recipe Path.
6- **Editor:** Area where the selected text will appear so you can anchor values to tags.
7- **Add Item:** Adds an item. The items in the properties area are used to define the tags that will be modified by the recipe and to anchor where the tag value will come from in the text file.
8- **Anchor Value:** Defines which value the tag will receive from the text file.
9- **Add First/Second/Third Anchor:** Defines the location the anchor value will be based on. The more anchors you use, the more accurate it will be in case the text has similar values.
10- **Tag:** Defines which tag will have its value modified by the recipe.
11- **Type:** Allows linking a single tag value or, when dealing with a table, you can also link an entire line, column or matrix to an array tag.
12- **Keep Indentation:** Respects the indentation of the document. If it contains spaces or tabulations before some anchors, this checkbox guarantees it will respect the original layout. When unchecked, it allows a recipe without the spaces and tabulations.
13- **Value Anchored:** As soon as an Anchor Value is added, it displays the value on that field.
14- **Anchors:** Contains the 3 anchors and the information about the location of those anchors. These anchors help the smart recipe to locate the anchor value. (i.e. "The trend

object allows customization. Change the top value to 20"). That sentence might be used in a recipe, but I don't want to set "20" in other recipes, so the text 20 in this recipe will be loaded and replaced when requested. Since it is a big recipe, it probably has other "20"s, so we need to make sure we will only replace "that 20". That 20 that appears in the same sentence as **trend** a few words behind and it also appears next to the **top** work. So, we would set "20" as the anchor value and then add 2 anchors to "**trend**" and to "**top**".



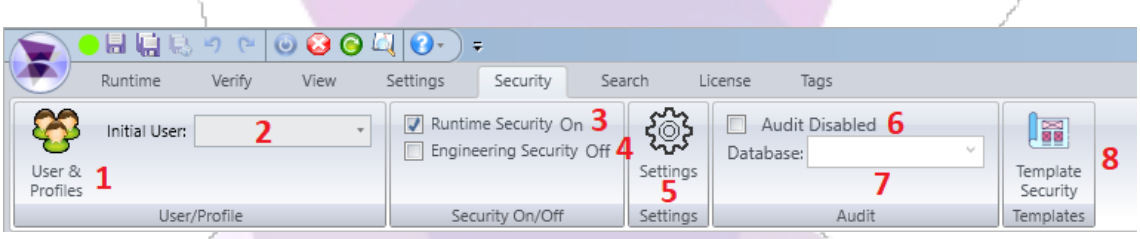
22. Security & Users

ADISRA SmartView allows the developer to create a mechanism to manage the access level of users in RunTime mode, that is, depending on the Profile and/or user settings, an application grants or denies actions, such as clicking a button, starting or stop an application, open task manager, keyboard shortcuts and others. It is also possible to protect the engineering environment, that is, if the application is opened in engineering mode, the developer must enter a password to access the application engineering.

For our hypothetical project, we are going to create profiles and users with specific privileges.

22.1. Knowing Security Settings

To access the security settings of an application, in the top menu, click on the "Security" option.



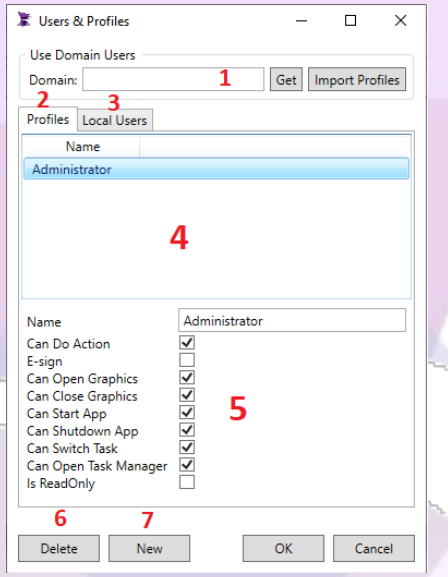
1- Opens the Profiles and Users settings;
2- Sets the initial user after running the runtime;
3- Enables or disables the RunTime Mode Security System;
4- Enables or disables the Engineering Mode security system, that is, once enabled, it will be necessary to enter a password;
5- Opens Viewer Security and Password settings;
6- Enables or disables Audit Trail;
7- Defines in which external database the Audit Trail electronic records will be stored;
8- It opens the Templates Security settings, that is, it is possible to enter a password to access a template developed in the application.

Now that we know what each option of the ADISRA SmartView Security System is, let's take a closer look at each option below.

22.2. Configuring Users & Profiles

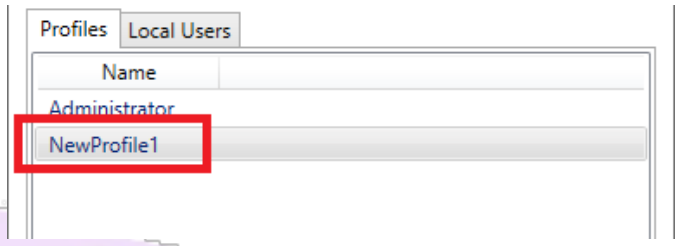
1. Click on the "User & Profiles" option.

Before continuing the step-by-step, let's get to know each configuration field of the “Users & Profiles” option.



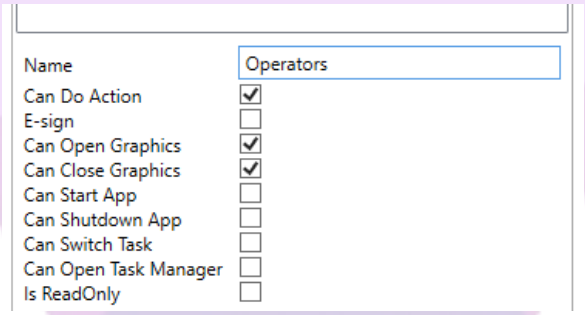
1- Allows you to use the user settings from the Windows OS;
2- Tab in which it shows all the “Profiles” of users;
3- Tab in which all the “Users” are shown;
4- Depends on open tab, shows all “Profiles” or application users;
5- Options to enable or disable user privileges “Profile” or the user selected above;
6- Deletes the selected “Profile” or “User”;
7- Creates a new “Profile” or “User”, depending on the Open tab.

2. Let's start creating and configuring the “Profiles” and “Users” of the application. First, note that by default, the application has a “profile” called “Administrator” in which it has several privileges granted. For that profile, keep all privileges. Next, let's create one more “Profile”. With the “Profiles” tab open, click on the “New” button.



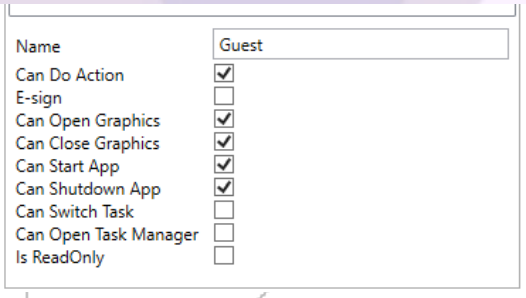
Note that a new Profile was created with the name “NewProfile1” and with some predefined privileges.

3. Change the name of the new Profile to “Operators” in the name field and click “Enter”. Then set the following privileges as shown in the image below:

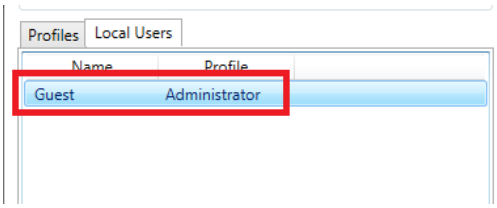


note: *After defining the privileges of the “Operators” Profile, any user associated with that profile will only be able to “Can Do Action”, Can Open Graphics” and “Can Close Graphics” in the application in RunTime mode.*

4. Create another Profile called “Guest” and set the following privileges:



5. Now, let's create the users and associate them with the “Profiles” created. Then, click on the “Local Users” tab. Note that there is already a user named “Guest”, which is associated with the “Administrator” profile, that is, all privileges of this user are defined by the profile in which the user was associated. In this case, the “Guest” user is associated with the “Administrator” Profile.



6. Change the user profile from “Guest” to “Guest”. To do this, click on the “Profile” option and select the “Guest” profile.

7. From now on, the “Guest” user will have the privileges of the “Guest” Profile, that is, only “Is ReadOnly”.

8. Now, let's create three more users by clicking on the “New” button, they are:

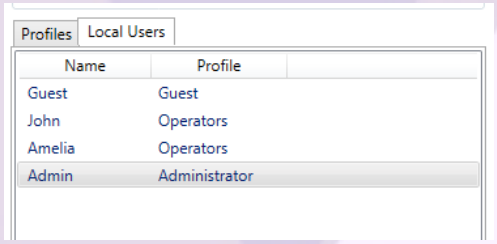
   Name: JohnLogin: JohnPassword: 123Profile: Operators

   Name: AmeliaLogin: ameliaPassword: 123Profile: Operators

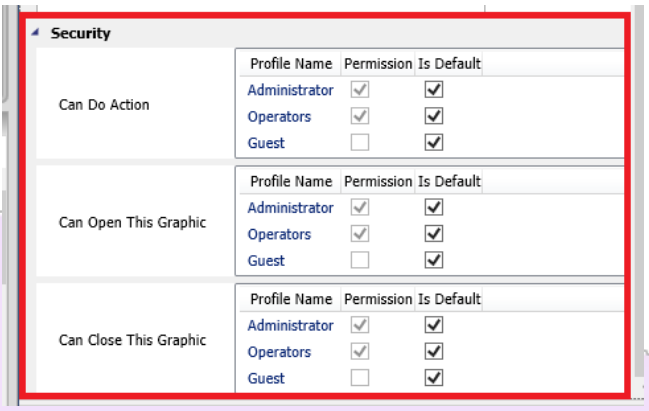
   Name: AdminLogin: adminPassword: 123Profile: Administrator

   note: *For the “Guest” user, select it and click the “Change Password” button, then keep the password fields blank and click OK. We will leave the user “Guest” without a password.*

Below is the expected result.



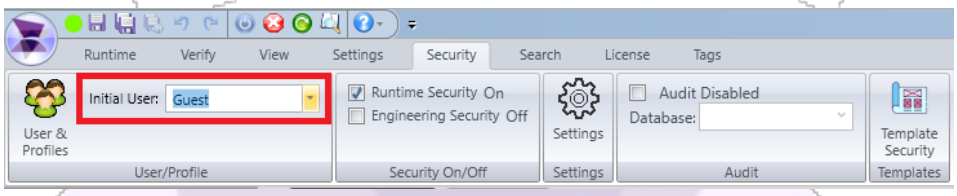
It is also possible to configure the privileges of “Profiles” directly on an application object. For example, open the graphic “scr01_Main” and then click on the “Alarm & Event” button. In the property list, notice that there is a property called “Security”, which allows you to grant privileges to “Can Do Action”, “Can Open This Graphic” and “Can Close This Graphic”.



Within each privilege it is possible to define whether it will respect the settings made in the "Users & Profiles" window or local settings. If the option "Is Default" is checked, it will respect the settings in the "Users & Profiles" window. If the "Permission" option is checked, it will respect the object's property settings. For our project, keep the settings as "Is Default".

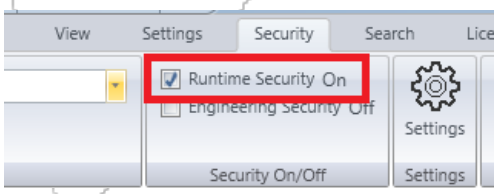
22.3. Setting the RunTime Initial User

After creating the "Profiles" and "Users", click the "OK" button. In the "Initial User" property, set the "Guest" user to start in RunTime mode.



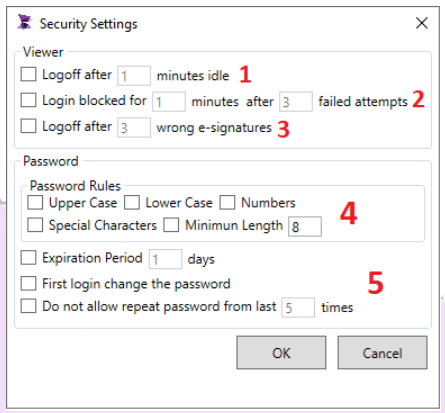
22.4. Configuring the RunTime and Engineering Mode Security System

1. In the next field "Security ON/OFF", keep the option "RunTime Security On" enabled for the security system to work in RunTime mode.



2. In the "Engineering Security Off" option, if you want to put a password on the application in engineering mode, enable this option.

3. Now let's get to know and configure the "Settings" option.



1- If enabled, Automatic user logoff after X minutes.
2- If enabled, blocks User Login by X Minutes after X failed attempts;
3- If enabled, Automatic user logoff after X electronic signature errors;
4- Define password rules, such as capital letter and lowercase, numbers, special characters and minimum characters in user passwords;
5- Sets rules for passwords, like, expires in X days, user will need to change the password in the first login and repeating the last X is not allowed previously used passwords.

22.5. Getting to Know the Audit Trail

The Audit trail Feature, known as (Audit Trail) is a detailed chronological record of all user actions in the application. Audit Trail is widely requested by food and pharmaceutical industries to ensure that all user actions are tracked.

This feature was developed to comply with FDA 21 CFR Part 11 regulations of the Food and Drug Administration (FDA), which establishes the criteria under which the agency considers electronic records and electronic signatures to be reliable, being equivalent to paper records and handwritten signatures. on paper.

An Electronic Record is any data that can be saved electronically and retrieved later.

An Electronic Signature is a specific type of Electronic Record that contains the following information:
- Timestamp;
- User name;
- Meaning of the signature.

SCADA software cannot claim to conform to the standard. The software must provide the tools necessary to allow a developer to create an application that is FDA 21 CFR Part 11 compliant.

SCADA does not "force" the developer to build an application that complies with the standard. Compliance is optional during application development.

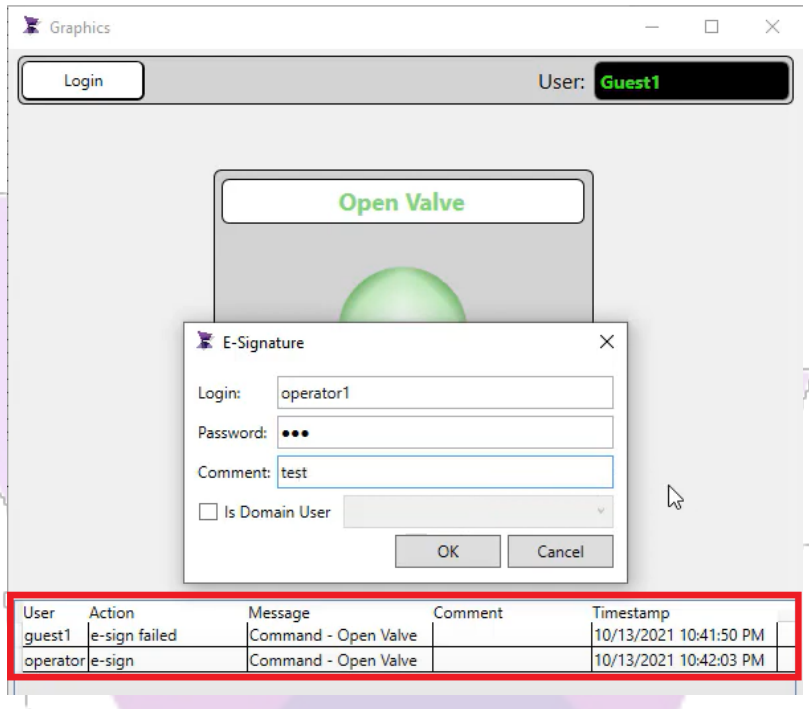
Therefore, to meet the requirements of FDA Standard 21 CFR Part 11, ADSIRA SmartView has the following functionalities:

- Controlled access to SCADA application: All users of the SCADA system can be properly registered with unique and personal passwords. Therefore, these users will be authentic and responsible for their actions within the system;
- Controlled access to SCADA application: All users of the SCADA system can be properly registered with unique and personal passwords. Therefore, these users will be authentic and responsible for their actions within the system;
- No one (not even the system administrator) has access to any user's password;
- Automatic user logoff: this functionality protects the system by blocking access to it if the user is not present, as this way no other person could impersonate the last user who logged in without having logged out properly;
- Records of occurrences in Audit Trail: critical actions within the SCADA system such as initialization, termination, equipment setpoint change, report generation, among others, are recorded in a Database;
- Audit Trail preview in RunTime Mode;
- Database connectivity (SQL Server, Oracle, Microsoft Access, etc.).

note[1]: *The standard does not mention whether electronic records must be stored in a standard database or in a proprietary format.*

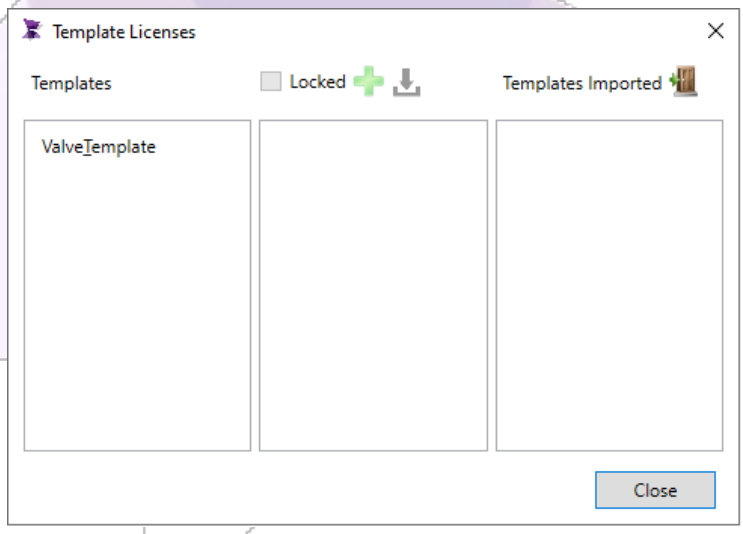
note[2]: *In our hypothetical project, we will not use this feature. For more information, visit our KnowledgeBase.*

The following is the audit trail of an example application.



22.6. Getting to Know Template Security

This feature allows the developer to set a password to access a specific template created in the application.



note: *We will not use this feature in our application.*

22.7. Configuring User Login and Logoff

After knowing and configuring the Users and Profiles of the application, we need to configure the Login and Logoff buttons of the graphic “scr01_Main”.

1. Open the “scr01_Main” graphic, then select the “Login” button and insert the following script in the “MouseUp” event:

```
SVSecurity.Login();
```

This function aims to open a login window requesting the User Login and password.

2. Then select the “Logoff” button and insert the following script in the “MouseUp” event:

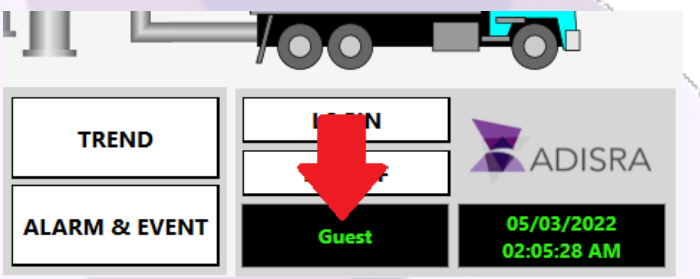
```
SVSecurity.logoff();
```

This function aims to open a login window requesting the User Login and password.

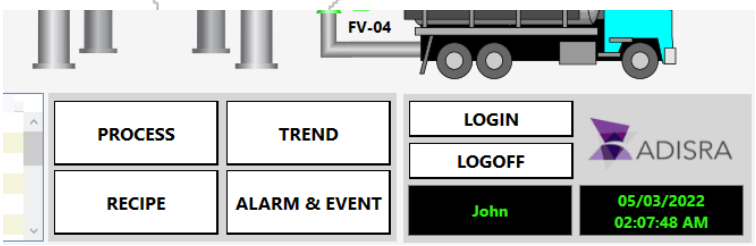
22.8. Testing the Security System

After knowing and configuring each option of the ADSIRA SmartView security system, let's test the application.

1. Save the application and start RunTime. Once started, note that the current user is the “Guest” user, which was configured to start along with RunTime.



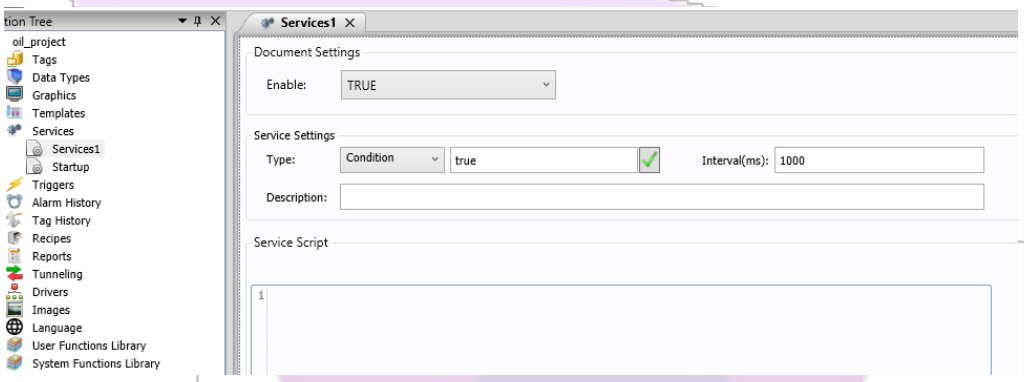
2. Now, let's test the “Login” and “Logoff” buttons. Login user “John”. Login is “John” and password “123”. Below is the expected result.



3. Then, log off the user “John”.

23. Document Services

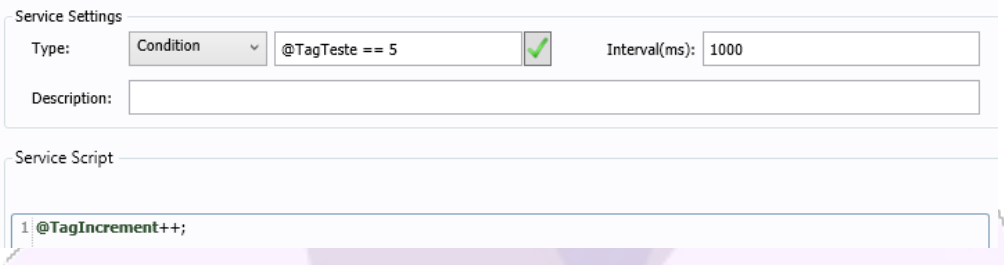
The Document Service aims to execute scripts, in which the execution can be of the "Condition" or "Trigger" type.



.

23.1. Condition Type

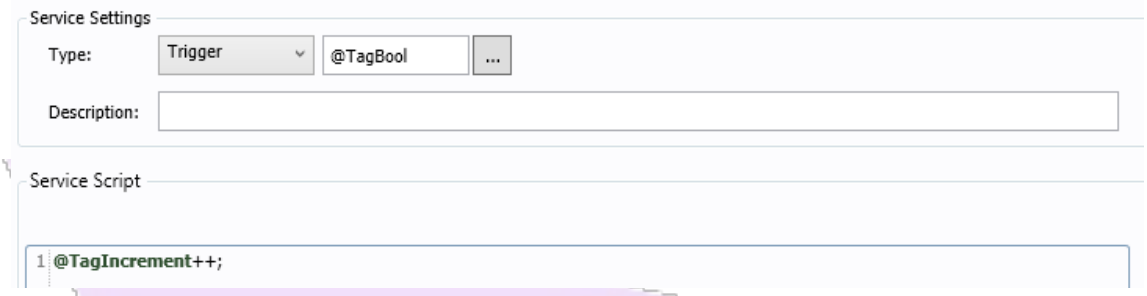
The "Condition" type executes a script if the condition is "True". Therefore, the script will be executed every "X" milliseconds, as long as the condition is "True".



According to the example above, whenever the condition "@TagTeste == 5" is satisfied, the script will be executed every 1000ms.

23.2. Trigger Type

The "Trigger" type executes a script if the value of the associated tag is changed from "False" to "True" or "True" to "False".

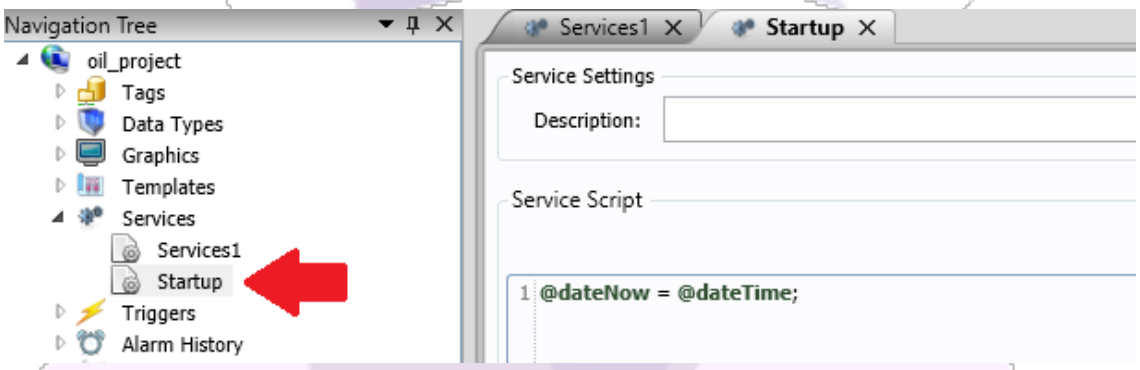


According to the example above, whenever the "TagBool" tag changes its value, the script will be executed. While in the "Condition" type the script will be executed every 1000ms, in the "Trigger" type the script will be executed only once until the tag has its value changed again.

note: *In the "Condition" type, the execution time can be changed as needed.*

23.3. Startup Type

By default, the application automatically creates a service called "Startup". The script inside this service will be executed once every time the RunTime is started.



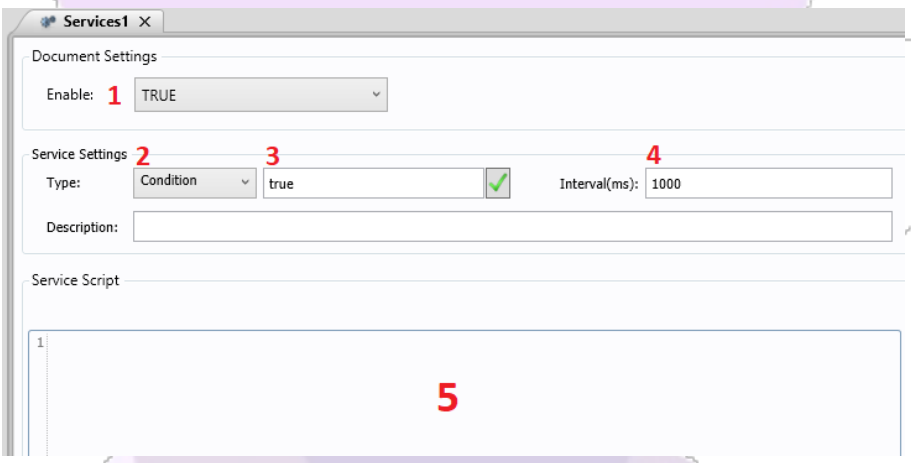
As per the example above, whenever the RunTime is started, the script "@dateNow = @dateTime;" will run only once.

note: *The "Startup" Document cannot be removed or renamed.*

23.4. Configuring the Condition Type

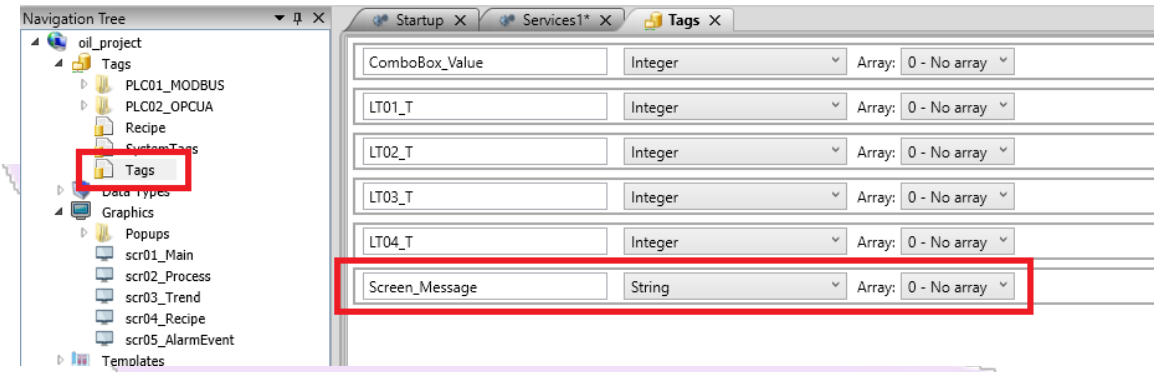
1. In the navigation tree, right-click on the "Services" document and then click on the "New Document" option and save the new document as "Services1".

2. Before continuing the step-by-step, let's get to know each configuration field of the "Services" document.



1- Enables or disables the Document "Services";
2- Defines the type of condition;
3- Field to insert the condition tag;
4- Defines the script execution interval;
5- Field to insert the script in C#.

3. Now let's configure our Document Services. To do this, open the "Tags" document and create a tag called "Screen_Message" of type "String". Save the "Tags" document and then return to the "Services1" document settings.



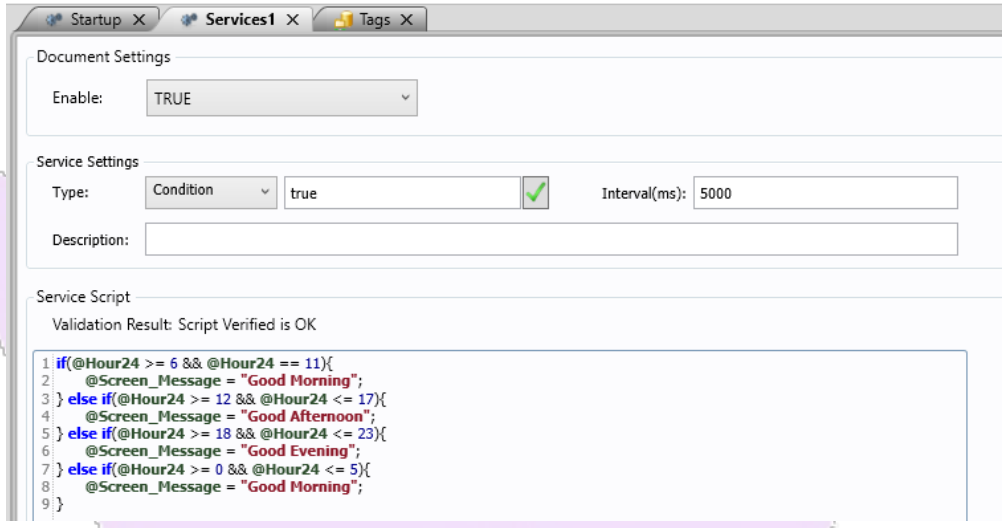
note: *The tag "Screen_Message" will be used in the script of Document "Services1".*

4. In the "Services1" document, change the following properties:

Enable: True
Type: Condition
Condition field: True
Interval(ms): 5000
Service Script:

```
if(@Hour24 >= 6 && @Hour24 == 11){
    @Screen_Message = "Good Morning";
} else if(@Hour24 >= 12 && @Hour24 <= 17){
    @Screen_Message = "Good Afternoon";
} else if(@Hour24 >= 18 && @Hour24 <= 23){
    @Screen_Message = "Good Evening";
} else if(@Hour24 >= 0 && @Hour24 <= 5){
    @Screen_Message = "Good Morning";
}
```

Below is the expected result.

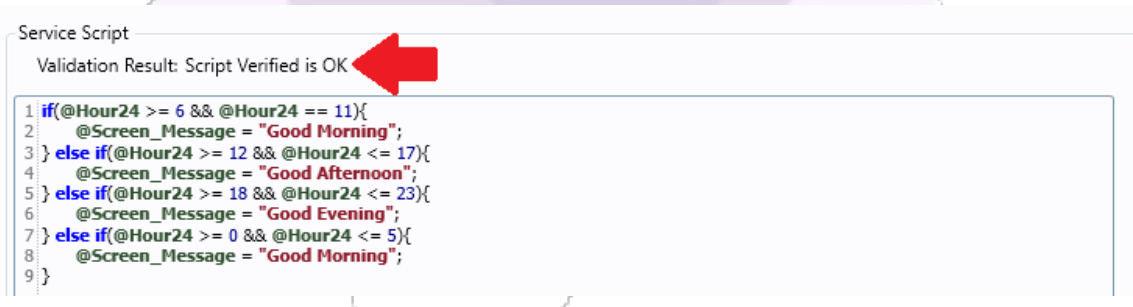


The above script will be executed every 5000ms once the RunTime is started, as the condition is set to “True”.

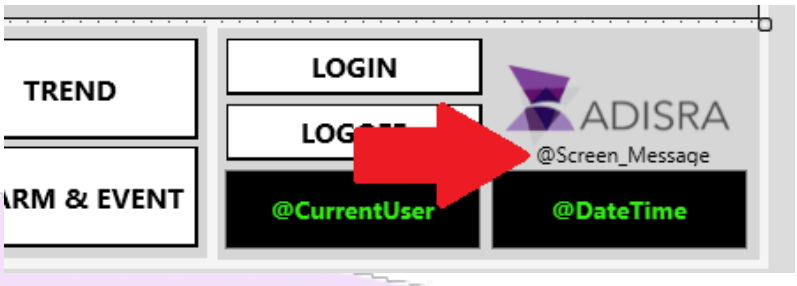
Basically, the script aims to assign the messages of “Good morning”, “Good afternoon” and “Good evening” depending on the time of the @Hour24 tag, which is a system tag.

Now, we need to add a Label object associated with the “Screen_Message” tag.

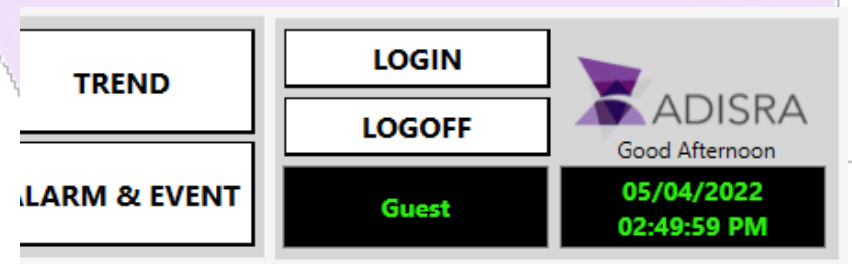
5. Before closing the “Services'” document, check the script for errors. To do this, press the keyboard keys “CTRL + S”. If there are no errors, a message will be shown above the script as shown in the image below.



6. Open the graphic “scr01_Main and insert a Label object and change the following properties: Location(827, 1545), Size(145, 15), Text(@Screen_Message, Segoe UI, 12) and Text Alignment(Center).

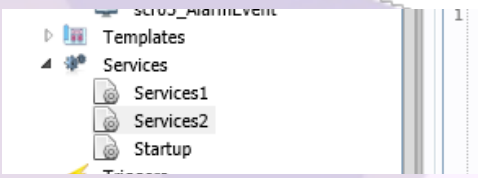


7. Now save and start RunTime. Below is the expected result.



23.5. Configuring the Trigger Type

1. In the navigation tree, right-click on the "Services" document and then click on the "New Document" option and save the new document as "Services2".



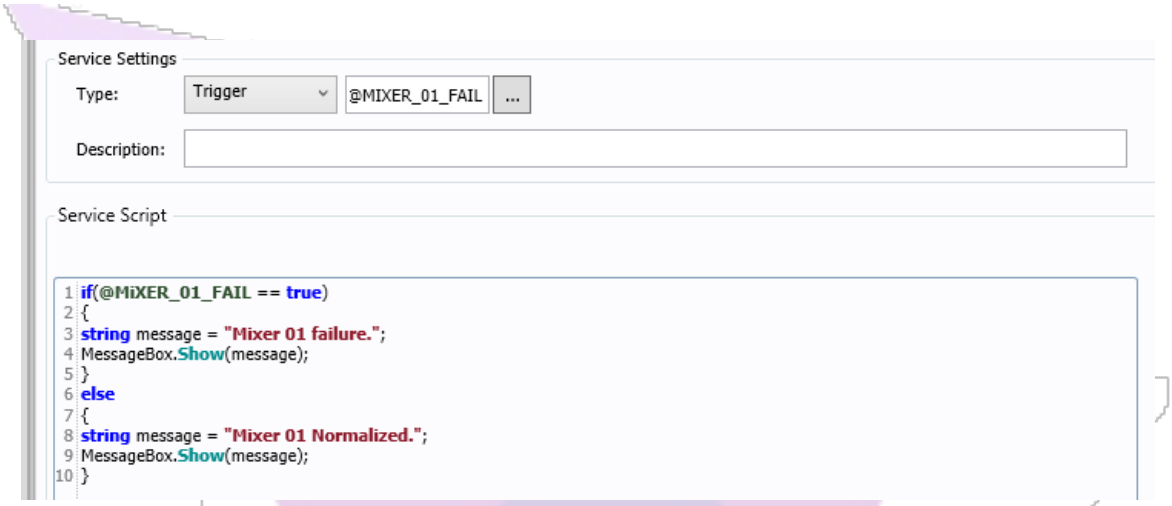
2. In the "Services2" document, change the following properties:

Enable: True
Type: trigger
Condition field: @MIXER_01_FAIL
Service Script:

```
if(@MiXER_01_FAIL == True)
{
string message = "Mixer 01 failure.";
MessageBox.Show(message);
}
else
{
string message = "Mixer 01 Normalized.";
```

```
MessageBox.Show(message);
}
```

Below is the expected result.

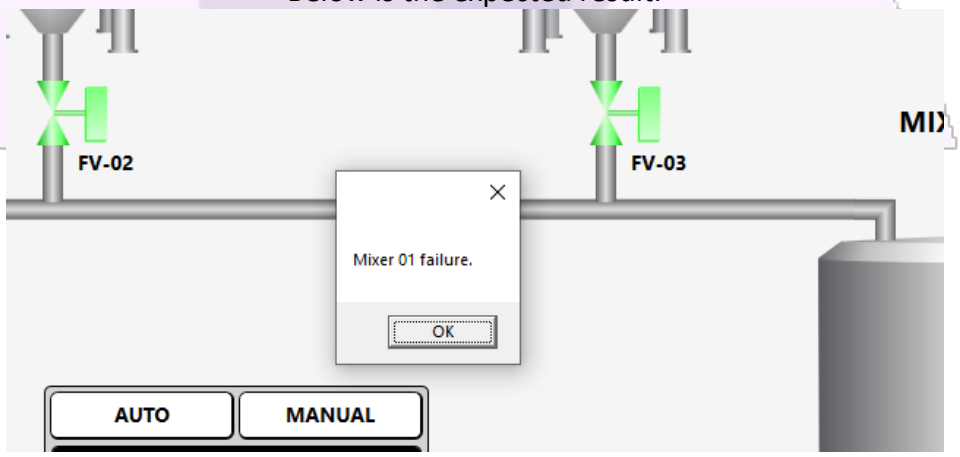


The above script will be executed whenever the @MIXER_01_FAIL tag changes the value from "False" to "True" or from "True" to "False".

Basically, the script aims to show a message box in the viewer. If the tag "@MIXER_01_FAIL" is "True", the message "Mixer 01 failure" will be displayed and if the tag is "False", the message "Mixer 01 Normalized" will be displayed.

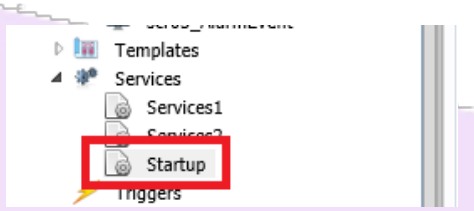
3. Save the application and start RunTime. Then change the value of the tag "@MIXER_01_FAIL" by the Data Watcher or the simulator.

Below is the expected result.



23.6. Configuring Service Startup

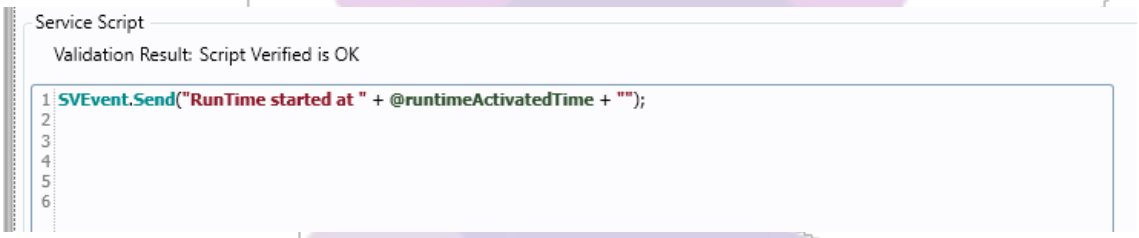
1. In the navigation tree, expand the “Services” item and click on the “Startup” document.



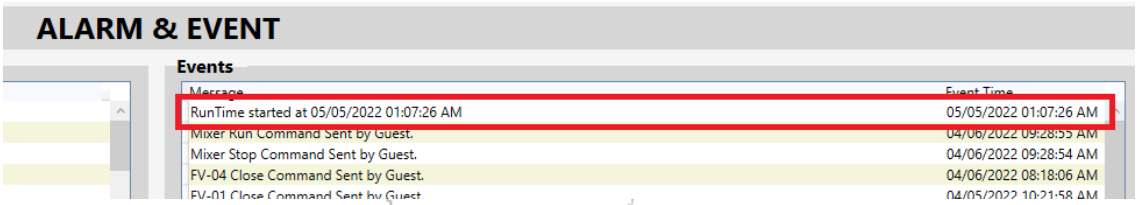
2. In the “Service Script” field, enter the following script:

   SVEvent.Send("RunTime started at " + @runtimeActivatedTime + "");

   Below is the expected result.



3. Now save the application and start RunTime. Access the “Alarm & Event” graphic and notice that the event was generated and displayed in the “AlarmEvent” object.

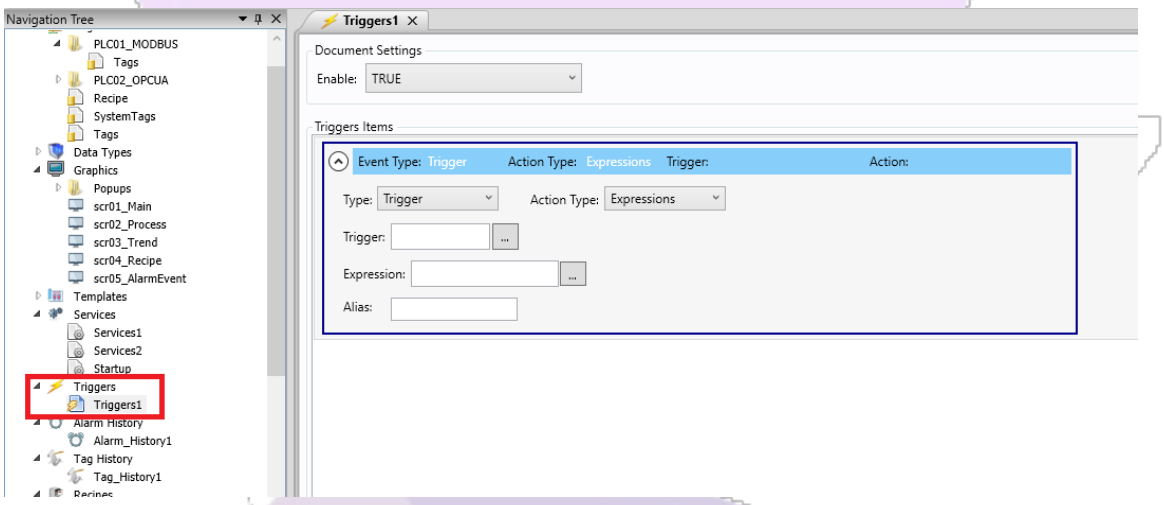


24. Trigger Document

Document Triggers is intended to run scripts, just like Document Services. However, this document has a few more features.

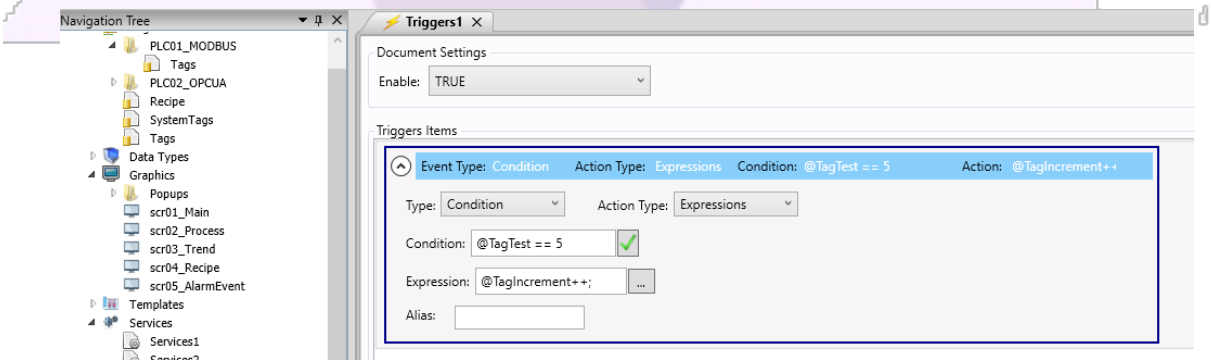
While Document Services has three execution types (Condition, Trigger and Startup), Document Triggers has four execution types (Condition, Trigger, Calendar and Interval).

It is also possible using Document Triggers to run Document Services.



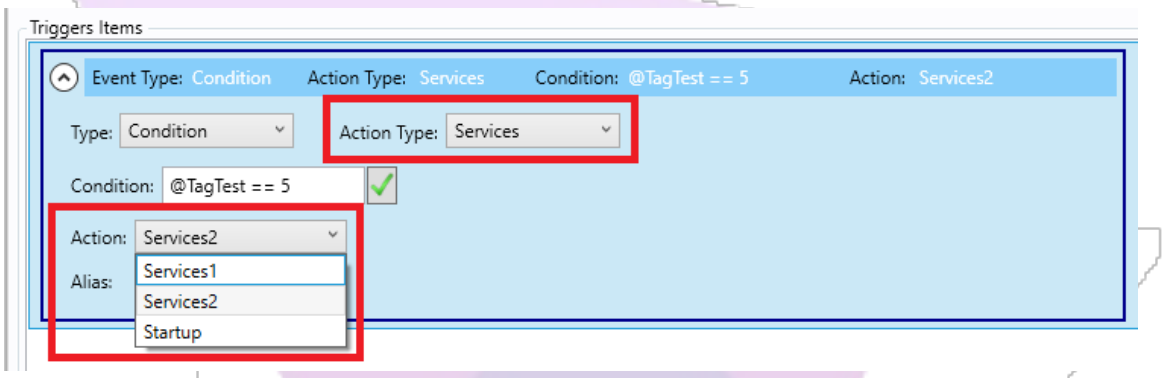
24.1. Condition Type

The “Condition” type executes a script if the condition is “True”. However, it is not possible to set the execution interval as in Document Services. By default, the execution interval is 300ms.



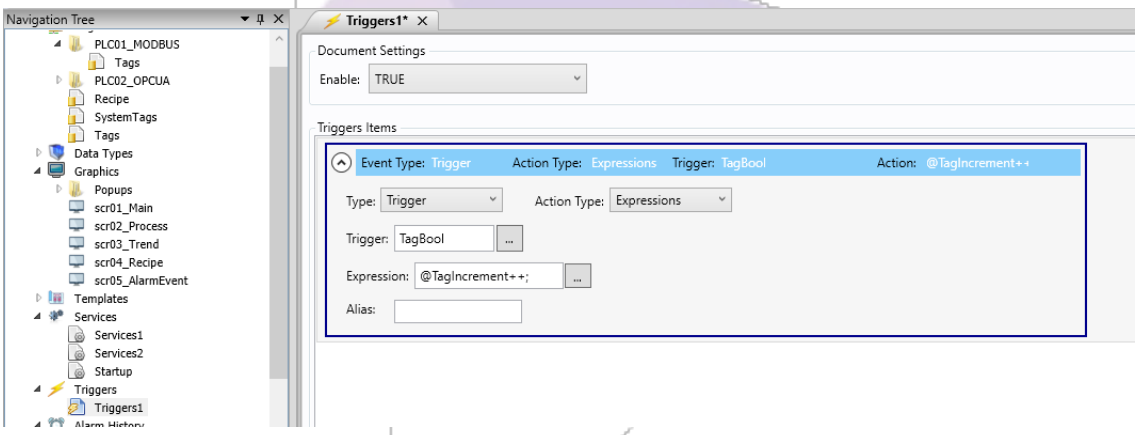
According to the example above, whenever the condition “@TagTeste == 5” is satisfied, the script will be executed every 300ms.

Also notice in the image below that there is a new field called “Action Type”, which allows executing a Services document from a Trigger document. If this option is selected, and a Document Services is defined, after the condition “@TagTest == 5” is satisfied, the Document Services will be executed.



24.2. Trigger Type

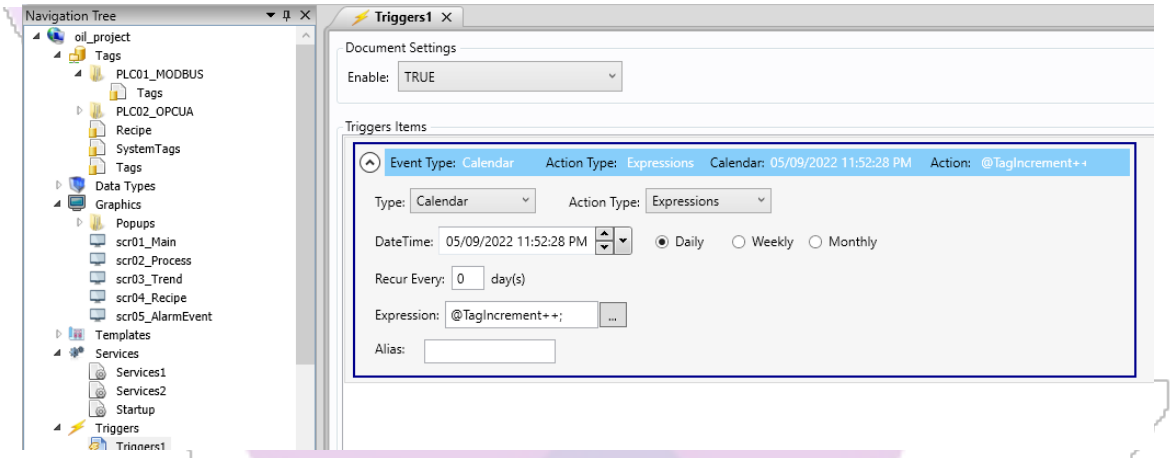
The “Trigger” type executes a script if the value of the associated tag is changed from “False” to “True” or “True” to “False”.



According to the example above, whenever the “TagBool” tag changes its value, the script will be executed. While in the “Condition” type the script will be executed every 300ms, in the “Trigger” type the script will be executed only once until the tag has its value changed again.

24.3. Calendar Type

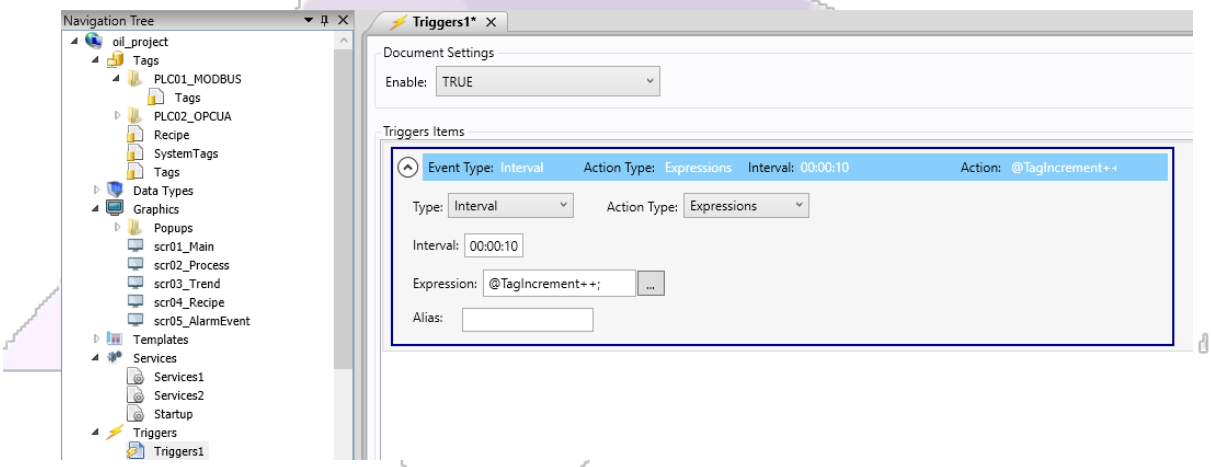
The “Calendar” type executes a script if the defined DateTime is satisfied. If the DateTime is satisfied, the script can be executed “daily”, weekly or “monthly”. The “Recur Every” field allows running the script repeatedly every “X” days, if it is filled with a value above one.



According to the example above, whenever the “DateTime” condition is satisfied, the script will be executed.

24.4. Interval Type

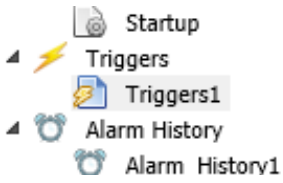
The “Interval” type delays the execution of a script, that is, we can treat it as a timer.



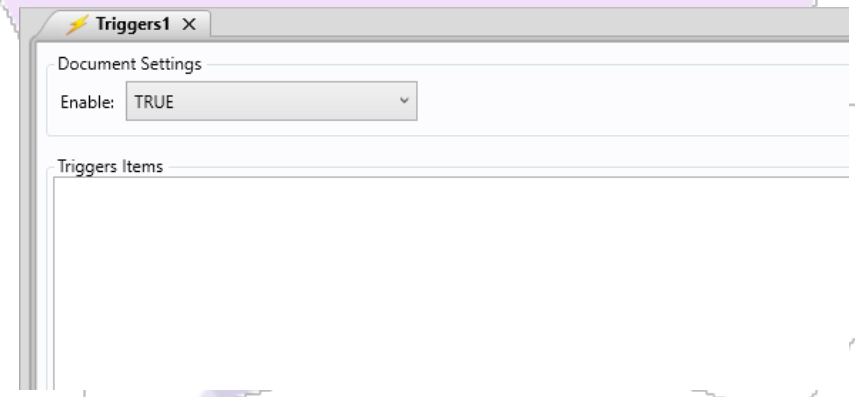
According to the example above, every “10 seconds” the script will be executed.

24.5. Configuring the Condition Type

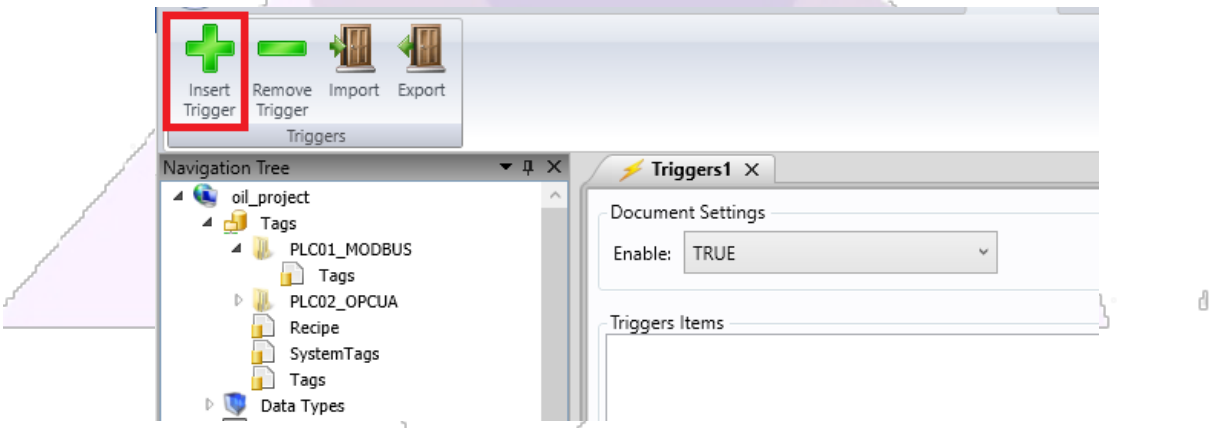
1. In the navigation tree, right-click on the “Triggers” document and then click on the “New Document” option and save the new document as “Triggers1”.



2. After creating the new document, note that within a single document it is possible to insert several triggers. Therefore, depending on the application, it is not necessary to create several Trigger Documents. It is possible to create a single Triggers Document in which you will have several Triggers items.



3. Now let's insert a new item by clicking the "Insert Trigger" button located at the top left of the engineering environment.



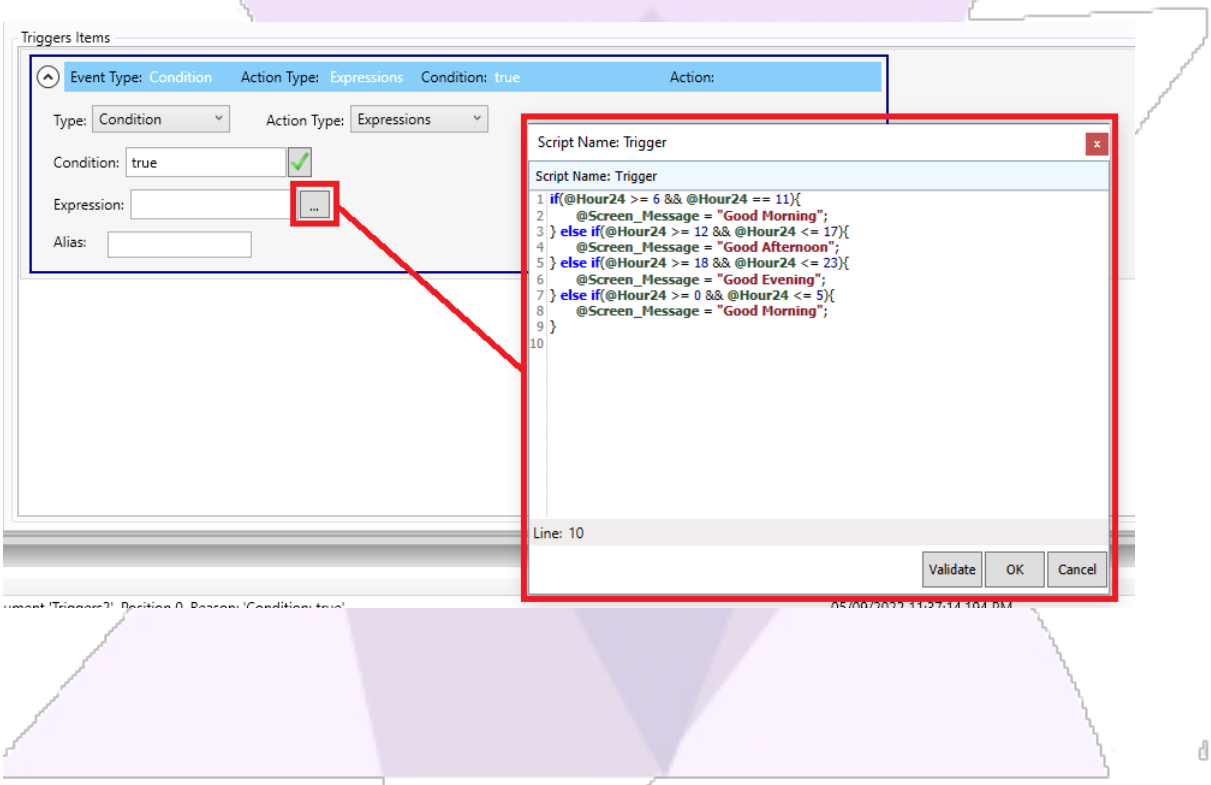
4. In the first item, change the following properties:

Type: Condition
ActionType: expression
Condition: True

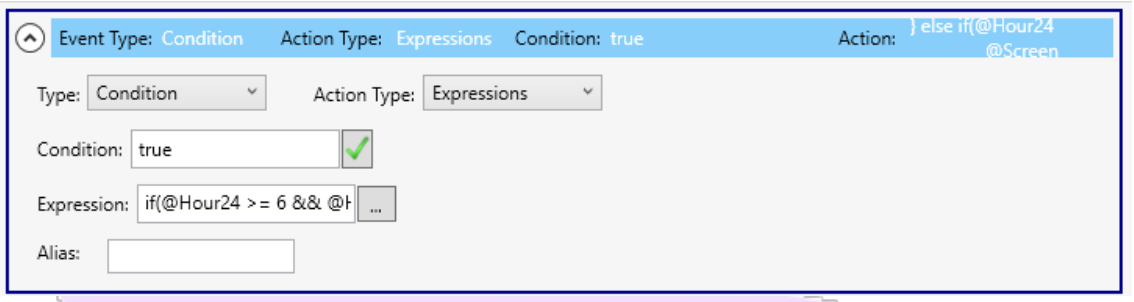
Express:

```
if(@Hour24 >= 6 && @Hour24 == 11){
      @Screen_Message = "Good Morning";
} else if(@Hour24 >= 12 && @Hour24 <= 17){
      @Screen_Message = "Good Afternoon";
} else if(@Hour24 >= 18 && @Hour24 <= 23){
      @Screen_Message = "Good Evening";
} else if(@Hour24 >= 0 && @Hour24 <= 5){
      @Screen_Message = "Good Morning";
}
```

note: *You can click the "..." button to open the script area.*



Below is the expected result.



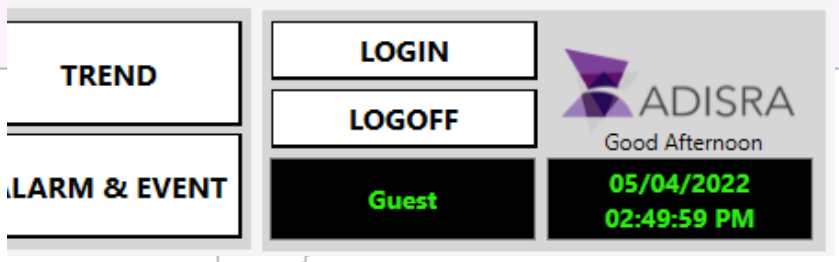
The above script will be executed every 300ms once the RunTime starts, as the condition is set to “True”.

Basically, the script aims to assign the messages of “Good morning”, “Good afternoon” and “Good evening” depending on the time of the @Hour24 tag, which is a system tag.

Now, we need to add a Label object associated with the “Screen_Message” tag.

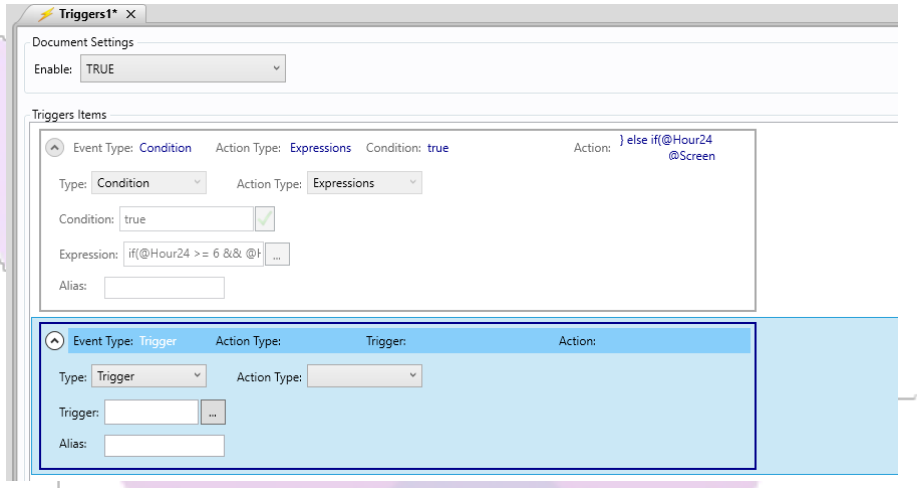
> ***note:*** *If we compare the Type “Condition” of the Document Service with the Document Trigger, it is possible to notice that the “Condition” of the Document Triggers will consume three times more CPU processing than the “Condition” of the Document Services. That's because the script is running every 300ms. Therefore, if the script does not need to be executed very fast, it is recommended to use the execution of a longer time interval, so as not to overload the application.*

5. Now, let's test the Trigger above. Save the application and then open the “Services1” document in which you have the same script configured above, and change the “Enable” property to False, as we want to visualize the script's operation by the Condition of the Triggers Document. Once that's done, start RunTime. Below is the expected result.



24.6. Configuring the Trigger Type

1. In the same Triggers Document called “Triggers1” insert one more item.

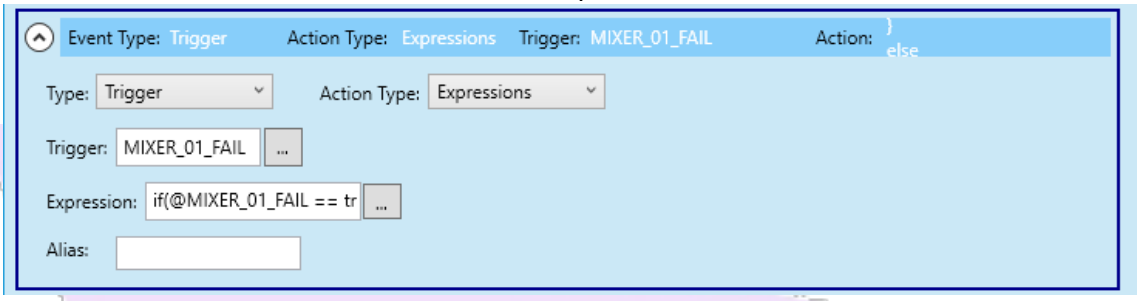


2. In item 02, change the following properties:

Type: Trigger
Action Type: Expression
Condition: True
Expression:

```
if(@MIXER_01_FAIL == True)
{
    string message = "Mixer 01 Failure.";
    MessageBox.Show(message);
}
else
{
    string message = "Mixer 01 Normalized.";
    MessageBox.Show(message);
}
```

Below is the expected result.

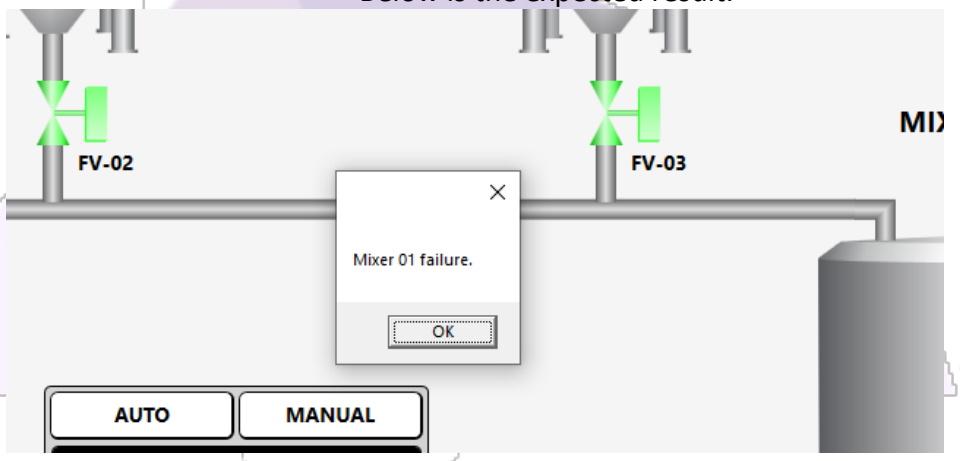


The above script will be executed whenever the @MIXER_01_FAIL tag changes the value from "False" to "True" or from "True" to "False".

Basically, the script aims to show a message box in the viewer. If the tag "@MIXER_01_FAIL" is "True", the message "Mixer 01 failure" will be displayed and if the tag is "False", the message "Mixer 01 Normalized" will be displayed.

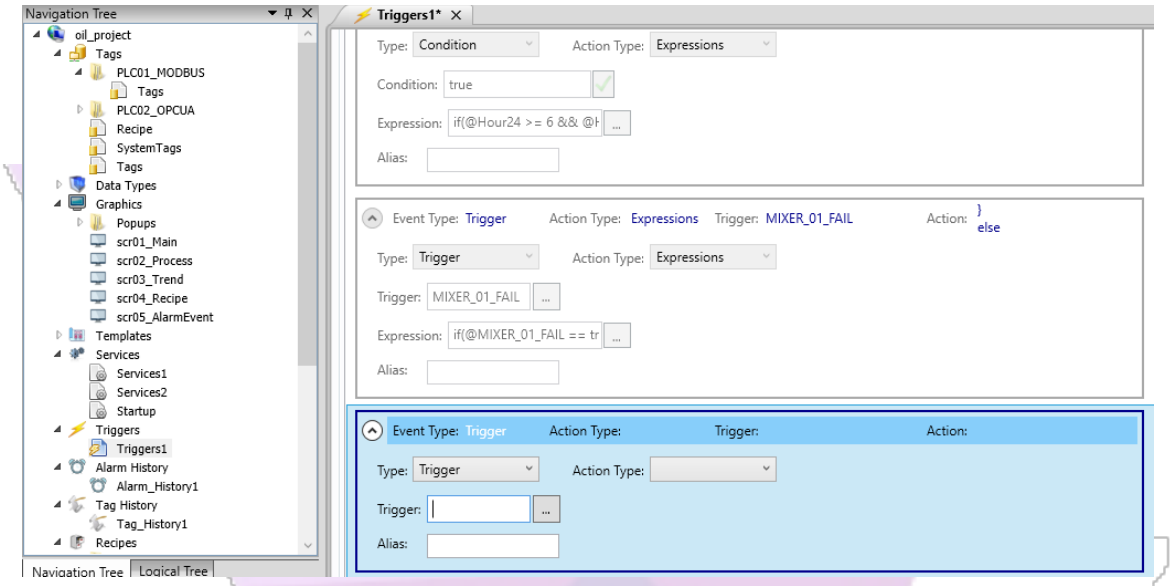
3. Now, let's test the Trigger above. Save the application and then open the "Services2" document in which you have the same script configured above, and change the "Enable" property to False, as we want to see how the script works by the Trigger of the Document Triggers. Once that's done, start RunTime.

Below is the expected result.



24.7. Configuring the Calendar Type

1. In the same Triggers Document called "Triggers1" insert one more item.

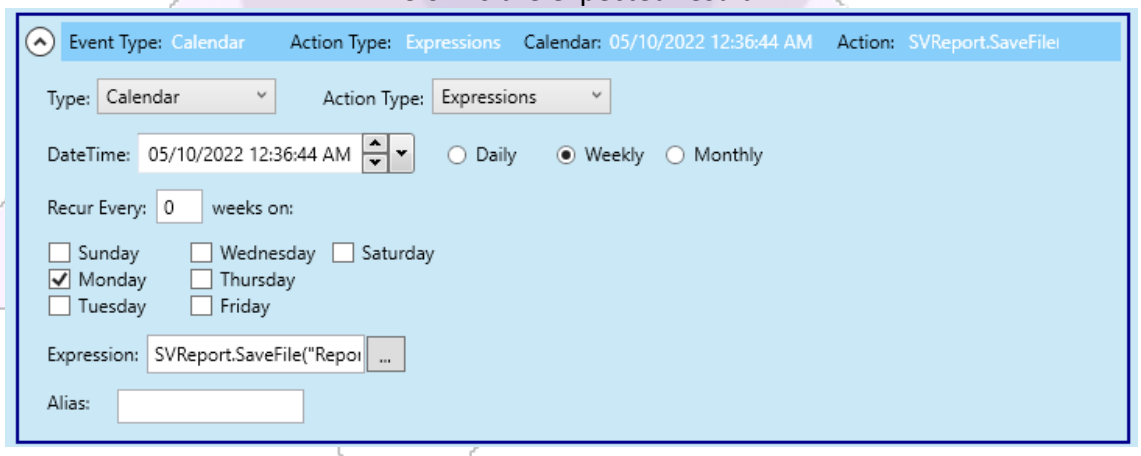


2. In item 03, change the following properties:

Type: Calendar
Action Type: Expression
DateTime: Weekly (Monday)
Expression:

SVReport.SaveFile("Reports1","C:\\Reports\\Report1.rtf");

Below is the expected result.

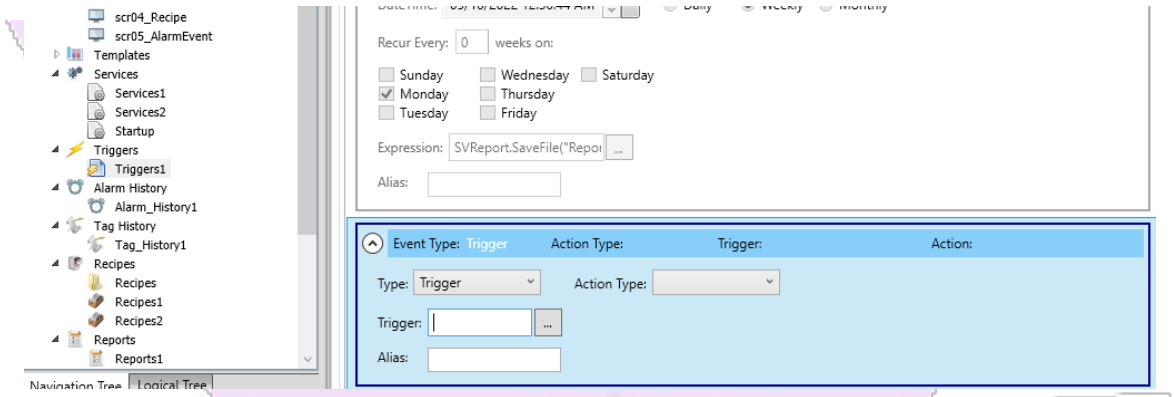


According to the above settings, whenever the "DateTime" condition is satisfied, that is, every Monday at the time defined in the DateTime field, the script will be executed.

3. This Trigger will not be possible to test because of the long time for the script to run.

24.8. Setting the Interval Type

1. In the same Triggers Document called “Triggers1” insert one more item.

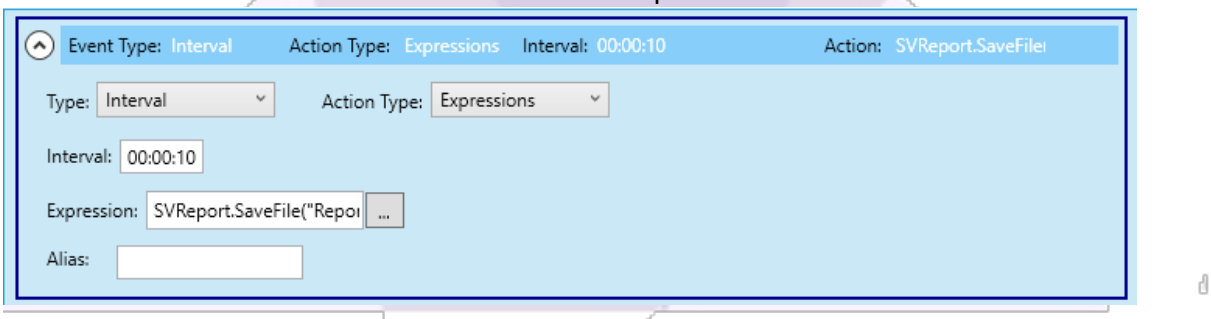


2. In item 04, change the following properties:

Type: Interval
Action Type: Expression
Interval: 00:00:10
Expression:

```
string path = (SVApplications.ProjectPath() +"Reports\\Reports1.rtf");
SVReport.SaveFile("Reports1", path);
```

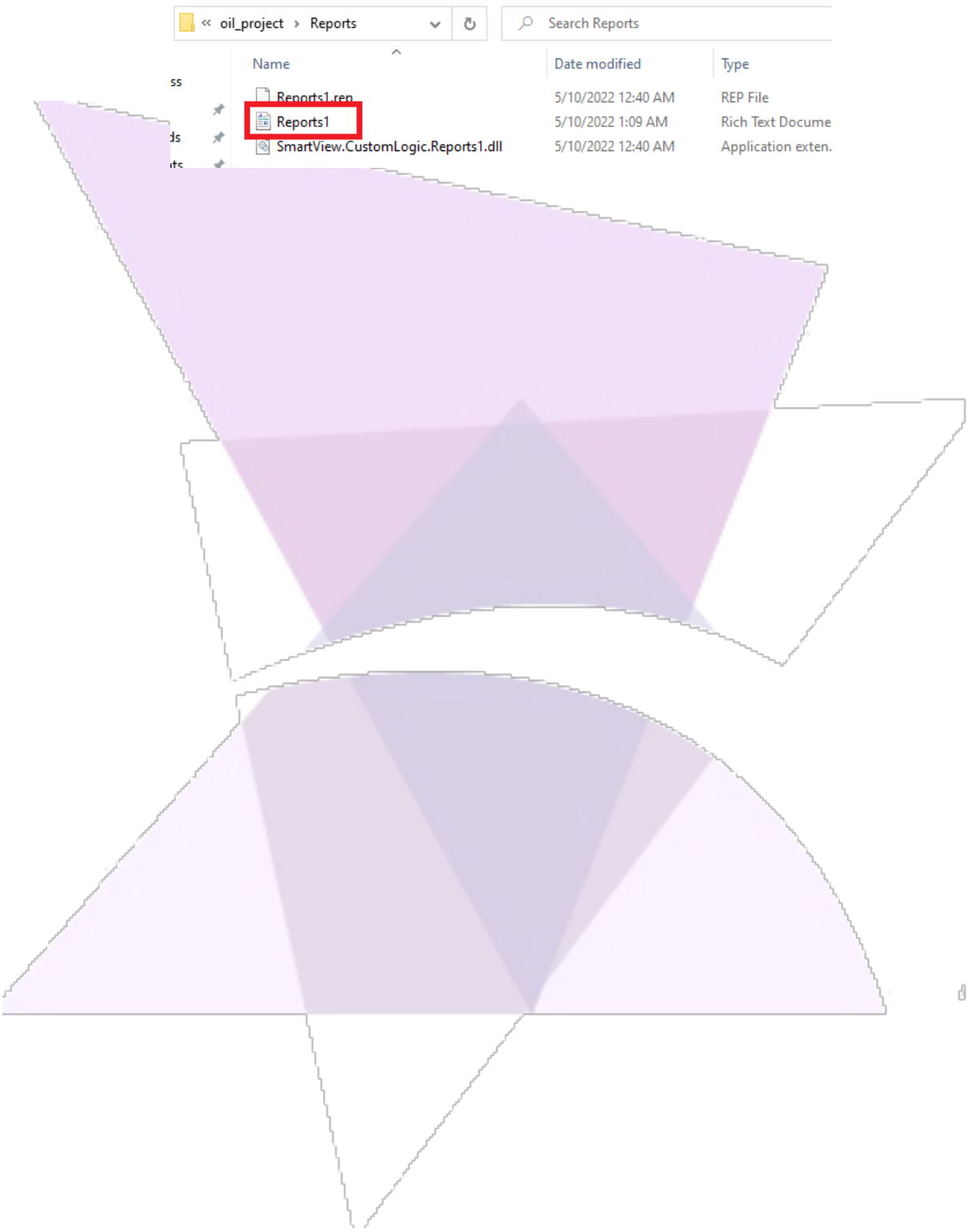
Below is the expected result.



The above script will be executed every 10 seconds, that is, the “Reports1” will be saved in the Report folder inside the main project folder.

3. Now, let's test the Trigger above. Save the application and start RunTime. Then open the configured path and wait for the “Report.rtf” file to be generated every 10 seconds. If nothing appears, try to update the folder by clicking the “F5” button on the keyboard.

Below is the expected result.

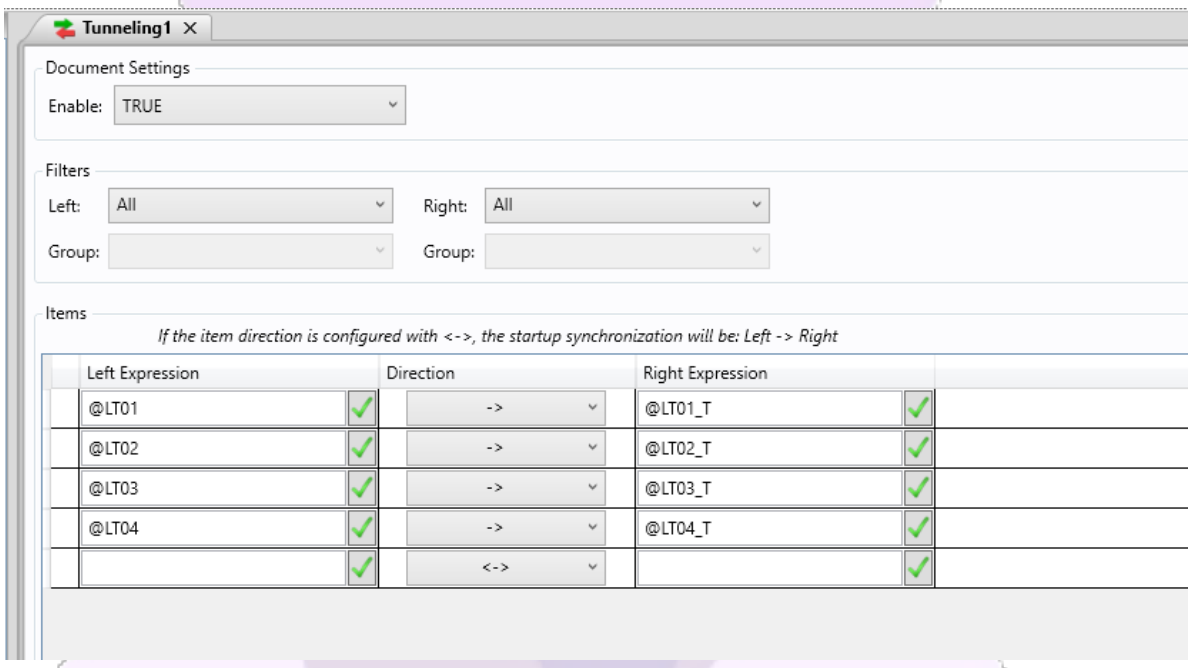

« oil_project › Reports
Search Reports
Name
Date modified
Type
Reports1.rep
5/10/2022 12:40 AM
REP File
Reports1
5/10/2022 1:09 AM
Rich Text Docume
SmartView.CustomLogic.Reports1.dll
5/10/2022 12:40 AM
Application exten.

25. Tunneling Document

The Tunneling Document aims to associate the value of a tag with another tag. For example, if you want to move the value from tag LT-01 to tag LT-02, you can use this document. Therefore, every time the LT-01 tag updates its value, Document Tunneling will update the LT-02 tag's value.

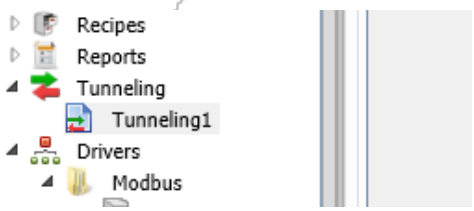
For our hypothetical project, we will add the Tunneling document for knowledge purposes only.

Below is a demonstration of how the “Tunneling” document will look.

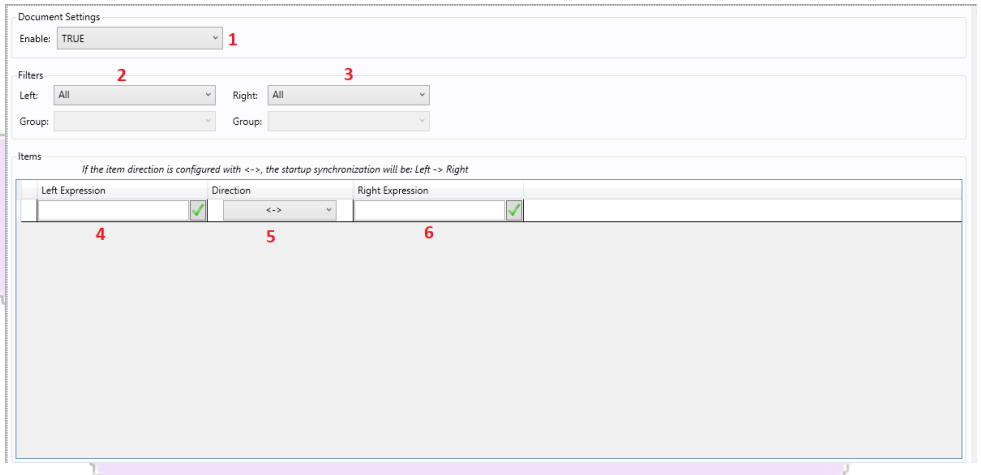


25.1. Configuring Document Tunneling

1. In the navigation tree, right-click on the “Tunneling” document and then on “New Document”. Save the new document as “Tunneling1”.



2. Before continuing the step-by-step, let's get to know each configuration field of the “Tunneling” document.



1- Enables or disables Document Tunneling;
2- Filters the "Left Expression" column tags;
3- Filters the "Right Expression" column tags;
4- Field to insert the first tag of the association;
5- Defines the meaning of the association;
6- Field to insert the second tag of the association.

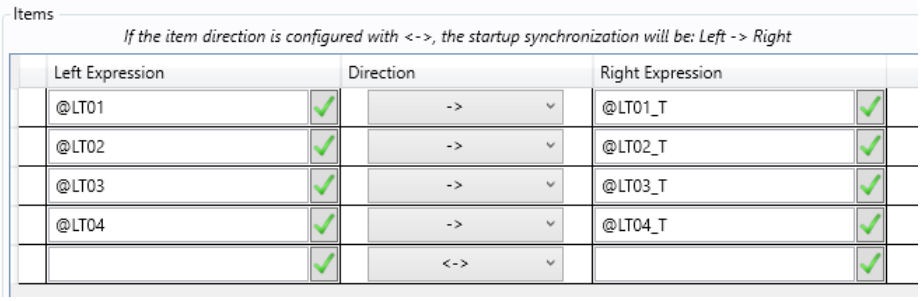
Now that we know each field of the "Tunneling" Document, let's create, configure and associate the tags in the document's tag list.

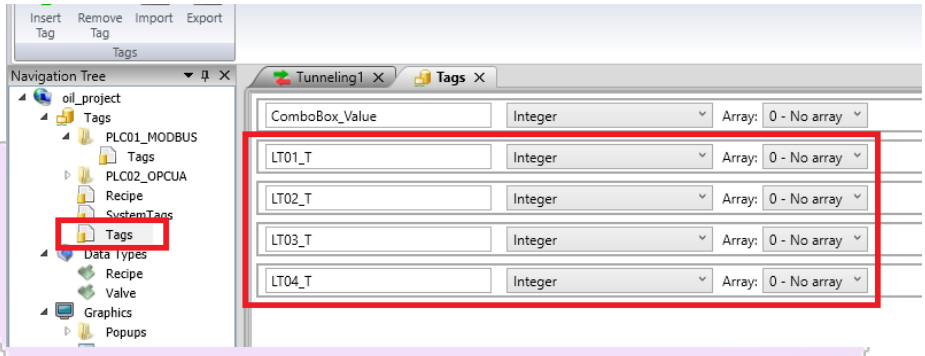
3. In the "Tunneling" document, change the following properties:

Enable: True
Filters Left: All
Filters Right: All

4. Now let's add the association tags. In the "Left Expression" column add the "LT's" tags and in the "Right Expression" column create the new tags shown below the "Integer" type in the "Tags" document. In the "Direction" column, select the " -> " direction for all associations.

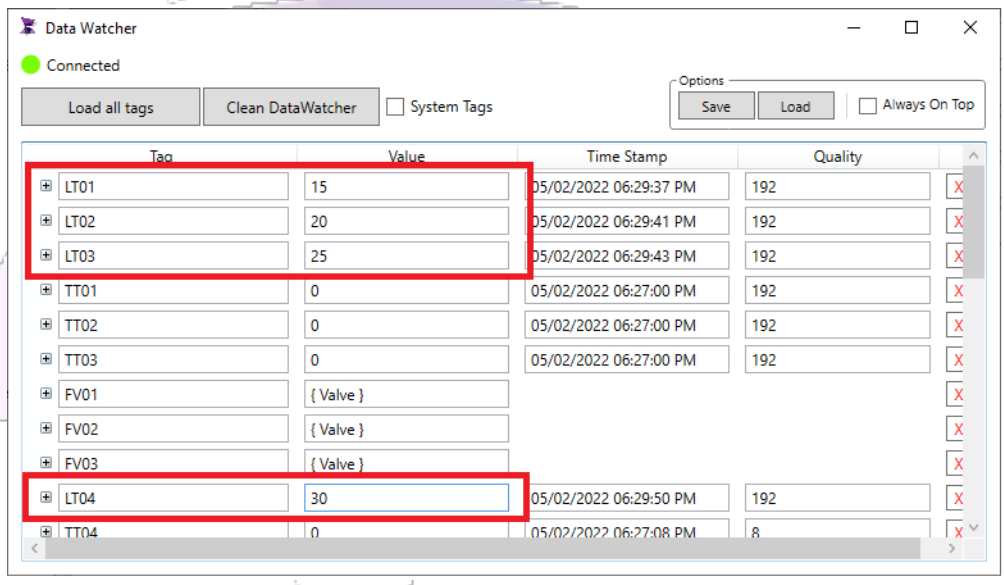
Below is the expected result.



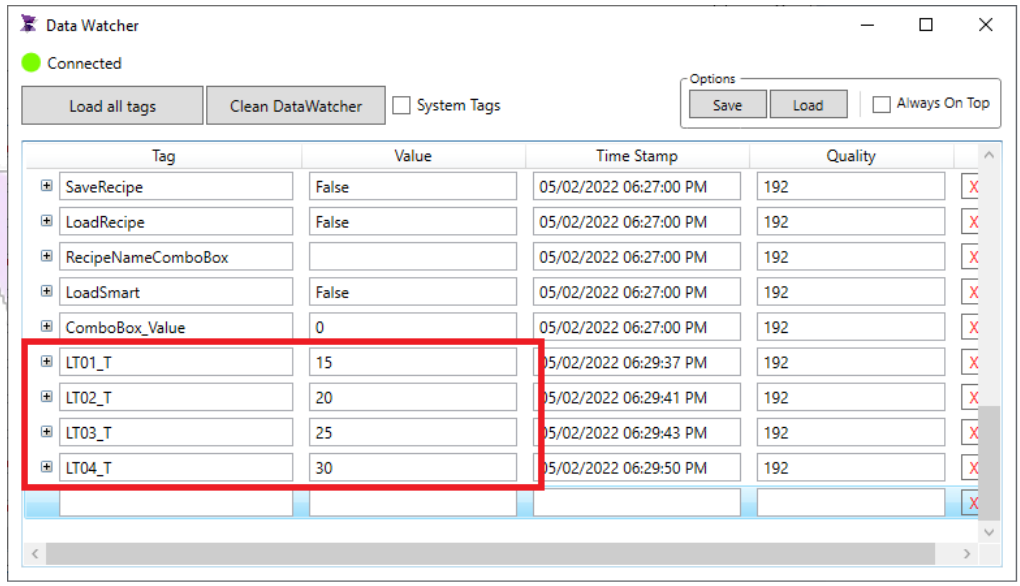


From now on, after starting the runtime, the value of Tags LT01 to LT04 will be associated with the new tags LT01_T to LT04_T. The association is made in the "Tag Change" of the value of the tags. Therefore, whenever the application is started, the first association will be from the "Left Expression" tags to the "Right Expression" tags. The next association will only be made when the tag value is updated.

5. Once the settings are done, let's test. Save the application and start RunTime. Open Data Watcher and click the "Load all tags" button. To test manually, it will be necessary to disable the communication drivers and save the application again. But if you want to test with the communication drivers enabled, change the values of the LT01 to LT04 tags directly on the simulators. Enter the following values:



6. Then look in Data Watcher for the new associated tags. All tags must have the same values, as Document Tunneling moved the values in the "Tag Change". Below is the expected result.



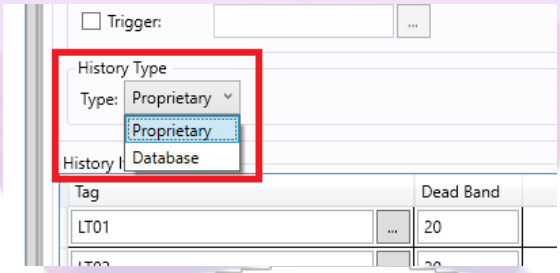
So, whenever you need to move the value of a tag to another tag, you can use Document Tunneling.

26. Database

26.1. What is a Database?

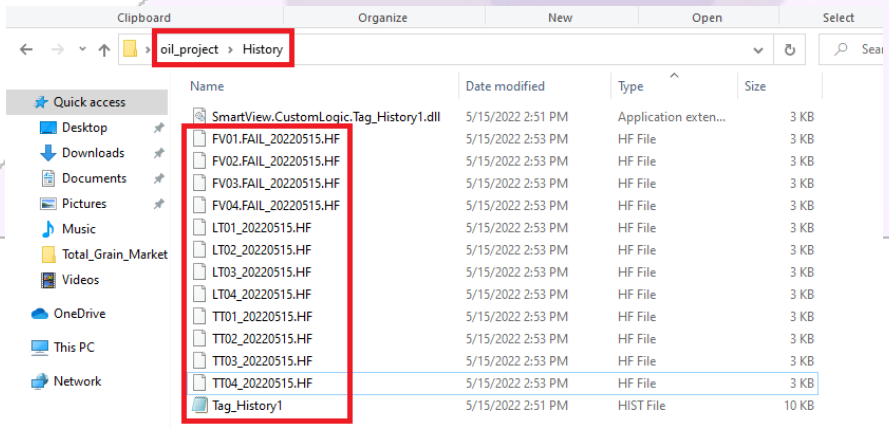
The Database is an organized collection of structured information. Database Software allows you to create, edit, maintain and record information, for example from ADISRA SmartView. Database example: SQL Server, MySQL, PostGres, SQLite3, Oracle, and Microsoft Access.

In addition to the Databases mentioned above, which are called external Databases, ADISRA SmartView has a proprietary Database, which was used throughout our project to store data from Tag History, Alarm History, and Events documents.

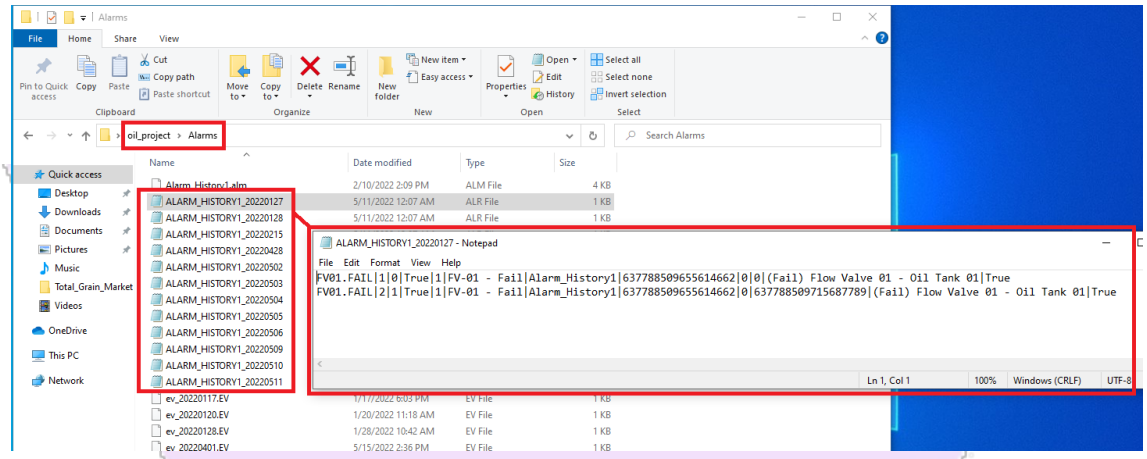


26.2. Proprietary Database

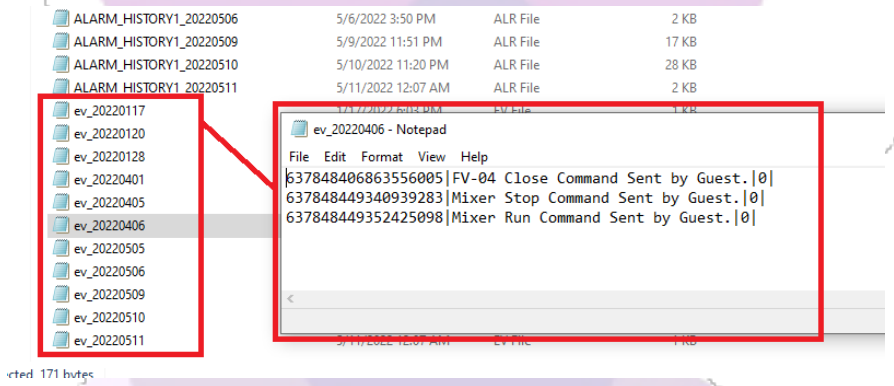
The Owner Database stores the data in the project folder. The stored data of the Tag History document can be found in the “oil_project/History” folder.



The stored data of the Alarm History document can be found in the “oil_project/Alarms” folder.



The data generated by the Event System can be found in the “oil_project/Alarms” folder.



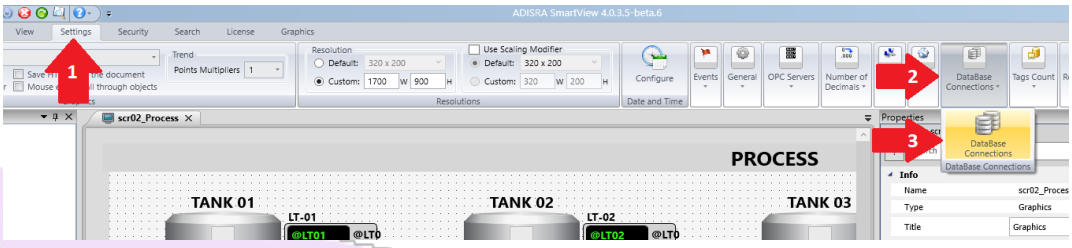
note[1]: *Data stored in the Owner Database can be queried by the Datagrid object.*

note[2]: *The proprietary Database does not have functions for SQL commands such as Select, Insert, Delete, etc.*

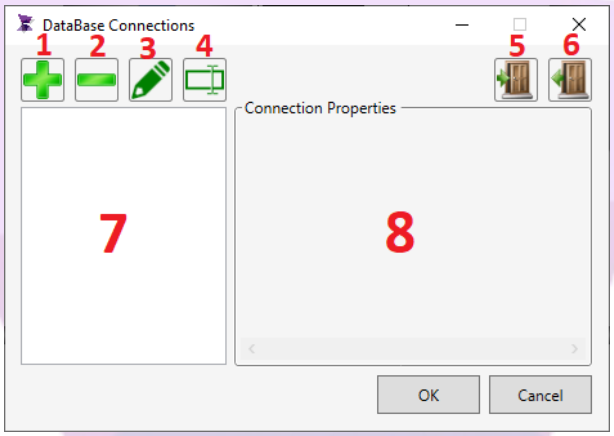
26.3. External Database

ADISRA SmartView allows you to connect to external databases such as SQL Server, MySQL, PostGres, SQLite3, Oracle and Microsoft Access. This connectivity will not be done in our hypothetical project, but for the sake of knowledge, below will be shown how to connect a SQLServer database.

1. On the top menu, click on “Settings”, “DataBase Connections” and then “DataBase Connections”.

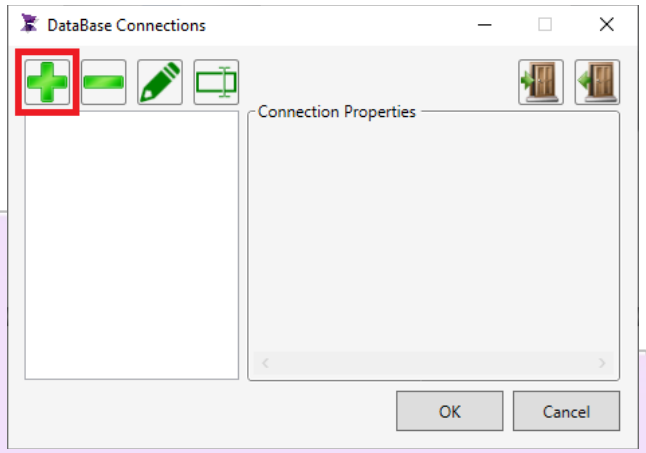


Below is the Database connections window. Before continuing, let's get to know the interface of this window.

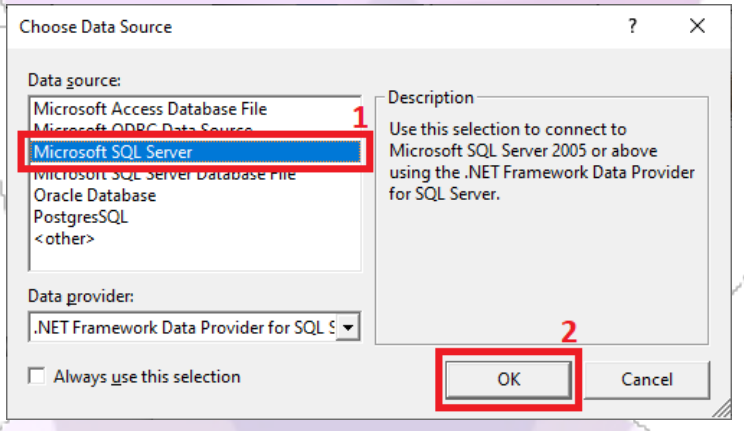


1- Adds a new connection;
2- Removes a selected connection;
3- Edit a selected connection;
4- Edit the name of the selected connection;
5- Import a configured connection;
6- Export a configured connection;
7- Shows all available connections;
8- Shows connection properties selected.

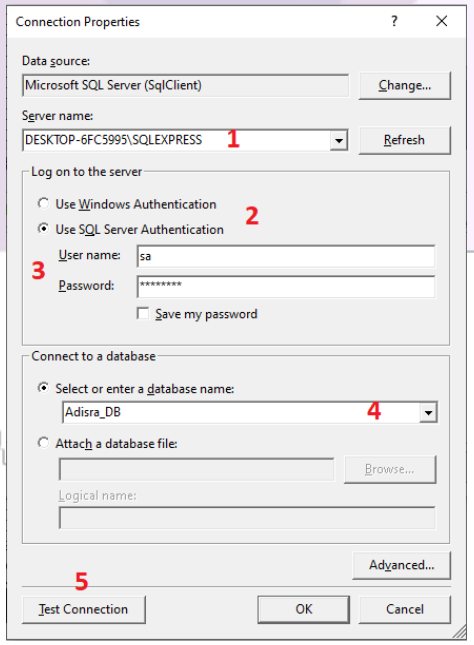
2- Click the “+” button to add a new connection.



3- Select the Data Source “Microsoft SQL Server” and then click “OK”.

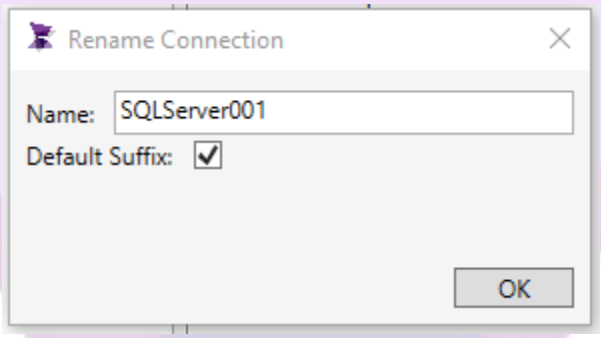


4- Now, fill in the required fields:

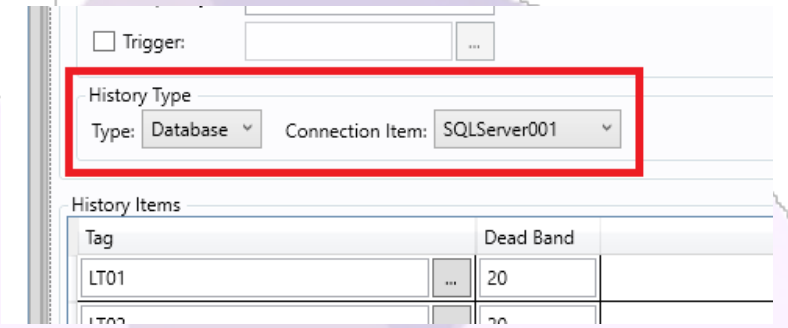


1. Server name: Enter the Server Name;
2. Choose the authentication type;
3. Enter the UserName and Password of the type of selected authentication;
4. Select the Database already created in SQL Server;
5. Test the Communication with the Database and then click OK.

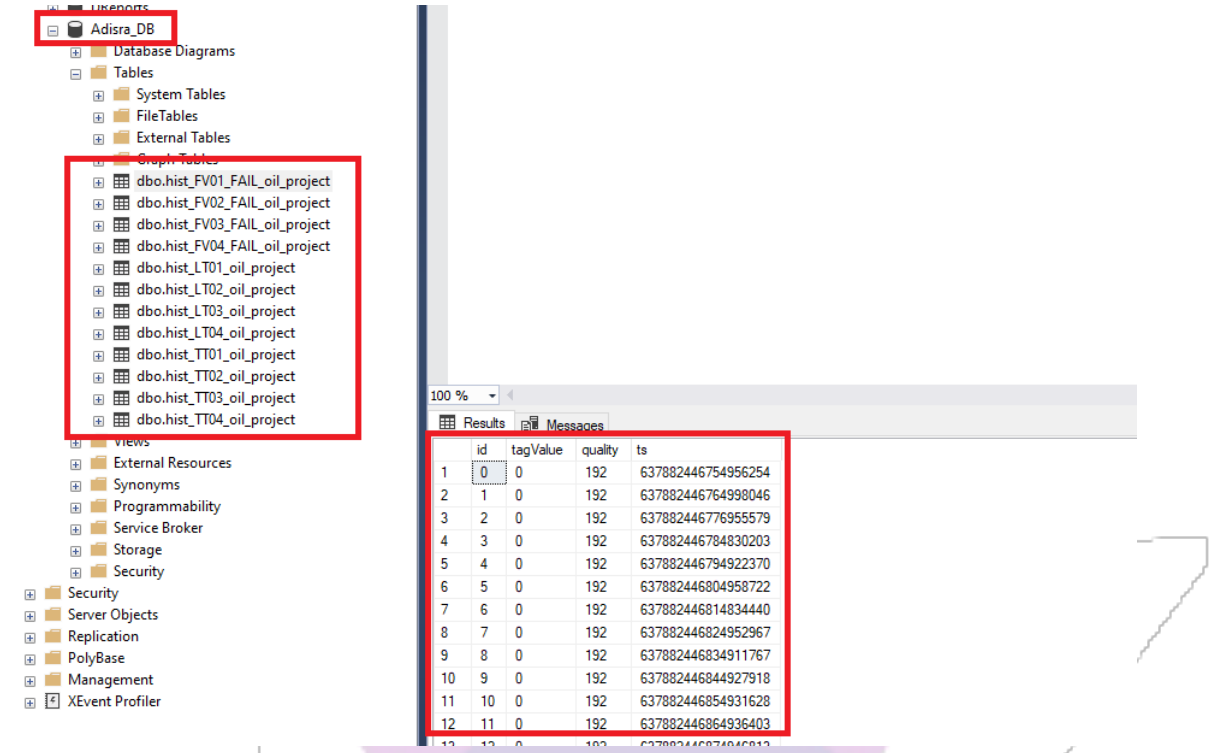
5- Then keep the connection name and click OK.



6- Ready! Now the SQL Server Database is connected with ADISRA SmartView. Now, access any document that is using the proprietary database and change it to the SQLServer001 database as shown in the image below.



From now on, all tags of the document “TagHistory1” will be history in the SQL Server database named “ADSIRA_DB”. It is worth mentioning that the data will only be stored in the SQL Database after starting the RunTime and the chosen “Save Modes” condition is satisfied. Tables will be generated automatically. Below is the expected result.


Adisra_DB
Database Diagrams
Tables
System Tables
FileTables
External Tables
dbo.hist_FV01_FAIL_oil_project
dbo.hist_FV02_FAIL_oil_project
dbo.hist_FV03_FAIL_oil_project
dbo.hist_FV04_FAIL_oil_project
dbo.hist_LT01_oil_project
dbo.hist_LT02_oil_project
dbo.hist_LT03_oil_project
dbo.hist_LT04_oil_project
dbo.hist_TT01_oil_project
dbo.hist_TT02_oil_project
dbo.hist_TT03_oil_project
dbo.hist_TT04_oil_project
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
100 %
Results
Messages
id tagValue quality ts
1 0 0 192 637882446754956254
2 1 0 192 637882446764998046
3 2 0 192 637882446776955579
4 3 0 192 637882446784830203
5 4 0 192 637882446794922370
6 5 0 192 637882446804958722
7 6 0 192 637882446814834440
8 7 0 192 637882446824952967
9 8 0 192 637882446834911767
10 9 0 192 637882446844927918
11 10 0 192 637882446854931628
12 11 0 192 637882446864936403

27. Web Viewer

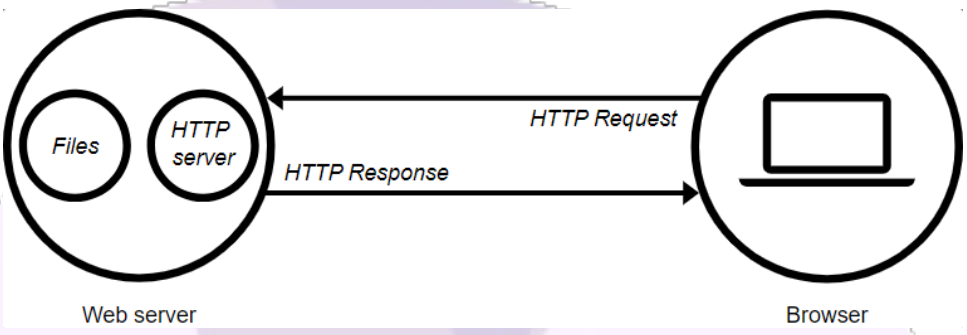
ADISRA SmartView allows you to convert Graphics from an application to Web pages (HTML5 and JavaScript). With this feature it is possible to view the graphics in RunTime Mode through a web browser. However, for this feature to work, it is necessary to use a Web Server.

27.1. What is Web Server?

The Web Server is a software responsible for storing files from a website or web application. These files make up the site (e.g. HTML documents, images, style sheets, and JavaScript files).

Basically, the Web Server uses the Client-Server concept. The web browser (Client), located on a local or remote machine, uses the HTTP or HTTPS protocol to request data from the Web Server (Server). In turn, the Server will send the requested data also via HTTP or HTTPS.

The most used Web Servers are: Microsoft-IIS, Apache and Nginx.



note: *Both Web Viewer and Viewer Local are called "Viewer Session".*

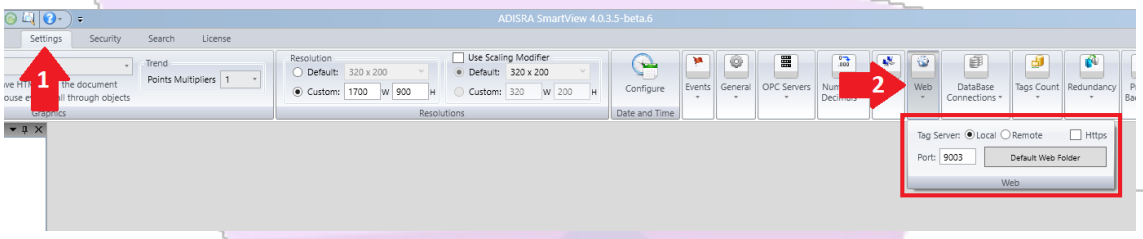
27.2. Generating Graphics for the Web

To view graphics on the web in RunTime mode, it is necessary to convert them into web pages (HTML5 and JavaScript). This conversion is done in the ADSIRA SmartView engineering environment.
It is worth mentioning that, after the conversion, the web pages will be generated, however, it will be necessary to install and configure a Web Server to view them in the web browser. However, this guide will not show you how to configure any Web Server, but only

the settings for generating web pages. For more information on how to configure a Web Server, visit our Knowledge Base.

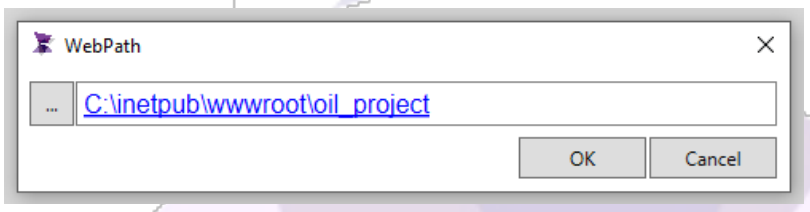
We recommend following the steps below to configure the generation of web pages:

1- On the upper tab of the engineering environment, click on the “Settings” option and then on the “Web” button.

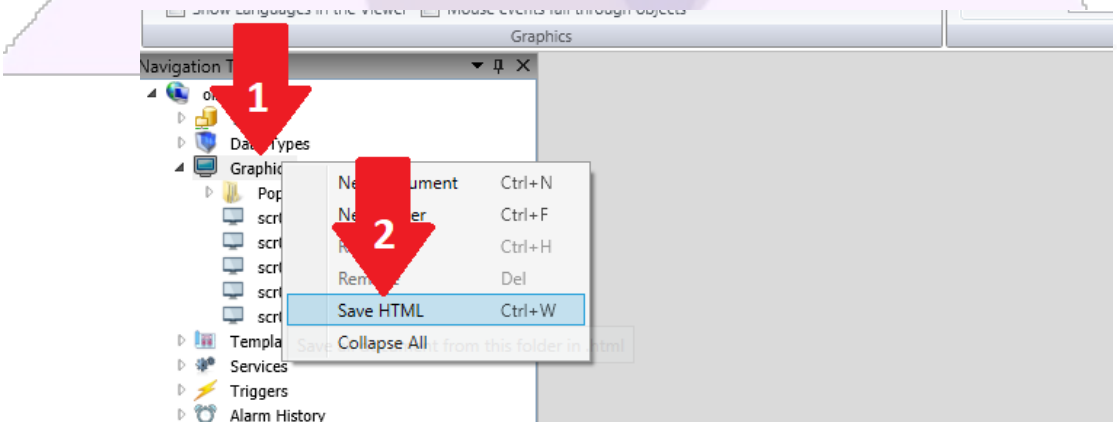


1. **Port:** It is the port used for the Web Server.
2. **Default Web Folder:** It is the folder where all web pages will be generated.

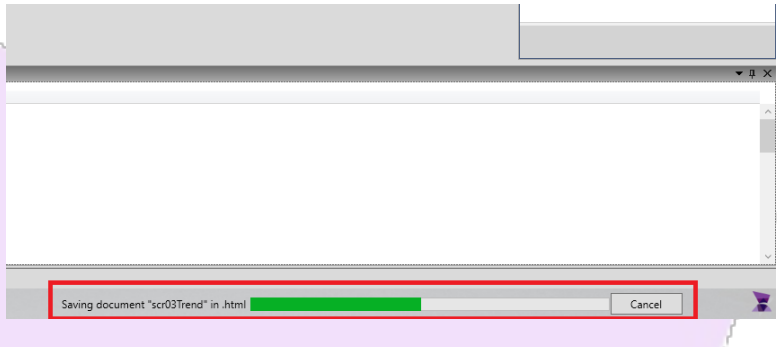
2- Then click on the “Default Web Folder” button. Below is the directory in which the web pages will be generated.



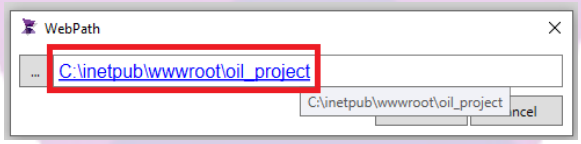
3- Now let's generate the WEB pages. After developing and saving the application, close the window above and right-click on the “Graphics” Document and then on the “Save HTML” option. All screens will be generated as web pages.



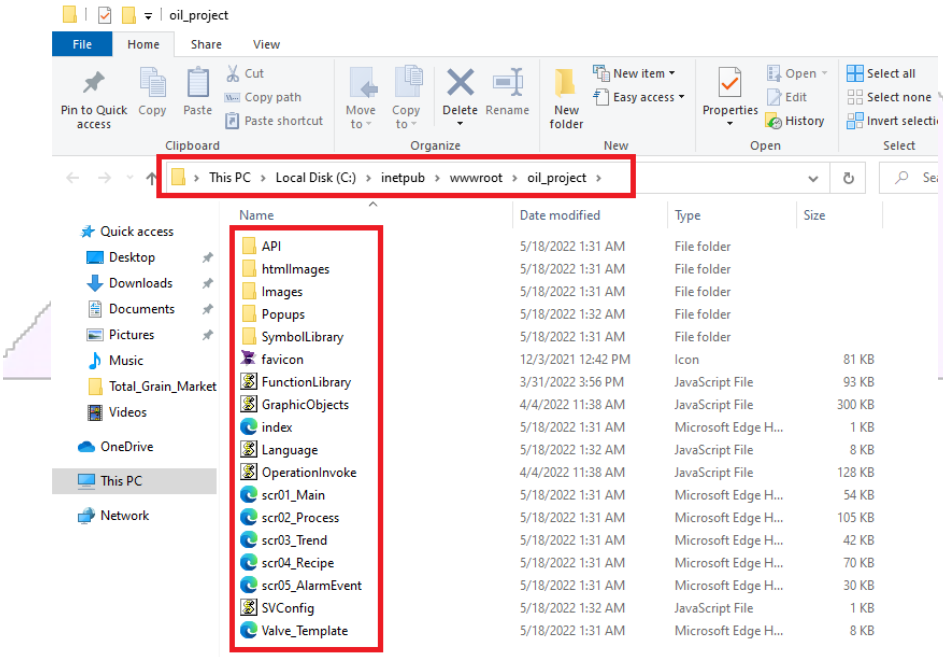
note: *In the lower right corner of the engineering environment, it is possible to follow the progress of the generation of web pages in the directory.*



4- After finishing generating the web pages, return to the web settings and then click on the directory below. A folder will open.



5- Ready, the web pages were generated. Now, choose install and configure a Web Server to view application screens.



note: *After generating the web pages and configuring the Web Server, run the "scr01_Main" page to view the application as it is done in Viewer Local. It is worth mentioning that the RunTime must be in "Run".*